МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ
по дисциплине «Разработка мобильных приложений»
для студентов направления
10.05.03 «Информационная безопасность автоматизированных систем»
Направленность (профиль)
«Разработка автоматизированных систем в защищенном исполнении»

Ставрополь

Содержание

ВВЕДЕНИЕ

Целью дисциплины «Разработка мобильных приложений» является формирование набора профессиональных компетенций будущих бакалавров по направлению подготовки 10.05.03 «Информационная безопасность автоматизированных систем».

Задачами дисциплины являются:

– ознакомление студентов с видами пакетов прикладных программ и классами программного обеспечения;

– обучение студентов постановке задач, корректному и эффективному использованию инструментальных средств;

– приобретение студентами навыков самостоятельного и последовательного применения пакетов прикладных программ в профессиональной деятельности;

– формирование логического мышления.

Лабораторная работа №1. Знакомство с Android Studio

Цель работы: Изучение интерфейса Android studio и создание первогопростого приложения.

Теоретическая часть

Необходимо вспомнить основные принципы ООП.

Существует огромное количество компонентов, из которых можно собрать приложение. В Android studio они представлены, как виджеты. Основные из них:

- TextView – поле, в котором будет отображаться текст;
- Button – кнопка;
- ProgressBar – индикатор прогресса;
- EditText – поле ввода данных;
- ChekBox – особый тип кнопки, который может быть в одном издвух состояний (checked или unchecked);
- RadioButton – подобный CheckBox тип кнопки, с одним исключением, что используется в RadioGroup, где Checked можно присвоить

лишь одному экземпляру RadioButton;

- Toast – небольшое всплывающее сообщение;
- ListView – список строк.

Ход лабораторной работы:

Процесс создания приложения состоит из следующих этапов.

1) Скачиваем и устанавливаем Android studio https://developer.android.com/studio

Процесс установки включает в себя многократное нажатие «Далее». Следует учесть, что имя пользователя компьютера должно быть только на английском языке. Дело в том, что среда Android Studio не поддерживает формат UTF-8, из-за чего возникнут проблемы во время установки.

Рисунок 1 - Главное окно установки

Установка является самой сложной частью программирования на Android. Установка разбивается на несколько этапов:

- установка основного компонента;
- установка SDK-менеджера;
- установка компонентов для сборки;
- установка эмулятора.

2) Запускаем Android studio и создаём приложение.

Android Studio обладает некоторыми предустановленными шаблонами, чтобы проектировать приложения. Сейчас мы рассмотрим самое простое приложение, поэтому, выберите Empty Activity.

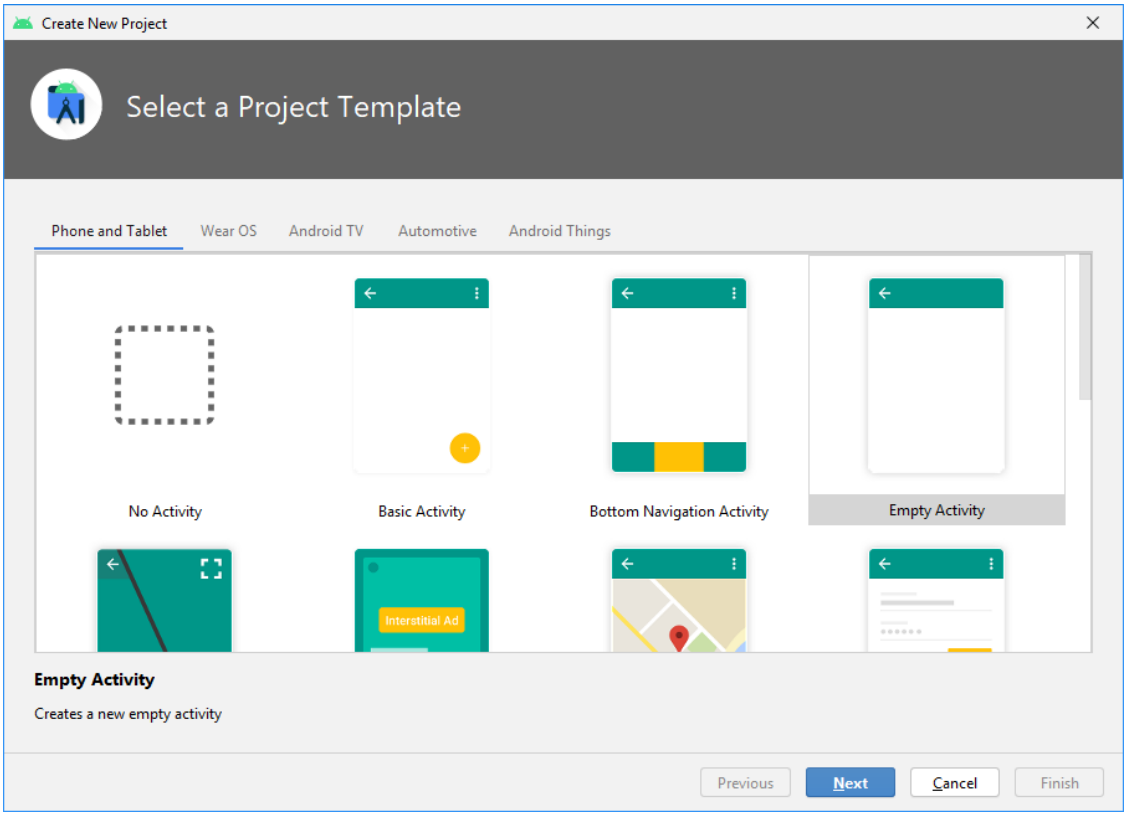


Рисунок 2 – Выбор макета приложения

Нажав «Next», перейдём к окну выбора места сохранения проекта.

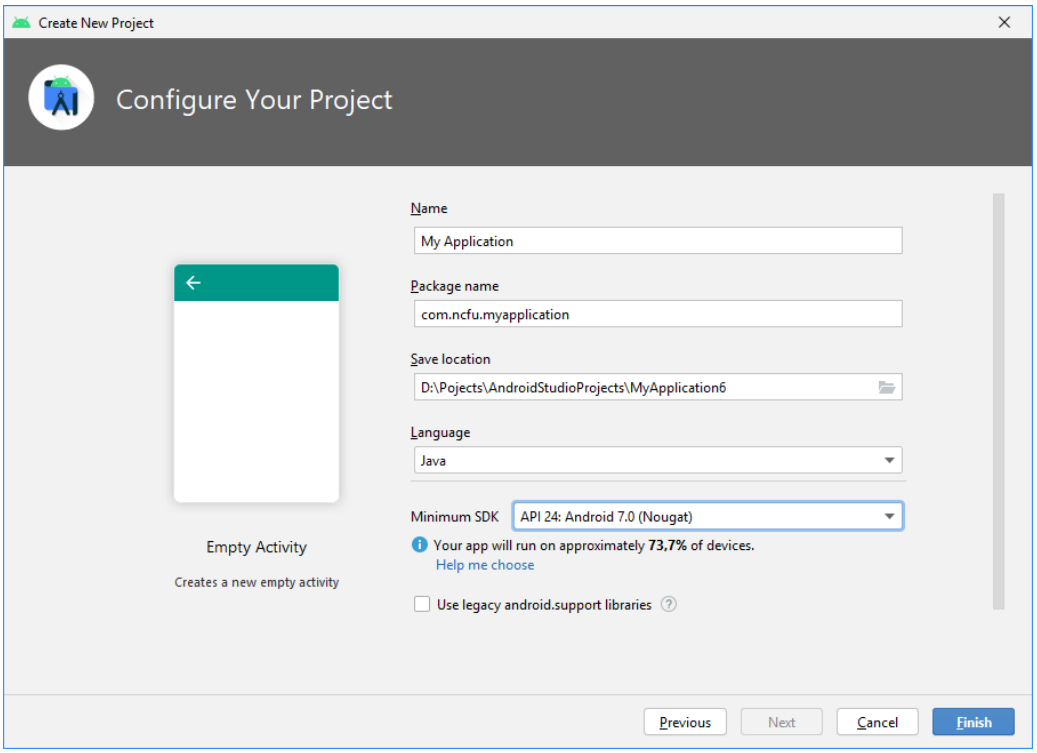


Рисунок 3 – Выбор места создания проекта

3) Далее нажимаем «Финиш».

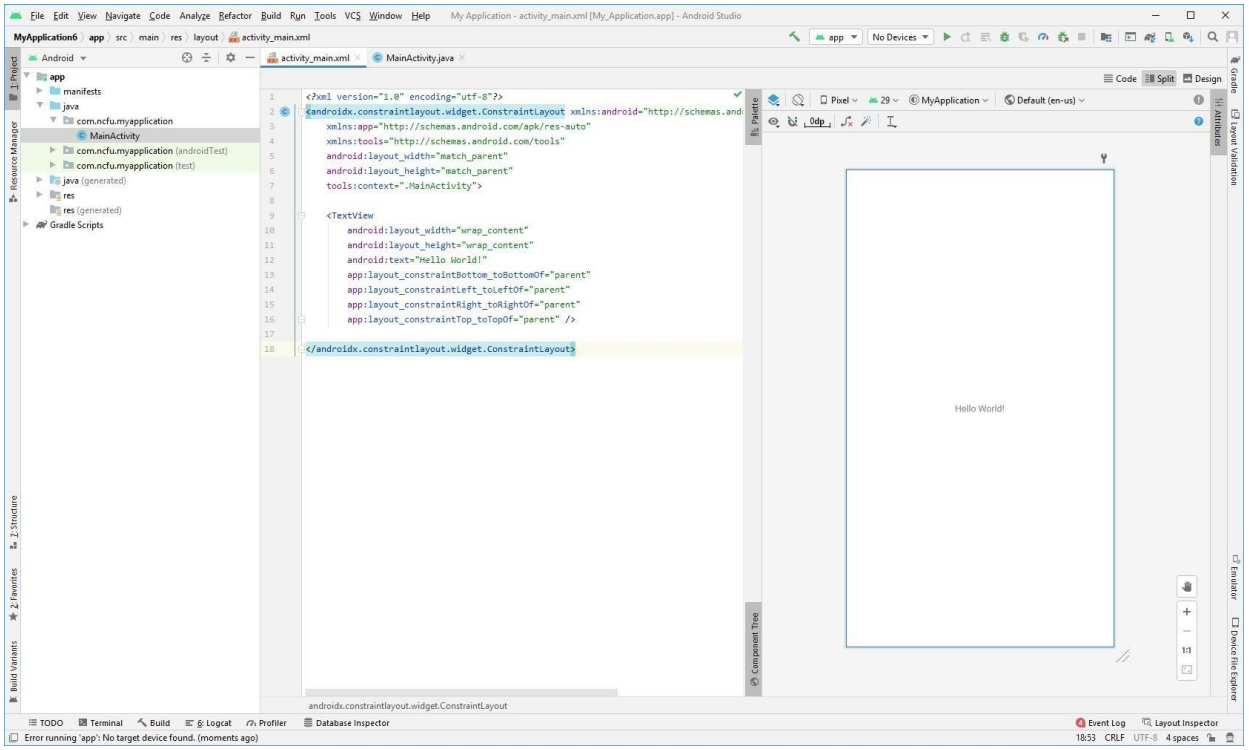


Рисунок 4 – Главное окно приложения

Прежде чем начать программировать, запустите приложение, чтобы проверить, что все компоненты были установлены правильно. В верхней строке меню выберите No Devices→AVD Manager.

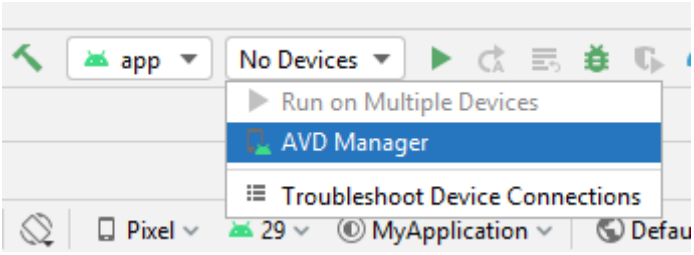


Рисунок 5 – Выбор виртуального устройства

Во время первого запуска необходимо выбрать эмулятор (Если Вы собираетесь тестировать своё приложение на своём андроид-устройстве, необходимо: включить свойства разработчика на устройстве, поставить галочку рядом с пунктом «Разрешить отладку по USB»).

Так как мы хотим проверить сейчас на эмульяторе, рассмотрим данный процесс подробнее.

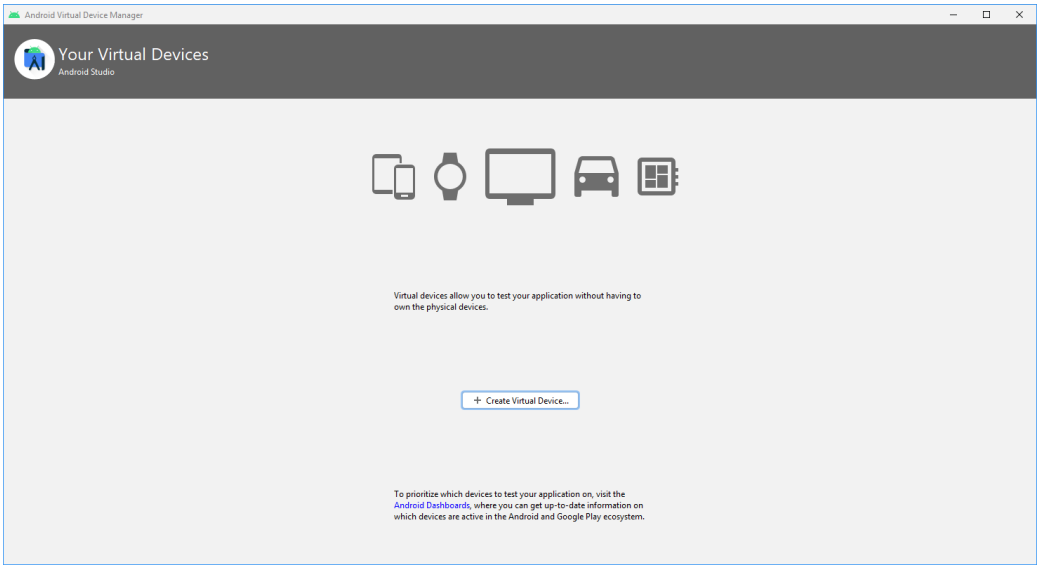


Рисунок 6 – Выбор типа устройства эмулятора

Нажав на «Create Virtual Device», выберите эмулятор под те характеристики, которые Вам более удобны. Например, создадим своё собственное устройство. Найдём в интернете любое устройство на Android (например, Samsung F700 Galaxy Z Flip 8). Перепишем характеристики в новом окне. Нажмите «New Hardware Profile».

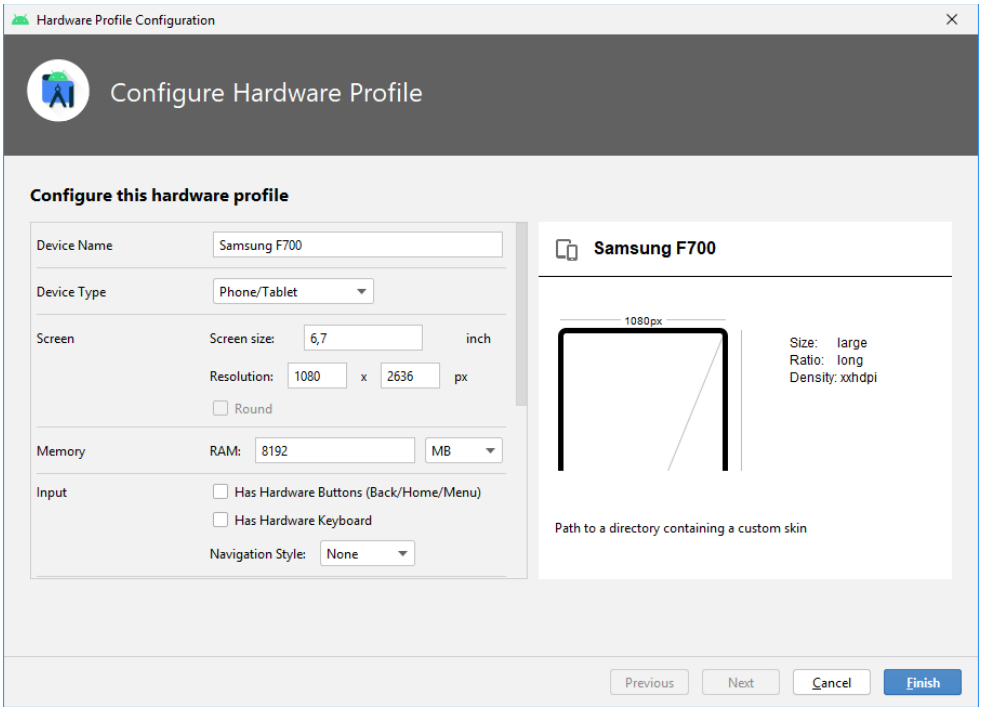


Рисунок 7 – Внесение изменений эмулятора

Теперь необходимо выбрать версию Android для нашего устройства. Скачаем последнюю актуальную версию Android 11.

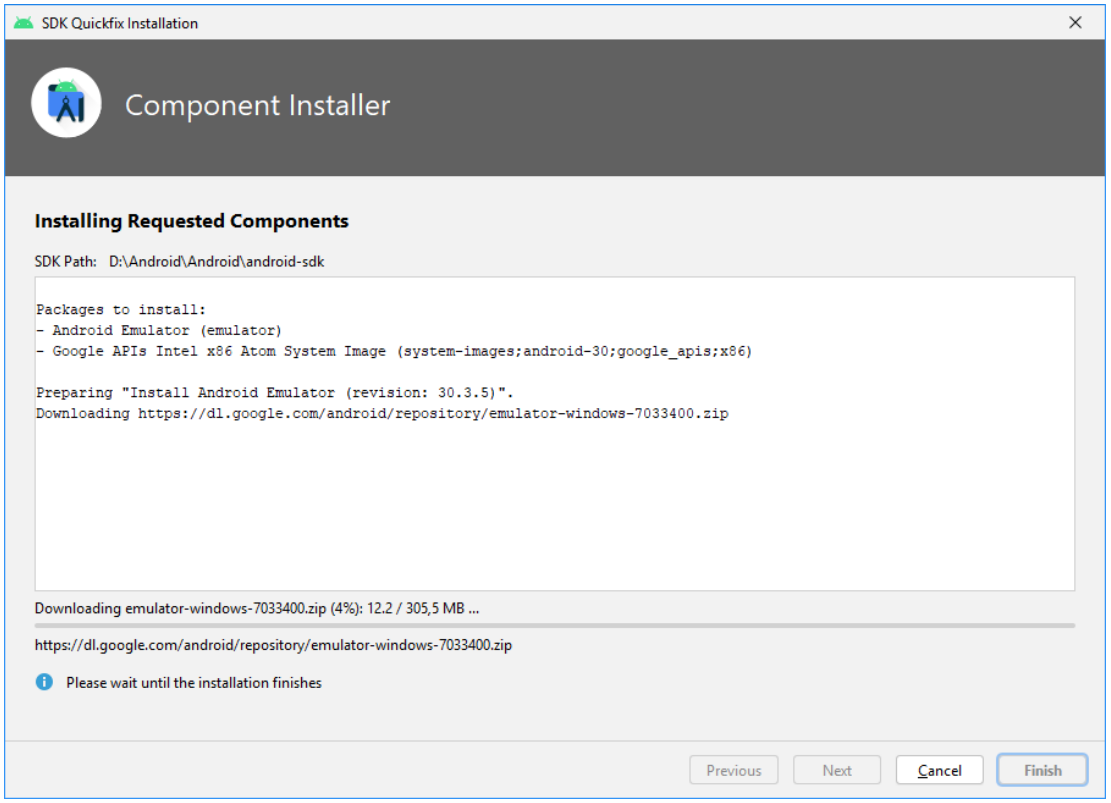


Рисунок 8 – Скачивание Android 11

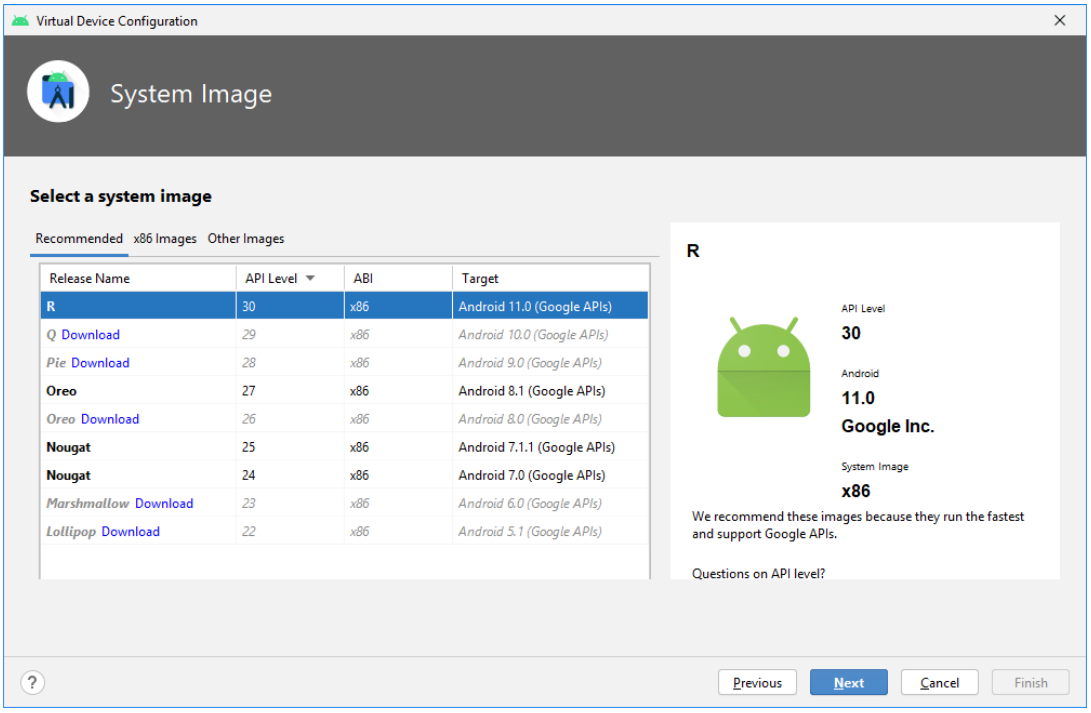


Рисунок 9 – Выбор версии Android

После создания эмулятора оно будет доступно в списке AVD.

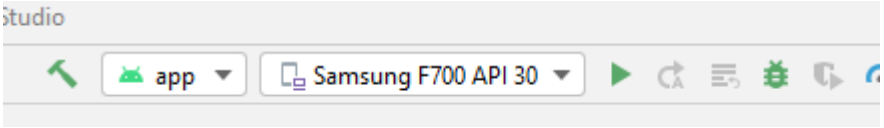


Рисунок 10 – Выбор устройства

Для запуска приложения нажмите комбинацию Shift+F10 или на зеленый треугольник.

Если никаких ошибок во время установки и запуска эмулятора не было, то результатом будет такое же окно, как на рисунке 11.

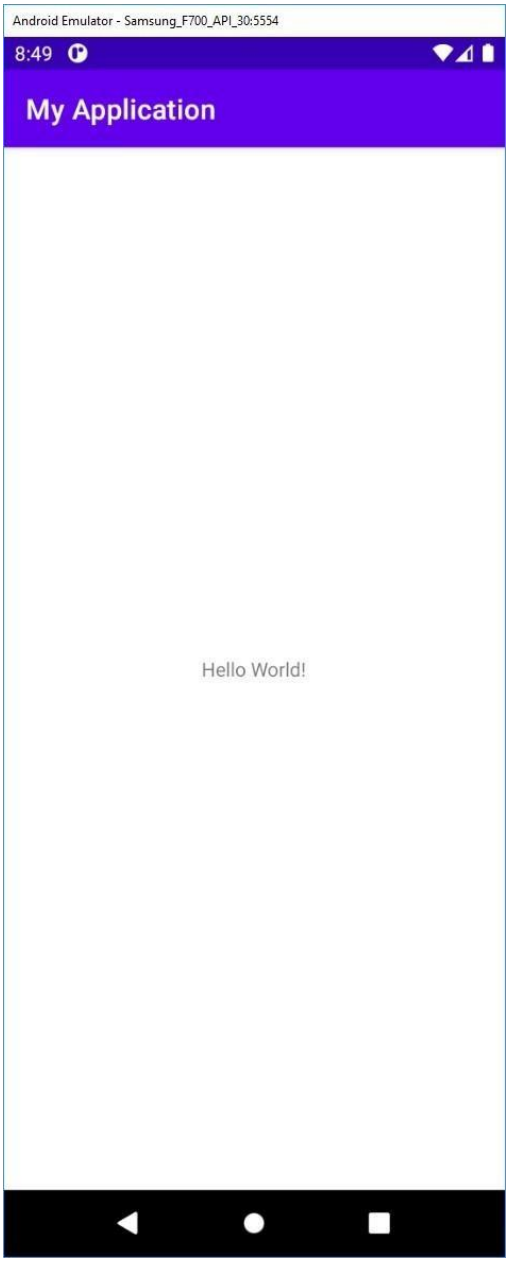


Рисунок 11 – Главное окно эмулятора

Интерфейс студии интуитивно понятен:

1) Дерево решений. Предоставляет доступ к различным файлам проекта. В папке app/java/com.example../ располагаются файлы на языке java. В папке app/res/layout/ файлы, которые отвечают за интерфейс вашей программы (активность). Папка app/res/drawable будет содержать файлы для рисования (подробнее во второй лабораторной работе). Папка app/res/values/ содержит файлы, в которых хранится информация о константах приложения (например, цвет фона, название приложения, и тд).

2) Панель управления проектом. Содержит стандартные возможности работы с проектом, такие как: открыть, сохранить, запустить и тд.

3) Рабочая область. Здесь будет располагаться Ваш код.

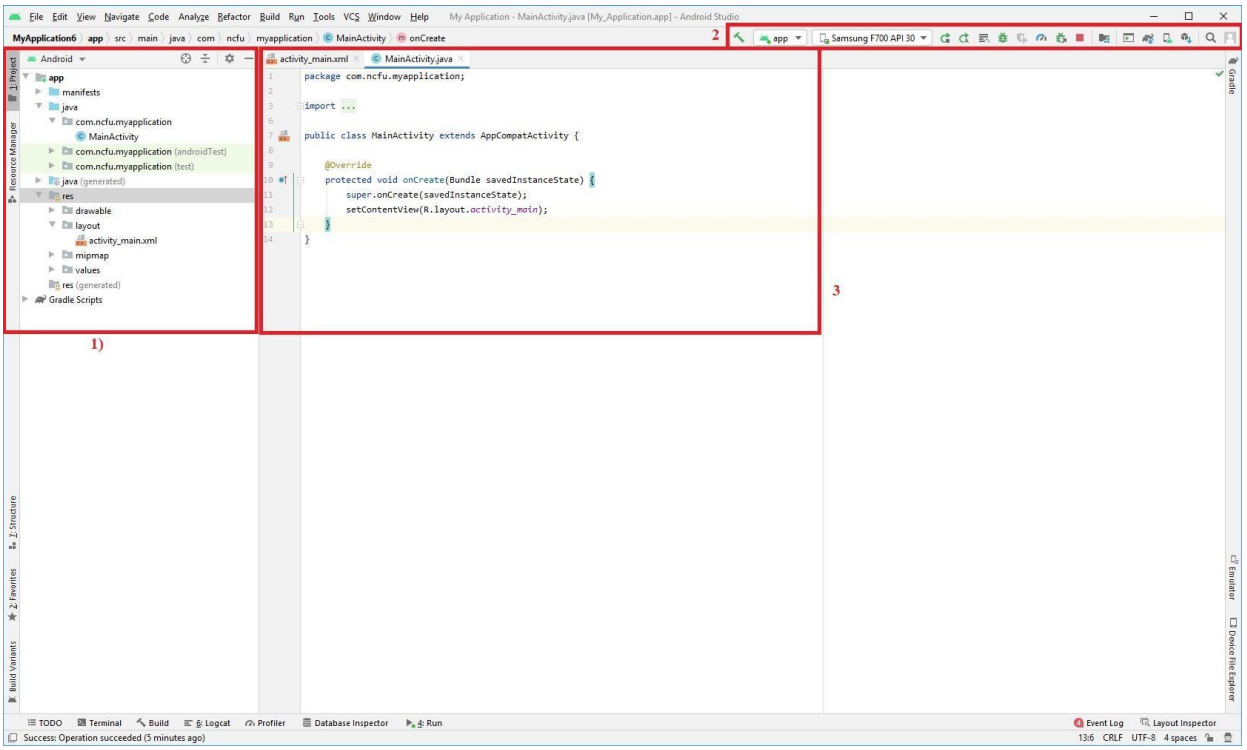


Рисунок 12 – Интерфейс Java

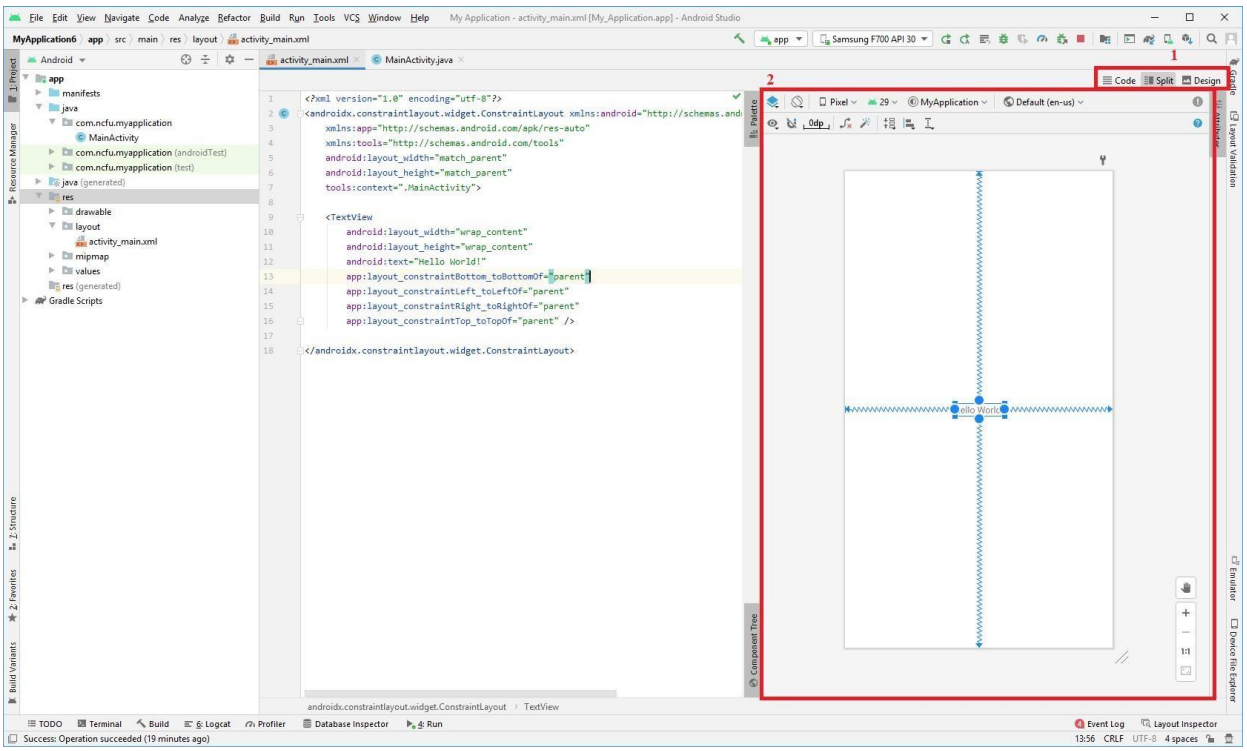


Рисунок 13 – Интерфейс xml

На рисунке 13 показан внешний вид студии во время работы с интерфейсом приложения.

