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
09.03.04 «Программная инженерия»
Направленность (профиль)
«Разработка и сопровождение программного обеспечения»

Ставрополь
2026

Содержание

ВВЕДЕНИЕ

Целью дисциплины «Практикум по разработке мобильных приложений» является формирование набора профессиональных компетенций будущих бакалавров по направлению подготовки 09.03.04 «Программная инженерия».

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.

Формируемые компетенции: ОПК-2, ОПК-6, ПК-2

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.

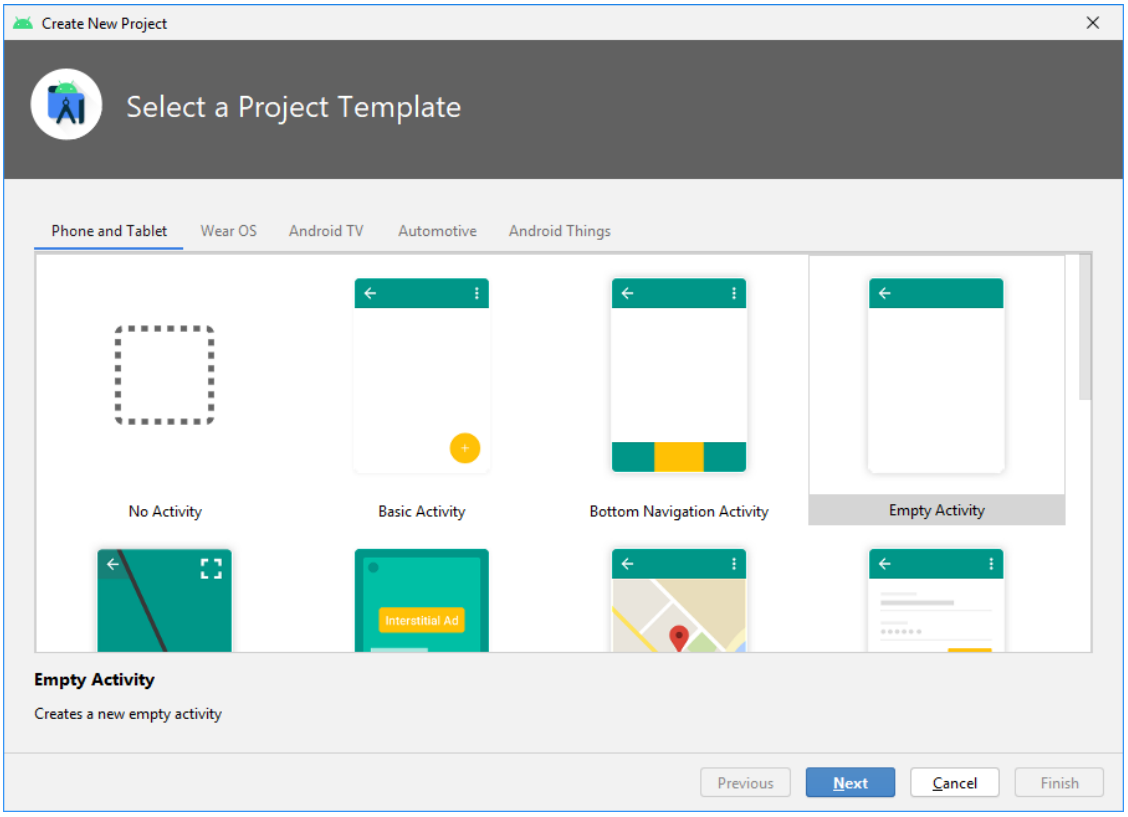


Рисунок 2 – Выбор макета приложения

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

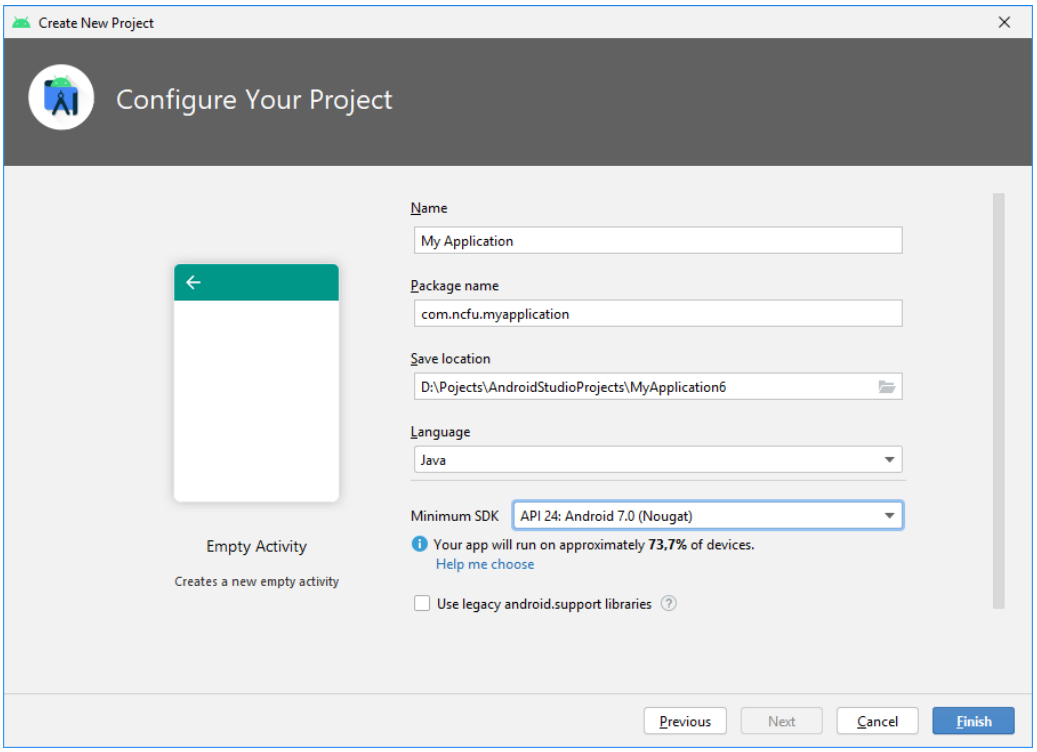


Рисунок 3 – Выбор места создания проекта

3) Далее нажимаем «Финиш».