1. Предпросмотр. Область, которая помогает увидеть расположение объектов, созданных с помощью языка xml. Здесь можно выбрать эмулятор, похожий на тот, который запускает приложение.

2. Кнопка, позволяющая выбрать, как будет выглядеть область предпросмотра.

Дизайн приложения разрабатывается с помощью языка XML. Процесс проектирования дизайна приложения отдаленно напоминает разработку web-страниц на языке HTML, однако об этом мы поговорим позже. Дизайн страниц (код на языке XML) хранится в файлах с расширением *.xml. Так, в обозревателе решений(Project) дизайн главной страницы соответствует файлу activity_main.xml. Однако о проектировании дизайна с помощью XML-разметки мы поговорим на следующих занятиях. Так как мы учимся в университете, нам главное добраться как можно ближе до истины. Поэтому

мы попробуем понять, что собой страница представляет на более низком уровне.

В java/com.example.<nameuser>.<nameproject>/ отобразится еще один похожий по названию файл MainActivity. Этот файл тоже соответствует главной странице приложения, и в нем находится код на языке Java.

Обычное бизнес-приложение Android (как WindowsPhone, и IOS) состоит из определенного набора страниц с возможностью навигации между ними. Каждой странице при проектировании соответствует определенный класс. Программирование на всех платформах в основном и заключается в проектировании классов страниц и взаимодействия между ними.

Итак, рассмотрим содержимое файла MainActivity.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Рисунок 14 – Листинг главного окна

Класс MainActivity определяет функционал главной страницы приложения (эта страница отображается при запуске приложения).

Еще раз обратите внимание. Мы разрабатываем класс MainActivity. При запуске приложения инфраструктура ОС создает объект этого класса. И после этого отображает его на экране телефона. Что значит отобразить класс? Это значит, что объект класса MainActivity обладает свойствами-характеристиками отображения (например, цвет фона, цвет рамки, наличие дочерних элементов и др.), а инфраструктура приложения опрашивает объект об этих свойствах и отображает их на экране.

Изменить название приложения можно двумя способами:

1) Перейти в AndroidManifest.xml (рис.13).

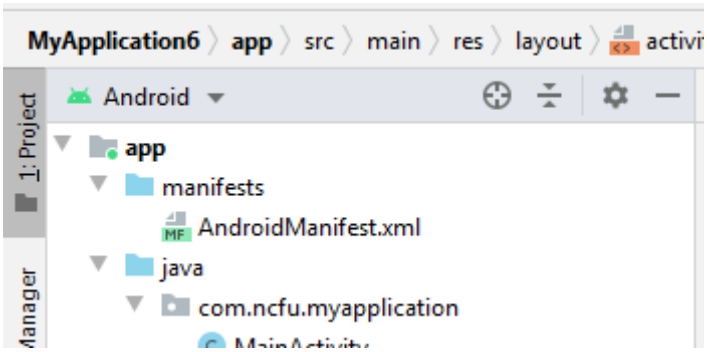


Рисунок 15 – Расположение AndroidManifest.xml в дереве решний

Нажмём на строку android:label="My application". Она изменится, как показано на рисунке.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ncfu.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Рисунок 16 –Содержимое файла AndroidManifest.xml

Держа ctrl нажмём на @string/app_name. Нас перекинет на страницу с ресурсами типа данных «строка». Все переменные, которые созданы в приложении будут храниться именно в ресурсах. Изменим значение на необходимое.

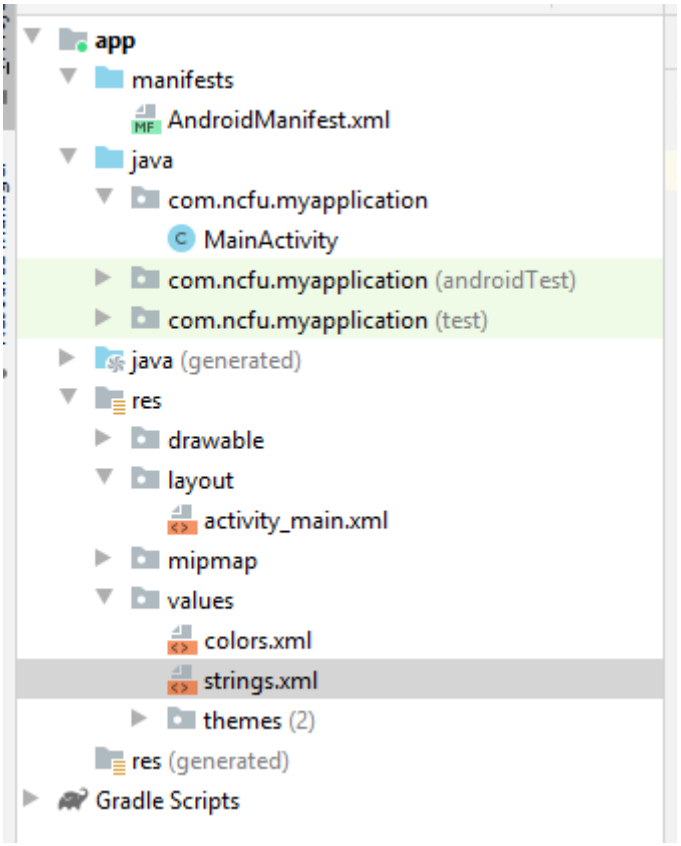


Рисунок 17 – Содержимое папки res (ресурсы)

```
<resources>
    <string name="app_name">My Application</string
</resources>
```

Рисунок 18 – Содержимое файла Strings

2) Внутри MainActivity.java с помощью метода setTitle(), где в параметре передаётся новое название приложения.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    setTitle("Тестовое название");
}
```

Рисунок 19 – Пример использования setTitle()

Следует отметить, что в языке программирования Java есть правила именования компонентов. Например, каждый класс должен всегда начинаться с большой буквы, в то время как все названия свойств и методов внутри в стиле CamelCase (первое слово с маленькое, а каждое последующее – с большой, например, setNameOfAnyObject()).

Добавление элементов на страницу приложения.

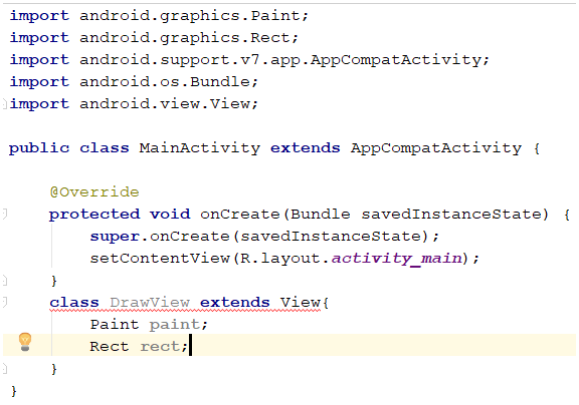
Страница приложения целиком и полностью состоит из элементов (текстовые блоки, кнопочки, галочки и многое другое). Каждый из этих блоков представляет собой также объект соответствующего класса.

При создании приложения, Android studio любезно разместила один из таких текстовых блоков (объектов) на странице без нашего ведома. Однако остается некая недосказанность – почему в описании класса эти элементы отсутствуют. Как нам их удалить и добавить те элементы, которые нам действительно нужны.

Попробуем создать новые элементы и разместить их на странице. Так как все элементы страницы являются объектами, для добавления новых элементов, нам нужно создавать новые объекты.

Поиграемся, для начала, с графическими примитивами. В android.graphics есть несколько графических классов, объекты которых мы будем использовать. Создаём класс DrawView, который наследуется от View. Конечно, каждый класс стоит создавать в новом файле. Но, так как для начала необходимо разобраться с простым рисованием, поместим новый класс в тот же файл, где расположена активность MainActivity. Более того, сделаем его вложенным классом.

Внутри определим свойства Paint (кисть) и Rect (прямоугольник).



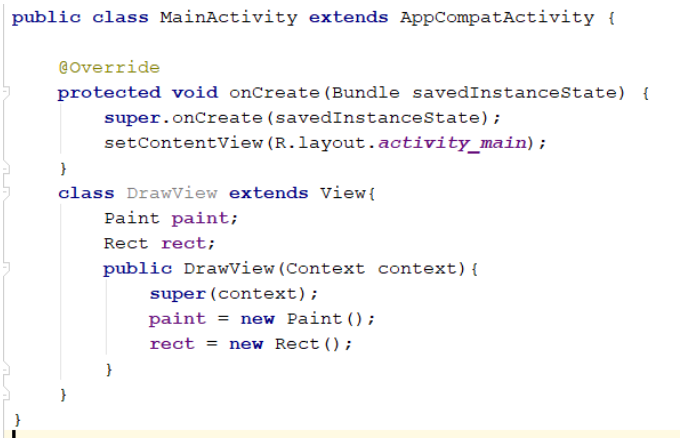
```
import android.graphics.Paint;
import android.graphics.Rect;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    class DrawView extends View{
        Paint paint;
        Rect rect;
    }
}
```

Рисунок 20 – Создание нового класса DrawView

На данном этапе выводит ошибку, которая будет исправлена после создания конструктора класса. Собственно, создадим конструктор.



```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    class DrawView extends View{
        Paint paint;
        Rect rect;
        public DrawView(Context context){
            super(context);
            paint = new Paint();
            rect = new Rect();
        }
    }
}
```

Рисунок 21 – Добавление конструктора

Теперь можно рисовать. Создадим метод, который будет «рисовать» за нас. Рисовать в воздухе не получится, поэтому необходимо создать «холст». Эту роль выполняет Canvas. Раскрасим наш холст и добавим его в контекст страницы. Класс DrawView наследуется от класса View (extends), в котором уже присутствует некоторое количество методов для «рисования». Чтобы не создавать заново всю инфраструктуру, воспользуемся уже существующими методами и переопределим их так, как нам удобно. Переопределяем метод OnDraw(), как показано на рисунке 20.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new DrawView( context: this));
        setTitle("Тестовое название");
    }

    class DrawView extends View{
        Paint p;
        Rect rect;

        public DrawView(Context context) {
            super(context);
            p = new Paint ();
            rect = new Rect();
        }
        @Override
        protected void onDraw(Canvas canvas){
            canvas.drawARGB( a: 80, r: 102, g: 104, b: 255);
        }
    }
}
```

Рисунок 22 – Создание метода для "рисования"

Метод drawARGB() закрашивает цвет заднего фона холста, где a (alpha) – насыщенность цвета, a r, g, b – количество красного, зелёного и синего цвета соответственно. Запустите приложение. Сделайте выводы. Чтобы нарисовать прямоугольник необходимо просто разместить его на канвасе (холсте). Воспользуемся методом drawRect(), где в параметрах передаются координаты отрезка диагонали прямоугольника (200, 150 – координаты левого верхнего угла прямоугольника, а 400, 200 – нижнего правого), и кисть.

```
@Override
protected void onDraw(Canvas canvas){
    canvas.drawARGB( a: 80, r: 102, g: 104, b: 255);
    p.setColor(Color.RED);
    canvas.drawRect( left: 200,  top: 105,  right: 400,  bottom: 200 ,p);
}
```

Рисунок 23 – Создание прямоугольника красного цвета

Чтобы нарисовать круг воспользуемся методом drawCircle(). Входные параметры: координаты центра (слева, сверху), радиус и кисть.

```
@Override
protected void onDraw(Canvas canvas){
    canvas.drawARGB( a: 80, r: 102, g: 104, b: 255);
    p.setColor(Color.RED);
    canvas.drawRect( left: 200,  top: 105,  right: 400,  bottom: 200 ,p);
    canvas.drawCircle( cx: 100,  cy: 200,  radius: 50, p);
}
```

Рисунок 24 – Создание круга красного цвета

Присутствует также возможность рисовать прямую (-ые) с помощью метода drawLine(-s) (просто точка(-и) - drawPoint(-s)). Например, на рисунке 23 показано создание отрезка, где координаты его вершин передаются в качестве массива.

```
@Override
protected void onDraw(Canvas canvas){
    canvas.drawARGB( a: 80, r: 102, g: 104, b: 255);
    p.setColor(Color.RED);
    canvas.drawRect( left: 200,  top: 105,  right: 400,  bottom: 200 ,p);
    canvas.drawCircle( cx: 100,  cy: 200,  radius: 50, p);
    p.setStrokeWidth(10);
    float[] points = new float[]{300, 200, 400, 200};
    canvas.drawLines(points, p);
}
```

Рисунок 25 – Создание отрезка

setStrokeWidth – метод, позволяющий установить толщину кисти.

Необходимо также учесть, что массив должен содержать обязательно количество точек кратное 4, ибо отрезок – две точки, соединенные между собой, а точка определяется координатами x и y.

Для более сложных фигур используется соответственно более сложный метод рисования. Path – позволяет рисовать фигуры каким угодно способом: это может быть и банальный многоугольник, а может и использовать кривые. Например, чтобы нарисовать треугольник нужно следующее:

```
Path path = new Path();
path.moveTo( x: 50,  y: 50);
path.lineTo( x: 0,  y: 100);
path.lineTo( x: 100,  y: 100);
path.lineTo( x: 50,  y: 50);

canvas.drawPath(path, p);
```

Рисунок 26 – Код треугольника

А для рисования кривых используется метод addArc(). В качестве параметров передаются координаты внутреннего прямоугольника, угол смещения и угол прорисовки. То есть, если нужно нарисовать полукруг с радиусом 50 px, выпуклый к низу, то воспользуемся следующим кодом:

```
Path path = new Path();
path.addArc(new RectF( left: 0,  top: 0,  right: 100,  bottom: 100),  startAngle: 0,  sweepAngle: 180);

canvas.drawPath(path, p);
```

Рисунок 27 – Первый пример кривой

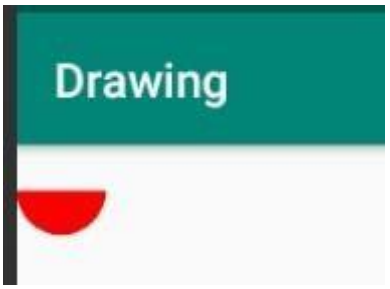


Рисунок 28 – Результат

Для вертикально-выпуклой кривой используем:

```
Path path = new Path();
path.addArc(new RectF( left: 0,  top: 0,  right: 100,  bottom: 100),  startAngle: 90,  sweepAngle: 180);

canvas.drawPath(path, p);
```

Рисунок 29 – Второй пример кривой

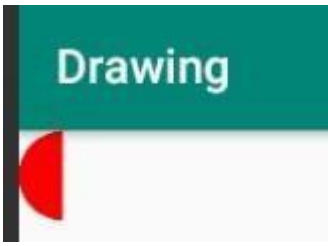


Рисунок 30 – Результат

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 (8,8.1,10) и программным продуктом Android Studio.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Задания: для выполнения лабораторной работы необходимо выполнить следующее:

1) Запустить и проверить, все ли работает правильно. При запуске сначала загрузится эмулятор мобильной платформы, вслед за чем на нем запустится ваше приложение. Это приложение является заготовкой и кроме как главную страницу ничего не покажет. Запомните, для того, чтобы остановить приложения не обязательно (более того не рекомендуется) закрывать эмулятор. При следующем запуске приложения, эмулятор уже будет загружен и запуск вашего приложения произойдет гораздо быстрее.

2) Указать в имени приложения авторство, а также на странице указать дату рождения, с помощью свойства страницы TextView.

3) Выполнить все описанные в лабораторной работе шаги. Запустите приложение. На экране должен отобразиться красный прямоугольник. Если этого не произошло, внимательно проверьте все выполненные шаги. Обратитесь к преподавателю за помощью. Выполните следующее задание согласно своему варианту. Если не указаны конкретные размеры, отталкиваться от того что ширина должны быть не меньше 150 пикселей. Площадь круга $\pi*r^2$

4) Создание фигур с помощью графических примитивов. Необходимо использовать минимум 3 цвета и ломаные.

Варианты для задания 3:

1. Зеленый прямоугольник, у которого высота в 2 раза больше ширины.
2. Синий круг, занимающий половину площади экрана.
3. Желтый прямоугольник, занимающий половину площади экрана.
4. Полупрозрачный зеленый овал, касающийся всех граней экрана.
5. Пурпурный квадрат, площадь которого в 2 раза больше периметра.
6. Розовый круг, площадь которого равна периметру.

7. Синий квадрат, периметр которого в 2 раза меньше периметра экрана.
8. Зеленый квадрат, площадь которого равна периметру.
9. Оранжевый квадрат, площадь которого в 2 раза меньше площади экрана.
10. Серый прямоугольник, с одинаковым расстоянием от граней экрана до граней прямоугольника.
11. Красный овал, с длиной вертикальной полуоси в 3 раза большей горизонтальной.
12. Белый круг, площадь которого в 2 раза больше периметра.
13. Желтый овал с одинаковым расстоянием от граней экрана до ближайшей точки на овале.
14. Розовый круг, площадь которого в 3 раза меньше площади экрана.

Варианты для задания 4:

1. Дом с дверью и окном.
2. Легковой автомобиль.
3. Грузовой автомобиль.
4. Логотип СКФУ.
5. Железная дорога, уходящая за горизонт.
6. Новогодняя елка.
7. Котик.
8. Велосипед.
9. Ножницы.
10. Куб с разноцветными гранями.
11. Солнце со смайлом.
12. Набор из 15 вложенных прямоугольников со случайными параметрами.
13. Набор из 15 вложенных кругов со случайными параметрами.
14. Кнопочный телефон.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчета приводятся выводы о проделанной работе.

В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Понятие виджета.
2. Графические примитивы.
3. Переопределение метода.
4. Наследование.

Список литературы, рекомендуемый к использованию по данной теме:

1. Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. Основы интернет-технологий: учебное пособие / Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. – М.: Архангельск: ИПЦ САФУ, 2013 – 366 с.; То же [Электронный ресурс]. - URL:http://biblioclub.ru/index.php?page=book_red&id=436379&sr=1
2. Соколова В. В. Разработка мобильных приложений: учебное пособие/ Соколова В. В. Томск: Издательство Томского политехнического университета, 2015 – 176с. То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book_red&id=442808&sr=1

Лабораторная работа 2. Разработка дизайна страницы с помощью языка XML. Менеджеры размещения.

Цель работы: Изучение XML разметки

Теоретическая часть:

На прошлой лабораторной работе мы рассмотрели объектную структуру главной страницы. Мы научились добавлять на страницу графические примитивы и даже управлять ими. Таким же образом возможно добавлять на страницу элементы управления: кнопки, галочки, текстовые блоки и т.д.

Очевидно, что проектировать дизайн приложения таким образом не самый лучший вариант. Поэтому используется язык разметки XML, который позволяет в гораздо более удобном виде управлять внешним видом страницы.

Очень важно понять, что язык XML не просто язык разметки, он позволяет декларативно создавать объекты и определять их свойства, вместо того чтобы делать это в коде, как мы делали на предыдущем занятии.

Рассмотрим код XML, созданный по умолчанию AS. Весь код состоит из отдельных элементов-тегов, заключенных в угловые скобки. Давайте разберем, что представляет собой тег.

```
<имя_тега имя_атрибута=”значение атрибута”>
    Содержимое тега
</имя_тега>
```

Содержимое тега представляет собой либо текст, либо вложенные теги:

```
<имя_тега имя_атрибута=”значение атрибута”>
    <имя_вложенного_тега>
        содержимое вложенного тега
    </имя_вложенного_тега>
</имя_тега>
```

Таким образом, XML код – набор вложенных тегов. Каждый тег позволяет описывать декларативным (описательным) способом классы и объекты, участвующие в приложении. Например, в корневом теге файла activity_main.xml описывается класс главной страницы MainActivity.java. Рассмотрим этот момент подробнее.

Именем корневого тега является тег RelativeLayout (В новых версиях Android Studio появился ConstraintLayout, который стоит заменить на RelativeLayout для выполнения данной лабораторной работы).

ReiativeLayout (относительная разметка) позволяет дочерним компонентам определять свою позицию относительно родительского компонента или относительно соседних дочерних элементов (по идентификатору элемента). В RelativeLayout дочерние элементы расположены так, что, если первый элемент расположен по центру экрана, другие элементы, выровненные относительно первого элемента, будут выровнены относительно центра экрана.

android:id="@+id/activity_main"

В этом свойстве указывается идентификатор разрабатываемой страницы.

Атрибуты с префиксом xmlns используются для описания используемых пространств имен. Обратите внимание на атрибут

xmlns:android="http://schemas.android.com/apk/res/android"

он создает псевдоним android для всего пространства имен, которое используется в имени корневого тега RelativeLayout.

Получается, что в корневом теге activity_main.xml идет речь о том же классе, что и в файле MainActivity.java. Более того с помощью XML вы можете разрабатывать класс страницы одновременно с разработкой его в исходном коде в файле MainActivity.java.

Остальные атрибуты корневого тега рассмотрим на дальнейших занятиях. Важное замечание: как и в любом XMLдокументе, здесь может

существовать только один корневой элемент, таким образом, вы можете описать в одном файле только один класс страницы, что вполне логично.

Следующий важный момент: содержимое корневого тега соответствует свойству страницы Content, которое мы использовали на предыдущей лабораторной работе. Каждый вложенный тег описывает какой-либо объект, аналогично тому, как мы создавали объекты в исходном коде с помощью конструктора.

Найдите среди вложенных тегов следующий:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"/>
```

Рисунок 1 - Виджет TextView

Этот тег не является прямым потомком корневого тега, и это логично, ведь свойство Content страницы не может хранить более одного объекта (вспоминаем лаб.1). Он находится в контейнере RelativeLayout. Об этих контейнерах и вложенностях мы поговорим далее, вспомним лишь, что в лаб. 1 мы использовали контейнер Canvas (холст).

Основные элементы, которые можно добавлять на страницу были перечислены в первой лабораторной работе. Однако, нам понадобится ещё один виджет – ImageView. Данный элемент позволяет размещать изображения/примитивы внутри себя. По сути, ImageView является прямоугольником, поэтому, в случае чего, ему можно поменять цвет фона и даже цвет примитива (свойства background и tint соответственно).

Рассмотрим основные параметры, которые можно применять к элементам разметки XML:

Название	**К каким элементам применять**	**Краткое описание**	**Примеры**
android:id	Все	Устанавливает уникальный идентификатор элемента, чтобы можно было найти его в java-коде	@+id/el1, @+id/textView
android:layout_width	Все	Устанавливает ширину элемента	10px, 18dp, match_parent (занимает всё доступное пространство родителя), wrap_content (устанавливает размеры в зависимости от содержимого)
android:layout_height	Все	Устанавливает высоту элемента	
android:layout-margin	Все	Устанавливает внешние отступы элемента относительно других	10dp, 150px, 200mm, 14pt
android:padding	Все	Устанавливает внутренний отступ у содержимого	
android:background	Все	Устанавливает цвет задней	#fff @color/colorPrimary

Название	К каким элементам применять	Краткое описание	Примеры
		поверхности элемента	
android:gravity	Все	Устанавливает выравнивание содержимого элемента по одной из сторон	left, right, center, bottom, top... Можно комбинировать: left|center
android:rotation	Все	Поворачивает элемент на указанное число градусов	90, 45, 180
android:text	TextView, EditText	Устанавливает текст в элементе	Любая строка
android:textSize		Устанавливает размер текста	Принято указывать размер в единицах sp.
android:visibility	Все	Скрывает или отображает элемент	Visible, invisible(просто скрывает), gone(удаляет, как будто бы и не было элемента на этом месте)
android:src	ImageView	Устанавливае	@drawable/

Название	К каким элементам применять	Краткое описание	Примеры
		т в качестве источника картинка внешний файл или графический примитив	ic_launcher_foreground

Контейнеры LinearLayout и Grid.

Мы уже говорили о том, что для размещения элементов на странице, необходимо использовать контейнеры.

Термин «контейнер», мы использовали в контексте того, данный элемента может содержать в себе другие элементы. На самом деле миссия у такого «контейнера» гораздо выше. Заключается она в определении схемы размещения и компоновки других элементов. Поэтому такие «контейнеры» часто называют менеджерами размещения.

Мы использовали контейнер типа «Холст», который позволяет размещать элементы произвольным образом с помощью присоединенных свойств (attached properties).

Такой подход называется абсолютным позиционированием, т.е. позиционирование по абсолютным координатам(относительно верхнего левого края экрана). С одной стороны, это дает бесконечную гибкость в расположении элементов на странице, с другой же значительно повышает трудоемкость размещения, т.к. вам необходимо рассчитывать координаты для каждого элемента. Поэтому менеджер размещения Canvas используется

только в исключительных случаях, например, рисования произвольных геометрических фигур (лаб. 1).

В случае разработки бизнес-приложений, используются стандартные элементы управления (кнопки, текстовые блоки, галочки и т.д.). Для их размещения чаще всего достаточно использовать простые макеты размещения, например, размещение одного элемента за другим горизонтально или вертикально. Для такого размещения используется менеджер LinearLayout.

Всего существует 5 основных менеджеров:

- LinearLayout (один за одним);
- RelativeLayout (располагать элементы относительно друг друга);
- ConstraintLayout (более совершенная версия RelativeLayout);
- GridLayout (сетка, позволяет помещать элементы в "ячейки");
- TableLayout (таблица, отличается от GridLayout тем, что необходимо задавать элементы в каждой ячейке);

Ход лабораторной работы

1. Создайте новый проект с произвольным именем.
2. Создадим два Drawable-файла с названиями “rectangle” и “circle”. Заменим исходный код:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="#00ff00"></solid>
</shape>
```

Листинг 1 - Содержимое файла rectangle.xml

```
shape  solid
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="#ff0000"></solid>
</shape>
```

Листинг 2 - Содержимое файла circle.xml

В наш xml файл добавим LinearLayout и заполним содержимым

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="100px"
        android:layout_height="100px"
        android:src="@drawable/rectangle"/>
    <ImageView
        android:layout_width="100px"
        android:layout_height="100px"
        android:src="@drawable/circle"/>
    <ImageView
        android:layout_width="100px"
        android:layout_height="100px"
        android:src="@drawable/rectangle"/>
</LinearLayout>
```

Листинг 3 - Содержимое файла Activity_main.xml

Результат действий представлен на рисунке1.

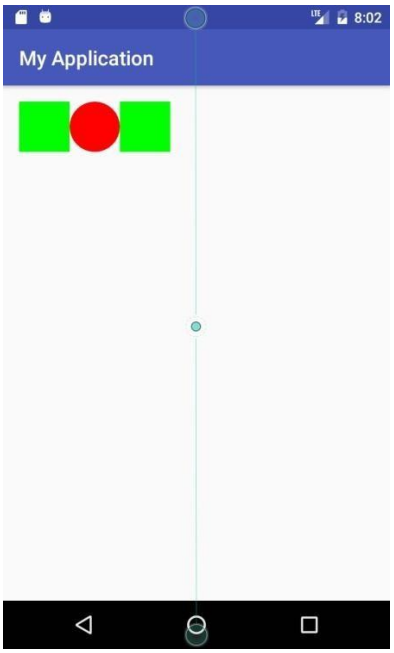


Рисунок 2 - Отображение результата на эмуляторе

3. По умолчанию LinearLayout размещает элементы горизонтально один за другим. Такое поведение можно изменить с помощью свойства android:orientation="vertical":

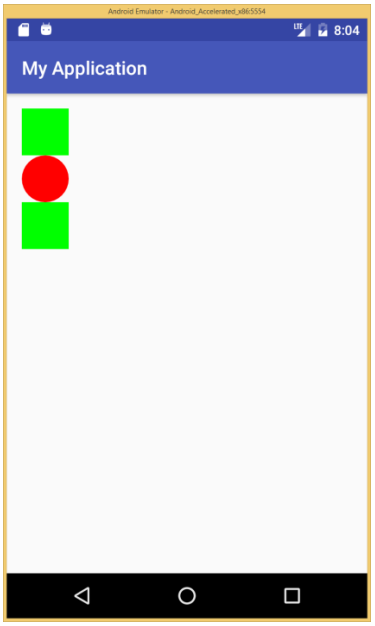


Рисунок 3 - Использование вертикальной ориентации

4. При использовании горизонтальной ориентации также можно изменить направление расположение элементов, «прилепив» их к правой границе экрана с помощью свойства layoutDirection:

```
android:layoutDirection="rtl"
```

Рисунок 4 - Пример использования RightToLeft направления

5. Менеджеры могут размещаться один в другом для более изощренных видов разметки. Например:

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"

    >
    <ImageView
        android:layout_width="100px"
        android:layout_height="100px"
        android:src="@drawable/rectangle"
        android:layout_gravity="center"/>
    <LinearLayout
        android:layout_gravity="center"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <ImageView
            android:layout_width="100px"
            android:layout_height="100px"
            android:src="@drawable/rectangle" />

        <ImageView
            android:layout_width="100px"
            android:layout_height="100px"
            android:src="@drawable/circle" />

        <ImageView
            android:layout_width="100px"
            android:layout_height="100px"
            android:src="@drawable/rectangle" />
    </LinearLayout>
</LinearLayout>
```

Листинг 3- Содержимое файла activity_main.xml

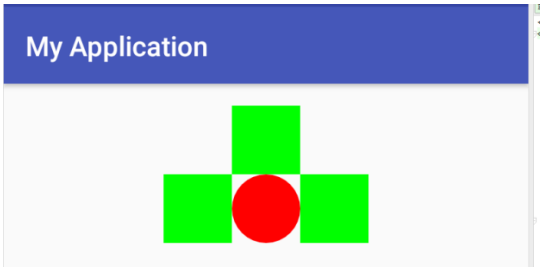


Рисунок 5 - Результат листинга 5

Не стоит переживать, что Android Studio выделяет область, где используются размеры коричневым цветом. Дело в том, что размеры у разных телефонов разные, следовательно, одни те же элементы на различных устройствах будут выглядеть по-разному. Бывают моменты, когда необходимо, чтобы элемент оставался всегда одинакового размера на разных устройствах, а иногда наоборот, чтобы подстраивалось под размеры экрана. Для первого случая следует указывать размеры в px, а во втором в dp.