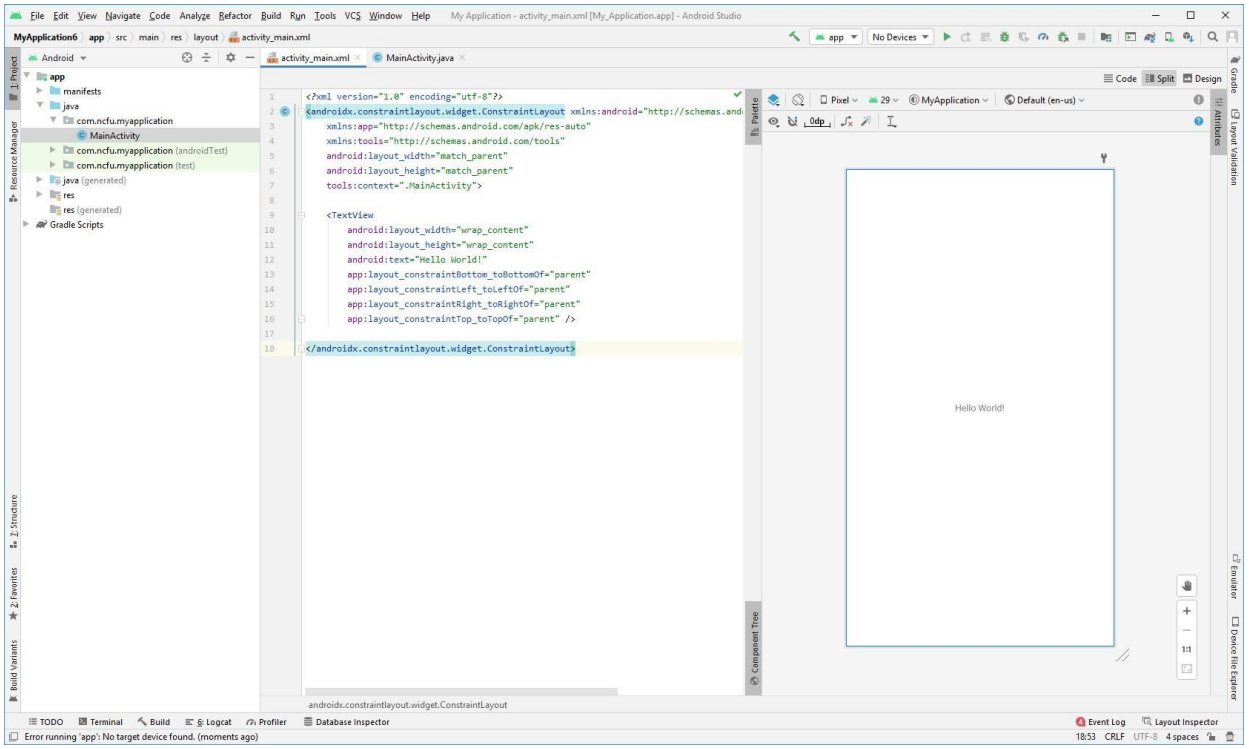


Рисунок 4 – Главное окно приложения

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

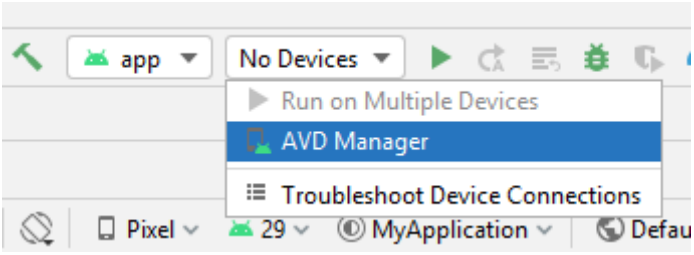


Рисунок 5 – Выбор виртуального устройства

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.

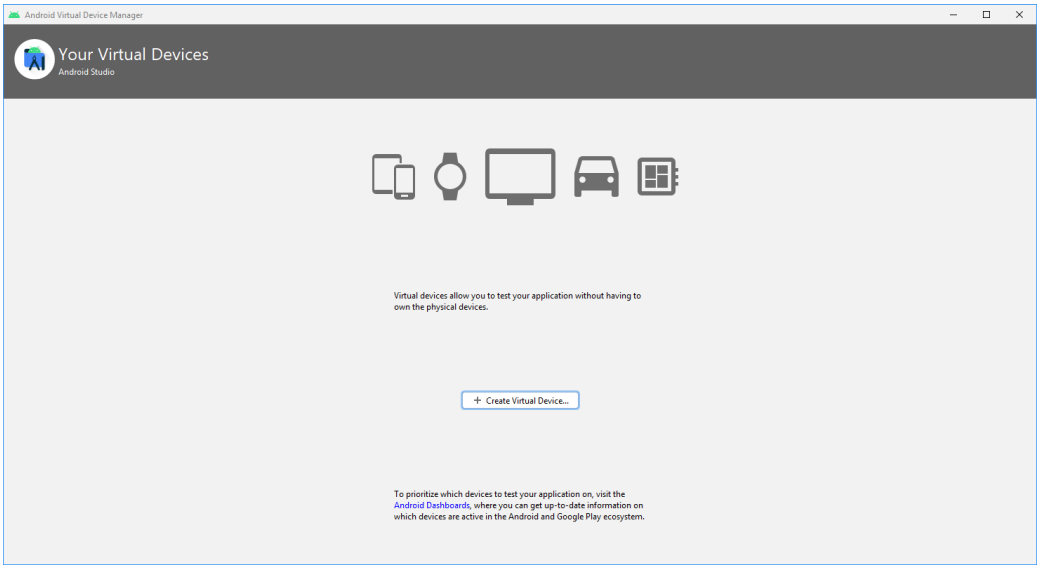


Рисунок 6 – Выбор типа устройства эмулятора

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».

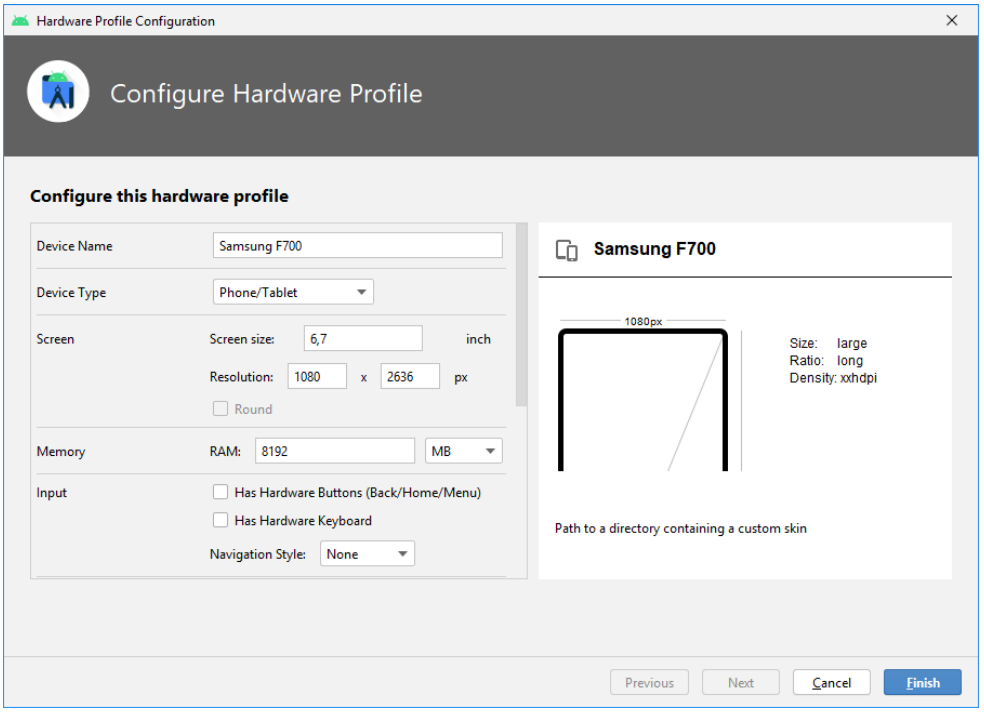


Рисунок 7 – Внесение изменений эмулятора

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

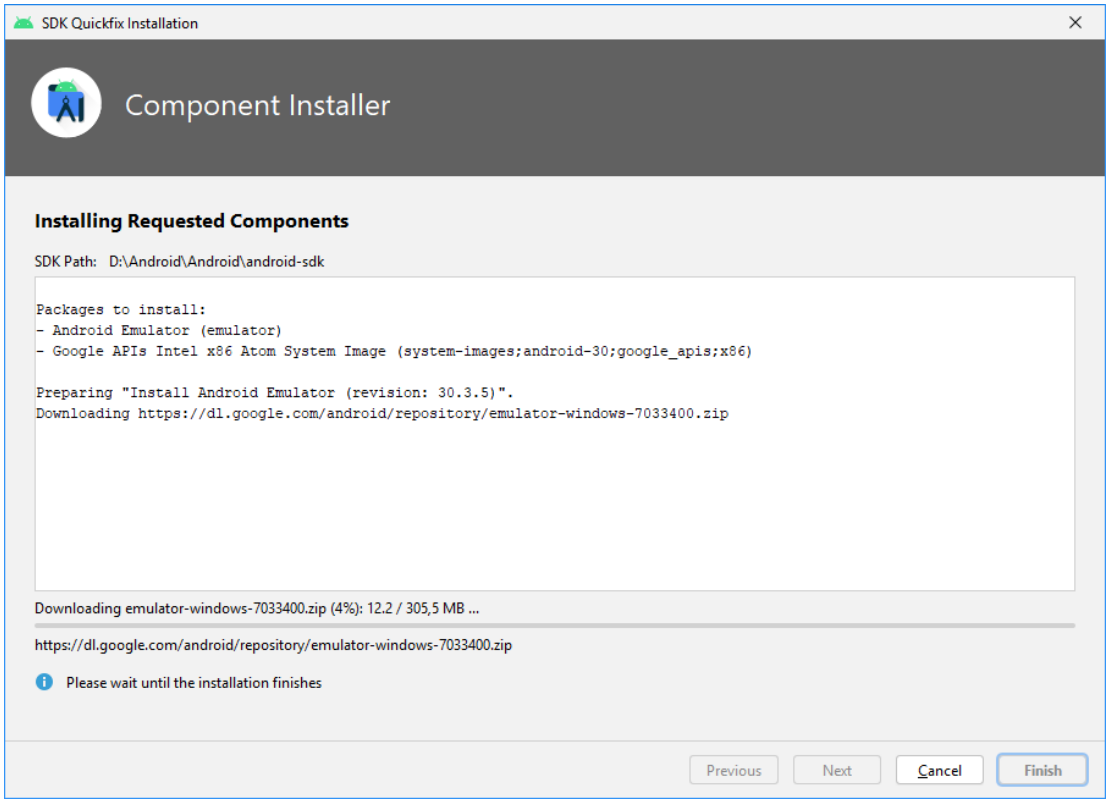


Рисунок 8 – Скачивание Android 11

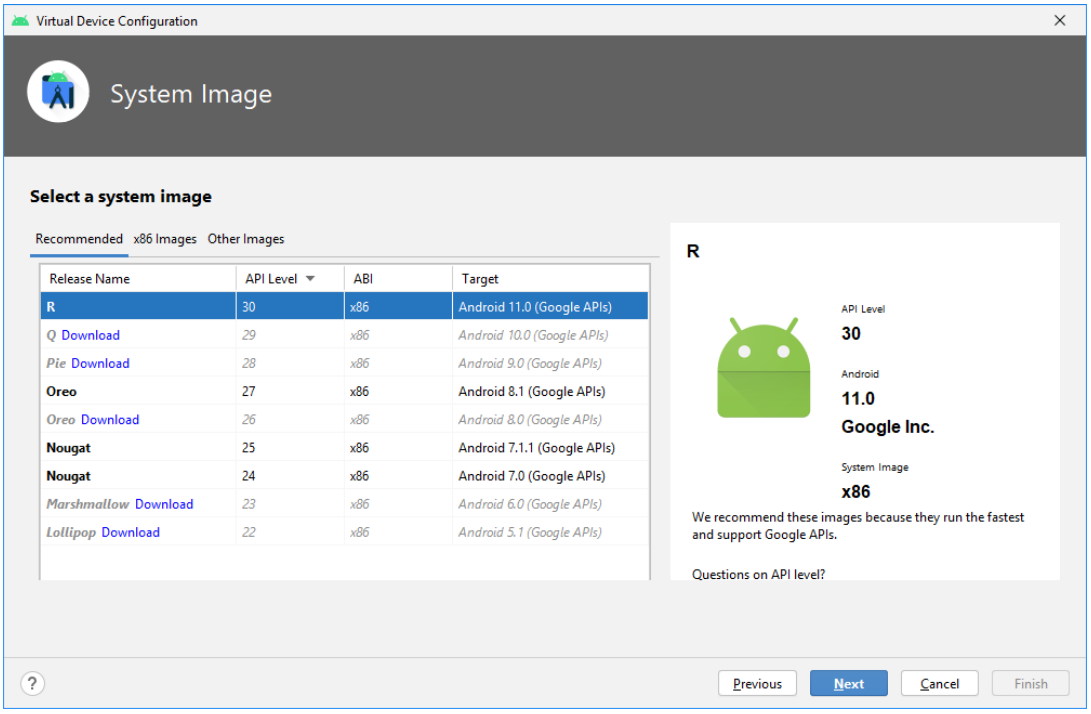


Рисунок 9 – Выбор версии Android

После создания эмулятора оно будет доступно в списке AVD.

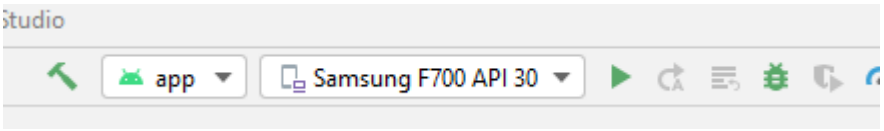


Рисунок 10 – Выбор устройства

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.

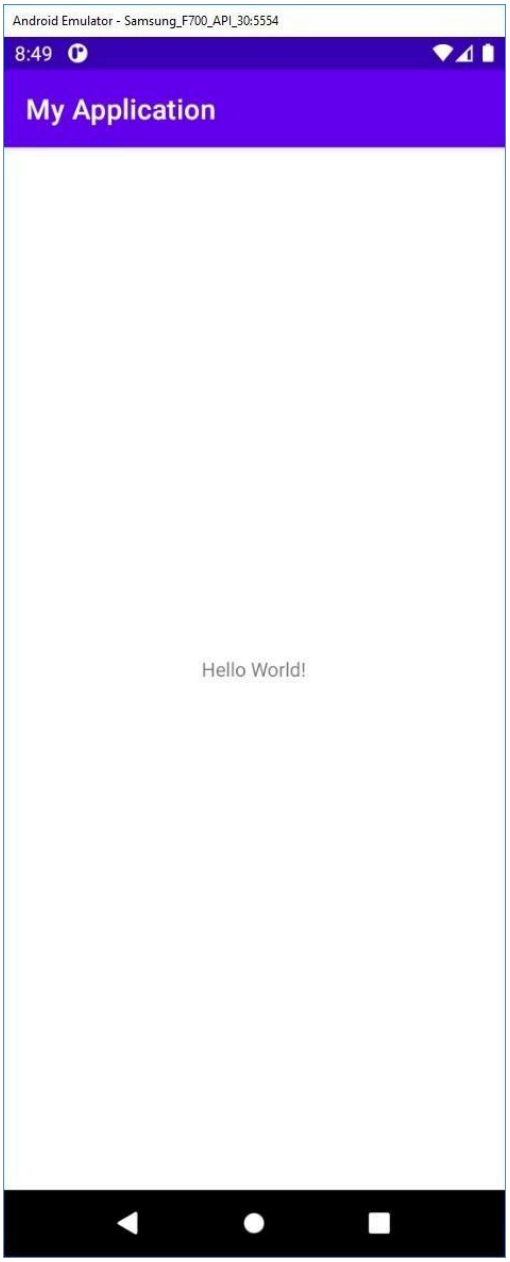


Рисунок 11 – Главное окно эмулятора

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.