Менеджер размещения Grid

Позволяет располагать элементы, используя строки и столбцы. При использовании менеджера размещения Grid разработчик определяет общую структуру сетки, а потом, используя присоединенные свойства, размещает элементы управления в ячейках сетки.

```
<GridLayout
    android:id="@+id/grid"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:columnCount="3"
    android:rowCount="3"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true">

    <Button android:text="1" />
    <Button android:text="2"/>
    <Button android:text="3" />
    <Button android:text="4" />
    <Button android:text="5" />
    <Button android:text="6" />
    <Button android:text="7" />
    <Button android:text="8" />
    <Button android:text="9" />
</GridLayout>
```

Рисунок 6 - Пример использования Grid-а

Обратите внимание на следующие моменты:

a. Размеры могут задаваться либо целыми числами, либо строкой «wrap_content», указывающей на то, что размер строки или столбца должен подстроиться под размер элемента в нем находящемся, либо строкой «match_parent», указывающей на то, что строка или столбец займет все оставшееся место на странице.

b. После определения структуры располагаются размещаемые на экране элементы. В данном случае это объекты типа Button (кнопка).

c. Определяя дочерним элементам ту или иную строку (столбец) следует помнить, что не нужно выходить за границу определяемого column_count и row_count. Например, если grid содержит всего 3 столбца, мы не сможем поместить его в колонку с номером 3 (индексы отсчитываются с 0 элемента, как в программировании).

Чтобы явно указать, в какую ячейку поместить объект, используются свойства layout_column (столбец) и layout_row (строка). Так как grid является подобием таблицы, присутствует возможность объединять ячейки. Для объединения строк используется layout_rowSpan, где передаётся количество ячеек для объединения, а для столбцов соответственно layout_columnSpan.

```
        <Button android:text="1" />
        <Button android:text="2" />
        <Button android:text="3" android:layout_column="3" />
        <Button
        <Button
        <Button android:text="6" />
        <Button android:text="7" />
        <Button android:text="8" />
        <Button android:text="9" />
    </GridLayout>
</RelativeLayout>
```

Column attribute (3) exceeds declared grid column count (3) more... (Ctrl+F1)

Листинг 4 - Пример неправильного обращения к индексам

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    GridLayout gr = (GridLayout)findViewById(R.id.grid);

    Button b1 = new Button(this);
    Button b2 = new Button(this);
    Button b3 = new Button(this);
    Button b4 = new Button(this);
    Button b5 = new Button(this);

    b1.setText("1");
    b2.setText("2");
    b3.setText("3");
    b4.setText("4");
    b5.setText("5");
    gr.addView(b1);
    gr.addView(b2);
    gr.addView(b3);
    gr.addView(b4);
    gr.addView(b5);
    gr.setRowCount(3);
    gr.setColumnCount(3);
}
```

Листинг 5 - Пример создания и использования Grid-a, используя java

```
<ImageView android:layout_columnSpan="2" />
```

Рисунок 7 - Пример объединения ячеек

Задания: для выполнения лабораторной работы необходимо выполнить следующее:

1. Реализуйте размещение элементов, как представлено на рисунке 5.
2. Нарисуйте на экране композицию, представленную на рисунке 8:

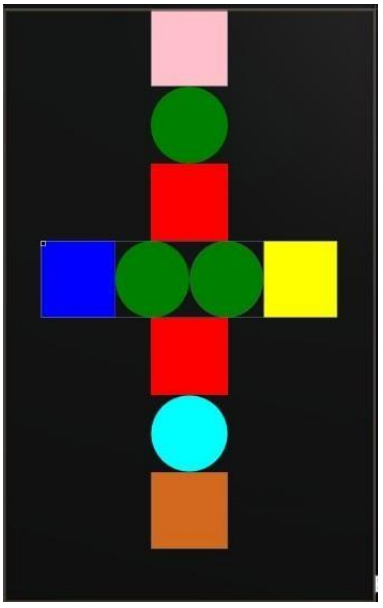


Рисунок 8 - Задание для проверки способностей

3. Выполните задание по варианту с помощью XML. При выполнении задания лабораторной работы необходимо использовать менеджеры разметки Grid и LinearLayout.

Варианты для задания 3:

1. Рисунок 9.
2. Рисунок 10.
3. Рисунок 11.
4. Рисунок 12.
5. Рисунок 13.
6. Рисунок 14.
7. Рисунок 15.
8. С помощью менеджеров размещения напишите букву Н с помощью кнопок на весь экран
9. С помощью менеджеров размещения выведите букву П из кнопок на весь экран
10. С помощью менеджеров размещения выведите букву Г из кнопок на весь экран

Рисунок 9

Рисунок 10

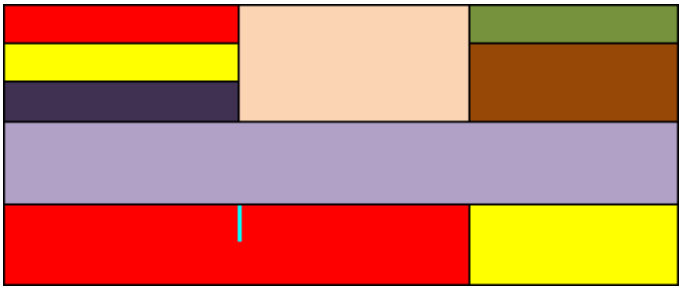


Рисунок 21

Рисунок 12

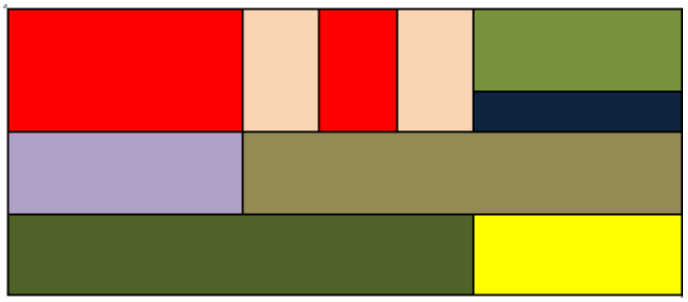


Рисунок 33

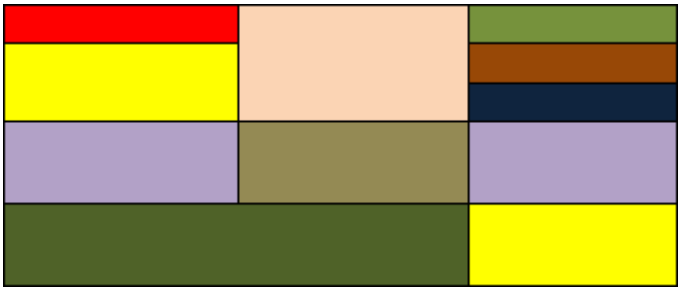


Рисунок 44

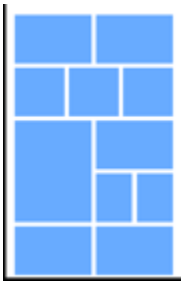


Рисунок 15

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 (8,8.1,10) и программными продуктами: AndroidStudio.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MSWord и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выровнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчета приводятся выводы о проделанной работе.

В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Язык XML
2. Тег
3. Менеджер размещения
4. Grid
5. LinearLayout

Список литературы, рекомендуемый к использованию по данной теме:

1. Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. Основы интернет-технологий: учебное пособие / Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. – М.: Архангельск: ИПЦ САФУ, 2013 – 366 с.; То же [Электронный ресурс]. - URL:http://biblioclub.ru/index.php?page=book_red&id=436379&sr=1
2. Соколова В. В. Разработка мобильных приложений: учебное пособие/ Соколова В. В. Томск: Издательство Томского политехнического

университета, 2015 – 176с. То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book_red&id=442808&sr=1

Лабораторная работа 3. Использование механизма событий Android.

Цель работы: Научиться разбираться с механизмом обработки клика.

Теоретическая часть:

Элементы управления – это элементы, которые располагаются на экране телефона и позволяют вам взаимодействовать с вашей программой. Они включают менеджеры размещений, кнопки, галочки, текстовые блоки, прогресс бар и многое др. О них было упомянуто в первой лабораторной работе.

На предыдущих лабораторных работах мы научились создавать некоторые графические элементы и элементы управления, а также размещать их на экране определенным образом. Теперь нам необходимо научиться обрабатывать взаимодействие пользователя с устройством. Например, при нажатии на кнопку, возможно, вам нужно будет выполнить какие-либо вычисления или отправить какие-либо данные на сервер для обработки. Для этого существует механизм событий. Событий, на которые мы можем реагировать достаточно много, при этом каждое событие привязано к конкретному элементу. Правильнее будет даже сказать, что у каждого элемента есть список событий, которые он может генерировать. При нажатии на кнопку генерируется событие *onClick.* Механизм событий позволяет вам подписываться на события. Эта подписка заключается в указании метода, который будет запускаться в ответ на генерацию события. Такой метод называется обработчиком события.

Code-behind

Как вы уже поняли, основной (статический) дизайн приложения разрабатывается с помощью *XML* в файле *activity_main.xml*. Это так называемое **внешнее представление** вашего приложения, т.е. то, что видит пользователь в момент запуска. В тоже время MainActivity.java используется для проектирования динамики приложения: в нем описывается обработка событий, выполнение вычислений, проектирование взаимодействия между

пользователем и программой. Это **внутреннее представление приложения**, скрытое от глаза пользователя. В связи с этим появилось понятие ***Code-Behind*** (с англ. «код позади», читается «**код бихайнд**»), которое обозначает именно код, который «обслуживает» внешнее представление. Этим понятием мы будем обозначать код, описывающий класс страницы с ее свойствами, методами и обработчиками событий.

Элементов управления множество, и каждый может генерировать определенный набор событий. Многие элементы имеют пересекающийся список событий в связи с тем, что они наследуют их от общих предков в иерархии наследования. Для того чтобы посмотреть какие события может генерировать элемент, можно использовать технологию IntelliSense. Для этого достаточно установить курсор на определение элемента в XML и нажать сочетание клавиш ctrl + пробел. Откроется список возможных атрибутов для использования.

Динамическое управление подпиской на события.

С помощью XML мы можем декларативно описать какой обработчик на какое событие реагирует. Однако, XML удобен в данном случае, лишь как точка входа в приложение, т.е. в качестве описания того, что пользователь увидит при запуске приложения. Когда же нам необходимо более гибкое динамическое поведение на помощь приходит code-behind. Динамическое поведение здесь подразумевает динамическую подписку на события и отписку от них. Например, когда при выполнении некоторых условий необходимо, чтобы обработчик не реагировал на событие (и при нажатии на кнопку ничего не происходило). К тому же code-behind позволяет подписывать на одно событие сразу несколько обработчиков, что в XML невозможно. В общем, давайте научимся работать с событиями в code-behind.

Ход лабораторной работы:

1. Создайте новый проект.
2. Разместите на главной странице под текстовым блоком, созданным по умолчанию, кнопку 175*75 с текстом «Кнопка». Оберните и

текст, и кнопку тегом <LinearLayout> с вертикальной ориентацией. Должно получиться следующим образом:

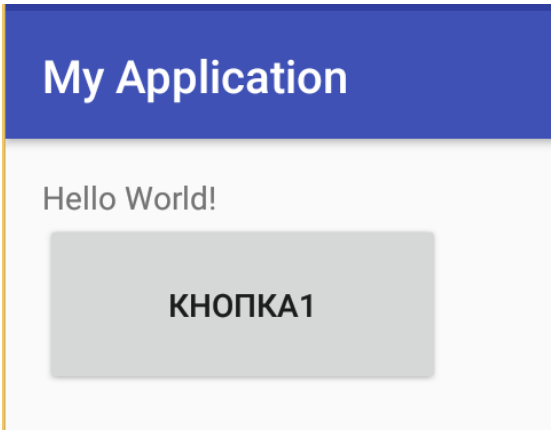


Рисунок 1 - Использование вертикальной ориентации

3. Зададим обработчик события *onClick*. Для этого добавим атрибут *onClick* к элементу *Button*. В отличие от Visualstudio, AndroidStudio не может сама генерировать обработчики событий. Поэтому, необходимо создать таковой:

```
public void onButtonClick(View v){
    Toast.makeText(MainActivity.this, "Нажата кнопка1", Toast.LENGTH_SHORT).show();
}
```

Рисунок 2 - Пример создания обработчика

4. Примените созданный обработчик для кнопки. Код кнопки в итоге будет выглядеть следующим образом:

```
<Button
    android:layout_width="175dp"
    android:layout_height="75dp"
    android:text="Кнопка1"
    android:onClick="onButtonClick"
    android:id="@+id/b1">

</Button>
```

Рисунок 3 - Исходный код кнопки

Значением атрибута *onClick*теперь является строка *onButtonClick*, которая является названием метода-обработчика класса *MainActivity*.

5. Чтобы при нажатии на кнопку вывести сообщение на экран, добавьте в этот метод следующее выражение:

```
Toast.makeText(this, "нажата кнопка", Toast.LENGTH_SHORT).show();
```

Рисунок 4 - Пример всплывающего события

Метод makeText – создаёт сообщение, где первым параметром указывается this, вторым – текст сообщения, далее идёт длительность сообщения: LENGTH_SHORT – устанавливает длительность 2.5 секунды, LENGTH_LONG – 3.5 секунды. Метод show() – заставляет показать созданное сообщение.

Запустите приложение, проверьте, как работает нажатие на кнопку.

Идентификаторы элементов управления

Вы можете давать имена элементам управления с помощью атрибута id:

```
android:id="@+id/b1"
```

Рисунок 5 - Присвоение идентификатора

В результате вы получите программный доступ к данному элементу в коде файла MainActivity.java под именем b1, например, можно задать текстовое значение кнопки (задается с помощью метода setText()):

```
Button b1 = (Button)findViewById(R.id.b1);
b1.setText("Кнопка1");
```

Рисунок 6 - Обращение по идентификатору

Если быть более точными при указании атрибута id, анализатор XML создает свойство в классе MainActivity с соответствующим идентификатором.

6. Выполним еще пару упражнений для закрепления. Удалите значение атрибута Clickв кнопке. Теперь чтобы добавить обработчик события, вы должны указать его имя. Вы можете сделать это вручную (просто вписать имя метода), либо с помощью технологии *IntelliSense (читается «ИнтелиСэнс»)*, нажав на сочетание клавиш Ctrl+Пробел (курсор в этом случае должен быть после первой кавычки значения атрибута). При использовании *IntelliSense,* вам представится выбор из существующих обработчиков.

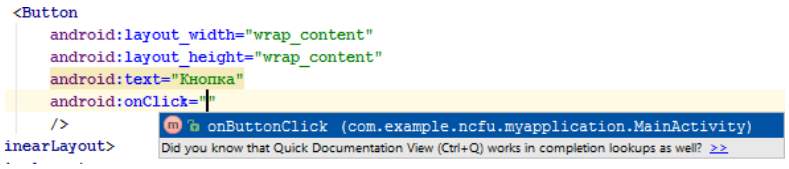


Рисунок 7 - Добавление обработчика в XML

7. Создайте новый обработчик и проделайте еще раз шаг 6, чтобы убедиться, что в качестве выбора вам представится уже 2 обработчика.

8. Создайте в классе MainActivityеще один метод с произвольным именем, но с такой же **сигнатурой**, как и метод onButtonClick.

Сигнатура метода представляет собой спецификацию метода, включающую информацию о типе возвращающего значения и типах его аргументов. Например, сигнатуру метода:

```
double getHeight(int a, string b)
{
    return 0;
}
```

Рисунок 8 – Пример 1

можно представить как **double (int, string)**. Сигнатура не включает в себя название метода и названия его аргументов, только их типы. Если говорят, что методы имеют одинаковую сигнатуру, значит методы возвращают значения одно и того же типа и принимают аргументы тех же типов в той же последовательности. Для указанного выше метода, такую же сигнатуру будет иметь следующий метод:

```
double calulateMass(int plotnost, string name)
{
    return plotnost*56/28 + 2390489230;
}
```

Рисунок 9 - Пример 2

Еще раз обратите внимание на то, что для сигнатуры не важны имя метода и имена его аргументов. Значение имеют только типы аргументов и тип.

9. Тело вашего метода должно содержать код, выводящий на экран сообщение с произвольным числом от 1 до 100.

10. Проделайте шаг 7 и убедитесь, что в предложенных вариантах обработчиков события *onClick* присутствует ваш новый метод. **Если вы создадите метод с другой сигнатурой, он в списке не отобразится.**

11. Рассмотрим, как можно сделать задание 3 с использованием аргумента *View.*

Взгляните еще раз на обработчик события, который подписан на событие *onClick* кнопки.

При генерации события вобработчик события передается ссылка на объект, генерировавший событие. Эта ссылка хранится в аргументе *view*. Но как видно из сигнатуры обработчика, этот аргумент имеет тип View, а кнопка имеет тип Button. Нам необходимо явное приведение типов:

```
Button b1  = (Button)v;
b1.setText("Нажалась");
```

Рисунок 10 - Изменение содержимого кнопки

В первой строке мы приводим ссылку к типу Buttoни сохраняем ее в переменную b1. Во второй строке мы обращаем к методуsetTextуже приведенного типа. Запись можно сократить следующим образом:

```
((Button)v).setText("Нажалась");
```

Рисунок 11 - Краткая запись

12. Создайте новый проект. Добавьте на главную страницу кнопку. Задайте ей имя b.

```
Button b  = (Button)findViewById(R.id.b);
```

Рисунок 12 - Использование кнопки в java

13. К сожалению, события в XMLи события в java подписываются по-разному.Если в XML необходимо передавать Viewи возвращать void, то здесь необходимо следовать правилам:

```
View.OnClickListener click1 = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(MainActivity.this, "Нажата кнопка", Toast.LENGTH_SHORT).show();
    }
};
```

Рисунок 13 - Создание простого события

Разберёмся подробнее. Во-первых, необходимо понять, что здесь событие – такой же тип данных, как, например, int или string. Следовательно, необходимо объявить переменную типа View.OnClickListener. Во-вторых, внутри необходимо переопределить метод onClick(). Например, если мы хотим, чтобы и в java и в XML при нажатии генерировались одинаковые события, то мы можем поместить вызов одного метода внутри другого:

```
View.OnClickListener click1 = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        onButtonClick(v);
        Toast.makeText(MainActivity.this, "Нажата кнопка", Toast.LENGTH_SHORT).show();
    }
};

public void onButtonClick(View v){
    ((Button)v).setText("Нажалась");
    Toast.makeText(MainActivity.this, "Нажата кнопка1", Toast.LENGTH_SHORT).show();
}
```

Рисунок 14 - Добавление события

14. В конструкторе страницы(code-behind) наберите:

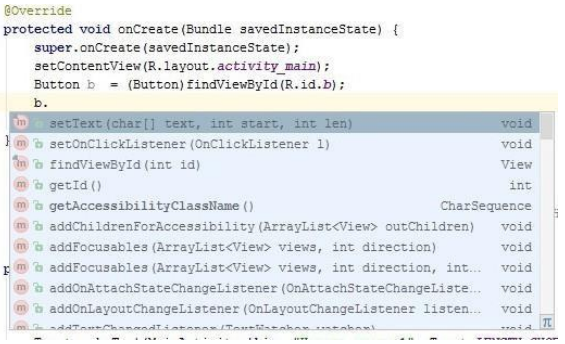


Рисунок 15 - Добавление слушателя

Выберите событие setOnClickListener. Нажмите Enter. Передайте в качестве параметра нашу объявленную переменнуюclick1.

```
Button b  = (Button)findViewById(R.id.b);
b.setOnClickListener(click1);
```

Рисунок 16 - Применение события

15. В сгенерированном методе выведите сообщение на экран о том, что у вас все получилось.

Итак, для подписки на событие в коде необходимо использовать следующую схему:

Объект.УстановитьСлушателя(имяОбработчика);

Пример: b.setOnClickListener(click1);

Для отписки от события используем следующую запись

Объект. УстановитьСлушателя(null);

Пример: b.setOnClickListener(null);

Задания:для выполнения лабораторной работы необходимо выполнить следующее:

1. Сделайте так, чтобы при нажатии на кнопку на экране отображалась ваша фамилия, как это вы делали на лабораторных работах 1 и 2.

2. Добавьте еще одну кнопку, и сделайте так, чтобы при нажатии на эту кнопку текст на ней менялся бы на текст вашей фамилии.

3. Создайте 3 кнопки. Сделайте так, чтобы при нажатии на одну из кнопок, выводилась ваша фамилия именно на той кнопке, которая была нажата (Примечание: необходимо для каждой кнопки добавить своё событие).

4. Выполните задание 3, используя аргумент view.

5. Добавьте в новое приложение 2 кнопки. Сделайте так, чтобы на экран выводилось сообщение с вашей фамилией либо при нажатии одной кнопки, либо при нажатии другой попеременно.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MSWord и оформлен согласно требованиям.
Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчета приводятся выводы о проделанной работе.

В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. CodeBehind
2. Событие
3. Слушатели

Список литературы, рекомендуемый к использованию по данной теме:

1. Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. Основы интернет-технологий: учебное пособие / Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. – М.: Архангельск: ИПЦ САФУ, 2013 – 366 с.; То же [Электронный ресурс]. - URL:http://biblioclub.ru/index.php?page=book_red&id=436379&sr=1
2. Соколова В. В. Разработка мобильных приложений: учебное пособие/ Соколова В. В. Томск: Издательство Томского политехнического университета, 2015 – 176с. То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book_red&id=442808&sr=1

Лабораторная работа № 4. Использование базы данных.

Цель работы: Научиться использовать встроенную базу данных

Теоретическая часть

В наше время практически не существует приложения, которое бы не использовало базы данных. Магазины, конструкторы, игры, всё упирается в их использование. В мобильных устройствах также существует способ хранить данные. СУБД (Система управления базами данных), используемая в телефонах называет SQLite. SQLite –упрощённая версия реляционной СУБД, обладающая базовыми возможностями полноценной СУБД.

В SQLite используются следующие типы данных:

- Null
- Integer
- Real
- Text
- Blob(бинарные)
- Numeric.

Несложно заметить, что SQLite не работает с логическими переменными Однако, сложно в наше время найти базу, которая не содержала бы значения true или false. Чтобы обойти данную "оплошность", такие значения следует хранить как обычный integer, где 1 выполняет роль true и 0 – false.

Для даты есть два способа хранения:

- Строка – ‘2013-03-27T07:58’ (Здесь используется стандартный формат даты, используемые в Америке, то есть год-месяц-день);
- Число – количество секунд с 1970-01-01T00:00:00.

Когда ваше приложение создаёт базу данных, она сохраняется в каталоге DATA/data/имя_пакета/databases/имя_базы.db.

Environment.getDataDirectory() возвращает путь к каталогу DATA.

Ход работы:

1. Для работы с SQLite необходимо создать «посредника», который бы делал всю работу сам, а мы бы просто пользовались его услугами. Таким посредником будет являться класс SQLHelper. Чтобы его создать, переходим к директории app/java и создадим в той же папке, где находится MainActivity, новый класс.

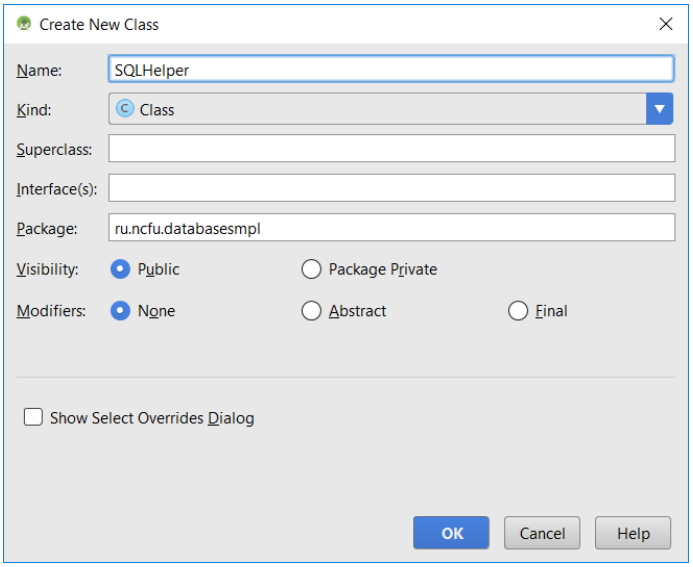


Рисунок 1 – Создание класса

SQLHelper должен наследоваться от SQLiteOpenHelper.

2. Создадим сразу же конструктор класса.

```
public SQLHelper(Context context) {
    super(context, "TrainBase", null, 1);
}
```

Рисунок 2 – Конструктор

Передаём в него контекст нашей страницы, название базы данных, фабрику и версию. Фабрика (CursorFactory) создаёт особые курсоры для БД. Вам это не пригодится. Версия – показывает текущую версию Вашей БД. Вначале версия будет равна 1. Если же, например, какие-то таблицы меняли свою структуру, то стоит повышать её версию на 1 (после первого изменения будет равно 2).

3. Переопределим методы onCreate() и onUpgrade().

```
@Override
public void onCreate(SQLiteDatabase db) {
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
}
```

Рисунок 3 – Перегрузка методов

Во время создания БД будет вызван метод onCreate(). В него следует поместить запрос на создание первоначальных таблиц.

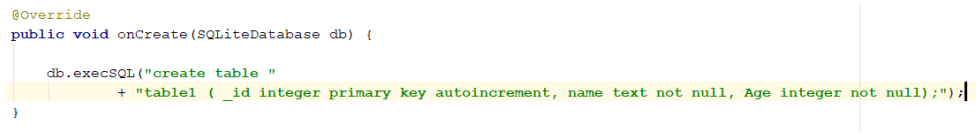


Рисунок 4 – Создание БД

В случае изменения структуры, будет вызван соответственно метод onUpgrade(). Он принимает 3 параметра: новую базу данных, номер старой версии и номер новой версии. Соответственно, можно использовать это для создания некоторых ограничений. Однако, в данном примере нам ничего не нужно менять.

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS table1");
    onCreate(db);
}
```

Рисунок 5 – Обновление БД

База данных не создаётся конкретно в этот момент, поэтому выполняется этот метод быстро. Непосредственное создание/открытие БД происходит после вызова одного из двух методов:

- getReadableDatabase() – этот метод проверяет, возможно ли получить доступ к базе данных, с целью прочесть данные. Если можно, то он вернёт базу, иначе, выдаст исключение.
- getWritableDatabase() – этот метод проверяет, возможно ли получить доступ к базе данных, с целью не только прочесть, но и записать данные в БД. Также, возвращает саму базу, если успешно.

Так как с БД порой случаются ошибки, связанные с нехваткой памяти или прав доступа, getWritableDatabase() может вызвать ошибку, которая закроет приложение. Чтобы предотвратить это, следует просто обернуть конструкцию в try/catch, вызвать getReadableDatabase() и сообщить пользователю, что не удалось подключиться для записи.

Для чтения/изменения данных обычно используются следующие методы:

- query(), rawQuery() – запрос на выборку данных;
- insert() – добавить данные в таблицу;
- delete() – удалить данные из таблицы;
- update() –обновить данные в таблице;
- execSQL(запрос) – запрос на изменение структуры таблицы.

Метод query() принимает 7 параметров:

- название таблицы;
- массив атрибутов (колонки);
- ограничение where (вместо фиксированных значений можно ставить “?”);
- массив динамических данных, используемых в where;
- groupBy;
- having;
- orderBy.

Если какой-либо из параметров Вам не нужен, необходимо добавить null.

```
int k = 4;
SQLiteDatabase db = this.getWritableDatabase();
return db.query("table1", new String[] { "_id",
                "FIO", "Age" }, "Age > ?", new String[]{String.valueOf(k)}
        , null, null, "Age");
```

Рисунок 6 – Пример запроса

Агрегатные функции также могут быть использованы в запросе.

```
SQLiteDatabase db = this.getWritableDatabase();
return db.query("table1", new String[] { "AVG(Age) AS averageAge" },null, null
        , null, null, null);
```

Рисунок 7 – Пример использования агрегатных функций

Альтернативой для query() является метод rawQuery(). Для него не нужно разбивать всё по параметрам, так как он принимает лишь строку запроса, как в обычной БД.

```
Cursor cursor = getReadableDatabase().
        rawQuery("select * from table1 where _id = ?", new String[] { Integer.toString(3) });
```

Рисунок 8 – использование rawQuery

Метод insert() принимает 3 параметра:

- название таблицы;
- Особый параметр для столбцов, которым будет присвоено значение null. Если необходимо вставить пустую строку в базу данных, то следует указать название хотя бы одного атрибута. Если строка не пустая, можно оставить это значение null.
- Объект типа ContentValues – собственно, пары значений название атрибута - значение атрибута.

```
ContentValues cv = new ContentValues();
cv.put("name", "Vasiliy");
cv.put("Age", "44");
db.insert("table1", null, cv);
```

Рисунок 9 – Пример insert

Метод update() принимает 4 параметра:

- Название таблицы;
- Данные ContentValues;
- Ограничение where;
- Динамические параметры для ограничения where.

```
ContentValues cv = new ContentValues();
cv.put("Age", "13");
db.update("table1", cv, "name = ?", new String[]{"Vasiliy"});
```

Рисунок 10 – Использование where

Последний метод, delete() принимает 3 параметра:

- Название таблицы;
- Ограничение where;
- Динамические параметры для ограничения where.