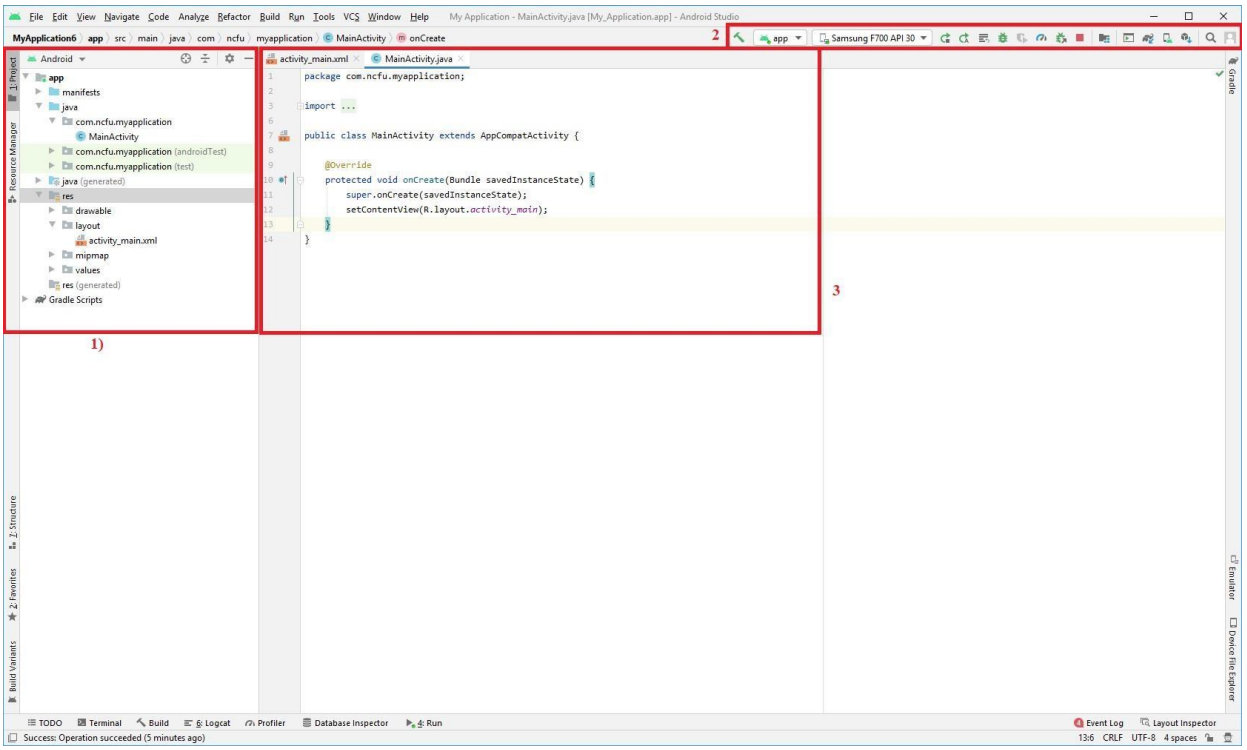


Рисунок 12 – Интерфейс Java

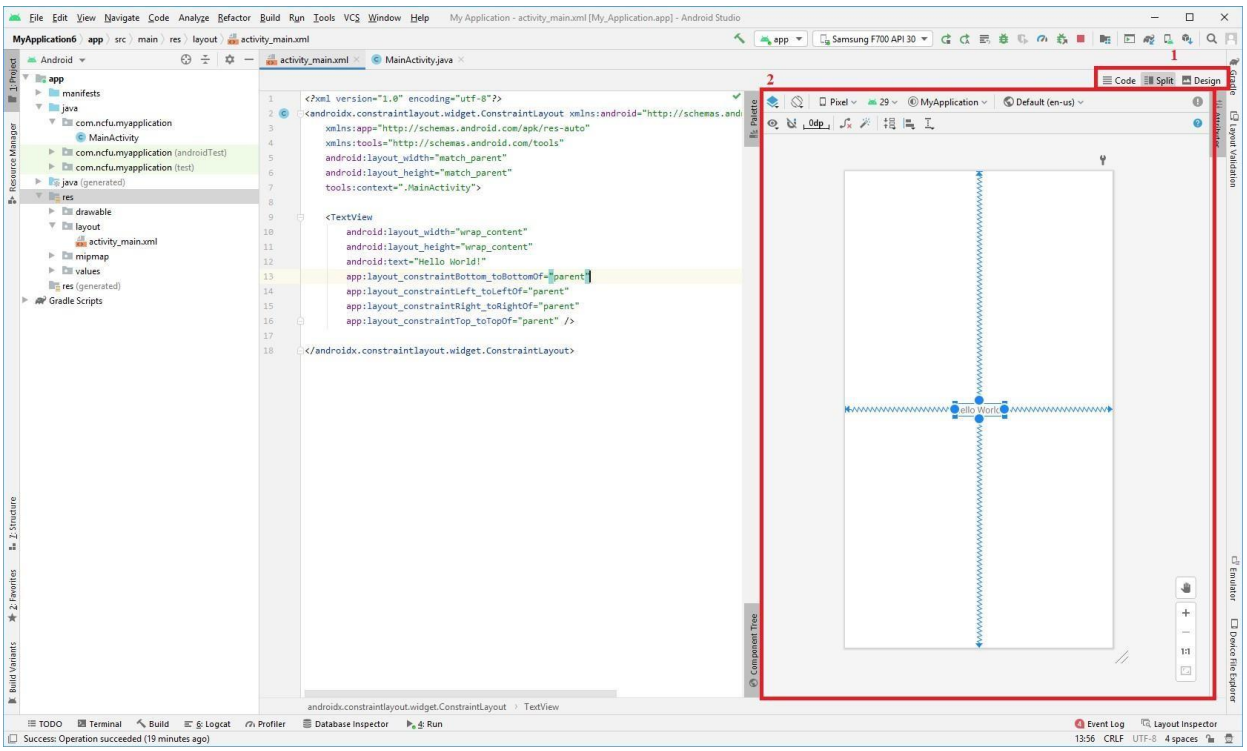


Рисунок 13 – Интерфейс xml

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).

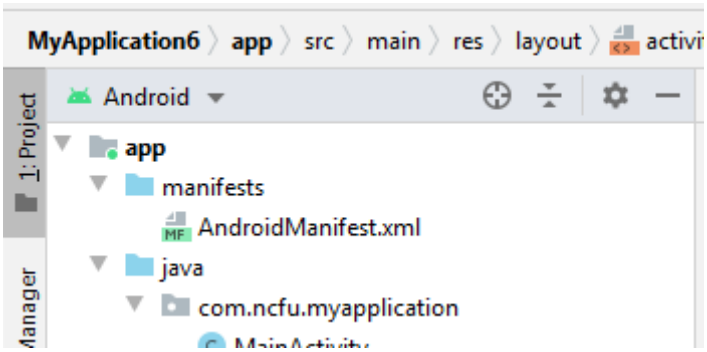


Рисунок 15 – Расположение AndroidManifest.xml в дереве решний

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.

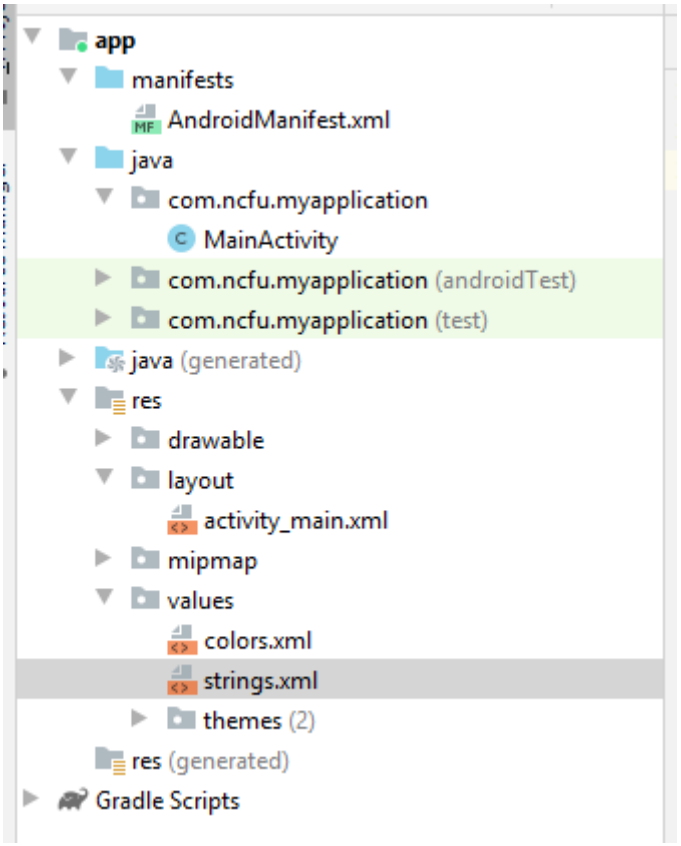


Рисунок 17 – Содержимое папки res (ресурсы)

```
<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, а 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

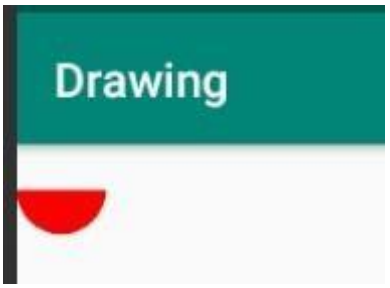


Рисунок 28 – Результат

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

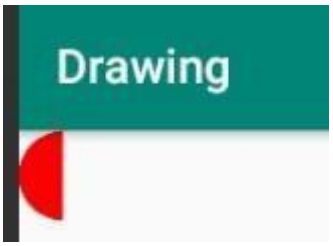


Рисунок 30 – Результат

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

Формируемые компетенции: ПК-2

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.

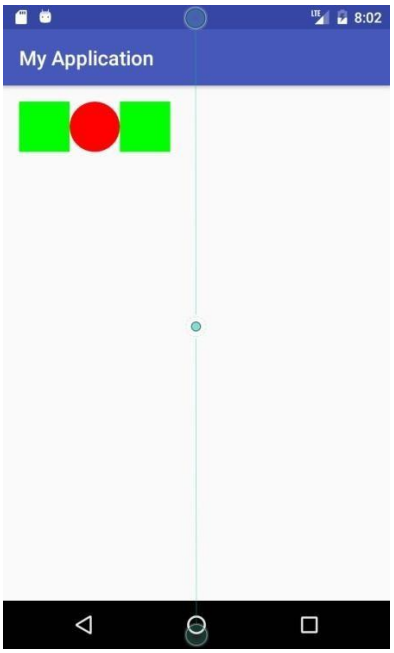


Рисунок 2 - Отображение результата на эмуляторе

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

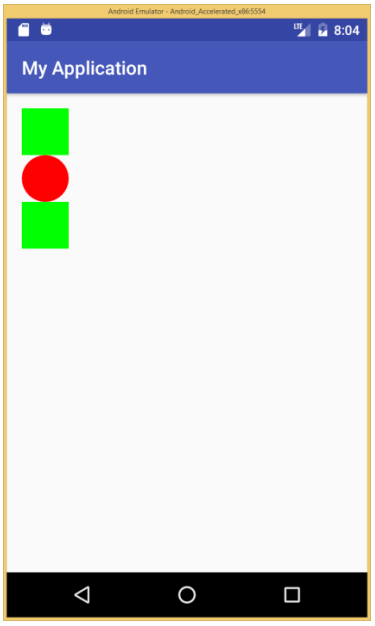


Рисунок 3 - Использование вертикальной ориентации

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

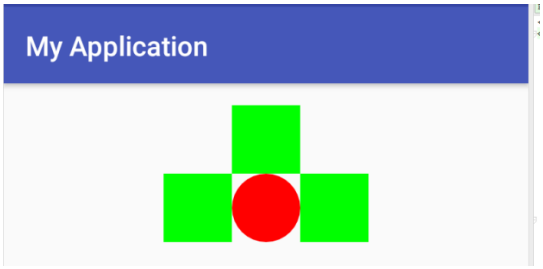


Рисунок 5 - Результат листинга 5

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-a

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:

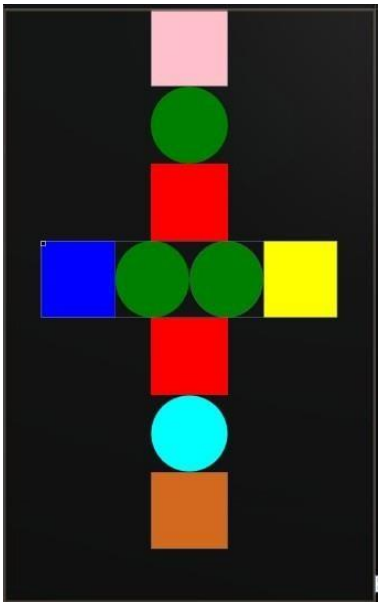


Рисунок 8 - Задание для проверки способностей

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

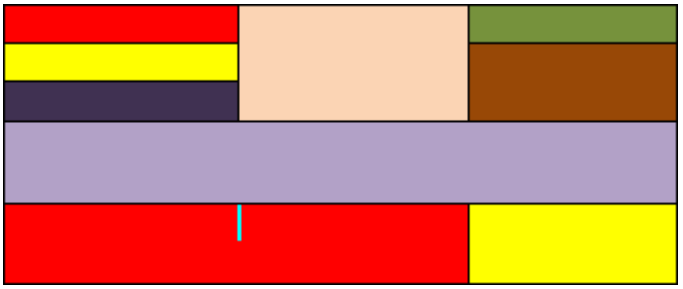