4. Возвращаемся в класс MainActivity. Создадим экземпляр посредника БД.

```
public class MainActivity extends AppCompatActivity {

    SQLHelper db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        db = new SQLHelper(this);
    }
}
```

Рисунок 11 –Создание объекта SQLHelper

5. В целях соблюдения логики, для каждой таблицы следует создавать свой класс (Модель), чтобы описать в нём атрибуты объектов. Создадим новый класс в той же папке с названием Person.

```
public class Person {
    public int _id;
    public String name;
    public int Age;
    public Person(int id, String n, int a){
        _id = id;
        name = n;
        Age = a;
    }
}
```

Рисунок 12 – Создание модели

6. Создадим метод, чтобы получить текущий список объектов в БД.

```
public Cursor getFullTable() {
    SQLiteDatabase db = this.getWritableDatabase();
    return db.query("table1", new String[] { "_id", "name", "Age" }, null,
            null, null, null, null);
}
```

Рисунок 13 – Получение всех объектов

7. Теперь обработаем полученный список и выведем его на дисплее телефона.

```
public class MainActivity extends AppCompatActivity {
    SQLHelper db;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        db = new SQLHelper( context: this);
    }
    public ArrayList<Person> getList(){
        ArrayList<Person> persons = new ArrayList<>();
        Cursor cursor  = db.getFullTable();
        if(cursor != null){
            while(cursor.moveToNext()){
                persons.add(new Person(cursor.getInt( i: 0), cursor.getString( i: 1), cursor.getInt( i: 2)));
            }
        }
        return persons;
    }
}
```

Рисунок 14 – Вывод на экран

ArrayList - динамический массив данных в языке java.

Методы getInt() и getString() автоматически преобразовывают форматы с БД в форматы для java. В скобках передаётся идентификатор атрибута, т.е., если мы выбираем Id и Name в таком порядке, то под 0 можно будет получить Id, а под 1 – имя. Иногда бывают случаи, когда неизвестно, что придёт под каким номером (например, если идёт выборка всех атрибутов). Для решения этой задачи используется метод getColumnIndex().

```
persons.add(new Person(cursor.getInt(cursor.getColumnIndex("_id")),
        cursor.getString(cursor.getColumnIndex("name")),
        cursor.getInt(2)));
```

Рисунок 15 – Пример использование getColumnIndex

В файле xml создадим LinearLayout, который будет отображать наш список людей.

```
<LinearLayout
    android:layout_width="368dp"
    android:layout_height="495dp"
    android:orientation="vertical"
    android:id="@+id/ll">
</LinearLayout>
```

Рисунок 16 – Создание менеджера для отображения данных

В методе onCreate() класса MainActivity вызовем созданный метод и результат поместим в LinearLayout.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    db = new SQLHelper(this);
    LinearLayout ll = (LinearLayout)findViewById(R.id.ll);
    LinearLayout row;
    ArrayList<Person> persons = getList();
    for (Person p: persons ) {
        row = new LinearLayout(this);
        row.setOrientation(LinearLayout.HORIZONTAL);

        TextView id = new TextView(this);
        id.setText(Integer.toString(p.Id));
        id.setWidth(120);

        TextView name = new TextView(this);
        name.setText(p.Name);
        name.setWidth(120);


        TextView Age = new TextView(this);
        Age.setText(Integer.toString(p.Age));
        name.setWidth(120);

        row.addView(id);
        row.addView(name);
        row.addView(Age);

        ll.addView(row);
    }
}
```

Рисунок 17 – Обработка полученных данных с базы

В результате, получится следующее.

Рисунок 18 – Результат выполнения

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 (8,8.1,10) и программным продуктом Android Studio.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Задания: для выполнения лабораторной работы необходимо выполнить следующее:

1. Реализовать структуру БД согласно варианту.
2. Создать методы для отображения и редактирования некоторых данных, взятых из БД.

Варианты

1. Приёмная комиссия университета.
2. Школа.
3. База таксистов.
4. Центр занятости.
5. Поликлиника.
6. Магазин электроники.
7. Магазин книг.
8. ЖКХ.
9. Железнодорожная станция.
10. Ветеринарная клиника.

11. Швейная компания.
12. Магазин продажи авто.
13. Служба ремонта телефонов.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выровнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчета приводятся выводы о проделанной работе.

В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Типы данных SQLite
2. Методы работы с SQLite
3. ООП
4. ContentValues

Список литературы, рекомендуемый к использованию по данной теме:

1. Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. Основы интернет-технологий: учебное пособие / Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. – М.: Архангельск: ИПЦ САФУ, 2013 – 366 с.; То же [Электронный ресурс]. - URL:http://biblioclub.ru/index.php?page=book_red&id=436379&sr=1
2. Соколова В. В. Разработка мобильных приложений: учебное пособие/ Соколова В. В. Томск: Издательство Томского политехнического университета, 2015 – 176с. То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php? page=book_red&id=442808&sr=1

Лабораторная работа 5. Работа с несколькими окнами.

Цель работы: Научиться создавать дополнительные окна приложения и передавать данные между ними.

Теоретическая часть

Обычно, приложение состоит из нескольких страниц, на каждой из которых происходит своё действие. Например, в играх, одна страница – главная, где отображается меню с доступными действиями. Затем, есть страница с настройками, о приложении, непосредственно сама игра и тд. Естественно, для каждого такого пункта меню создаётся собственный layout.

Чтобы создать новый layout есть два способа.

Первый способ – вручную.

1. В папке res/layout создаём новый файл.

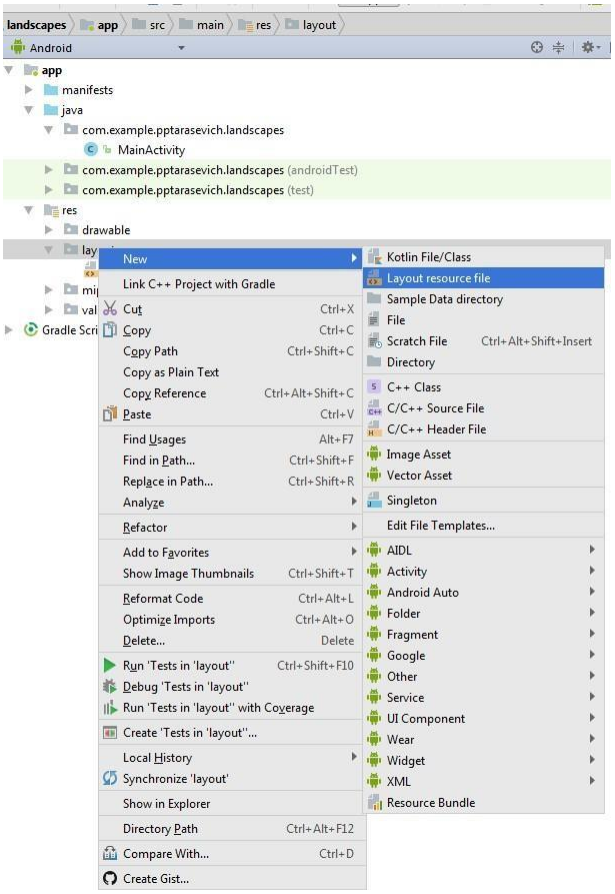


Рисунок 1 – Создание нового layout

2. Меняем содержимое этого файла, чтобы проверить, что попали куда надо.

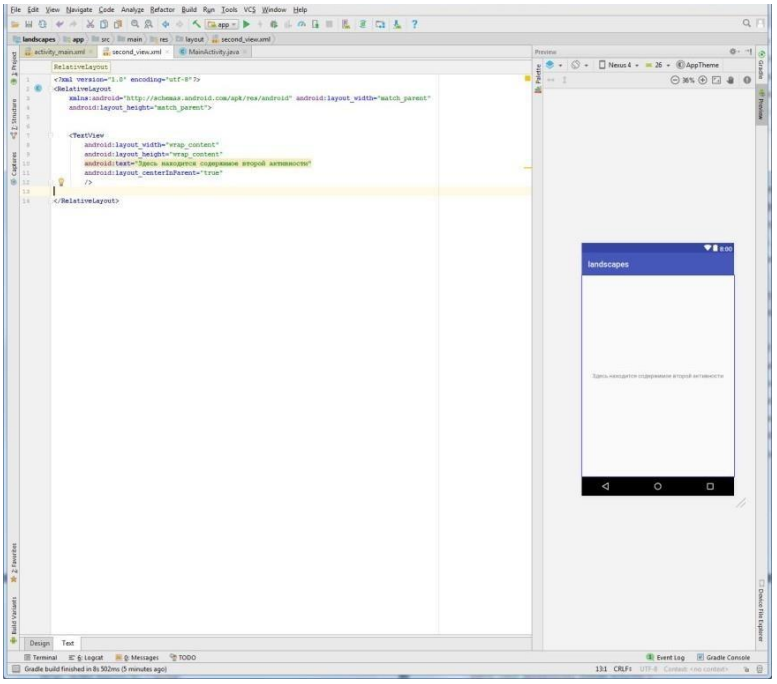


Рисунок 2 – Содержимое layout

3. Теперь создадим java-класс, который будет связан с layout-ом, также, как и MainActivity связан с activity_main.xml.

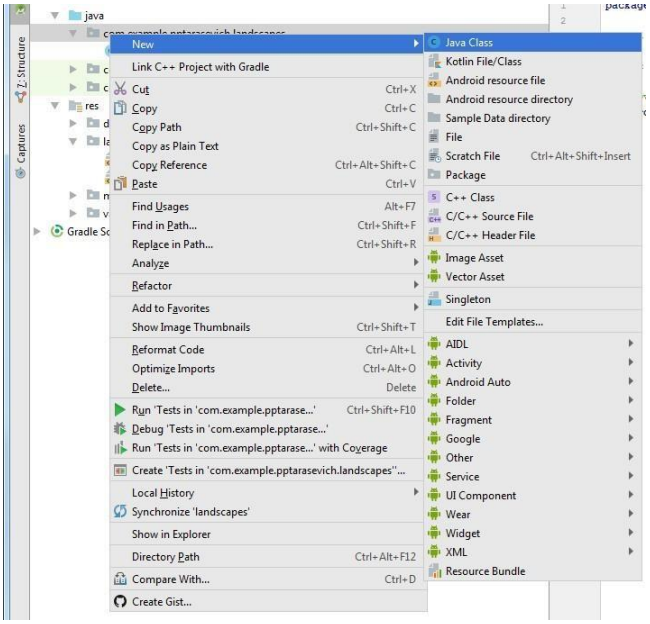


Рисунок 3 – Добавление класса

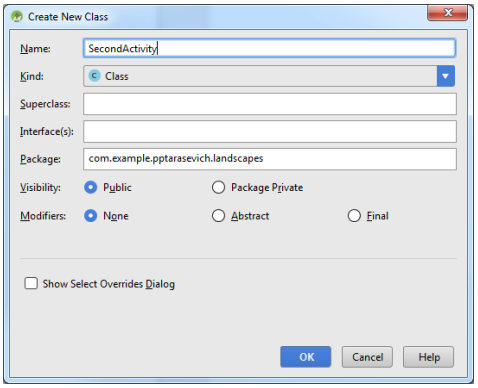


Рисунок 4 – название класса

4. Перегружаем метод onCreate, как и в MainActivity.java (можно просто скопировать), и задаём содержимое нашим layout-ом.

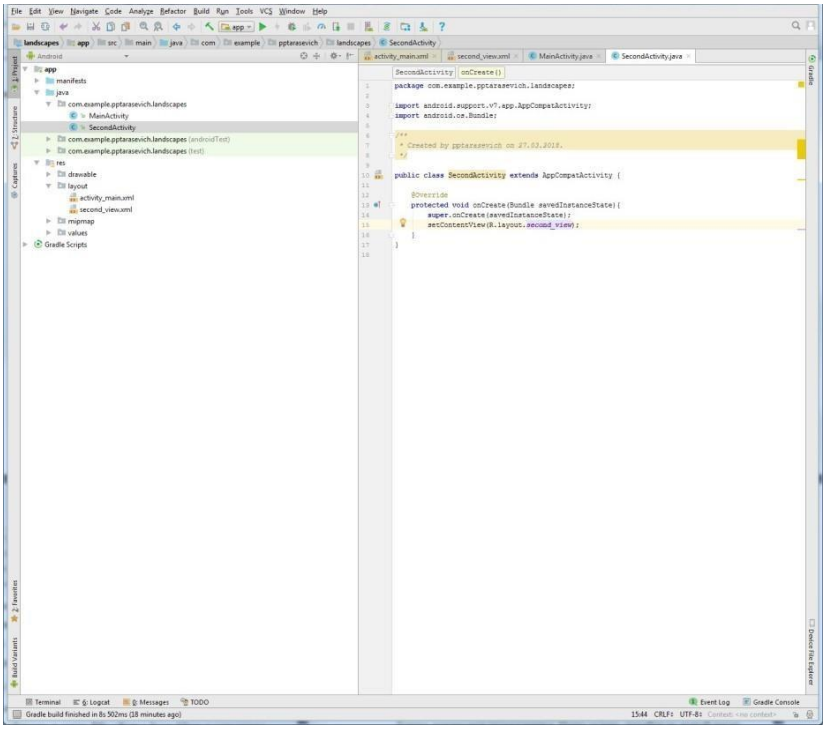


Рисунок 5 – Перегрузка onCreate()

5. Казалось бы, что ещё нужно. Но нет, необходимо зарегистрировать layout в файле AndroidManifest.xml. Если не сделать этого, то приложение закроется с ошибкой.

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="landscapes"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".SecondActivity"
        android:label="Вторая страница">
    </activity>
</application>
```

Рисунок 6 – Содержимое AndroidManifest.xml

6. Последний пункт – создадим метод для непосредственно перехода между активностями.

Intent – "намерение", которое будет служить ссылкой для перехода между активностями. Первым параметром указываем контекст, с какой активности перейти, а вторым – класс, который отвечает за активность, куда необходимо перейти. Затем, просто вызываем метод startActivity() с нашим intent-ом.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void btnClick(View view){
        Intent intent = new Intent(getApplicationContext(), SecondActivity.class);
        startActivity(intent);
    }
}
```

Рисунок 7 – Переход к новому Intent

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Переход на другую активность"
    android:onClick="btnClick"
    />
```

Рисунок 8 – Код кнопки для перехода

Второй способ создания layout-а гораздо проще. Для этого просто перейти во вкладку File, и выберите пункты, как показано на рисунке.

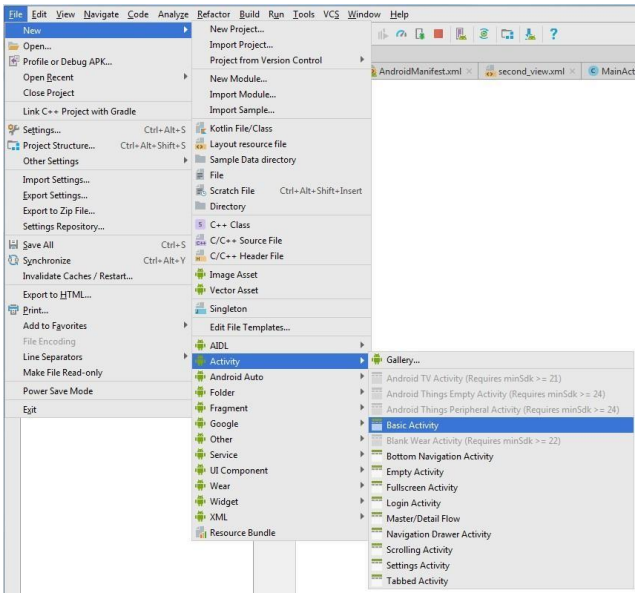


Рисунок 9 – Упрощённый способ создания

Изменим название активности на нужное.

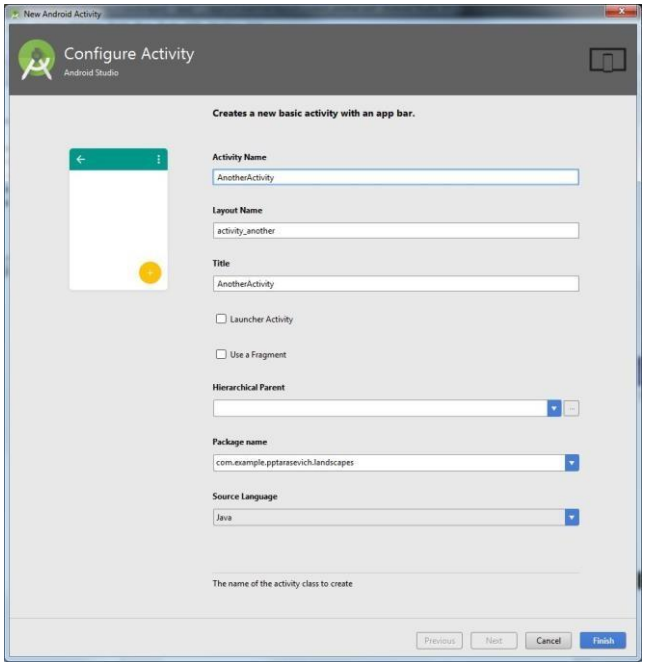


Рисунок 10 – Мастер по созданию нового Intent

Данное действие создало 2 файла и модифицировало AndroidManifest.

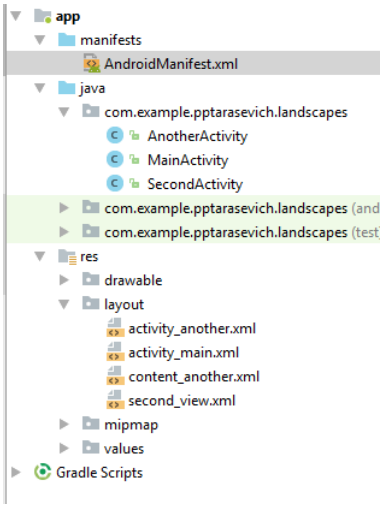


Рисунок 11 – Дерево решений

Рассмотрим поближе, что именно произошло. activity_another.xml содержит внешний вид новой активности. Удалим содержимое, кроме одной строчки.

```
<include layout="@layout/content_another" />
```

Рисунок 12 – Пример частичного layout

Данная строка позволяет добавлять одно представление внутри другого почти так же, как и ImageView. Отличие в том, что ImageView содержит всего лишь один файл или одну фигуру (чаще всего), а include позволяет добавить полноценную разметку со своими виджетами.

content_another.xml – является вложенной активностью, которую можно будет использовать в нескольких частях Вашего приложения.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Вложенное представление"/>
```

Рисунок 13 – Содержимое вложенного layout

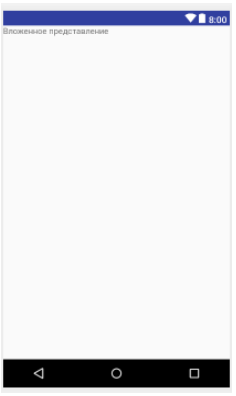


Рисунок 14 – Пример использования include

AnotherActivity.java также содержит код, созданный по умолчанию. Нам это не понадобится, поэтому смело удаляем всё незнакомое содержимое.

Используем данную активность в качестве страницы настроек. Данная страница, обычно, нужна для того, чтобы использовать одни и те же данные на всех страницах. Например, добавим RadioGroup, где будем хранить "сложность" игры.

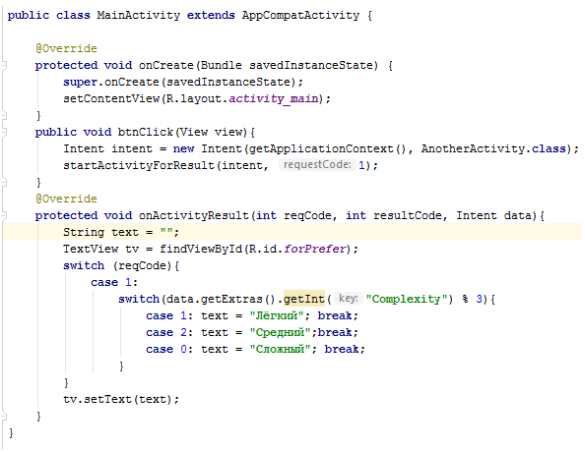
```
<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/rbtn">
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Легкий"
        />
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Средний" />

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Сложный" />
</RadioGroup>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_below="@+id/rbtn"
    android:onClick="onClick"
    android:text="Сохранить" />
```

Рисунок 15 – Содержимое второго Intent

Теперь передадим выбранный параметр в MainActivity. Для начала, изменим файл MainActivity.java таким образом, чтобы он мог получить результат от нашего Intent-а.



```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void btnClick(View view){
        Intent intent = new Intent(getApplicationContext(), AnotherActivity.class);
        startActivityForResult(intent,  requestCode: 1);
    }
    @Override
    protected void onActivityResult(int reqCode, int resultCode, Intent data){
        String text = "";
        TextView tv = findViewById(R.id.forPrefer);
        switch (reqCode){
            case 1:
                switch(data.getExtras().getInt( key: "Complexity") % 3){
                    case 1: text = "Лёгкий"; break;
                    case 2: text = "Средний";break;
                    case 0: text = "Сложный"; break;
                }
        }
        tv.setText(text);
    }
}
```

Рисунок 16 – Содержимое MainActivity.java

Сейчас нам нужно не просто запустить Activity, а получить какой-то результат оттуда, поэтому мы используем метод startActivityForResult(). Первым параметром передаём сам Intent, куда будем переходить, а вторым – так называемый requestCode, который нужен для того, чтобы однозначно понять, с какой страницы мы вернулись.

Переопределяя onActivityResult мы проверяем, какой requestCode приходит. Так как мы указали выше, что он равен 1, то его и сравниваем в первом case.

```
public class AnotherActivity extends AppCompatActivity {
    RadioGroup rbtn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_another);
        rbtn = findViewById(R.id.rbtn);
        rbtn.clearCheck();
    }

    public void onClick(View v) {
        Intent intent = new Intent();
        intent.putExtra( name: "Complexity", rbtn.getCheckedRadioButtonId());
        setResult(RESULT_OK, intent);
        finish();
    }
}
```

Рисунок 17 – Содержимое AnotherActivity.java

Чтобы записать или получить общие данные используется putExtra() и getExtra() соответственно. setResult() позволяет уточнить результат выполнения Intent-а. Например, если всё хорошо, то передаём RESULT_OK, если же возникла ошибка, то передаём RESULT_CANCELED. А finish() завершает выполнение текущего Intent-а, и возвращается к MainActivity, где в первую очередь вызывается onActionResult().

Задание: для выполнения лабораторной работы необходимо создать 2 дополнительных окна: одно будет в качестве страницы настроек, а другое – отображать информацию об авторе.

Варианты:

1. Изменение цвета фона на главной странице (не менее 3).
2. Изменение цвета текста на главной странице.
3. Изменение размера текста на главной странице.
4. Изменение количества прямоугольников на главной странице.
5. Изменение количества кругов на главной странице.

6. Изменение размеров игрового поля (3x3, 4x4, 5x5), состоящего из квадратов.
7. Смена стиля кнопок (не менее трех разных стилей).
8. Смена изображения в ImageView.
9. Смена типа фигур (круг, овал, кольцо) в ImageView.
10. Изменение размеров фигур на главной странице.

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MSWord и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчета приводятся выводы о проделанной работе.

В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Intent.

2. Перегрузка методов.

Список литературы, рекомендуемый к использованию по данной теме:

1. Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. Основы интернет-технологий: учебное пособие / Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. – М.: Архангельск: ИПЦ САФУ, 2013 – 366 с.; То же [Электронный ресурс]. - URL:http://biblioclub.ru/index.php?page=book_red&id=436379&sr=1
2. Соколова В. В. Разработка мобильных приложений: учебное пособие/ Соколова В. В. Томск: Издательство Томского политехнического университета, 2015 – 176с. То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php? page=book_red&id=442808&sr=1

Лабораторная работа 6. Таймеры и секундомеры. Логирование.

Цель работы: Научиться использовать отладочную информацию

Теоретическая часть:

Важным инструментом разработчика является отладка приложения. Пока Вы не создали приложение в идеальном виде, оно может иногда «вылетать», «обзываться» или вообще даже не запускаться. Как уже опытные программисты Вы должны знать, что такое средства отладки (точки останова, шаги с обходом и тд). При работе с AndroidStudio есть несколько нюансов. Самый главный – приложение в режиме отладки нужно запускать сразу. Что это значит: в Visual Studio Вы можете поставить точку останова когда приспичит, и она сразу же остановит приложение, когда до неё дойдёт очередь. В Android Studio же Вы должны сразу запускать приложение в режиме отладки, чтобы можно было расставлять точки останова и работать с ними. Для того чтобы запустить приложение в этом режиме, необходимо найти кнопку Debug:

После этого можно смело расставлять точки останова в любом месте приложения.

Тем не менее, точки останова не всегда удобно использовать. Бывают ситуации, когда нужные Вам данные «спрятаны» внутри объекта, который в свою очередь тоже является частью другого объекта. В таком случае, проще всего использовать Логи (журнал). В них Вы можете записывать всё, что Вам нужно: от обычной информации о переменных и объектах до отображения ошибок.

Естественно, данные Логи не предназначены для «простых смертных», поэтому, добраться к ним может только программист.

Вообще, Логи делятся на 5 основных типов:

- Info – простое информационное сообщение;
- Warning – предупреждение – ещё не ошибка, но уже стоит действовать аккуратно
- Error – ошибка – что-то в процессе выполнения пошло не так, поэтому, нужно предупредить самого себя;
- Debug – отладка – сообщения, связанные с отладкой приложения;
- Verbose – Огромный лог для подробного описания того, что произошло.

Для того чтобы записать что-то в журнал, нужно использовать соответствующий метод. В Android Studio для каждого из перечисленных типов Логов есть свой метод, который называется первой буквой определяющего слова: Например, Log.i() – для информационных сообщений, Log.e() для ошибок и тд.

Для каждого из методов есть 2 варианта, но в обоих Вы можете использовать теги для их быстрого поиска в журнале:

- С ошибкой (то есть, как только запишется Ваше сообщение в Лог, то приложение выдаст ошибку);
- Без ошибки.

```
Log.i( tag: "Тэг",  msg: "Сообщение");
```

Первым параметром идёт тэг. Он нужен для того, чтобы быстро найти нужное сообщение в журнале. Второй – непосредственно текст сообщения. Здесь можно передать какое-нибудь строковое значение переменной.

Чтобы посмотреть журнал во время работы Вашего приложения, можно найти панель, которая называется Logcat. Чтобы её открыть, нужно выбрать View→ToolWindows→Logcat. Использовать данные Логи Вы сможете при выполнении следующих заданий.

Порой возникает необходимость выполнить какое-то действие спустя некоторое время. Или же наоборот начать отслеживание выполнения

действия, чтобы потом посчитать, сколько времени ушло на его выполнение. Разберёмся по порядку.

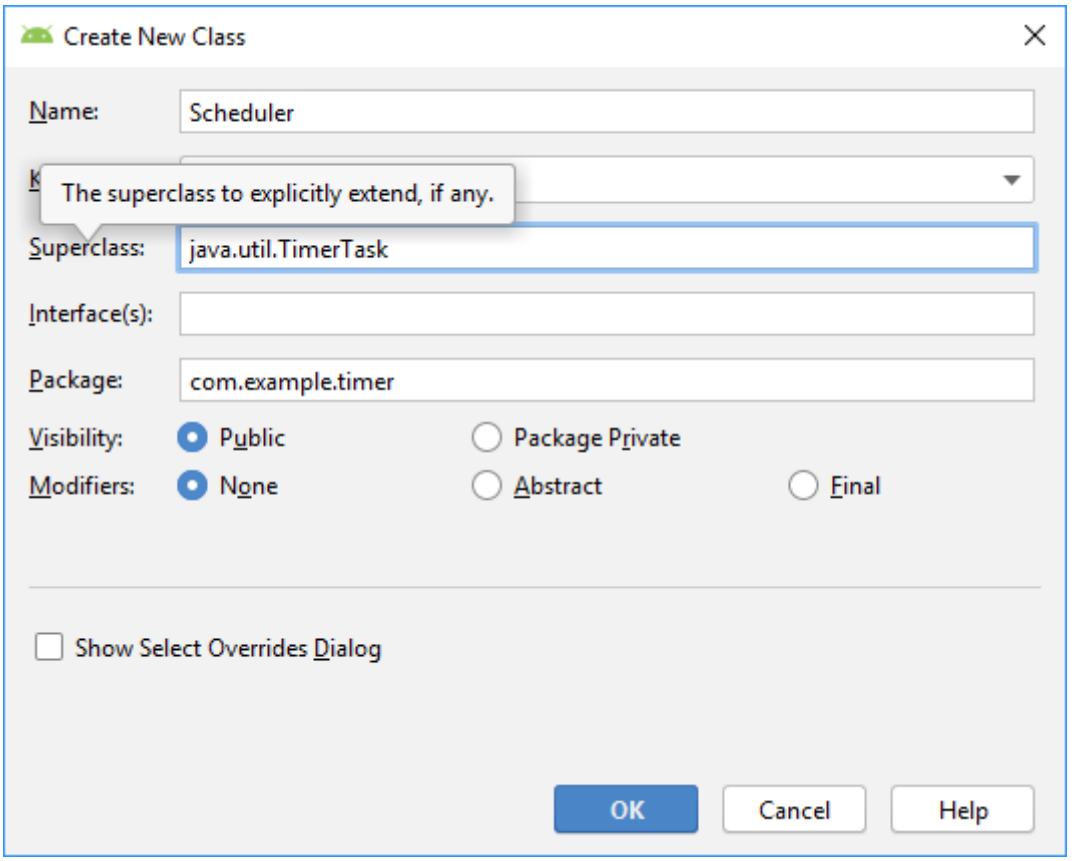
Таймер

Таймеры состоят из двух частей. Первое – то, что должно выполниться. То есть, действия, которые должны совершиться по истечении какого-то времени. Второе – это непосредственно время, спустя которое должно выполниться действие.

TimerTask – относится к первой части программирования таймера. Именно этот класс отвечает за выполнение действия. Timer – вторая часть. По странному стечению обстоятельств, таймер является частью самого себя.

Рассмотрим простой пример:

1) Создаём новое приложение.
2) В папке Java создаём новый класс:



3) Добавляем внутри нашего класса переопределённый метод run.
4) Метод должен содержать следующий код:

```
public class Scheduler extends TimerTask {
    @Override
    public void run() {
        Log.i( tag: "Информация",  msg: "Таймер сработал");
    }
}
```

5) В MainActivity создаём Экземпляр класса Timer, который будет работать по нашему планировщику.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Log.i( tag: "Тэг",  msg: "Сообщение");
    Timer timer = new Timer();
    Scheduler sch = new Scheduler();
    timer.schedule(sch,  delay: 1000);
}
```

6) Запустим приложение и убедимся, что код выполнился спустя какое-то время. Откройте вкладку logcat в приложении и убедитесь, что там есть запись, которую мы добавили.

```
10212-10272/com.example.myapplication I/Информация: Таймер сработал
```

Секундомер.

Для реализации секундомера можно использовать два способа:

- Встроенный класс Chronometer;
- Timer.