Рисунок 21

Рисунок 12

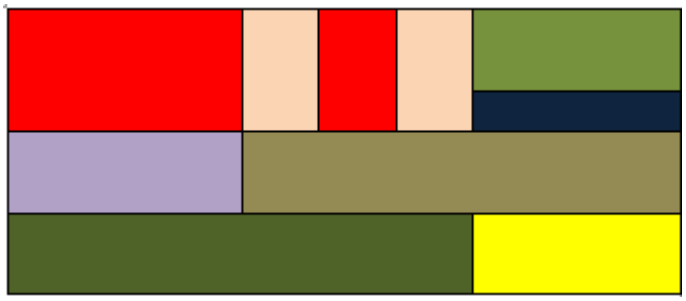


Рисунок 33

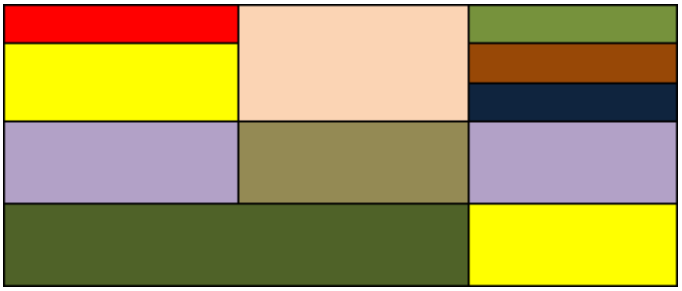


Рисунок 44

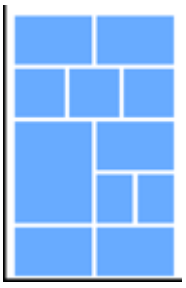


Рисунок 15

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.

Формируемые компетенции: ОПК-2, ОПК-6, ПК-2

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:

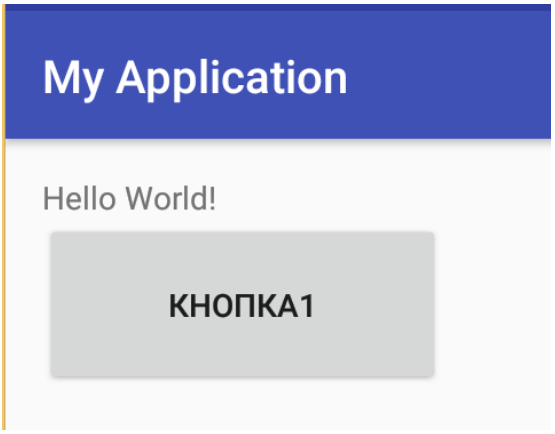


Рисунок 1 - Использование вертикальной ориентации

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.

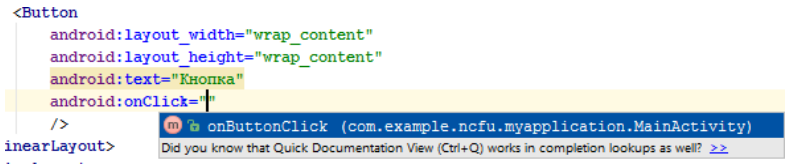


Рисунок 7 - Добавление обработчика в XML

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 Buttoni 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:

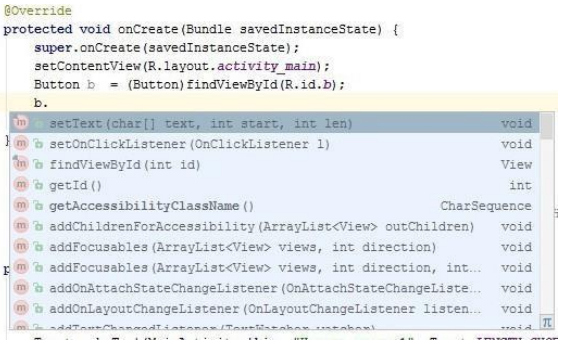


Рисунок 15 - Добавление слушателя

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

Формируемые компетенции: ОПК-2, ОПК-6, ПК-2

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.

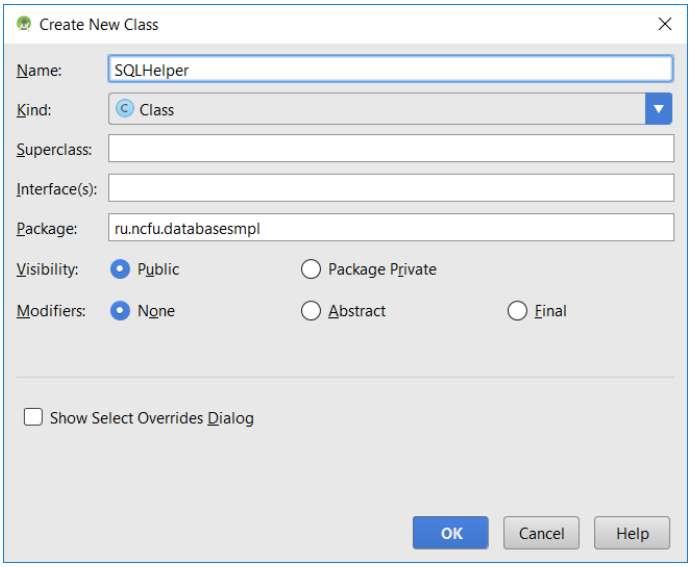


Рисунок 1 – Создание класса

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.

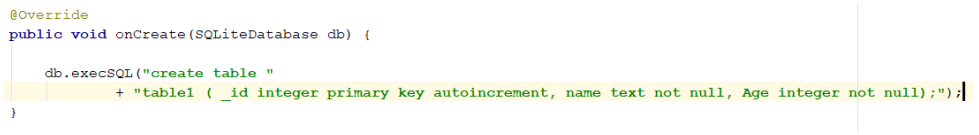


Рисунок 4 – Создание БД

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.

Формируемые компетенции: ОПК-2, ОПК-6, ПК-2

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.

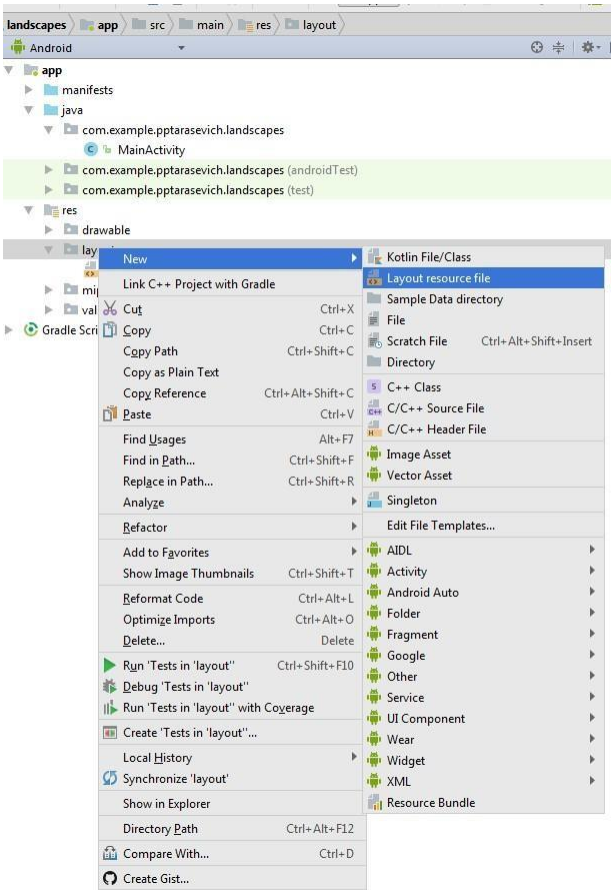


Рисунок 1 – Создание нового layout

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

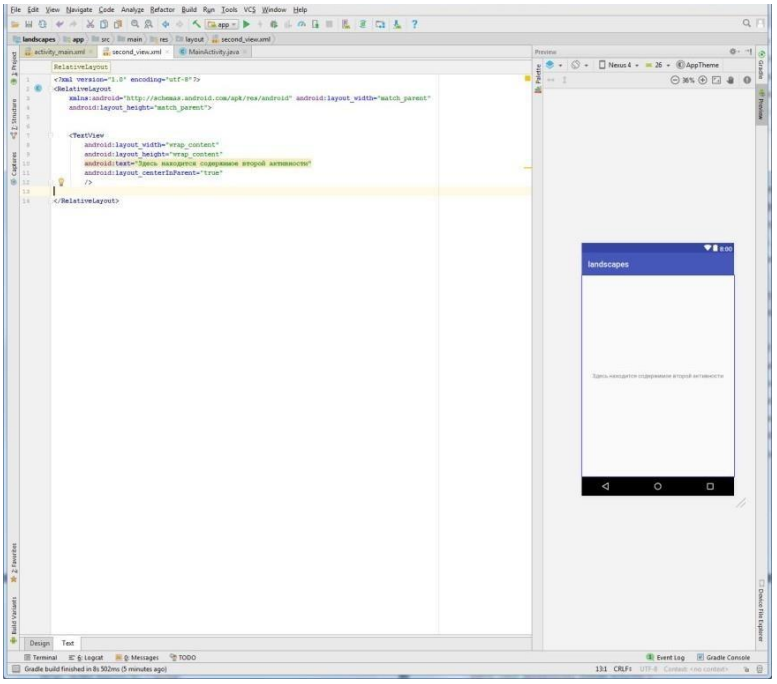


Рисунок 2 – Содержимое layout

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

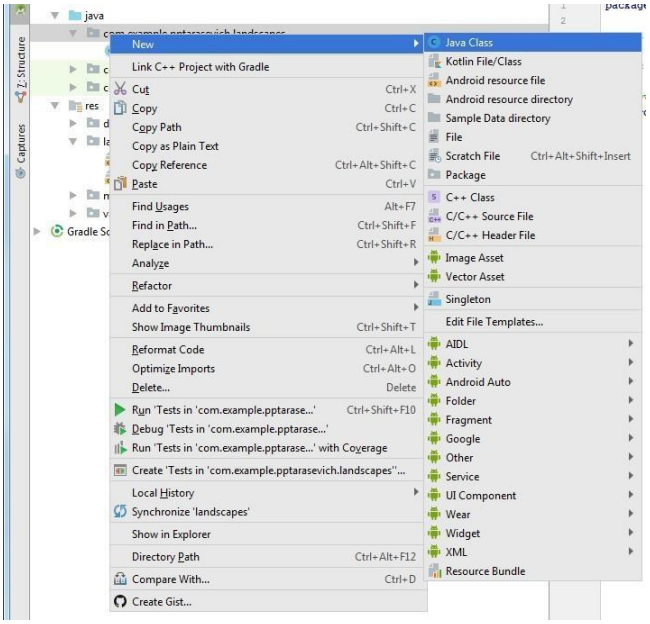


Рисунок 3 – Добавление класса

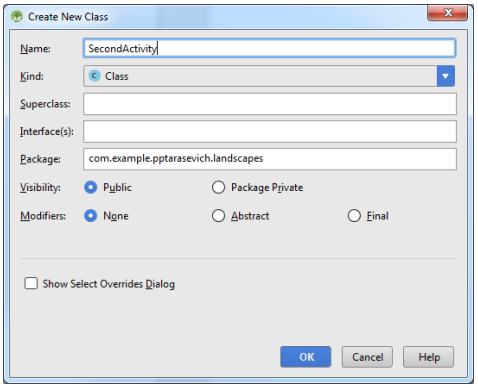


Рисунок 4 – название класса

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

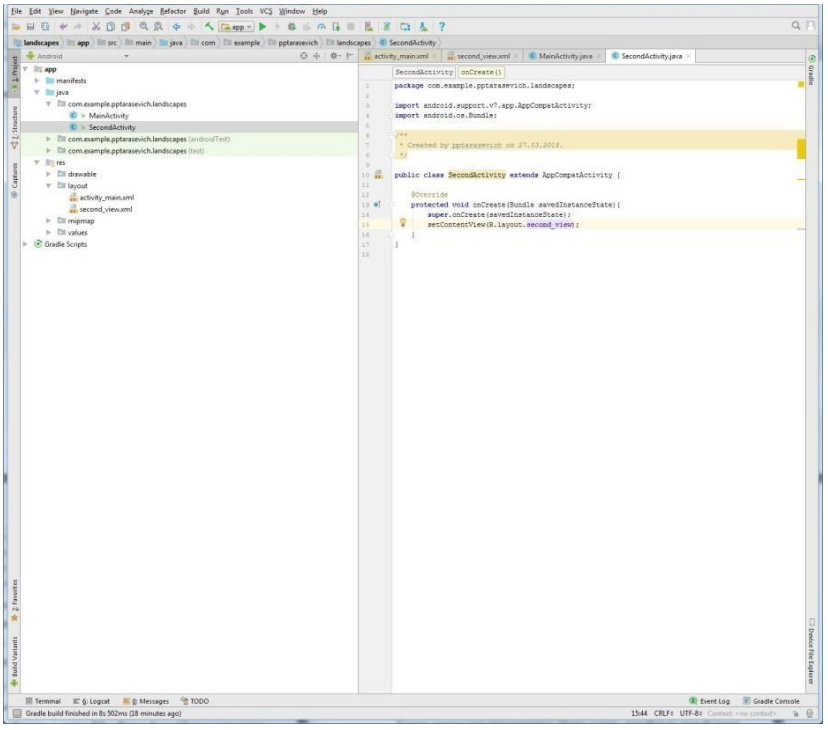


Рисунок 5 – Перегрузка onCreate()

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.

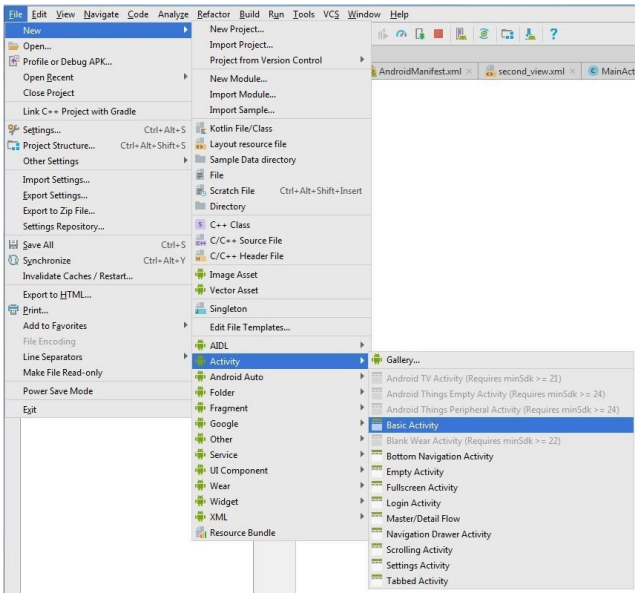


Рисунок 9 – Упрощённый способ создания

Изменим название активности на нужное.

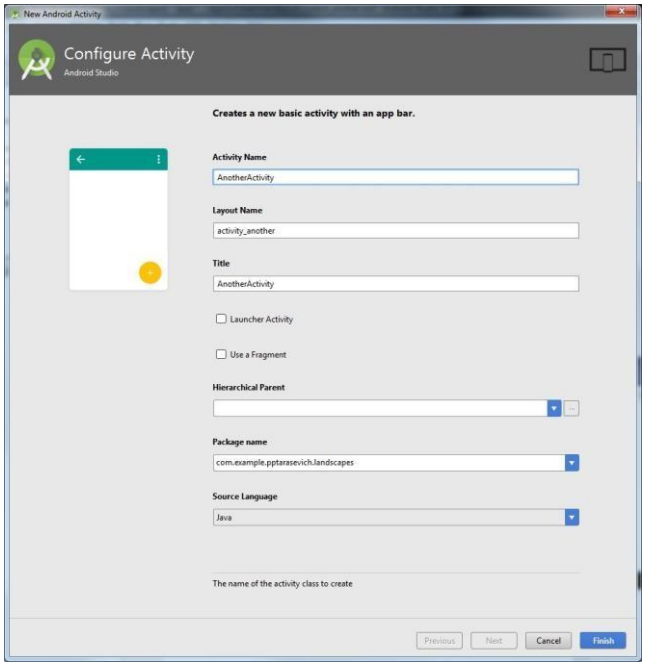


Рисунок 10 – Мастер по созданию нового Intent

Данное действие создало 2 файла и модифицировало AndroidManifest.

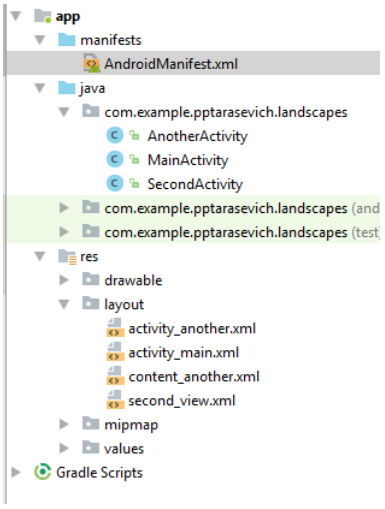


Рисунок 11 – Дерево решений

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

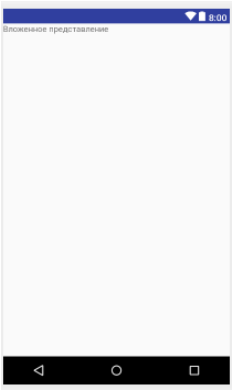


Рисунок 14 – Пример использования include

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.