Самый простой способ – использовать Chronometer. Зачем изобретать велосипед, если он уже есть. Рассмотрим следующий пример: Нам нужно написать простой секундомер с кнопками «Запуск», «Стоп» и «Очистить». Каждую секунду отображать в TextView текущее значение секундомера.

1) Создадим второе окно в нашем приложении и добавим две кнопки на главной странице:

- Начать отсчёт таймера.
- Переход к секундомеру.

2) Переместим код таймера в обработчик нажатия по первой кнопке.

```
public void click1(View v){
    Timer timer = new Timer();
    Scheduler sch = new Scheduler();
    timer.schedule(sch,  delay: 1000);
}
```

3) Создадим ещё один обработчик для перехода на другое окно с секундомером.

```
public void click2(View v){
    Intent intent = new Intent( packageContext: this, Main2Activity.class);
    startActivity(intent);
}
```

4) Создадим следующую разметку на странице секундомера.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".Main2Activity">
    <Button
        android:layout_width="200dp"
        android:layout_height="50dp"
        android:onClick="start"
        android:text="Запуск" />

    <Button
        android:layout_width="200dp"
        android:layout_height="50dp"
        android:onClick="stop"
        android:text="Стоп" />

    <Button
        android:layout_width="200dp"
        android:layout_height="50dp"
        android:onClick="clear"
        android:text="Очистить" />

    <Chronometer
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/chronometer" />

</LinearLayout>
```

5) И напишем обработчики всех кликов:

```java
public class Main2Activity extends AppCompatActivity {

    Chronometer chr;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        chr = (Chronometer) findViewById(R.id.chronometer);
        chr.setBase(SystemClock.elapsedRealtime());
    }

    public void start(View v) {
        chr.start();
    }

    public void stop(View v) {
        chr.stop();
    }

    public void clear(View v) {
        chr.setBase(SystemClock.elapsedRealtime());
    }
}
```

Во время работы Вашего приложения оно может непроизвольно «вылетать». Это происходит из-за критической ошибки. Чтобы узнать, что за ошибка привела к этому, нужно снова обратиться к Logcat. Давайте, создадим такую ошибку и попробуем её найти в журнале:

```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Log.i( tag: "Тэг",  msg: "Сообщение");
    Log.e( tag: "ФАТАЛЬНАЯ ОШИБКА",  msg: "Всё пропало!");
}
```

```
10833-10833/com.example.myapplication W/e.myapplicatio: Accessing hidden method Landroid/view/ViewGroup;->makeOptionalFitsSystemWindows()V (greylist, reflection, allowed)
10833-10833/com.example.myapplication I/Тэг: Сообщение
10833-10833/com.example.myapplication E/ФАТАЛЬНАЯ ОШИБКА: Всё пропало!
10833-10871/com.example.myapplication D/HostConnection: HostConnection::get() New Host Connection established 0x7b53f2649780, tid 10871
```

Критические ошибки обычно хранят информацию обо всех файлах, которые её вызвали, поэтому, найти их в журнале достаточно просто: будет много красного текста. Чтобы правильно использовать логирование ошибок, необходимо помещать их в конструкцию try catch.

Задание

С помощью таймера и логов решить задачу в соответствии с вариантом:

1) С шагом 3 посчитать сумму всех простых чисел в диапазоне от 0 до 100. Каждый шаг должен выполняться в новом «тике». Вывести результат в журнале.

2) Каждую секунду в течение минуты отображать в логах следующий элемент из последовательности Фибоначчи, начиная с 1.

3) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} ax^2 + b \text{, при } x < 0 \text{ и } b \neq 0; \\ \dfrac{x - a}{x - c} \text{, при } x > 0 \text{ и } b = 0; \\ \dfrac{x}{c} \text{ в остальных случаях.} \end{cases}$$

4) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} \dfrac{1}{ax} - b, \text{при } x + 5 < 0 \text{ и } c = 0; \\ \dfrac{x - a}{x}, \text{при } x + 5 > 0 \text{ и } b \neq 0; \\ \dfrac{10x}{c - 4} \text{ в остальных случаях.} \end{cases}$$

5) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} ax^2 + bx + c \text{, при } a < 0 \text{ и } c \neq 0; \\ \dfrac{-a}{x - c}, \quad \text{при } a > 0 \text{ и } c = 0; \\ a(x + c) \text{ в остальных случаях.} \end{cases}$$

6) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} -ax - c \text{ , при } c < 0 \text{ и } x \neq 0; \\ \dfrac{x - a}{-c} \text{ , при } c > 0 \text{ и } x = 0; \\ \dfrac{bx}{c - a} \text{ в остальных случаях.} \end{cases}$$

7) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} a - \dfrac{x}{10 + b} \text{ , при } x < 0 \text{ и } b \neq 0; \\ \dfrac{x - a}{x - c} \text{ , при } x > 0 \text{ и } b = 0; \\ 3x + \dfrac{2}{c} \text{ в остальных случаях.} \end{cases}$$

8) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} ax^2 + b^2x \text{ , при } c < 0 \text{ и } b \neq 0; \\ \dfrac{x + a}{x + c} \text{ , при } c > 0 \text{ и } b = 0; \\ \dfrac{x}{c} \text{ в остальных случаях.} \end{cases}$$

9) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} -ax^2 - b \text{ , при } x < 5 \text{ и } c \neq 0; \\ \dfrac{x - a}{x} \text{ , при } x > 5 \text{ и } c = 0; \\ \dfrac{-x}{c} \text{ в остальных случаях.} \end{cases}$$

10) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} -ax^2, \text{при } c < 0 \text{ и } a \neq 0; \\ \dfrac{x-a}{cx}, \text{при } c > 0 \text{ и } a = 0; \\ \dfrac{-x}{c} \text{ в остальных случаях.} \end{cases}$$

11) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} ax^2 + b^2x, \text{при } a < 0 \text{ и } x \neq 0; \\ x - \dfrac{a}{x-c}, \text{при } a > 0 \text{ и } x = 0; \\ 1 + \dfrac{x}{c} \text{ в остальных случаях.} \end{cases}$$

12) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} ax^2 - bx + c, \text{при } x < 3 \text{ и } b \neq 0; \\ \dfrac{x-a}{x-c}, \text{при } x > 3 \text{ и } b = 0; \\ \dfrac{x}{c} \text{ в остальных случаях.} \end{cases}$$

13) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} ax^2 + \dfrac{b}{c}, \text{при } x < 1 \text{ и } c \neq 0; \\ \dfrac{x-a}{(x-c)^2}, \text{при } x > 1.5 \text{ и } c = 0; \\ \dfrac{x^2}{c^2} \text{ в остальных случаях.} \end{cases}$$

14) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} ax^3 + b^2 + c \text{ , при } x < 0.6 \text{ и } b + c \neq 0; \\ \dfrac{x - a}{x - c} \text{ , при } x > 0.6 \text{ и } b + c = 0; \\ \dfrac{x}{c} + \dfrac{x}{a} \text{ в остальных случаях.} \end{cases}$$

15) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} ax^2 + b \text{ , при } x - 1 < 0 \text{ и } b - x \neq 0; \\ \dfrac{x - a}{x} \text{ , при } x - 1 > 0 \text{ и } b + x = 0; \\ \dfrac{x}{c} \text{ в остальных случаях.} \end{cases}$$

16) После каждой итерации вывести значение функции F в определённый момент времени (x в течение одной минуты меняет своё значение от -30 до 30):

$$F = \begin{cases} -ax^3 - b \text{ , при } x + c < 0 \text{ и } a \neq 0; \\ \dfrac{x - a}{x - c} \text{ , при } x + c > 0 \text{ и } a = 0; \\ \dfrac{x}{c} + \dfrac{c}{x} \text{ в остальных случаях.} \end{cases}$$

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MSWord и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен поширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчета приводятся выводы о проделанной работе.

В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1. Таймер.

2. Chronometer.

3. Виды и описание логов.

Список литературы, рекомендуемый к использованию по данной теме:

1. Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. Основы интернет-технологий: учебное пособие / Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. – М.: Архангельск: ИПЦ САФУ, 2013 – 366 с.; То же [Электронный ресурс]. - URL:http://biblioclub.ru/index.php?page=book_red&id=436379&sr=1
2. Соколова В. В. Разработка мобильных приложений: учебное пособие/ Соколова В. В. Томск: Издательство Томского политехнического университета, 2015 – 176с. То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php? page=book_red&id=442808&sr=1

Лабораторная работа 7. Локализация и списки.

Цель работы: научиться использовать ресурсы и менять содержимое в зависимости от локализации телефона.

Теоретическая часть

Стандартное бизнес-приложение может быть использовано не только в России. Поэтому, чтобы зарубежные пользователи могли также беспрепятственно работать с Вашим приложением, можно добавить локализацию. На первой лабораторной работе мы нашли файл со строковыми ресурсами, куда выносятся общие для приложения строки. Располагается он папке values:

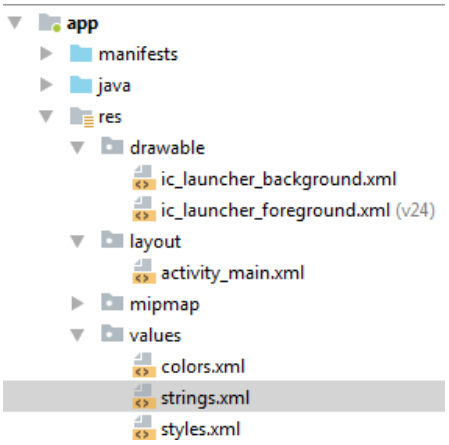


Рисунок 1 –Расположение папки values

Здесь мы можем добавлять общие значения для определённых типов. Например, в strings.xml хранится список общих строк для всего приложения. Как Вы могли заметить, если на кнопке или TextView жёстко прописать содержимое строки, то она будет выделяться коричневым цветом. Это не считается ошибкой, но является предупреждением. Лучше выводить все значения свойства android:text в файл strings.xml, так же, как и цвета в файл colors.xml.

В первую очередь необходимо разграничить ресурсы для различных стран. Поэтому, для каждой страны создадим соответствующую папку values с постфиксом краткого наименования. Так, для России создадим values-ru.

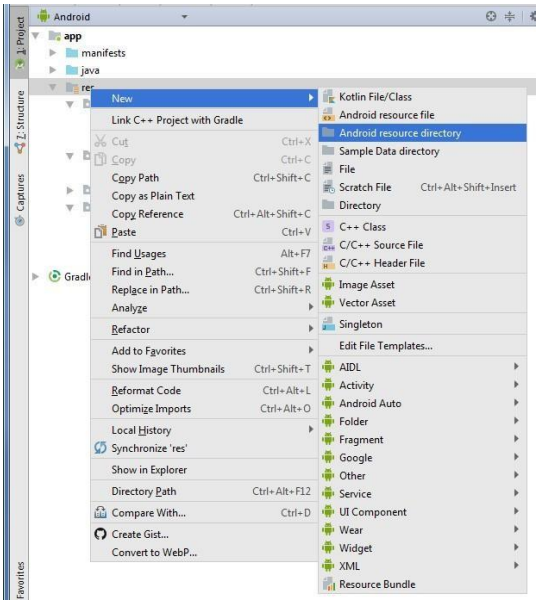


Рисунок 2 – Добавление директории

В открывшемся меню, в списке ограничений, выберем пункт Locale и нажмём кнопку, выделенную на рисунке ниже.

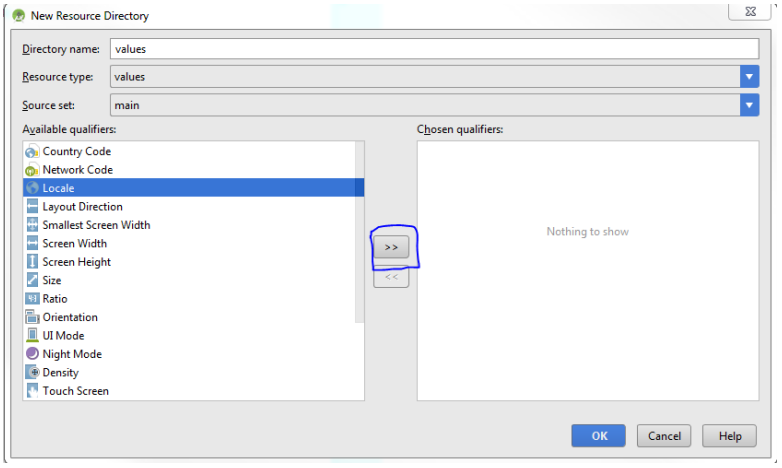


Рисунок 3 – Добавление локализаци

Найдём в списке Россию и нажмём OK.

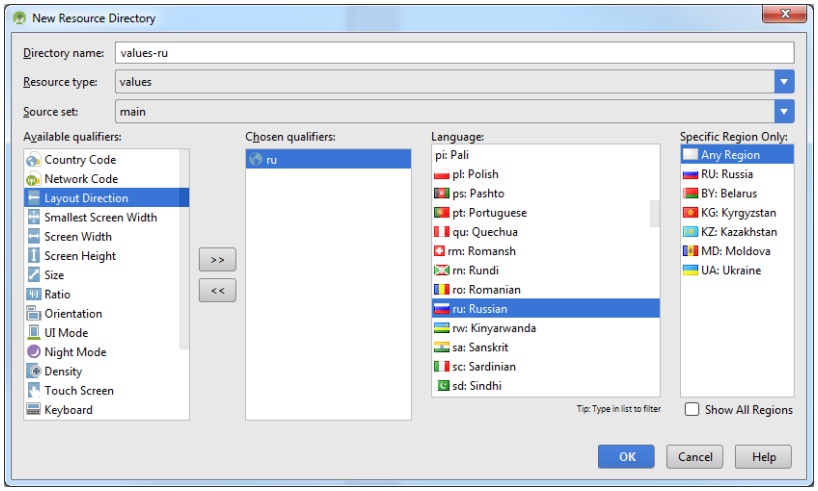


Рисунок 4 – Добавление локализации Россия

Созданная папка не будет отображаться. Чтобы добавить в неё файлы следует перейти в режим project (рис. 5).

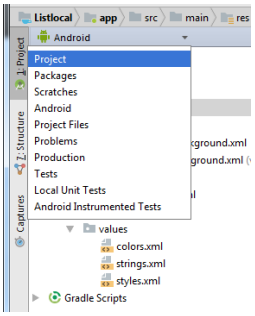


Рисунок 5 – Переключение вида

Теперь, раскроем папку с названием проекта и создадим в папке values-ru новый Values resource file.

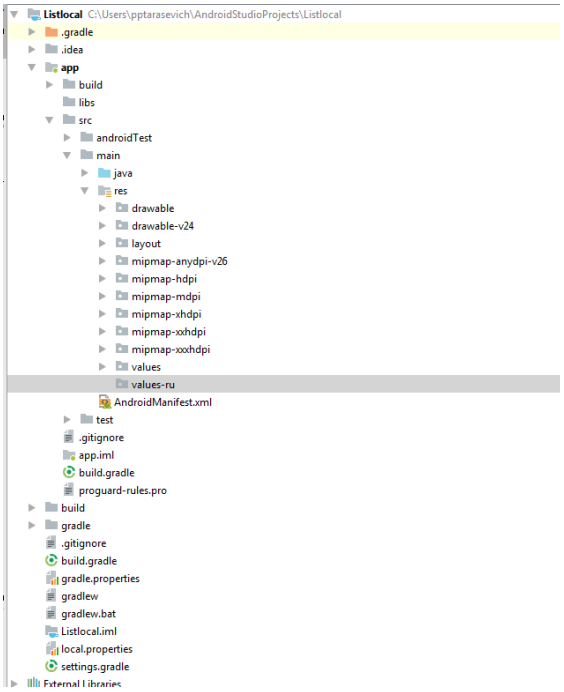


Рисунок 6 –Создание нового файла

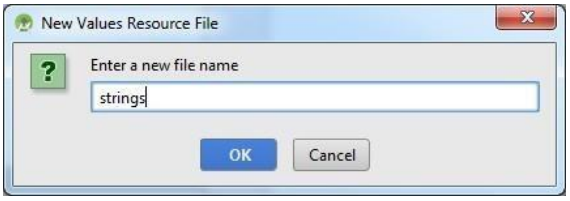


Рисунок 7 –Название файла ресурсов

Добавим строку.

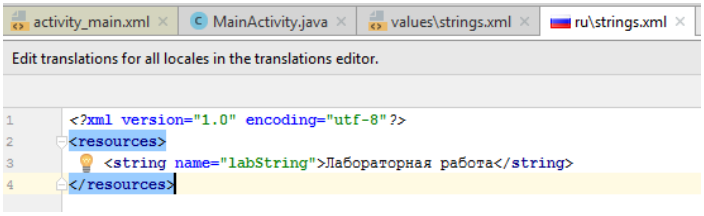


Рисунок 8 – Добавление ресурса

Теперь, если вернуться в режим Android, то вместо файла strings.xml появится папка, которая будет хранить уже два файла: один по умолчанию и второй, созданный только что.

Создадим теперь английский вариант того же самого файла и добавим строку с тем же именем.

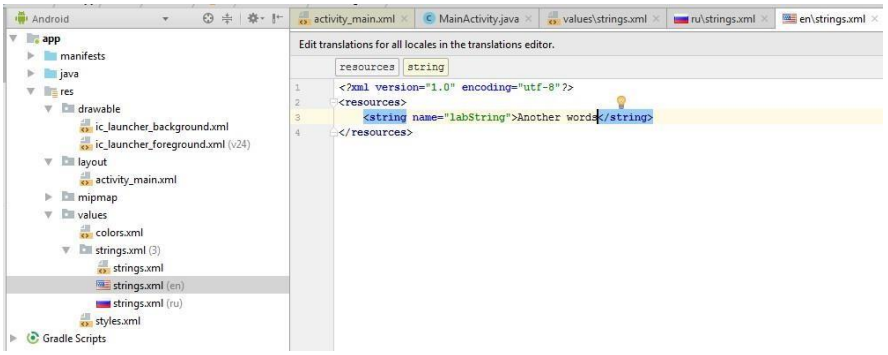


Рисунок 9 – Добавление английской локализации

Добавив в activity_main.xml TextView проверим результаты.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/labString"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Рисунок 10 – Установка ресурса

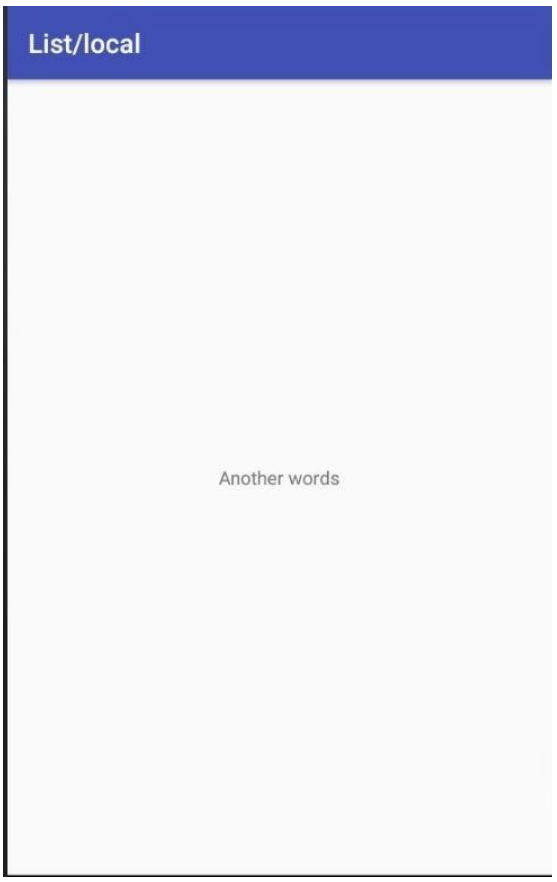


Рисунок 11 –Пример русского языка

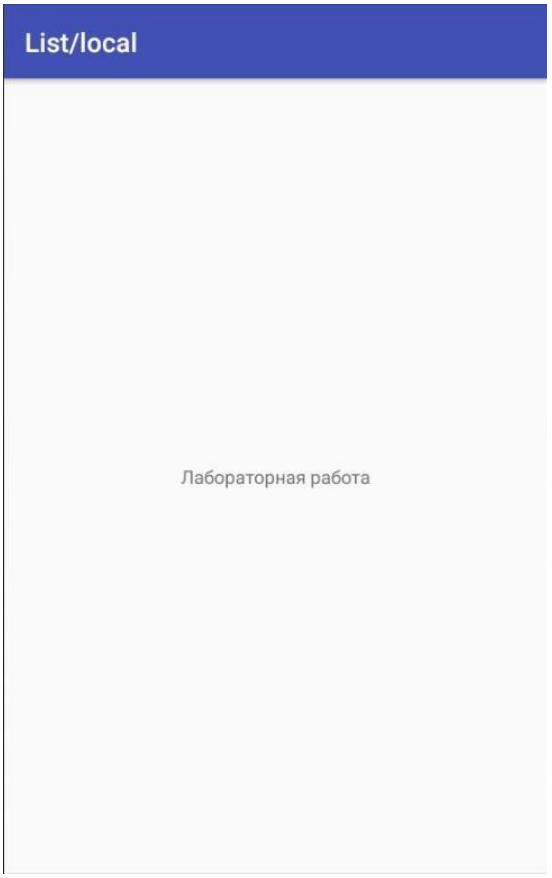


Рисунок 12 – Пример английского языка

Когда мы добавляем разметку в xml можно тут же и проверить, как будет отображаться строка для различных локализаций. Для этого в Preview есть пункт Language, где можно выбрать необходимую локализацию и просмотреть её отображение.

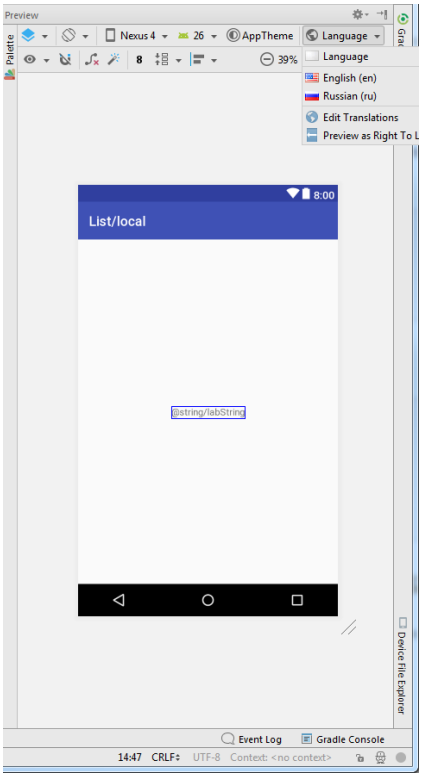


Рисунок 13 –Предпросмотр локализации

Теперь перейдём ко второму пункту лабораторной работы – списки. Списки используются повсеместно, будь то отображение контактов в телефоне, список товаров, список дел и тд. Отображать такой список с помощью TextView будет неудобно и уж тем более неправильно.

Для этой цели существует отличный виджет – ListView.

```
<ListView
    android:id="@+id/list"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

</ListView>
```

Рисунок 14 – Создание ListView

Смысл присутствия такого списка заключается в динамике. То есть, мы можем динамически удалять и добавлять элементы. В качестве посредника выступает ArrayAdapter, который содержит ссылку на источник объектов и ссылку на разметку. В качестве источника могут выступать обычные массивы, список, база данных. Чтобы заполнить ListView данными, например, из массива, используется следующий код.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onBtnClick(View v) {
        String[] array = new String[]{"Первый пункт", "Второй пункт", "Третий пункт"};
        ArrayAdapter<String> adapter = new ArrayAdapter(getApplicationContext(), android.R.layout.simple_list_item_1, array);
        ((ListView)findViewById(R.id.list)).setAdapter(adapter);
    }
}
```

Рисунок 15 – Заполнение с использованием масива

Первый параметр – это контекст страницы, затем идёт ссылка на разметку (она есть по умолчанию), а затем ссылка на источник данных. В конце мы должны применить для ListView созданный адаптер.

Также, в качестве источника могут выступать ресурсы. Это очень полезно, если используется локализация в приложении. Пример такого использования представлен ниже.

```
<resources>
    <string name="app_name">List/local</string>
    <string-array name="stringArray">
        <item>Строка 1</item>
        <item>Строка 2</item>
        <item>Строка 3</item>
        <item>Строка 4</item>
        <item>Строка 5</item>
    </string-array>
</resources>
```

Рисунок 16 –Ввод ресурсов

```
public void onBtnClick(View v) {
    String[] array = getResources().getStringArray(R.array.stringArray);
    ArrayAdapter<String> adapter = new ArrayAdapter(getApplicationContext(), android.R.layout.simple_list_item_1, array);
    ((ListView)findViewById(R.id.list)).setAdapter(adapter);
}
```

Рисунок 17 –Назначение ресурсов для ListView

Теперь нужно обработать выбор пункта из списка. Например, чтобы значение выделенной строки отображалось в TextView, изменим так, как показано на рисунке.

```
public void onBtnClick(View v) {
    String[] array = getResources().getStringArray(R.array.stringArray);
    ArrayAdapter<String> adapter = new ArrayAdapter(getApplicationContext(), android.R.layout.simple_list_item_1, array);
    ((ListView)findViewById(R.id.list)).setAdapter(adapter);
    ((ListView)findViewById(R.id.list)).setOnItemClickListener(new AdapterView.OnItemClickListener(){
        @Override
        public void onItemClick(AdapterView<?> parent, View v, int position, long id)
        {
            ((TextView)findViewById(R.id.forText)).setText(String.valueOf(parent.getItemAtPosition(position)));
        }
    });
}
```

Рисунок 18 – Обработка нажатия

В качестве последнего пункта рассмотрим, какие есть формы ввода в Android Studio. Для взаимодействия с пользователем есть несколько способов. Самый простой способ – это кнопка. То есть, при нажатии на определённую кнопку будет срабатывать определённое действие. Тем не

менее, иногда необходимо получить от пользователя неопределённую информацию. Для этого существуют поля ввода.

Самым распространённым пример служит EditText. EditText очень похож на input в html. Но, если в html абсолютно всё является input-ом, то здесь не так. Сюда, обычно, вносится некоторая текстовая информация. При этом, с помощью особого атрибута inputType можно ограничить символы, доступные пользователю. Существуют следующие ограничения:

- plainText (Значение по умолчанию) –любые символы можно вводить в данном поле;
- number – огранивает ввод только цифровыми значениями. То есть можно вставить только числа. Можно дополнительно ограничить только положительными числами (numberSigned) и возможность вводить числа с плавающей точкой (numberDecimal);
- textPassword – все символы будут заменены звёздочками (как в html). При этом можно ограничить для ввода только цифр, если добавить number (numberPassword);
- textEmailAddress – можно вносить данные только согласно шаблону (<name>@<subdomain>.<domain>);
- phone – используется для ввода телефона с возможностью добавления символов «*» и «#»;
- time – можно вводить цифры и символ «:»;
- date – можно использовать для ввода даты (цифры, «.», «-»);

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPersonName"/>
```

Рисунок 19 – Пример поля ввода

В html есть такой компонент, как textarea, который позволяет вводить текст, состоящий из нескольких строк. Здесь тоже есть такая возможность. Для этого в EditText inputType указывается как multiLine. Причём, выбрав данный тип, можно указать количество строк для отображение, указав атрибут lines.

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textMultiLine"
    android:lines="3"
    />
```

Рисунок 20 – Пример добавления многострочного поля ввода

Если Вы хотите помочь пользователю подсказкой, то можно указать так называемый placeholder. То есть некоторый текст, который будет отображаться, пока пользователь не начнёт вводить значение. В AndroidStudio данный атрибут носит название hint.

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textMultiLine"
    android:lines="3"
    android:hint="Введите имя"
    />
```


Рисунок 21 – Добавление замещающего текста

У текстовых полей есть атрибут imeOptions, с помощью которого настраиваются параметры для текущего метода ввода. Например, когда EditText получает фокус и отображается виртуальная клавиатура, эта клавиатура содержит кнопку «Next» (Далее), если атрибут imeOptions содержит значение actionNext. Если пользователь касается этой кнопки, фокус перемещается к следующему компоненту, который принимает пользовательский ввод. Если компонент EditText получает фокус и на виртуальной клавиатуре появляется кнопка «Done» (Готово), значит, использовался атрибут imeOptions со значением actionDone. Как только пользователь касается этой кнопки, система скрывает виртуальную клавиатуру.

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPersonName"
    android:hint="Введите имя"
    android:imeOptions="actionNext"
    />
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPersonName"
    android:hint="Введите фамилию"
    android:imeOptions="actionNext"
    />
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberSigned"
    android:hint="Введите возраст"
    android:imeOptions="actionDone"
    />
```

Рисунок 22 – Дополнительные возможности использования клавиатуры

Атрибут maxLength позволяет ограничить количество символов, доступных для ввода в это поле.

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPersonName"
    android:hint="Введите имя"
    android:imeOptions="actionNext"
    android:maxLength="40"
    />
```

Рисунок 23 – Ограничение количества символов

Ранее можно было запретить пользователю вводить данные с помощью атрибута editable, который схож с enabled в html. Сейчас данный атрибут устарел. Если же у Вас есть необходимость запретить пользователю вводить некоторые данные в поле (например, пока не выберет, что согласен с условиями), можно использовать атрибут focusable.

```
android:focusable="false"
```

Рисунок 24 – Запрет на редактирование

AndroidStudio не предоставляет атрибутов для валидации, как, например, атрибут pattern у input в html. Тем не менее, такая задача порой возникает у разработчиков. Для решения подобной задачи можно воспользоваться слушателем textChangeListener.

```
editText.addTextChangedListener(new TextWatcher() {
    int len = 0;

    @Override
    public void afterTextChanged(Editable s) {
        String str = editText.getText().toString();
        if (str.length() == 4 && len < str.length()) {//len check for backspace
            editText.append("-");
        }
    }

    @Override
    public void beforeTextChanged(CharSequence arg0, int arg1, int arg2, int arg3) {

        String str = editText.getText().toString();
        len = str.length();
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
    }


});
```

Рисунок 25 – Пример замены удалённых символов знаком «-»

В различные моменты времени мы можем получить значение того, что ввёл пользователь и обработать в соответствии с требованиями, которые мы указали для данного поля. Хорошо с этим справляются регулярные выражения.

Таблица 1.1 – регулярные выражения

Выражение	Описание
\d [0-9]	Одна цифра от 0 до 9.
\D [^0-9]	Любой символ кроме цифры.
\s	Пробел.
[A-Z]	Только заглавная латинская буква.
[A-Za-z]	Только латинская буква в любом регистре.

[А-Яа-яЁё]	Только русская буква в любом регистре.
[A-Za-zA-Яа-яЁё]	Любая буква русского и латинского алфавита.
[0-9]{3}	Три цифры.
[A-Za-z]{6,}	Не менее шести латинских букв.
[0-9]{,3}	Не более трёх цифр.
[0-9]{5,10}	От пяти до десяти цифр.
^[a-zA-Z]+$	Любое слово на латинице.
^[А-Яа-яЁё\s]+$	Любое слово на русском включая пробелы.
^[0-9]+$	Любое число.
[0-9]{6}	Почтовый индекс.
\d+(,\d{2})?	Число в формате 1,34 (разделитель запятая).
\d+(\.\d{2})?	Число в формате 2.10 (разделитель точка).
\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}	IP-адрес

Например

```
@Override
public void afterTextChanged(Editable s) {
    String str = editText.getText().toString();
    str.matches(String.valueOf(new Regex( pattern: "7-[0-9]{3}-[0-9]{3}-[0-9]{4}")));

}
```

Рисунок 26 – добавление паттерна ввода номера телефона

Данный патерн осуществляет следующую валидацию: номер телефона должен начинаться с 7, затем символ дефис. После этого допустимо 3 символа цифры от 0 до 9. Затем, опять дефис и 3 любые цифры, дефис и ещё 4 цифры

Для ввода даты и времени помимо EditText можно использовать интерфейс, предлагаемые AndroidStudio. Данными решениями являются DatePicker и TimePicker. Они позволяют пользователю не вводить цифры

вручную, а выбрать соответствующее значение с помощью визуального оформления в виде часов/календаря.

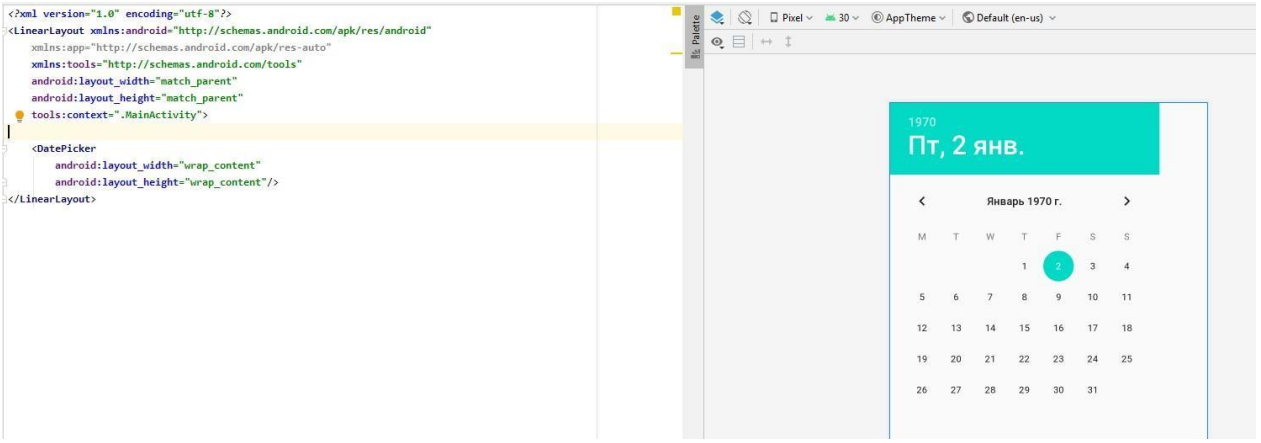


Рисунок 27 – Использование DatePicker

Вначале можно спрятать его, а по нажатию на определённую клавишу сделать видимым

```
dp.setVisibility(View.VISIBLE);
```

Рисунок 28 – изменение атрибута видимости

Также, в AnroidStudio можно использовать такие компоненты, как Checkbox и RadioButton. С RadioButton Вы уже познакомились в лабораторной работе 5. Checkbox отличается только лишь тем, что можно одновременно выбрать несколько значений из списка.

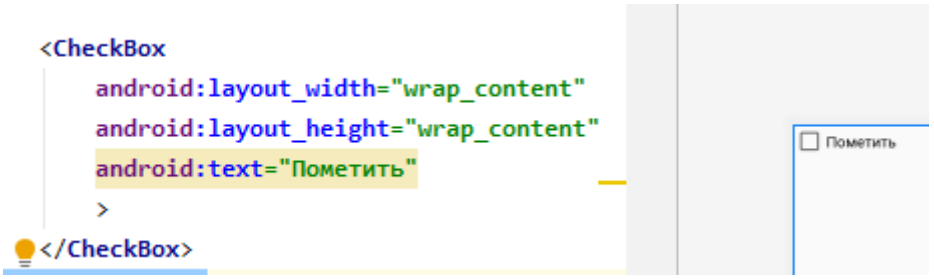


Рисунок 29 – Простейший пример Checkbox

Для выпадающего списка используется компонент Spinner:

```
        String[] array = new String[]{"Первый пункт","Второй пункт","Третий пункт"};
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(getApplicationContext(),
android.R.layout.simple_list_item_1, array);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
 {
                Toast.makeText(getApplicationContext(),  text: "Выбран " + view.toString(),
Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });
```

Рисунок 30 – Пример добавления выпадающего списка

Задания:

1. Создать список в файле ресурсов согласно варианту.
2. Создать локализованные ресурсы согласно варианту.
3. Добавить страницу для регистрации с использованием валидации:
    a. ФИО – только кириллические буквы, дефис и пробелы;
    b. Логин – только латиница;
    c. Email – валидный формат email-адрес;
    d. Номер телефона – в международной формате с вводом кода страны;
    e. Пароль;
    f. Повтор пароля – введенное значение должно совпадать с паролем;
    g. Дата рождения – валидная дата, не ранее 1900 года;
    h. Выпадающее меню для выбора из списка согласно варианту:
    i. Согласие на обработку персональных данных – должно быть отмечено.

В случае несоответствия любым требованиям выводится сообщение об ошибке и заносится в лог, поля с ошибками выделяются.

Варианты:

1. Список одногруппников (английский, русский).
2. Список имен котов (русский, казахский).
3. Список продуктов (русский, чешский).
4. Список преподавателей (русский, бурятский).
5. Список ингредиентов для пиццы (русский, итальянский).

6. Список ингредиентов для борща (русский, украинский).
7. Список заклинаний из Гарри Поттера (русский, английский).
8. Список периферийных устройств (русский, немецкий).
9. Список хоз. Инструментов (русский, вьетнамский).
10. Список мебели (русский, польский).

Содержание отчета: отчет по лабораторной работе должен быть выполнен в редакторе MSWord и оформлен согласно требованиям. Требования по форматированию: Шрифт TimesNewRoman, интервал – полуторный, поля левое – 3 см., правое – 1,5 см., верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выровнен по ширине.

Отчет должен содержать титульный лист с темой лабораторной работы, цель работы и описанный процесс выполнения вашей работы. В конце отчета приводятся выводы о проделанной работе.

В отчет необходимо вставлять скриншоты выполненной работы и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

Контрольные вопросы:

1.

Список литературы, рекомендуемый к использованию по данной теме:

1. Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. Основы интернет-технологий: учебное пособие / Пархимович М. Н. , Липницкий А. А. , Некрасова В. А. – М.: Архангельск: ИПЦ САФУ, 2013 – 366 с.; То же [Электронный ресурс]. - URL:http://biblioclub.ru/index.php?page=book_red&id=436379&sr=1
2. Соколова В. В. Разработка мобильных приложений: учебное пособие/ Соколова В. В. Томск: Издательство Томского политехнического университета, 2015 – 176с. То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php? page=book_red&id=442808&sr=1

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ
по дисциплине «Разработка мобильных приложений»
для студентов направления
10.05.03 «Информационная безопасность автоматизированных систем»
Направленность (профиль)
«Разработка автоматизированных систем в защищенном исполнении»

Ставрополь

Практическая работа №1. Анализ энергопотребления мобильного приложения и оптимизация работы Android-приложений

Цель работы

Цель работы: изучить методы анализа энергопотребления мобильных приложений в среде Android Studio, освоить использование Android Profiler для мониторинга системной нагрузки, исследовать влияние фоновых процессов и сетевой активности на расход заряда аккумулятора, а также изучить методы оптимизации работы Android-приложений с использованием механизмов WakeLock, Doze и группировки сетевых запросов.

Теоретическая часть

Одной из ключевых особенностей мобильных устройств является ограниченный объём энергетических ресурсов. В отличие от персональных компьютеров мобильные устройства функционируют от аккумуляторных батарей, поэтому эффективность использования энергии является важнейшим показателем качества мобильного приложения. Неправильно спроектированное приложение может создавать чрезмерную нагрузку на процессор, активно использовать сеть, постоянно обращаться к датчикам устройства и выполнять большое количество фоновых операций, что приводит к быстрому разряду аккумулятора и снижению производительности устройства.

Операционная система Android содержит встроенные механизмы контроля энергопотребления. Одним из основных средств анализа производительности приложений является Android Profiler — инструмент среды Android Studio, предназначенный для мониторинга загрузки процессора, использования памяти, сетевой активности и состояния энергопотребления приложения в реальном времени. Android Profiler позволяет разработчику определить, какие действия приложения создают наибольшую нагрузку на устройство, а также выявить потенциальные ошибки проектирования.

Серьёзное влияние на энергопотребление оказывает использование фоновых процессов. Многие приложения продолжают работать даже после закрытия пользовательского интерфейса, выполняя синхронизацию данных, обновление информации, отправку телеметрии или отслеживание местоположения

пользователя. Для предотвращения перехода устройства в спящий режим Android предоставляет механизм WakeLock. Данный механизм позволяет приложению временно удерживать устройство в активном состоянии для завершения определённых операций. Однако длительное удержание WakeLock приводит к постоянной активности процессора и значительному расходу энергии. По этой причине разработчик обязан минимизировать время использования блокировки и освобождать её сразу после завершения необходимых действий.

В современных версиях Android используется система интеллектуального энергосбережения Doze. Данный режим автоматически активируется при длительном бездействии устройства и ограничивает выполнение фоновых задач, сетевую активность и синхронизацию приложений. Основная цель режима Doze заключается в увеличении времени автономной работы устройства. При этом приложения, разработанные без учёта особенностей режима энергосбережения, могут работать нестабильно или некорректно выполнять фоновые операции.

Дополнительным фактором повышенного энергопотребления являются частые сетевые запросы. Каждый сетевой обмен активирует сетевой модуль устройства, что требует дополнительных затрат энергии. Если приложение выполняет большое количество отдельных запросов через короткие промежутки времени, расход аккумулятора существенно возрастает. Для уменьшения нагрузки применяется механизм группировки сетевых запросов. Данный подход позволяет объединять несколько операций передачи данных в один сетевой сеанс, снижая количество обращений к сети и повышая энергоэффективность мобильного приложения.

Для специалистов в области информационной безопасности анализ энергопотребления мобильных приложений имеет дополнительное значение. Вредоносное программное обеспечение может скрыто использовать вычислительные ресурсы устройства, выполнять несанкционированную передачу данных, осуществлять скрытый сбор информации или использовать устройство для майнинга криптовалюты. Подобная активность часто сопровождается повышенным энергопотреблением и аномальной фоновой

активностью. Поэтому специалист должен уметь выявлять признаки неэффективной или потенциально вредоносной работы мобильных приложений.

Практическая часть

Постановка задачи

В рамках практической работы студенту необходимо выполнить комплексное исследование энергопотребления Android-приложения с использованием встроенных средств анализа Android Studio. Работа направлена на изучение факторов, влияющих на расход аккумулятора мобильного устройства, а также на получение практических навыков оптимизации фоновой активности приложения.

В процессе выполнения задания студент должен создать тестовый проект мобильного приложения либо использовать готовое Android-приложение с выраженной фоновой активностью. В качестве объекта исследования могут использоваться приложения, связанные с обновлением данных через сеть, обработкой сообщений, воспроизведением мультимедиа, геолокацией или периодической синхронизацией информации.

На первом этапе работы необходимо подготовить среду разработки и подключить мобильное устройство либо эмулятор Android. После запуска приложения студент должен открыть инструмент Android Profiler и изучить показатели работы приложения в реальном времени. Особое внимание необходимо уделить активности центрального процессора, объёму используемой памяти, количеству сетевых запросов и изменениям уровня энергопотребления во время работы приложения.

В ходе наблюдения студент должен определить, какие действия приложения создают повышенную нагрузку на систему. Для этого необходимо последовательно выполнять различные пользовательские сценарии: запуск приложения, переход между экранами, обновление данных, работа с сетью, выполнение фоновых операций и сворачивание приложения. Полученные результаты необходимо проанализировать и определить потенциальные причины повышенного расхода энергии.

Следующим этапом работы является исследование фоновой активности приложения. Студент должен изучить, как приложение продолжает выполнять задачи после сворачивания пользовательского интерфейса. Необходимо проанализировать, какие системные процессы продолжают функционировать в фоновом режиме, насколько активно приложение использует процессор и сеть, а также изменяется ли уровень энергопотребления при длительной работе программы.

После этого необходимо изучить механизм WakeLock. В рамках задания студент должен исследовать ситуации, в которых приложение удерживает устройство в активном состоянии. Требуется определить, насколько длительно приложение использует системные ресурсы без перехода устройства в спящий режим, а также оценить влияние данной активности на расход аккумулятора.

Далее необходимо выполнить оптимизацию работы приложения путём сокращения времени удержания WakeLock и уменьшения количества фоновых операций.

Особое внимание следует уделить режиму энергосбережения Doze. Студент должен перевести устройство в режим ожидания и проанализировать поведение приложения при ограничении фоновой активности. Необходимо определить, какие функции приложения продолжают работать, какие задачи приостанавливаются системой и каким образом изменяется сетевое взаимодействие. В ходе анализа требуется оценить, соответствует ли работа приложения рекомендациям Android по энергосбережению.

Следующим этапом является исследование сетевой активности приложения. Студенту необходимо определить количество сетевых обращений, выполняемых приложением в процессе работы. После анализа необходимо выполнить оптимизацию сетевого взаимодействия путём группировки запросов и уменьшения частоты передачи данных. Важно оценить, как изменение логики сетевого взаимодействия влияет на энергопотребление устройства и общую производительность приложения.

После выполнения оптимизации студент должен повторно провести анализ приложения с использованием Android Profiler и сравнить полученные результаты с первоначальными показателями. В процессе сравнения

необходимо определить, удалось ли снизить нагрузку на процессор, уменьшить сетевую активность и сократить энергопотребление приложения.

По результатам выполнения практической работы студент должен подготовить отчёт, содержащий:

- описание исследуемого приложения;
- результаты анализа энергопотребления;
- описание выявленных проблем;
- описание выполненных оптимизаций;
- сравнительный анализ показателей до и после оптимизации;
- выводы о влиянии фоновой активности и сетевых операций на расход аккумулятора мобильного устройства.

Варианты заданий

№ варианта	**Содержание задания**
1	Выполнить анализ энергопотребления музыкального проигрывателя, работающего в фоновом режиме, определить влияние воспроизведения мультимедиа на расход аккумулятора и предложить способы оптимизации работы приложения.
2	Исследовать влияние постоянного использования GPS-модуля на энергопотребление мобильного устройства и определить методы снижения нагрузки при работе с геолокацией.
3	Проанализировать работу чат-приложения с частыми сетевыми запросами и выполнить оптимизацию передачи сообщений с использованием группировки сетевых операций.
4	Исследовать энергопотребление приложения облачного хранилища при автоматической синхронизации файлов и предложить методы уменьшения фоновой активности.
5	Выполнить анализ работы приложения

	видеонаблюдения, определить влияние потоковой передачи видео на энергопотребление и предложить способы оптимизации.
6	Исследовать влияние push-уведомлений и фоновой синхронизации на время автономной работы мобильного устройства.
7	Проанализировать работу мобильного приложения, выполняющего периодическую передачу телеметрических данных на сервер, и реализовать механизм пакетной передачи информации.
8	Исследовать энергопотребление новостного приложения с автоматическим обновлением контента и определить оптимальную частоту синхронизации данных.
9	Выполнить анализ работы фитнес-приложения при использовании датчиков движения, GPS и фоновой активности.
10	Исследовать влияние постоянной работы фоновых сервисов на производительность и энергопотребление Android-устройства.
11	Проанализировать влияние частых обращений к локальной базе данных на энергопотребление мобильного приложения.
12	Исследовать работу приложения потокового аудио и определить способы уменьшения нагрузки на сеть и аккумулятор устройства.
13	Выполнить исследование поведения мобильного приложения в режиме Doze и определить ограничения, накладываемые системой Android на фоновую активность.
14	Исследовать энергопотребление приложения

	загрузки файлов и предложить методы оптимизации передачи данных через сеть.
15	Проанализировать сетевую активность мобильного браузера при открытии веб-страниц и определить факторы повышенного расхода энергии.
16	Разработать механизм адаптивного обновления данных в мобильном приложении в зависимости от уровня заряда аккумулятора устройства.
17	Исследовать влияние различных режимов работы мобильного устройства на энергопотребление приложения с фоновой синхронизацией.
18	Выполнить анализ энергопотребления мобильного IoT-клиента, осуществляющего постоянный обмен данными с удалённым сервером.
19	Исследовать влияние очереди сетевых запросов на производительность и расход энергии мобильного приложения.
20	Выполнить сравнительный анализ энергопотребления двух Android-приложений с различной организацией фоновой активности.

Контрольные вопросы

1. Что представляет собой Android Profiler?
2. Для чего используется анализ энергопотребления мобильного приложения?
3. Какие параметры можно отслеживать с помощью Android Profiler?
4. Что такое фоновая активность приложения?
5. Как фоновая активность влияет на расход аккумулятора?
6. Для чего используется механизм WakeLock?
7. Какие проблемы возникают при неправильном использовании WakeLock?
8. Что представляет собой режим Doze?
9. В каких случаях Android активирует режим Doze?

10. Какие ограничения накладываются на приложения в режиме энергосбережения?
11. Почему частые сетевые запросы увеличивают энергопотребление?
12. Что такое группировка сетевых запросов?
13. Каким образом оптимизация сетевой активности снижает расход энергии?
14. Какие задачи выполняют фоновые сервисы Android?
15. Почему необходимо ограничивать работу приложения в фоновом режиме?
16. Какие признаки могут указывать на повышенное энергопотребление приложения?
17. Как можно определить источник повышенной нагрузки на систему?
18. Какие методы оптимизации энергопотребления используются в Android?
19. Почему анализ энергопотребления важен для специалистов по информационной безопасности?
20. Какие последствия может вызвать некорректная работа фоновых процессов мобильного приложения?

Список литературы

1. Билл Филлипс, Крис Стюарт, Кристин Марсикано, Брайан Гарднер Android-программирование для профессионалов. — СПб.: Питер, 2023. — 912 с.
2. Google Developers. Android NDK Guides [Электронный ресурс]. — Режим доступа: Android NDK Guides
3. Google Developers. JNI Tips [Электронный ресурс]. — Режим доступа: JNI Tips
4. Liu P., Li L., Yan Y. Identifying and Characterizing Silently-Evolved Methods in the Android API [Электронный ресурс]. — Режим доступа: arXiv.org
5. Google Developers. ConstraintLayout [Электронный ресурс]. — Режим доступа: ConstraintLayout Documentation
6. Google Developers. Test your app on Android [Электронный ресурс]. — Режим доступа: Test your app on Android

Практическая работа №2. Создание мобильного приложения, реагирующего на встряхивание устройства и изменение ориентации экрана

Цель работы

Цель работы: изучить принципы работы датчиков мобильного устройства в операционной системе Android, освоить методы обработки событий изменения положения смартфона в пространстве, научиться реализовывать реакцию приложения на встряхивание устройства и изменение ориентации экрана, а также исследовать особенности обработки событий жизненного цикла Android-приложений.

Теоретическая часть

Современные мобильные устройства оснащаются большим количеством встроенных датчиков, обеспечивающих взаимодействие программного обеспечения с физическим окружением пользователя. Использование сенсоров позволяет мобильным приложениям реагировать на движение устройства, изменение положения в пространстве, ускорение, наклон и другие физические параметры. Благодаря этому разработчики могут создавать интерактивные приложения, системы навигации, игровые приложения, фитнес-трекеры, средства дополненной реальности и иные программные решения.

Одним из наиболее распространённых датчиков является акселерометр. Данный сенсор измеряет ускорение устройства по трём координатным осям и позволяет определить изменение положения смартфона в пространстве. На основе показаний акселерометра приложение может обнаруживать встряхивание устройства, наклон, изменение ориентации экрана и другие движения пользователя. Работа с акселерометром требует постоянного получения и анализа данных, поступающих от сенсора в режиме реального времени.

Событие встряхивания устройства (shake) широко используется в мобильных приложениях. Например, встряхивание может применяться для обновления данных, отмены действия, переключения режимов работы приложения или активации дополнительных функций. Для определения встряхивания приложение анализирует резкие изменения ускорения по нескольким осям. При

этом важно учитывать чувствительность обработки данных, поскольку слишком высокая чувствительность может приводить к ложным срабатываниям, а слишком низкая — к отсутствию реакции на действия пользователя.

Другим важным аспектом работы мобильных приложений является обработка изменения ориентации экрана. Android автоматически изменяет ориентацию пользовательского интерфейса при повороте устройства между портретным и альбомным режимами. При этом система может полностью пересоздавать активность приложения, что влияет на сохранение данных интерфейса и состояние программы. Если разработчик не учитывает особенности жизненного цикла Android-приложения, при изменении ориентации возможно потеря пользовательских данных, сброс состояния интерфейса или повторный запуск отдельных компонентов приложения.

Жизненный цикл Activity представляет собой последовательность состояний приложения во время его работы. При изменении ориентации экрана Android может вызывать уничтожение и повторное создание Activity. В связи с этим разработчику необходимо реализовывать механизмы сохранения состояния приложения, чтобы пользователь не терял введённые данные или текущий прогресс работы.

Использование датчиков мобильного устройства требует рационального управления ресурсами. Постоянное обращение к сенсорам может увеличивать нагрузку на процессор и ускорять расход заряда аккумулятора. Поэтому Android-приложения должны корректно регистрировать и отключать обработчики датчиков в зависимости от текущего состояния приложения. Особенно важно освобождать ресурсы при сворачивании программы или переходе приложения в фоновый режим.

Для специалистов в области информационной безопасности работа с сенсорами представляет отдельный интерес. Датчики мобильного устройства могут использоваться не только легитимными приложениями, но и вредоносным программным обеспечением. Например, акселерометр может применяться для скрытого анализа поведения пользователя, определения характера его действий или косвенного сбора информации о перемещении устройства. Поэтому

разработчик должен учитывать вопросы безопасности и конфиденциальности при проектировании мобильных приложений, использующих сенсоры.

Практическая часть

Постановка задачи

В рамках практической работы студенту необходимо разработать Android-приложение, способное реагировать на встряхивание мобильного устройства и изменение ориентации экрана. Работа направлена на изучение принципов взаимодействия приложения с аппаратными датчиками смартфона, обработку событий изменения положения устройства в пространстве, а также исследование особенностей жизненного цикла Android-приложений.

На первом этапе выполнения работы студенту необходимо подготовить среду разработки Android Studio и создать новый проект мобильного приложения. После создания проекта требуется изучить структуру Android-приложения, определить основные компоненты пользовательского интерфейса и подготовить элементы, отображающие результаты работы датчиков.

Далее необходимо исследовать работу встроенных сенсоров мобильного устройства. В процессе выполнения задания студент должен изучить доступные датчики Android-устройства, определить наличие акселерометра и ознакомиться с принципами получения данных от сенсоров. Особое внимание следует уделить обработке событий изменения ускорения устройства и анализу показаний датчиков по различным координатным осям.

Следующим этапом является реализация механизма обнаружения встряхивания устройства. Студенту необходимо организовать получение данных от акселерометра и определить критерии, при которых приложение будет считать движение устройства встряхиванием. После обнаружения встряхивания приложение должно выполнять определённое действие, например отображать уведомление, изменять содержимое интерфейса, запускать анимацию или обновлять данные.

В процессе выполнения работы необходимо исследовать чувствительность реакции приложения на движение устройства. Студент должен определить, как изменение параметров обработки данных влияет на точность определения

встряхивания, а также проанализировать вероятность ложных срабатываний при обычном использовании смартфона.

После реализации обработки встряхивания необходимо изучить изменение ориентации экрана мобильного устройства. Студент должен исследовать поведение приложения при переходе между портретным и альбомным режимами отображения. Требуется определить, как изменяется пользовательский интерфейс приложения при повороте устройства и каким образом Android управляет жизненным циклом Activity во время изменения конфигурации экрана.

Особое внимание необходимо уделить сохранению состояния приложения при изменении ориентации. В рамках задания студент должен проанализировать, сохраняются ли данные интерфейса после поворота устройства, и реализовать механизмы предотвращения потери информации. Необходимо исследовать, какие методы Android позволяют сохранять текущее состояние пользовательского интерфейса и восстанавливать его после повторного создания Activity.

Следующим этапом работы является анализ производительности приложения и использования системных ресурсов. Студент должен исследовать, как постоянная работа с датчиками влияет на загрузку процессора и расход заряда аккумулятора. Требуется определить необходимость отключения обработки сенсоров при сворачивании приложения и проанализировать последствия неправильного управления обработчиками датчиков.

В ходе выполнения работы студенту необходимо протестировать приложение на различных сценариях использования: встряхивание устройства различной интенсивности, медленные наклоны смартфона, многократные повороты экрана, сворачивание и повторный запуск приложения. По результатам тестирования требуется определить устойчивость работы приложения и корректность обработки событий.

После завершения практической части студент должен подготовить отчёт, содержащий:

- описание разработанного приложения;
- описание используемых датчиков мобильного устройства;

- анализ работы механизма обнаружения встряхивания;
- результаты исследования изменения ориентации экрана;
- описание проблем, возникших при обработке жизненного цикла Activity;
- результаты тестирования приложения;
- выводы о работе мобильных сенсоров и особенностях обработки конфигурационных изменений в Android.

Варианты заданий

№ варианта	**Содержание задания**
1	Разработать приложение, которое при встряхивании устройства изменяет цвет фона пользовательского интерфейса.
2	Реализовать приложение, выполняющее обновление отображаемых данных после встряхивания смартфона.
3	Создать приложение, реагирующее на встряхивание запуском анимации элементов интерфейса.
4	Разработать приложение, изменяющее ориентацию расположения элементов интерфейса при повороте устройства.
5	Реализовать приложение для отображения текущей ориентации устройства и состояния акселерометра.
6	Создать приложение, выполняющее очистку текстового поля после встряхивания смартфона.
7	Разработать приложение с автоматическим изменением размера элементов интерфейса при смене ориентации экрана.
8	Реализовать мобильное приложение, изменяющее изображение на экране при обнаружении встряхивания устройства.
9	Создать приложение, выполняющее подсчёт

	количества встряхиваний устройства за определённый промежуток времени.
10	Разработать приложение, сохраняющее данные пользовательского интерфейса после поворота экрана.
11	Реализовать приложение, отображающее предупреждение при слишком резком движении устройства.
12	Создать приложение, изменяющее расположение кнопок и текстовых элементов в зависимости от ориентации экрана.
13	Разработать приложение с функцией случайного выбора значения после встряхивания смартфона.
14	Реализовать приложение, реагирующее на изменение положения устройства запуском различных режимов интерфейса.
15	Создать приложение для тестирования чувствительности акселерометра мобильного устройства.
16	Разработать приложение, сохраняющее результаты работы датчиков после пересоздания Activity.
17	Реализовать приложение, изменяющее параметры отображения графических элементов при наклоне устройства.
18	Создать приложение, анализирующее интенсивность движения смартфона и отображающее результаты на экране.
19	Разработать приложение с адаптацией интерфейса под портретный и альбомный режимы отображения.
20	Реализовать приложение, выполняющее различные

	действия в зависимости от силы встряхивания устройства.

Контрольные вопросы

1. Что представляет собой акселерометр мобильного устройства?
2. Какие данные предоставляет акселерометр Android-приложению?
3. Для чего используется обработка встряхивания устройства?
4. Что такое сенсоры мобильного устройства?
5. Какие виды датчиков используются в Android-устройствах?
6. Что такое изменение ориентации экрана?
7. Какие режимы ориентации поддерживаются Android?
8. Что происходит с Activity при повороте устройства?
9. Что представляет собой жизненный цикл Activity?
10. Почему необходимо сохранять состояние приложения при изменении конфигурации экрана?
11. Какие проблемы могут возникнуть при пересоздании Activity?
12. Почему важно отключать обработчики датчиков при сворачивании приложения?
13. Как использование сенсоров влияет на энергопотребление устройства?
14. Какие действия может выполнять приложение при обнаружении встряхивания?
15. Что такое ложное срабатывание сенсора?
16. Какие методы используются для фильтрации данных акселерометра?
17. Почему работа с датчиками важна для мобильных приложений?
18. Какие риски информационной безопасности связаны с использованием сенсоров?
19. Почему необходимо тестировать приложение на различных сценариях движения устройства?
20. Какие особенности необходимо учитывать при разработке адаптивного пользовательского интерфейса Android-приложения?

Список литературы

7. Билл Филлипс, Крис Стюарт, Кристин Марсикано, Брайан Гарднер Android-программирование для профессионалов. — СПб.: Питер, 2023. — 912 с.
8. Google Developers. Android NDK Guides [Электронный ресурс]. — Режим доступа: Android NDK Guides
9. Google Developers. JNI Tips [Электронный ресурс]. — Режим доступа: JNI Tips
10. Liu P., Li L., Yan Y. Identifying and Characterizing Silently-Evolved Methods in the Android API [Электронный ресурс]. — Режим доступа: arXiv.org
11. Google Developers. ConstraintLayout [Электронный ресурс]. — Режим доступа: ConstraintLayout Documentation
12. Google Developers. Test your app on Android [Электронный ресурс]. — Режим доступа: Test your app on Android

Практическая работа №3. Реализация загрузки данных из сети без блокировки интерфейса с использованием Coroutines, AsyncTask и RxJava

Цель работы

Цель работы: изучить методы организации асинхронного выполнения операций в Android-приложениях, освоить способы загрузки данных из сети без блокировки пользовательского интерфейса, исследовать особенности использования Kotlin Coroutines, AsyncTask и RxJava, а также получить практические навыки разработки отзывчивых мобильных приложений.

Теоретическая часть

Современные мобильные приложения активно взаимодействуют с сетевыми ресурсами. Получение данных с удалённых серверов используется в социальных сетях, мессенджерах, банковских приложениях, системах мониторинга, облачных сервисах и других мобильных решениях. При этом выполнение сетевых операций связано с задержками передачи данных, временем ожидания ответа сервера и нестабильностью сетевого соединения. Если сетевой запрос выполняется в основном потоке приложения, пользовательский интерфейс перестаёт реагировать на действия пользователя, что приводит к снижению удобства использования программы и возникновению ошибок выполнения.

В операционной системе Android существует разделение между основным потоком приложения и фоновыми потоками выполнения. Основной поток, также называемый UI-потоком, отвечает за отображение интерфейса и обработку пользовательских действий. Любые длительные операции, включая загрузку файлов, обращение к сети, обработку изображений или выполнение сложных вычислений, должны выполняться вне основного потока. Нарушение данного требования может привести к блокировке интерфейса и возникновению ошибки Application Not Responding (ANR).

Одним из исторически первых механизмов выполнения фоновых задач в Android являлся класс AsyncTask. Данный механизм позволял выполнять операции в фоновом потоке и передавать результаты выполнения в пользовательский интерфейс. AsyncTask широко использовался в ранних

версиях Android, однако со временем был признан устаревшим из-за ограничений в управлении потоками, сложностей обработки ошибок и риска возникновения утечек памяти. Несмотря на это, понимание принципов работы AsyncTask остаётся важным для анализа устаревших Android-приложений и поддержки legacy-кода.

Современным подходом к организации асинхронного выполнения задач в Android являются Kotlin Coroutines. Coroutines позволяют выполнять длительные операции без блокировки интерфейса, упрощают управление потоками и делают асинхронный код более понятным и структурированным. Использование корутин позволяет разработчику организовывать последовательное выполнение операций без создания большого количества вложенных обработчиков и обратных вызовов. Кроме того, Coroutines обеспечивают более безопасное управление жизненным циклом компонентов Android-приложения.

Ещё одним распространённым инструментом асинхронной обработки данных является библиотека RxJava. Данная технология основана на реактивном программировании и позволяет организовывать обработку потоков данных, событий и сетевых запросов. RxJava предоставляет разработчику механизмы подписки на изменения данных, комбинирования потоков и обработки асинхронных событий. Однако использование RxJava требует более глубокого понимания реактивного программирования и может усложнять структуру приложения при неправильном проектировании.

При выполнении сетевых запросов особое значение имеет обработка ошибок. Мобильные устройства могут работать в условиях нестабильного интернет-соединения, ограничения скорости передачи данных или полного отсутствия доступа к сети. Поэтому приложение должно корректно реагировать на ошибки соединения, тайм-ауты и некорректные ответы сервера. Важным элементом пользовательского интерфейса является отображение состояния загрузки данных, уведомление пользователя об ошибках и обеспечение повторного выполнения запроса.

Для специалистов в области информационной безопасности асинхронная обработка данных имеет практическое значение при анализе поведения

мобильных приложений. Некорректная реализация фоновых потоков может приводить к утечкам данных, неконтролируемой передаче информации, зависанию приложения или возможности эксплуатации уязвимостей. Кроме того, злоумышленники могут использовать фоновые процессы для скрытого выполнения вредоносных действий без ведома пользователя.

Практическая часть

Постановка задачи

В рамках практической работы студенту необходимо разработать Android-приложение, выполняющее загрузку данных из сети без блокировки пользовательского интерфейса. Работа направлена на изучение механизмов асинхронного выполнения операций, исследование принципов работы фоновых потоков и получение практических навыков разработки отзывчивых мобильных приложений.

На первом этапе выполнения задания студенту необходимо подготовить среду разработки Android Studio и создать новый проект мобильного приложения. После создания проекта требуется разработать пользовательский интерфейс приложения, содержащий элементы для запуска сетевого запроса, отображения состояния загрузки и вывода полученных данных.

Далее необходимо исследовать поведение приложения при выполнении сетевых операций в основном потоке. Студент должен проанализировать, каким образом длительное ожидание ответа от сервера влияет на отзывчивость интерфейса, и определить причины возникновения блокировки пользовательского интерфейса. В ходе анализа необходимо обратить внимание на задержки обработки пользовательских действий, зависание элементов интерфейса и возможное появление ошибок выполнения приложения.

Следующим этапом работы является реализация асинхронной загрузки данных. В зависимости от выбранного варианта задания студент должен использовать Kotlin Coroutines, AsyncTask либо RxJava для организации выполнения сетевых операций в фоновом режиме. Необходимо обеспечить корректную обработку запросов без блокировки основного потока приложения.

После реализации механизма асинхронной загрузки требуется организовать

отображение состояния выполнения операции. Студент должен реализовать индикацию загрузки данных, уведомление пользователя об успешном завершении операции и вывод сообщений об ошибках сетевого взаимодействия. Особое внимание следует уделить обработке ситуаций отсутствия интернет-соединения, длительного ожидания ответа сервера и некорректного формата полученных данных.

В ходе выполнения практической работы студент должен исследовать взаимодействие асинхронных операций с жизненным циклом Android-приложения. Необходимо определить, как приложение ведёт себя при сворачивании, повороте экрана или закрытии Activity во время выполнения сетевого запроса. Требуется проанализировать риски возникновения утечек памяти и ошибок при неправильном управлении фоновыми задачами.

Следующим этапом является анализ производительности приложения и эффективности выбранного механизма асинхронной обработки данных. Студент должен определить, насколько быстро приложение реагирует на действия пользователя при выполнении сетевых операций, а также сравнить удобство реализации различных подходов к организации фоновых задач.

Особое внимание необходимо уделить вопросам безопасности сетевого взаимодействия. В рамках практической работы требуется исследовать риски передачи данных по сети, необходимость обработки исключительных ситуаций и влияние фоновых операций на стабильность работы мобильного приложения.

После завершения разработки студент должен провести тестирование приложения в различных условиях работы сети: стабильное подключение, медленное соединение, временное отсутствие доступа к интернету и повторное подключение к сети. Полученные результаты необходимо проанализировать и определить устойчивость работы приложения при изменении сетевых условий.

По результатам выполнения практической работы студент должен подготовить отчёт, содержащий:

- описание разработанного приложения;
- описание выбранного механизма асинхронной обработки данных;
- результаты анализа работы пользовательского интерфейса;
- описание реализации сетевой загрузки;

- анализ обработки ошибок сетевого взаимодействия;
- результаты тестирования приложения при различных условиях сети;
- выводы о преимуществах асинхронного выполнения операций в Android-приложениях.

Варианты заданий

№ варианта	**Содержание задания**
1	Разработать приложение для загрузки текстовых данных из сети с использованием Kotlin Coroutines и отображением состояния загрузки.
2	Реализовать приложение для получения списка новостей с удалённого сервера без блокировки пользовательского интерфейса.
3	Создать приложение для загрузки изображений из сети с использованием RxJava и анализом производительности интерфейса.
4	Разработать приложение для получения данных о погоде через сетевой API с асинхронной обработкой запросов.
5	Реализовать приложение для отображения списка пользователей, получаемого с удалённого сервера в фоновом режиме.
6	Создать приложение, выполняющее асинхронную загрузку JSON-данных и отображение информации в интерфейсе.
7	Разработать приложение для загрузки данных из REST API с использованием Kotlin Coroutines и обработкой ошибок сети.
8	Реализовать приложение, выполняющее фоновую загрузку изображений и отображение прогресса выполнения операции.

9	Создать приложение для асинхронного получения данных о курсах валют с удалённого сервера.
10	Разработать приложение, выполняющее сетевые запросы с использованием AsyncTask и анализом ограничений legacy-подхода.
11	Реализовать приложение для загрузки списка сообщений с сервера при сохранении отзывчивости интерфейса.
12	Создать приложение, выполняющее повторную отправку сетевого запроса при возникновении ошибки соединения.
13	Разработать приложение для асинхронной синхронизации данных между локальным хранилищем и сервером.
14	Реализовать приложение с возможностью отмены сетевого запроса во время выполнения операции.
15	Создать приложение для тестирования поведения интерфейса при медленном сетевом соединении.
16	Разработать приложение, отображающее индикатор выполнения во время загрузки данных из сети.
17	Реализовать приложение для параллельной загрузки нескольких наборов данных с использованием RxJava.
18	Создать приложение для анализа влияния сетевых операций на производительность Android-приложения.
19	Разработать приложение с автоматическим обновлением данных через определённые интервалы времени без блокировки интерфейса.
20	Реализовать приложение, сравнивающее

	производительность Kotlin Coroutines, AsyncTask и RxJava при выполнении сетевых запросов.

Контрольные вопросы

1. Что представляет собой основной поток Android-приложения?
2. Почему сетевые операции нельзя выполнять в UI-потоке?
3. Что такое ошибка Application Not Responding (ANR)?
4. Для чего используются фоновые потоки выполнения?
5. Что представляет собой AsyncTask?
6. Почему AsyncTask считается устаревшим механизмом?
7. Какие преимущества имеют Kotlin Coroutines?
8. Что представляет собой RxJava?
9. Что такое реактивное программирование?
10. Какие проблемы могут возникнуть при неправильной работе фоновых потоков?
11. Почему важно учитывать жизненный цикл Activity при выполнении асинхронных операций?
12. Что такое утечка памяти в Android-приложении?
13. Какие ошибки могут возникать при сетевом взаимодействии?
14. Почему необходимо обрабатывать исключительные ситуации при загрузке данных?
15. Как асинхронные операции влияют на отзывчивость интерфейса?
16. Какие элементы интерфейса используются для отображения состояния загрузки?
17. Почему важно тестировать приложение при нестабильном интернет-соединении?
18. Какие риски информационной безопасности связаны с сетевыми запросами?
19. В чём различие между последовательным и параллельным выполнением операций?
20. Какие современные технологии используются для асинхронной обработки данных в Android?

Список литературы

13. Билл Филлипс, Крис Стюарт, Кристин Марсикано, Брайан Гарднер Android-программирование для профессионалов. — СПб.: Питер, 2023. — 912 с.

14. Google Developers. Android NDK Guides [Электронный ресурс]. — Режим доступа: Android NDK Guides

15. Google Developers. JNI Tips [Электронный ресурс]. — Режим доступа: JNI Tips

16. Liu P., Li L., Yan Y. Identifying and Characterizing Silently-Evolved Methods in the Android API [Электронный ресурс]. — Режим доступа: arXiv.org

17. Google Developers. ConstraintLayout [Электронный ресурс]. — Режим доступа: ConstraintLayout Documentation

18. Google Developers. Test your app on Android [Электронный ресурс]. — Режим доступа: Test your app on Android

Практическая работа №4. Создание простого видеоплеера с использованием ExoPlayer

Цель работы

Цель работы: изучить принципы воспроизведения мультимедийного контента в Android-приложениях, освоить использование библиотеки ExoPlayer для создания видеоплеера, исследовать особенности потокового воспроизведения видео, а также получить практические навыки разработки мультимедийных мобильных приложений.

Теоретическая часть

Современные мобильные приложения активно используют мультимедийные технологии для воспроизведения аудио- и видеоконтента. Видео применяется в образовательных системах, социальных сетях, онлайн-кинотеатрах, системах видеонаблюдения, корпоративных сервисах и других мобильных решениях. Для корректного воспроизведения мультимедиа Android-приложения должны эффективно взаимодействовать с системными ресурсами устройства, обеспечивать стабильную работу интерфейса и учитывать особенности обработки потоковых данных.

В операционной системе Android существует несколько способов воспроизведения видео. Ранние версии платформы использовали стандартный компонент VideoView, однако его функциональность ограничена базовыми возможностями воспроизведения мультимедиа. Для создания более гибких и производительных мультимедийных приложений Google разработала библиотеку ExoPlayer, предоставляющую расширенные возможности работы с аудио- и видеоданными.

ExoPlayer представляет собой современную библиотеку для воспроизведения мультимедийного контента в Android-приложениях. Основным преимуществом ExoPlayer является поддержка потокового воспроизведения видео, адаптивной загрузки контента, различных форматов мультимедиа и гибкого управления воспроизведением. Библиотека позволяет организовывать воспроизведение локальных файлов, сетевых потоков, видеоданных из облачных сервисов и онлайн-платформ.

Одной из важных особенностей работы видеоплееров является потоковая передача данных. При потоковом воспроизведении видеофайл не загружается полностью на устройство пользователя, а передаётся частями по мере просмотра. Такой подход позволяет экономить память устройства и ускоряет начало воспроизведения. Однако потоковая передача требует стабильного сетевого соединения и корректной обработки буферизации данных.

Важным элементом мультимедийного приложения является управление жизненным циклом видеоплеера. В процессе работы Android-приложения пользователь может сворачивать программу, изменять ориентацию экрана, блокировать устройство или переключаться между приложениями. Видеоплеер должен корректно реагировать на данные события: приостанавливать воспроизведение, освобождать системные ресурсы и сохранять текущее состояние воспроизведения.

Особое значение имеет обработка ошибок воспроизведения. При работе с сетевыми видеопотоками могут возникать проблемы соединения, ошибки загрузки файлов, недостаточная скорость передачи данных или несовместимость мультимедийного формата. Поэтому приложение должно информировать пользователя о возникших ошибках и обеспечивать корректное восстановление воспроизведения.

Использование мультимедиа связано с повышенной нагрузкой на процессор, память и сетевые ресурсы устройства. Неправильная реализация видеоплеера может приводить к перегреву устройства, быстрому разряду аккумулятора и нестабильной работе приложения. По этой причине разработчик должен учитывать вопросы производительности и оптимизации использования системных ресурсов.

Для специалистов в области информационной безопасности мультимедийные приложения представляют интерес с точки зрения передачи данных через сеть и обработки внешнего контента. Видеопотоки могут использоваться для скрытой передачи информации, внедрения вредоносного содержимого или эксплуатации уязвимостей мультимедийных библиотек. Поэтому при разработке приложений необходимо учитывать безопасность сетевого взаимодействия и проверку источников мультимедийных данных.

Практическая часть

Постановка задачи

В рамках практической работы студенту необходимо разработать Android-приложение с функцией воспроизведения видео с использованием библиотеки ExoPlayer. Работа направлена на изучение механизмов воспроизведения мультимедийного контента, исследование особенностей потоковой передачи данных и получение практических навыков разработки мобильных мультимедийных приложений.

На первом этапе выполнения задания студенту необходимо подготовить среду разработки Android Studio и создать новый проект мобильного приложения. После создания проекта требуется изучить структуру приложения и подготовить пользовательский интерфейс, содержащий область отображения видео, элементы управления воспроизведением и информационные компоненты интерфейса.

Следующим этапом является подключение библиотеки ExoPlayer и исследование её возможностей. Студент должен изучить основные компоненты библиотеки, определить принципы организации воспроизведения мультимедиа и подготовить приложение к работе с видеоконтентом. Особое внимание необходимо уделить взаимодействию пользовательского интерфейса с мультимедийным проигрывателем.

После настройки среды воспроизведения необходимо реализовать загрузку и отображение видеоконтента. В рамках задания студент должен организовать воспроизведение видеофайла из локального хранилища устройства либо из сетевого источника. Требуется обеспечить корректное отображение видеопотока и управление воспроизведением мультимедиа.

Далее необходимо реализовать базовые функции управления видеоплеером. Студент должен обеспечить возможность запуска, приостановки и остановки воспроизведения видео, а также исследовать работу элементов управления мультимедийным контентом. В процессе тестирования необходимо определить, насколько корректно приложение реагирует на действия пользователя.

Особое внимание следует уделить обработке жизненного цикла Android-

приложения. Студенту необходимо исследовать поведение видеоплеера при сворачивании приложения, изменении ориентации экрана и повторном открытии программы. Требуется определить, сохраняется ли текущее состояние воспроизведения и корректно ли освобождаются системные ресурсы при завершении работы приложения.

Следующим этапом работы является анализ потокового воспроизведения видео. Студент должен исследовать работу приложения при различных скоростях интернет-соединения, определить влияние буферизации на качество воспроизведения и проанализировать устойчивость работы видеоплеера при нестабильной сети.

В процессе выполнения задания необходимо исследовать использование системных ресурсов мобильного устройства. Требуется определить влияние воспроизведения видео на загрузку процессора, использование оперативной памяти и расход заряда аккумулятора. Особое внимание следует уделить корректному освобождению ресурсов видеоплеера после завершения работы приложения.

Дополнительно студент должен исследовать обработку ошибок воспроизведения. В рамках задания необходимо определить поведение приложения при отсутствии доступа к сети, повреждении видеофайла или попытке воспроизведения неподдерживаемого мультимедийного формата. Требуется обеспечить корректное информирование пользователя о возникших проблемах.

После завершения разработки необходимо провести комплексное тестирование приложения на различных сценариях использования: воспроизведение локального видео, потоковое воспроизведение из сети, изменение ориентации устройства, сворачивание приложения, повторное открытие программы и длительное воспроизведение мультимедиа.

По результатам выполнения практической работы студент должен подготовить отчёт, содержащий:

- описание разработанного видеоплеера;
- описание возможностей библиотеки ExoPlayer;
- результаты тестирования воспроизведения мультимедиа;

- анализ работы приложения при различных сетевых условиях;
- результаты исследования использования системных ресурсов;
- описание выявленных проблем и способов их устранения;
- выводы об особенностях разработки мультимедийных Android-приложений.

Варианты заданий

№ варианта	**Содержание задания**
1	Разработать видеоплеер для воспроизведения локальных видеофайлов с базовыми элементами управления воспроизведением.
2	Реализовать приложение для потокового воспроизведения видео из сети с использованием ExoPlayer.
3	Создать видеоплеер с возможностью автоматического продолжения воспроизведения после сворачивания приложения.
4	Разработать приложение для воспроизведения образовательных видеоматериалов с отображением информации о видео.
5	Реализовать видеоплеер с поддержкой изменения ориентации экрана без потери текущей позиции воспроизведения.
6	Создать приложение для анализа влияния качества видео на производительность мобильного устройства.
7	Разработать видеоплеер с отображением индикатора буферизации при потоковом воспроизведении.
8	Реализовать приложение для воспроизведения видео с автоматическим переключением качества

	потока.
9	Создать видеоплеер с функцией повторного воспроизведения видео после завершения просмотра.
10	Разработать приложение с отображением списка доступных видеозаписей и возможностью выбора контента.
11	Реализовать видеоплеер с анализом поведения приложения при нестабильном интернет-соединении.
12	Создать приложение для воспроизведения видеоконтента в фоновом режиме.
13	Разработать видеоплеер с функцией отображения текущего времени воспроизведения и длительности видео.
14	Реализовать приложение с поддержкой потокового воспроизведения видеоданных из удалённого сервера.
15	Создать видеоплеер, автоматически приостанавливающий воспроизведение при сворачивании приложения.
16	Разработать приложение для тестирования работы ExoPlayer с различными форматами видеофайлов.
17	Реализовать видеоплеер с возможностью переключения между несколькими видеопотоками.
18	Создать приложение для анализа энергопотребления при длительном воспроизведении видео.
19	Разработать видеоплеер с отображением

	сообщений об ошибках загрузки мультимедиа.
20	Реализовать приложение, сравнивающее воспроизведение локального и потокового видеоконтента.

Контрольные вопросы

1. Что представляет собой библиотека ExoPlayer?
2. Какие преимущества ExoPlayer имеет по сравнению с VideoView?
3. Что такое потоковое воспроизведение мультимедиа?
4. Почему потоковое воспроизведение требует стабильного интернет-соединения?
5. Какие функции выполняет видеоплеер в Android-приложении?
6. Что такое буферизация видеоданных?
7. Почему необходимо управлять жизненным циклом видеоплеера?
8. Какие проблемы могут возникнуть при неправильном освобождении ресурсов мультимедийного проигрывателя?
9. Как изменение ориентации экрана влияет на воспроизведение видео?
10. Какие системные ресурсы активно используются при воспроизведении мультимедиа?
11. Почему воспроизведение видео влияет на расход аккумулятора устройства?
12. Какие ошибки могут возникать при потоковом воспроизведении видео?
13. Что происходит при потере сетевого соединения во время воспроизведения?
14. Почему важно обрабатывать ошибки мультимедийного воспроизведения?
15. Какие виды мультимедийного контента поддерживает ExoPlayer?
16. Почему необходимо тестировать приложение на различных сетевых условиях?
17. Какие риски информационной безопасности связаны с обработкой мультимедийных данных?
18. Что такое адаптивное качество видеопотока?
19. Почему необходимо анализировать производительность

мультимедийного приложения?

20. Какие особенности необходимо учитывать при разработке Android-приложений для работы с видео?

Список литературы

19. Билл Филлипс, Крис Стюарт, Кристин Марсикано, Брайан Гарднер Android-программирование для профессионалов. — СПб.: Питер, 2023. — 912 с.

20. Google Developers. Android NDK Guides [Электронный ресурс]. — Режим доступа: Android NDK Guides

21. Google Developers. JNI Tips [Электронный ресурс]. — Режим доступа: JNI Tips

22. Liu P., Li L., Yan Y. Identifying and Characterizing Silently-Evolved Methods in the Android API [Электронный ресурс]. — Режим доступа: arXiv.org

23. Google Developers. ConstraintLayout [Электронный ресурс]. — Режим доступа: ConstraintLayout Documentation

24. Google Developers. Test your app on Android [Электронный ресурс]. — Режим доступа: Test your app on Android

Практическая работа №5. Написание Unit-тестов для бизнес-логики Android-приложения с использованием JUnit

Цель работы

Цель работы: изучить основы модульного тестирования Android-приложений, освоить использование библиотеки JUnit для написания Unit-тестов, исследовать методы проверки бизнес-логики мобильных приложений, а также получить практические навыки автоматизированного тестирования программного обеспечения.

Теоретическая часть

Разработка современного программного обеспечения невозможна без обеспечения качества и надёжности программного кода. Одним из основных методов контроля корректности работы приложения является тестирование. Тестирование позволяет выявлять ошибки, проверять правильность выполнения алгоритмов, анализировать устойчивость приложения и обеспечивать стабильность программного продукта при изменении исходного кода.

В процессе разработки мобильных приложений особое значение имеет автоматизированное тестирование. В отличие от ручного тестирования, автоматические тесты позволяют многократно проверять корректность работы отдельных компонентов приложения без участия пользователя. Это особенно важно при сопровождении крупных проектов, содержащих большое количество взаимосвязанных модулей.

Одним из наиболее распространённых видов автоматизированного тестирования являются модульные тесты (Unit-тесты). Unit-тестирование представляет собой проверку отдельных частей программного кода — функций, методов или классов — независимо от остальных компонентов приложения. Основная задача модульного тестирования заключается в проверке бизнес-логики программы и подтверждении корректности выполнения отдельных операций.

Бизнес-логика приложения включает алгоритмы обработки данных, вычисления, проверки условий, обработку пользовательской информации и принятие решений внутри программы. Ошибки в бизнес-логике могут

приводить к некорректным расчётам, потере данных, нарушению безопасности и неправильной работе приложения. Поэтому проверка логики обработки информации является важным этапом разработки программного обеспечения.
Для написания модульных тестов в Java и Kotlin широко используется библиотека JUnit. Данный инструмент позволяет создавать тестовые сценарии, автоматически выполнять проверки и анализировать результаты тестирования. JUnit обеспечивает механизм сравнения ожидаемого и фактического результата работы программы, а также предоставляет средства организации тестовых наборов и автоматического запуска тестов.
В процессе модульного тестирования приложение рассматривается как совокупность независимых компонентов. Каждый тест должен проверять только один конкретный аспект работы программы. Такой подход упрощает поиск ошибок и повышает качество программного кода. Важным принципом Unit-тестирования является изолированность тестов — результат выполнения одного теста не должен зависеть от результатов других проверок.
Особое внимание при разработке тестов уделяется проверке граничных ситуаций и обработки ошибок. Приложение должно корректно реагировать на некорректные входные данные, пустые значения, переполнение диапазонов и другие исключительные ситуации. Поэтому качественное тестирование включает не только проверку корректных сценариев работы, но и анализ поведения программы при возникновении ошибок.
Для специалистов в области информационной безопасности автоматизированное тестирование имеет особое значение. Наличие ошибок в бизнес-логике может приводить к возникновению уязвимостей, обходу механизмов авторизации, нарушению контроля доступа или утечке конфиденциальных данных. Регулярное тестирование позволяет своевременно выявлять потенциальные проблемы безопасности и снижать риск эксплуатации программных ошибок злоумышленниками.

Практическая часть

Постановка задачи

В рамках практической работы студенту необходимо разработать набор Unit-

тестов для проверки бизнес-логики Android-приложения с использованием библиотеки JUnit. Работа направлена на изучение принципов модульного тестирования, исследование методов автоматической проверки программного кода и получение практических навыков обеспечения качества мобильных приложений.

На первом этапе выполнения задания студенту необходимо подготовить среду разработки Android Studio и создать новый проект мобильного приложения либо использовать существующий проект с реализованной бизнес-логикой. После подготовки проекта требуется изучить структуру приложения и определить компоненты, подлежащие тестированию.

Далее необходимо исследовать бизнес-логику приложения и определить функциональные элементы, требующие проверки. В процессе анализа студент должен выделить методы обработки данных, вычислительные операции, механизмы проверки условий, алгоритмы работы с пользовательским вводом и иные элементы, влияющие на корректность работы приложения.

Следующим этапом является подготовка среды модульного тестирования. Студент должен изучить структуру тестовых каталогов Android-проекта, назначение Unit-тестов и принципы их автоматического запуска. Особое внимание необходимо уделить разделению тестируемого кода и тестовых сценариев.

После подготовки среды требуется разработать набор тестов для проверки корректной работы бизнес-логики приложения. В процессе выполнения задания необходимо реализовать тестовые сценарии для проверки правильности вычислений, обработки входных данных, проверки условий и корректного выполнения отдельных операций приложения.

Особое внимание следует уделить проверке граничных случаев и обработки ошибок. Студент должен исследовать поведение приложения при вводе некорректных данных, использовании пустых значений, превышении допустимых диапазонов и других нестандартных ситуациях. Требуется определить, насколько корректно приложение реагирует на ошибочные сценарии использования.

В ходе выполнения практической работы необходимо исследовать

изолированность тестов и независимость отдельных проверок друг от друга. Студент должен определить, влияет ли выполнение одного теста на результаты остальных проверок, а также проанализировать устойчивость тестового набора при повторном запуске.

Следующим этапом является анализ результатов тестирования. Требуется определить, какие тесты выполняются успешно, какие проверки завершаются ошибками и какие проблемы выявлены в бизнес-логике приложения. При обнаружении ошибок студент должен проанализировать причины возникновения проблем и предложить способы их устранения.

Дополнительно необходимо исследовать влияние изменений программного кода на результаты тестирования. В рамках задания студент должен внести изменения в бизнес-логику приложения и повторно выполнить тестирование для анализа корректности работы тестового набора. Требуется определить, насколько эффективно тесты позволяют обнаруживать изменения поведения программы.

Особое внимание следует уделить вопросам качества программного обеспечения и безопасности мобильных приложений. Студент должен проанализировать, каким образом ошибки бизнес-логики могут влиять на безопасность приложения, обработку пользовательских данных и стабильность работы программного обеспечения.

После завершения разработки тестов необходимо провести комплексное тестирование приложения и подготовить отчёт, содержащий:

- описание исследуемой бизнес-логики;
- описание структуры Unit-тестов;
- результаты выполнения тестовых сценариев;
- анализ проверки граничных ситуаций;
- описание выявленных ошибок и способов их устранения;
- результаты повторного тестирования после изменения кода;
- выводы о роли автоматизированного тестирования в разработке Android-приложений.

Варианты заданий

№ варианта	Содержание задания
1	Разработать набор Unit-тестов для проверки корректности вычислений калькулятора мобильного приложения.
2	Реализовать тестирование механизма авторизации пользователя с проверкой корректности обработки логина и пароля.
3	Создать Unit-тесты для проверки обработки пользовательского ввода в форме регистрации.
4	Разработать тесты для проверки алгоритма вычисления скидок в мобильном интернет-магазине.
5	Реализовать тестирование механизма проверки сложности пользовательского пароля.
6	Создать Unit-тесты для проверки обработки пустых и некорректных входных данных.
7	Разработать тестовый набор для проверки сортировки и фильтрации данных в мобильном приложении.
8	Реализовать тестирование алгоритма расчёта статистических показателей в приложении аналитики.
9	Создать Unit-тесты для проверки работы механизма поиска информации в приложении.
10	Разработать тесты для проверки корректности обработки исключительных ситуаций в бизнес-логике.
11	Реализовать тестирование механизма ограничения доступа пользователей к функционалу приложения.
12	Создать Unit-тесты для проверки корректности обработки финансовых операций в мобильном приложении.

13	Разработать тестовый набор для проверки логики работы системы уведомлений.
14	Реализовать тестирование алгоритма обработки текстовых данных и поиска совпадений.
15	Создать Unit-тесты для проверки работы приложения при вводе данных вне допустимого диапазона.
16	Разработать тесты для проверки корректности сохранения пользовательских настроек.
17	Реализовать тестирование бизнес-логики мобильного приложения для управления задачами.
18	Создать Unit-тесты для проверки алгоритма обработки сетевых данных перед отображением в интерфейсе.
19	Разработать тестовый набор для анализа корректности обработки временных и календарных данных.
20	Реализовать сравнительный анализ качества приложения до и после внедрения Unit-тестирования.

Контрольные вопросы

1. Что представляет собой модульное тестирование?
2. Для чего используются Unit-тесты?
3. Что такое бизнес-логика приложения?
4. Какие задачи решает библиотека JUnit?
5. Почему важно автоматизированное тестирование программного обеспечения?
6. Чем Unit-тестирование отличается от ручного тестирования?
7. Что такое изолированность тестов?
8. Почему тесты не должны зависеть друг от друга?
9. Какие ошибки могут возникать в бизнес-логике приложения?
10. Что такое граничные условия тестирования?

11. Почему необходимо проверять обработку некорректных данных?
12. Какие преимущества предоставляет автоматический запуск тестов?
13. Как изменения программного кода влияют на результаты тестирования?
14. Что такое регрессионное тестирование?
15. Почему важно анализировать результаты тестов после изменения логики приложения?
16. Какие риски информационной безопасности могут возникать из-за ошибок бизнес-логики?
17. Почему Unit-тесты повышают качество программного обеспечения?
18. Какие компоненты Android-приложения чаще всего подвергаются модульному тестированию?
19. Что может произойти при отсутствии автоматизированного тестирования?
20. Какие этапы включает процесс разработки и выполнения Unit-тестов?

Список литературы

25. Билл Филлипс, Крис Стюарт, Кристин Марсикано, Брайан Гарднер Android-программирование для профессионалов. — СПб.: Питер, 2023. — 912 с.
26. Google Developers. Android NDK Guides [Электронный ресурс]. — Режим доступа: Android NDK Guides
27. Google Developers. JNI Tips [Электронный ресурс]. — Режим доступа: JNI Tips
28. Liu P., Li L., Yan Y. Identifying and Characterizing Silently-Evolved Methods in the Android API [Электронный ресурс]. — Режим доступа: arXiv.org
29. Google Developers. ConstraintLayout [Электронный ресурс]. — Режим доступа: ConstraintLayout Documentation
30. Google Developers. Test your app on Android [Электронный ресурс]. — Режим доступа: Test your app on Android

Практическая работа №6. Заполнение карточки мобильного приложения в Google Play Console

Цель работы

Цель работы: изучить принципы публикации мобильных приложений в Google Play Console, освоить процесс заполнения карточки приложения, исследовать требования Google Play к оформлению приложений, а также получить практические навыки подготовки Android-приложения к публикации и распространению.

Теоретическая часть

Публикация мобильного приложения является завершающим этапом разработки программного обеспечения для платформы Android. После создания и тестирования приложения разработчику необходимо подготовить программный продукт к размещению в магазине приложений Google Play. Для управления публикацией Android-приложений используется сервис Google Play Console, предоставляющий инструменты загрузки приложений, настройки распространения, анализа статистики и взаимодействия с пользователями.
Карточка приложения представляет собой набор информационных материалов, отображаемых пользователю в магазине Google Play. Именно карточка приложения формирует первое впечатление пользователя о программном продукте и влияет на решение об установке приложения. Некачественно оформленная карточка может значительно снизить количество установок даже при высоком качестве самого приложения.
Одним из основных элементов карточки приложения является название. Название должно быть информативным, отражать назначение приложения и соответствовать требованиям Google Play. Запрещается использовать вводящие в заблуждение названия, чрезмерное количество ключевых слов, специальные символы и элементы, нарушающие правила платформы.
Важную роль играет текстовое описание приложения. Краткое описание отображается пользователю в результатах поиска и должно лаконично передавать основное назначение программы. Полное описание предназначено для детального ознакомления пользователя с функциональностью приложения,

его возможностями и особенностями. При составлении описания необходимо учитывать требования читаемости текста, отсутствие ложной информации и соответствие фактическим возможностям программного продукта.
Неотъемлемой частью карточки приложения являются графические материалы. Google Play требует наличие иконки приложения, скриншотов интерфейса и, при необходимости, рекламных изображений. Графические материалы должны соответствовать установленным требованиям по размеру, формату и качеству. Скриншоты должны демонстрировать реальный интерфейс приложения и отражать основные возможности программного продукта.
Дополнительно Google Play Console требует указания категории приложения, возрастных ограничений, контактной информации разработчика и политики конфиденциальности. Политика конфиденциальности является обязательным элементом для приложений, работающих с пользовательскими данными, сетевыми сервисами, геолокацией или разрешениями устройства. Отсутствие необходимых сведений может привести к отклонению приложения при проверке.
Особое внимание Google уделяет вопросам безопасности и защиты пользовательских данных. Разработчик обязан указывать, какие данные собирает приложение, каким образом они используются и передаются ли сторонним сервисам. В современных версиях Google Play Console реализован раздел Data Safety, предназначенный для информирования пользователей о механизмах обработки данных внутри приложения.
Для специалистов в области информационной безопасности публикация мобильного приложения связана с необходимостью соблюдения требований безопасности программного обеспечения. Ошибки при заполнении карточки приложения, отсутствие политики конфиденциальности или некорректное описание работы с пользовательскими данными могут привести к нарушению требований платформы и блокировке приложения. Поэтому разработчик должен внимательно относиться к вопросам документирования функциональности и обработки информации.

Практическая часть

Постановка задачи

В рамках практической работы студенту необходимо подготовить карточку мобильного приложения для публикации в Google Play Console. Работа направлена на изучение требований магазина Google Play, исследование процесса оформления мобильного приложения и получение практических навыков подготовки программного продукта к распространению.

На первом этапе выполнения задания студенту необходимо изучить структуру Google Play Console и ознакомиться с основными разделами, связанными с публикацией Android-приложений. Требуется определить, какие сведения необходимы для создания карточки приложения и какие элементы являются обязательными при публикации программного продукта.

Далее необходимо подготовить информацию о разрабатываемом приложении. В процессе выполнения задания студент должен определить назначение приложения, его основные функции, категорию распространения и предполагаемую целевую аудиторию. Особое внимание следует уделить формированию информативного и корректного описания программного продукта.

Следующим этапом работы является разработка текстового содержимого карточки приложения. Студенту необходимо подготовить название приложения, краткое описание и полное описание функциональности программы. Требуется обеспечить соответствие текста требованиям Google Play, отсутствие вводящей в заблуждение информации и корректное описание возможностей приложения.

После подготовки текстового описания необходимо разработать графические материалы для карточки приложения. В рамках задания студент должен подготовить иконку приложения, скриншоты пользовательского интерфейса и дополнительные изображения для публикации. Требуется обеспечить соответствие графических материалов требованиям качества, читаемости и визуального оформления.

Особое внимание необходимо уделить разделу конфиденциальности и обработки пользовательских данных. Студент должен определить, какие данные использует приложение, требуется ли предоставление политики

конфиденциальности и каким образом необходимо описывать работу приложения с пользовательской информацией. В процессе анализа необходимо учитывать требования Google Play к защите персональных данных пользователей.

Следующим этапом является заполнение параметров публикации приложения. Студенту необходимо определить возрастные ограничения, категорию приложения, регион распространения и параметры доступа пользователей к приложению. Требуется исследовать влияние данных параметров на отображение приложения в магазине Google Play.

В процессе выполнения практической работы необходимо исследовать требования Google Play к безопасности мобильных приложений. Студент должен проанализировать возможные причины отклонения приложения при публикации, определить запрещённые элементы оформления и исследовать правила модерации программного обеспечения.

Дополнительно необходимо изучить особенности публикации обновлений мобильного приложения. В рамках задания студент должен определить, какие элементы карточки приложения могут изменяться после публикации и каким образом осуществляется обновление информации о программном продукте.

После завершения заполнения карточки приложения студент должен провести анализ качества подготовленных материалов и оценить соответствие карточки приложения требованиям магазина Google Play. Требуется определить, насколько информативно представлено приложение и какие элементы оформления могут повлиять на заинтересованность пользователей.

По результатам выполнения практической работы студент должен подготовить отчёт, содержащий:

- описание разработанного мобильного приложения;
- подготовленные текстовые материалы карточки приложения;
- описание графических материалов;
- результаты анализа требований Google Play;
- описание параметров публикации приложения;
- анализ раздела конфиденциальности и безопасности данных;
- выводы об особенностях подготовки Android-приложений к публикации.

Варианты заданий

№ варианта	Содержание задания
1	Подготовить карточку приложения для мобильного калькулятора с описанием функциональности и графическими материалами.
2	Разработать материалы публикации для приложения учёта личных расходов с указанием обработки пользовательских данных.
3	Создать карточку приложения для мобильного менеджера задач с оформлением скриншотов интерфейса.
4	Подготовить публикационные материалы для фитнес-приложения с описанием работы с датчиками устройства.
5	Разработать карточку образовательного приложения с указанием возрастных ограничений и категории распространения.
6	Создать материалы публикации для приложения потокового воспроизведения мультимедиа.
7	Подготовить карточку приложения для мобильного мессенджера с анализом требований конфиденциальности.
8	Разработать описание и графические материалы для приложения прогноза погоды.
9	Создать карточку приложения интернет-магазина с анализом обработки персональных данных пользователей.
10	Подготовить публикационные материалы для приложения навигации и геолокации.
11	Разработать карточку приложения системы

	электронного обучения.
12	Создать материалы публикации для мобильного приложения новостного агрегатора.
13	Подготовить карточку приложения для облачного хранилища файлов с анализом требований безопасности.
14	Разработать публикационные материалы для мобильного банковского приложения.
15	Создать карточку игрового приложения с оформлением графических материалов и описанием игрового процесса.
16	Подготовить материалы публикации для приложения обработки фотографий.
17	Разработать карточку приложения мониторинга сетевой активности мобильного устройства.
18	Создать публикационные материалы для приложения системы видеонаблюдения.
19	Подготовить карточку приложения управления умным домом с анализом требований конфиденциальности.
20	Разработать сравнительный анализ карточек нескольких мобильных приложений из Google Play и определить особенности их оформления.

Контрольные вопросы

1. Что представляет собой Google Play Console?
2. Для чего используется карточка мобильного приложения?
3. Какие элементы включает карточка приложения в Google Play?
4. Почему важно корректно оформлять описание приложения?
5. Какие требования предъявляются к названию приложения?
6. Для чего используются скриншоты интерфейса?
7. Почему графические материалы влияют на популярность приложения?

8. Что такое политика конфиденциальности?
9. В каких случаях приложение обязано иметь политику конфиденциальности?
10. Какие данные необходимо указывать в разделе Data Safety?
11. Почему Google Play уделяет внимание обработке пользовательских данных?
12. Какие причины могут привести к отклонению приложения при публикации?
13. Что такое возрастные ограничения приложения?
14. Почему важно правильно выбирать категорию приложения?
15. Какие сведения о разработчике необходимо указывать в Google Play Console?
16. Что такое модерация мобильного приложения?
17. Какие риски информационной безопасности связаны с публикацией мобильных приложений?
18. Почему необходимо анализировать требования платформы перед публикацией приложения?
19. Какие элементы карточки приложения можно изменять после публикации?
20. Как качество оформления карточки влияет на количество установок приложения?

Список литературы

31. Билл Филлипс, Крис Стюарт, Кристин Марсикано, Брайан Гарднер Android-программирование для профессионалов. — СПб.: Питер, 2023. — 912 с.
32. Google Developers. Android NDK Guides [Электронный ресурс]. — Режим доступа: Android NDK Guides
33. Google Developers. JNI Tips [Электронный ресурс]. — Режим доступа: JNI Tips
34. Liu P., Li L., Yan Y. Identifying and Characterizing Silently-Evolved Methods in the Android API [Электронный ресурс]. — Режим доступа: arXiv.org

35. Google Developers. ConstraintLayout [Электронный ресурс]. — Режим доступа: ConstraintLayout Documentation

36. Google Developers. Test your app on Android [Электронный ресурс]. — Режим доступа: Test your app on Android

Практическая работа №7. Создание гибкого пользовательского интерфейса с помощью ConstraintLayout

Цель работы

Цель работы: изучить принципы построения адаптивных пользовательских интерфейсов в Android-приложениях, освоить использование ConstraintLayout для размещения элементов интерфейса, исследовать методы создания гибкой структуры экранов под различные размеры устройств, а также получить практические навыки разработки современных мобильных интерфейсов.

Теоретическая часть

Пользовательский интерфейс является одним из важнейших компонентов мобильного приложения. Именно интерфейс обеспечивает взаимодействие пользователя с программным обеспечением и влияет на удобство использования приложения. При разработке Android-приложений особое значение имеет адаптация интерфейса под различные размеры экранов, разрешения устройств и ориентации дисплея.

Современные Android-устройства обладают большим разнообразием характеристик. Смартфоны и планшеты отличаются диагональю экрана, плотностью пикселей, соотношением сторон и особенностями отображения элементов интерфейса. В связи с этим разработчик мобильного приложения должен создавать интерфейс, способный корректно отображаться на различных устройствах без потери функциональности и удобства работы.

На ранних этапах развития Android для построения интерфейсов активно использовались LinearLayout, RelativeLayout и другие контейнеры размещения элементов. Однако при создании сложных интерфейсов использование большого количества вложенных контейнеров приводило к усложнению структуры приложения, снижению производительности и затруднению адаптации интерфейса под разные устройства.

Для решения данной проблемы в Android была разработана система ConstraintLayout. ConstraintLayout представляет собой современный контейнер размещения элементов интерфейса, основанный на использовании ограничений и взаимосвязей между компонентами. Вместо создания большого количества вложенных контейнеров разработчик определяет зависимости между

элементами, что позволяет строить сложные и гибкие интерфейсы с минимальной вложенностью структуры.

Основным принципом работы ConstraintLayout является использование ограничений (constraints). Ограничения определяют положение элемента относительно других компонентов интерфейса, границ экрана или специальных направляющих линий. Благодаря этому элементы автоматически адаптируются к изменению размеров экрана и ориентации устройства.

Одним из преимуществ ConstraintLayout является возможность создания адаптивных интерфейсов без дублирования экранов под различные разрешения устройств. Кроме того, уменьшение количества вложенных контейнеров положительно влияет на производительность приложения и скорость отображения интерфейса.

Важным элементом адаптивного дизайна является поддержка различных ориентаций экрана. Android-приложение должно корректно отображать интерфейс как в портретном, так и в альбомном режиме. ConstraintLayout позволяет гибко изменять расположение компонентов интерфейса при изменении конфигурации экрана.

При разработке пользовательских интерфейсов необходимо учитывать не только внешний вид приложения, но и удобство взаимодействия пользователя с элементами управления. Интерфейс должен быть понятным, логичным и обеспечивать быстрый доступ к основным функциям приложения. Неправильное расположение элементов, перегруженность экрана и отсутствие адаптации под различные устройства могут значительно ухудшить пользовательский опыт.

Для специалистов в области информационной безопасности интерфейс мобильного приложения также имеет значение. Некорректно спроектированный интерфейс может способствовать ошибкам пользователя, неправильной обработке данных или скрытию важной информации о действиях приложения. Кроме того, элементы интерфейса должны обеспечивать безопасное взаимодействие пользователя с программным обеспечением и предотвращать случайное выполнение критических операций.

Практическая часть

Постановка задачи

В рамках практической работы студенту необходимо разработать Android-приложение с гибким пользовательским интерфейсом с использованием ConstraintLayout. Работа направлена на изучение принципов построения адаптивных интерфейсов, исследование возможностей ConstraintLayout и получение практических навыков разработки современных мобильных интерфейсов для устройств с различными параметрами экрана.

На первом этапе выполнения задания студенту необходимо подготовить среду разработки Android Studio и создать новый проект мобильного приложения. После создания проекта требуется изучить структуру XML-разметки Android-приложения и определить назначение контейнеров размещения элементов интерфейса.

Следующим этапом является исследование принципов работы ConstraintLayout. Студент должен изучить механизм создания ограничений между элементами интерфейса, определить способы выравнивания компонентов и исследовать методы адаптации интерфейса под различные размеры экранов.

После изучения основных возможностей ConstraintLayout необходимо разработать структуру пользовательского интерфейса приложения. В процессе выполнения задания студент должен определить расположение текстовых элементов, кнопок, изображений, полей ввода и других компонентов интерфейса. Особое внимание следует уделить логике размещения элементов и удобству взаимодействия пользователя с приложением.

Далее необходимо реализовать взаимосвязи между компонентами интерфейса с использованием ограничений ConstraintLayout. Студент должен исследовать, каким образом изменение ограничений влияет на расположение элементов при изменении размеров экрана и ориентации устройства. Требуется обеспечить корректное отображение интерфейса на различных Android-устройствах.

Особое внимание необходимо уделить адаптации интерфейса под портретный и альбомный режимы отображения. В рамках задания студент должен проанализировать поведение элементов интерфейса при изменении ориентации устройства и определить необходимость перестройки структуры экрана для

различных режимов отображения.

Следующим этапом работы является исследование производительности пользовательского интерфейса. Студент должен определить влияние вложенности контейнеров на скорость отображения интерфейса и сравнить структуру ConstraintLayout с альтернативными способами организации экранов Android-приложения.

В процессе выполнения практической работы необходимо исследовать визуальную устойчивость интерфейса при изменении размеров текста, масштабировании элементов и использовании устройств с различной плотностью пикселей. Требуется определить, насколько корректно интерфейс адаптируется к различным условиям отображения.

Дополнительно студент должен проанализировать вопросы удобства использования интерфейса. Необходимо определить, насколько логично расположены элементы управления, достаточно ли удобно взаимодействовать с приложением и соответствует ли интерфейс современным требованиям мобильного дизайна.

После завершения разработки интерфейса студент должен провести тестирование приложения на различных конфигурациях устройств и подготовить отчёт, содержащий:

- описание структуры пользовательского интерфейса;
- анализ использования ConstraintLayout;
- результаты тестирования интерфейса на различных устройствах;
- описание адаптации интерфейса под разные ориентации экрана;
- анализ производительности интерфейса;
- описание выявленных проблем отображения;
- выводы об особенностях разработки адаптивных Android-интерфейсов.

Варианты заданий

№ варианта	**Содержание задания**
1	Разработать интерфейс мобильного калькулятора с адаптацией под различные размеры экранов.
2	Создать экран авторизации с гибким

	расположением текстовых полей и кнопок.
3	Разработать интерфейс новостного приложения с адаптацией под портретный и альбомный режимы.
4	Создать интерфейс приложения интернет-магазина с отображением списка товаров и элементов управления.
5	Разработать экран профиля пользователя с динамическим изменением расположения элементов.
6	Создать интерфейс мобильного мессенджера с адаптивным расположением сообщений и элементов ввода текста.
7	Разработать интерфейс приложения прогноза погоды с использованием изображений и текстовых блоков.
8	Создать адаптивный экран воспроизведения мультимедиа с элементами управления.
9	Разработать интерфейс приложения учёта задач с поддержкой различных размеров экрана.
10	Создать экран регистрации пользователя с корректным отображением элементов при изменении ориентации устройства.
11	Разработать интерфейс приложения мониторинга сетевой активности с отображением статистических данных.
12	Создать адаптивный интерфейс банковского приложения с отображением информации о счетах и операциях.
13	Разработать интерфейс мобильного приложения для системы электронного обучения.
14	Создать экран настроек приложения с гибким

	расположением элементов управления.
15	Разработать интерфейс приложения управления умным домом с адаптацией под планшеты и смартфоны.
16	Создать интерфейс приложения видеонаблюдения с отображением нескольких областей контента.
17	Разработать адаптивный экран отображения статистики и графиков.
18	Создать интерфейс файлового менеджера с поддержкой различных конфигураций экрана.
19	Разработать интерфейс фитнес-приложения с отображением информации о тренировках и активности пользователя.
20	Выполнить сравнительный анализ интерфейсов, созданных с использованием ConstraintLayout и других контейнеров Android.

Контрольные вопросы

1. Что представляет собой ConstraintLayout?
2. Для чего используются ограничения (constraints) в Android-интерфейсах?
3. Какие преимущества имеет ConstraintLayout по сравнению с LinearLayout и RelativeLayout?
4. Почему важно создавать адаптивные пользовательские интерфейсы?
5. Какие проблемы возникают при использовании большого количества вложенных контейнеров?
6. Что такое адаптация интерфейса под различные размеры экрана?
7. Почему необходимо учитывать плотность пикселей устройства?
8. Как изменение ориентации экрана влияет на интерфейс Android-приложения?
9. Какие элементы пользовательского интерфейса используются в Android-приложениях?
10. Почему производительность интерфейса зависит от структуры разметки?

11. Что такое XML-разметка Android-приложения?
12. Какие задачи решает ConstraintLayout при построении сложных интерфейсов?
13. Почему важно тестировать интерфейс на различных устройствах?
14. Какие проблемы могут возникать при масштабировании элементов интерфейса?
15. Что такое адаптивный дизайн мобильного приложения?
16. Почему удобство интерфейса влияет на качество мобильного приложения?
17. Какие риски могут возникать при неправильном проектировании интерфейса?
18. Почему необходимо учитывать доступность интерфейса для пользователей?
19. Какие методы используются для выравнивания элементов в ConstraintLayout?
20. Какие особенности необходимо учитывать при разработке Android-интерфейсов для планшетов и смартфонов?

Список литературы

37. Билл Филлипс, Крис Стюарт, Кристин Марсикано, Брайан Гарднер Android-программирование для профессионалов. — СПб.: Питер, 2023. — 912 с.
38. Google Developers. Android NDK Guides [Электронный ресурс]. — Режим доступа: Android NDK Guides
39. Google Developers. JNI Tips [Электронный ресурс]. — Режим доступа: JNI Tips
40. Liu P., Li L., Yan Y. Identifying and Characterizing Silently-Evolved Methods in the Android API [Электронный ресурс]. — Режим доступа: arXiv.org
41. Google Developers. ConstraintLayout [Электронный ресурс]. — Режим доступа: ConstraintLayout Documentation
42. Google Developers. Test your app on Android [Электронный ресурс]. — Режим доступа: Test your app on Android

Практическая работа №8. Настройка Android NDK в проекте Android Studio
Цель работы

Цель работы: изучить принципы использования Android Native Development Kit (NDK) при разработке мобильных приложений, освоить процесс настройки NDK в Android Studio, исследовать особенности взаимодействия Java/Kotlin-кода с нативными библиотеками на C/C++, а также получить практические навыки интеграции нативного кода в Android-приложение.

Теоретическая часть

Современные Android-приложения преимущественно разрабатываются с использованием языков Java и Kotlin. Однако в ряде случаев разработчику требуется использование нативного кода, выполняемого непосредственно на уровне операционной системы и аппаратной платформы устройства. Для решения данной задачи используется Android Native Development Kit (NDK) — набор инструментов, позволяющий интегрировать код на языках C и C++ в Android-приложения.

Использование NDK позволяет повысить производительность отдельных компонентов приложения, реализовать сложные вычислительные алгоритмы, использовать существующие библиотеки на C/C++, а также обеспечить взаимодействие с низкоуровневыми системными функциями. Нативный код активно применяется в игровых движках, системах обработки мультимедиа, криптографических приложениях, средствах анализа данных и специализированном программном обеспечении.

Взаимодействие между Java/Kotlin и нативным кодом осуществляется через механизм JNI (Java Native Interface). JNI представляет собой интерфейс взаимодействия между виртуальной машиной Java и нативными библиотеками. С помощью JNI Android-приложение может вызывать функции, написанные на C/C++, передавать данные между различными языками программирования и получать результаты выполнения нативных операций.

Одним из преимуществ использования нативного кода является высокая производительность вычислений. Нативные библиотеки выполняются напрямую процессором устройства без промежуточной виртуальной среды, что

позволяет ускорить выполнение ресурсоёмких операций. Однако использование NDK связано с повышенной сложностью разработки, необходимостью ручного управления памятью и риском возникновения критических ошибок.

При работе с нативным кодом особое значение имеет совместимость приложения с различными архитектурами процессоров Android-устройств. Современные смартфоны могут использовать ARM, ARM64, x86 и другие архитектуры. Поэтому разработчик должен учитывать особенности компиляции нативных библиотек под разные платформы.

Важным этапом интеграции NDK является настройка системы сборки проекта. Android Studio использует инструменты Gradle и CMake для компиляции нативного кода и включения библиотек в Android-приложение. Корректная настройка системы сборки обеспечивает успешную компиляцию проекта и совместимость Java/Kotlin-компонентов с нативными библиотеками.

Использование нативного кода связано с вопросами стабильности и безопасности программного обеспечения. Ошибки в C/C++-коде могут приводить к переполнению буфера, утечкам памяти, аварийному завершению приложения и другим критическим проблемам. В отличие от Java и Kotlin, нативный код не обладает встроенными механизмами автоматического управления памятью, что требует особого внимания со стороны разработчика.

Для специалистов в области информационной безопасности использование NDK представляет особый интерес. Многие вредоносные Android-приложения применяют нативные библиотеки для сокрытия вредоносной активности, обхода механизмов защиты и усложнения анализа программного кода. Кроме того, ошибки в нативных компонентах могут приводить к появлению серьёзных уязвимостей безопасности. Поэтому специалист должен понимать принципы работы NDK и особенности анализа Android-приложений с использованием нативного кода.

Практическая часть

Постановка задачи

В рамках практической работы студенту необходимо выполнить настройку Android NDK в проекте Android Studio и исследовать особенности интеграции

нативного кода в Android-приложение. Работа направлена на изучение принципов взаимодействия Java/Kotlin-компонентов с библиотеками C/C++, а также получение практических навыков настройки среды разработки для работы с нативными технологиями.

На первом этапе выполнения задания студенту необходимо подготовить среду разработки Android Studio и изучить структуру Android-проекта. После подготовки проекта требуется ознакомиться с назначением Android NDK, особенностями использования нативного кода и принципами взаимодействия Android-приложения с библиотеками C/C++.

Следующим этапом является установка и настройка компонентов Android NDK. Студент должен исследовать структуру SDK Manager, определить назначение инструментов NDK и CMake, а также подготовить среду разработки для компиляции нативных библиотек. Требуется проанализировать, каким образом Android Studio организует сборку проектов с использованием нативного кода.

После настройки среды необходимо исследовать структуру проекта Android-приложения с поддержкой NDK. В процессе выполнения задания студент должен определить расположение нативных файлов, особенности конфигурации Gradle и назначение файлов сборки CMake. Особое внимание следует уделить организации взаимодействия между Java/Kotlin-кодом и библиотеками C/C++.

Далее необходимо реализовать подключение нативной библиотеки к Android-приложению. Студент должен исследовать механизм вызова функций нативного кода из Android-приложения, определить способы передачи данных между различными языками программирования и проанализировать особенности использования JNI.

Особое внимание необходимо уделить процессу компиляции нативного кода под различные архитектуры мобильных устройств. В рамках задания студент должен определить, каким образом Android Studio формирует библиотеки для различных типов процессоров и как архитектурные особенности влияют на совместимость приложения.

Следующим этапом работы является исследование производительности и стабильности приложения с использованием нативного кода. Студент должен

определить влияние нативных библиотек на скорость выполнения операций, использование памяти и устойчивость работы Android-приложения.

В процессе выполнения практической работы необходимо исследовать возможные ошибки при интеграции NDK. Требуется проанализировать проблемы совместимости библиотек, ошибки сборки проекта, некорректную работу JNI и последствия неправильного управления памятью в нативном коде.

Дополнительно студент должен исследовать вопросы безопасности Android-приложений с использованием NDK. В рамках задания необходимо определить потенциальные риски использования нативного кода, проанализировать возможные уязвимости и исследовать способы защиты нативных библиотек от анализа и модификации.

После завершения настройки и тестирования проекта студент должен провести анализ работы Android-приложения с поддержкой NDK и подготовить отчёт, содержащий:

- описание назначения Android NDK;
- описание процесса настройки среды разработки;
- анализ структуры Android-проекта с нативными библиотеками;
- результаты исследования взаимодействия Java/Kotlin и C/C++;
- описание выявленных проблем сборки и интеграции;
- анализ вопросов безопасности нативного кода;
- выводы об особенностях использования NDK в Android-разработке.

Варианты заданий

№ варианта	**Содержание задания**
1	Настроить Android NDK в проекте мобильного калькулятора и исследовать взаимодействие Java-кода с нативными функциями.
2	Реализовать Android-приложение с использованием нативной библиотеки для обработки строковых данных.
3	Настроить проект с использованием CMake и исследовать процесс сборки нативных

	библиотек.
4	Разработать приложение с использованием JNI для выполнения математических вычислений в нативном коде.
5	Настроить Android NDK для проекта обработки изображений и исследовать влияние нативного кода на производительность.
6	Реализовать Android-приложение с использованием нативной библиотеки для работы с файловой системой.
7	Настроить проект для компиляции нативных библиотек под различные архитектуры процессоров Android-устройств.
8	Исследовать влияние ошибок управления памятью в C/C++ на стабильность Android-приложения.
9	Разработать приложение с использованием нативного кода для обработки массивов данных.
10	Настроить Android NDK в проекте мультимедийного приложения и исследовать обработку аудио- или видеоданных.
11	Реализовать Android-приложение с использованием JNI для обмена данными между Kotlin и C++.
12	Исследовать влияние нативных библиотек на размер APK-файла Android-приложения.
13	Настроить проект с использованием нескольких нативных библиотек и исследовать их взаимодействие.
14	Разработать Android-приложение с использованием нативного кода для выполнения

	криптографических операций.
15	Исследовать вопросы безопасности Android-приложений с использованием Android NDK.
16	Настроить Android-проект для поддержки ARM и x86 архитектур мобильных устройств.
17	Реализовать приложение с использованием нативного кода для анализа сетевых данных.
18	Исследовать ошибки сборки Android-проекта при использовании Android NDK и способы их устранения.
19	Разработать Android-приложение с использованием JNI для обработки пользовательского ввода.
20	Выполнить сравнительный анализ производительности Java/Kotlin-кода и нативного C/C++-кода в Android-приложении.

Контрольные вопросы

1. Что представляет собой Android NDK?
2. Для чего используется нативный код в Android-приложениях?
3. Какие языки программирования поддерживает Android NDK?
4. Что такое JNI?
5. Каким образом осуществляется взаимодействие Java/Kotlin и C/C++?
6. Какие преимущества имеет использование нативного кода?
7. Какие риски связаны с использованием C/C++ в Android-приложениях?
8. Почему управление памятью в нативном коде требует особого внимания?
9. Что такое CMake?
10. Для чего используется Gradle при работе с Android NDK?
11. Почему необходимо учитывать архитектуры процессоров Android-устройств?
12. Какие архитектуры мобильных процессоров поддерживает Android?
13. Какие ошибки могут возникать при использовании JNI?

14. Почему нативный код может приводить к снижению стабильности приложения?
15. Какие проблемы безопасности связаны с использованием Android NDK?
16. Почему вредоносные приложения используют нативные библиотеки?
17. Какие преимущества даёт использование нативного кода для производительности приложения?
18. Почему необходимо тестировать приложение на различных архитектурах устройств?
19. Какие проблемы могут возникать при сборке Android-проекта с использованием NDK?
20. Какие особенности необходимо учитывать при разработке Android-приложений с использованием нативного кода?

Список литературы

43. Билл Филлипс, Крис Стюарт, Кристин Марсикано, Брайан Гарднер Android-программирование для профессионалов. — СПб.: Питер, 2023. — 912 с.
44. Google Developers. Android NDK Guides [Электронный ресурс]. — Режим доступа: Android NDK Guides
45. Google Developers. JNI Tips [Электронный ресурс]. — Режим доступа: JNI Tips
46. Liu P., Li L., Yan Y. Identifying and Characterizing Silently-Evolved Methods in the Android API [Электронный ресурс]. — Режим доступа: arXiv.org
47. Google Developers. ConstraintLayout [Электронный ресурс]. — Режим доступа: ConstraintLayout Documentation
48. Google Developers. Test your app on Android [Электронный ресурс]. — Режим доступа: Test your app on Android