```java
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.

```java
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. A 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

Формируемые компетенции: ПК-7

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.

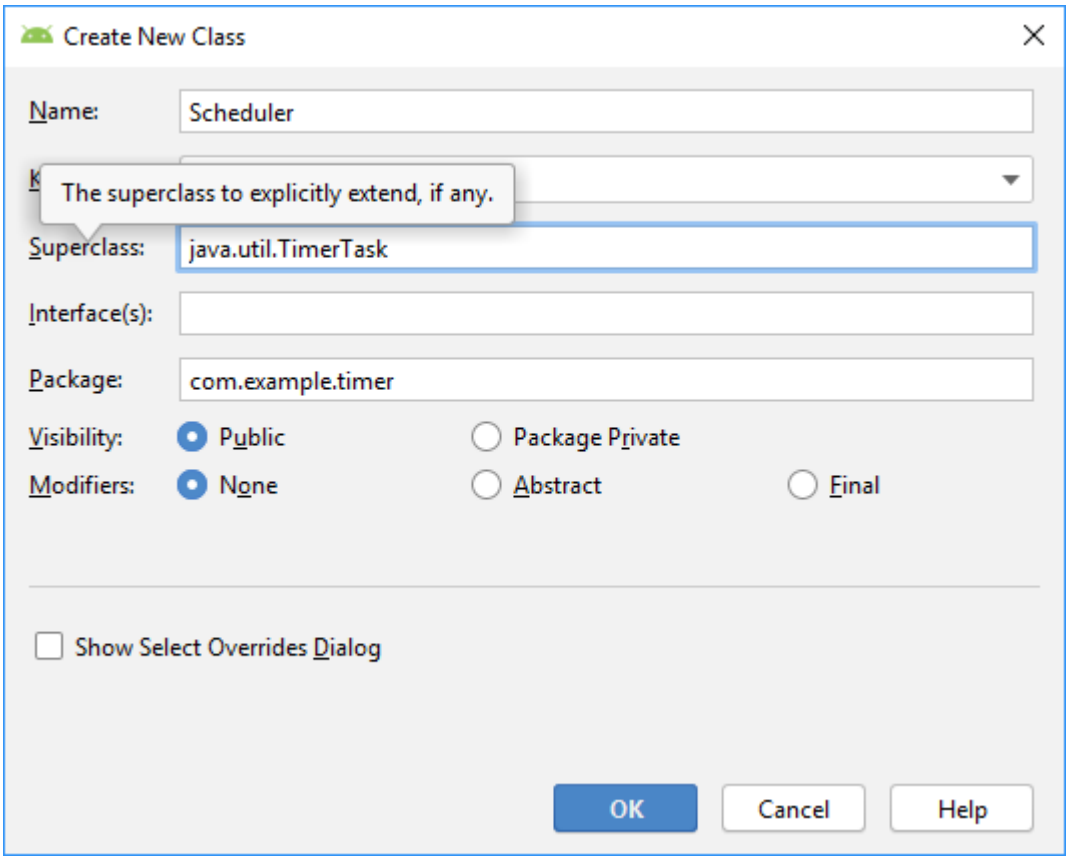
Таймер

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.

Формируемые компетенции: ОПК-2, ОПК-6, ПК-2

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:

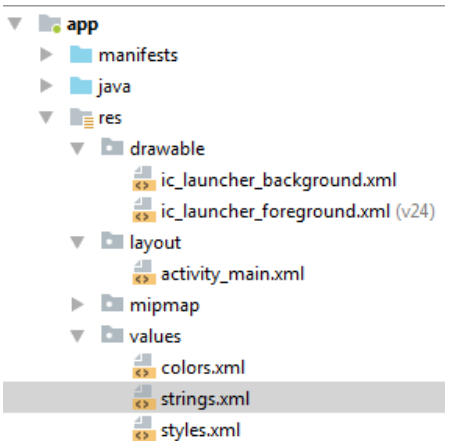


Рисунок 1 –Расположение папки values

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.

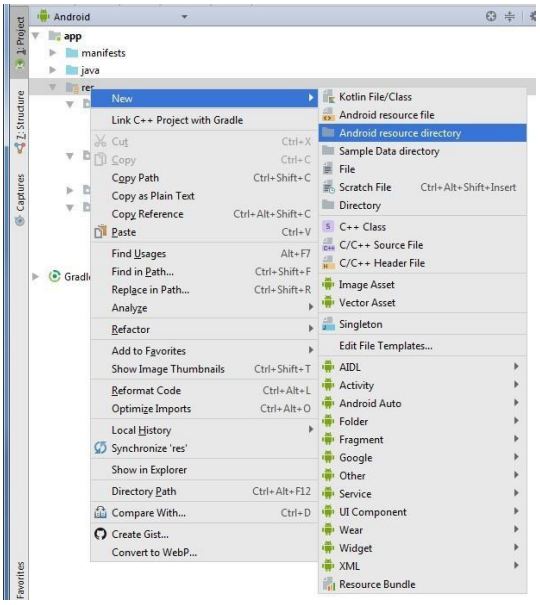


Рисунок 2 – Добавление директории

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

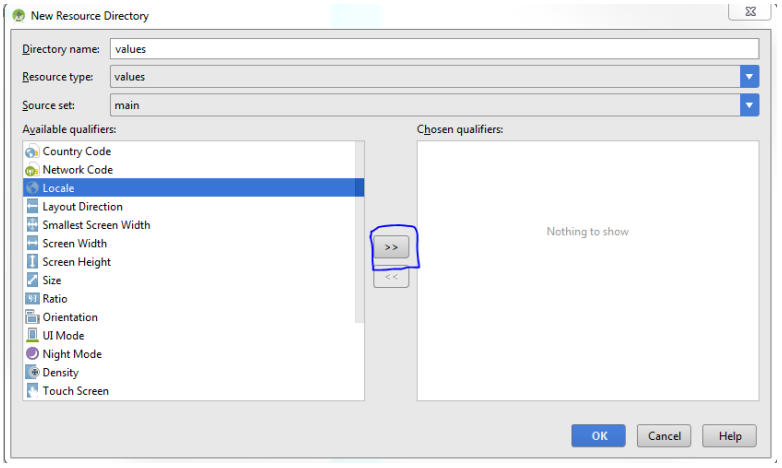


Рисунок 3 – Добавление локализаци

Найдём в списке Россию и нажмём OK.

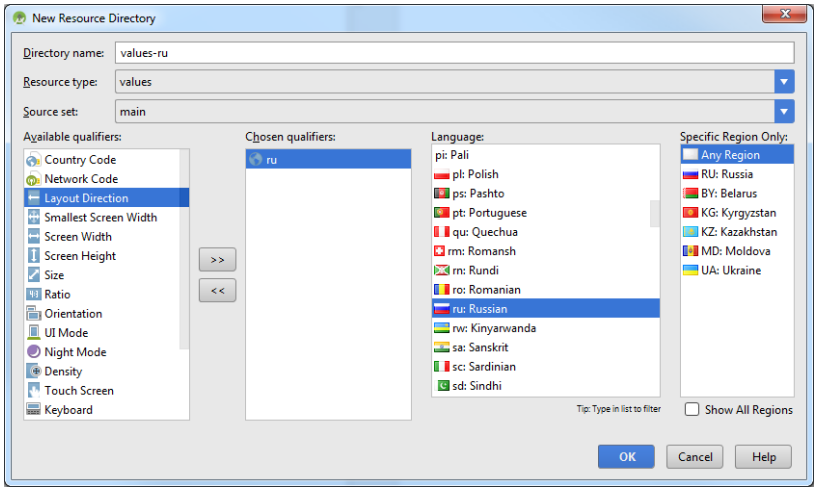


Рисунок 4 – Добавление локализации Россия

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

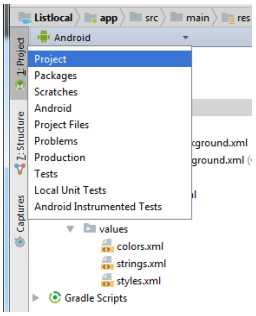


Рисунок 5 – Переключение вида

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

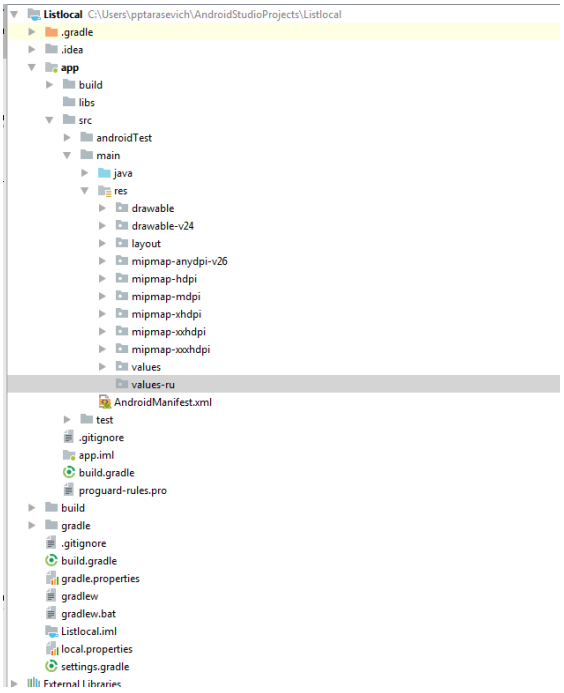


Рисунок 6 –Создание нового файла

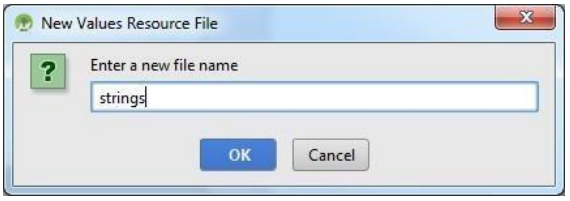


Рисунок 7 –Название файла ресурсов

Добавим строку.

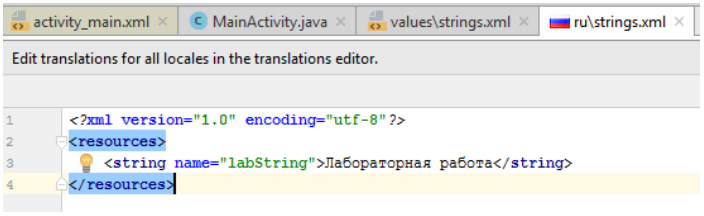


Рисунок 8 – Добавление ресурса

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.

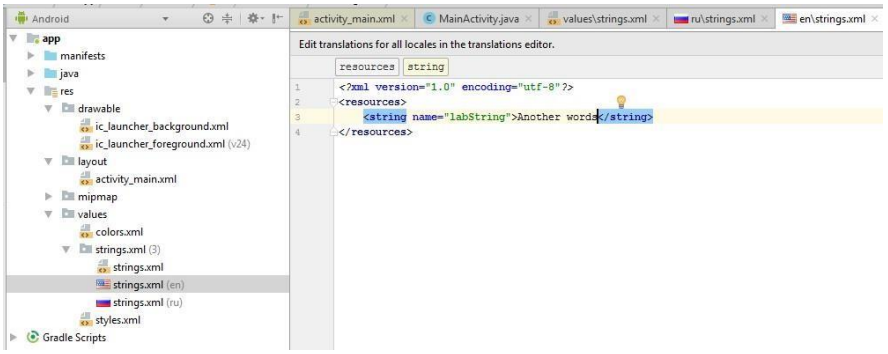


Рисунок 9 – Добавление английской локализации

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

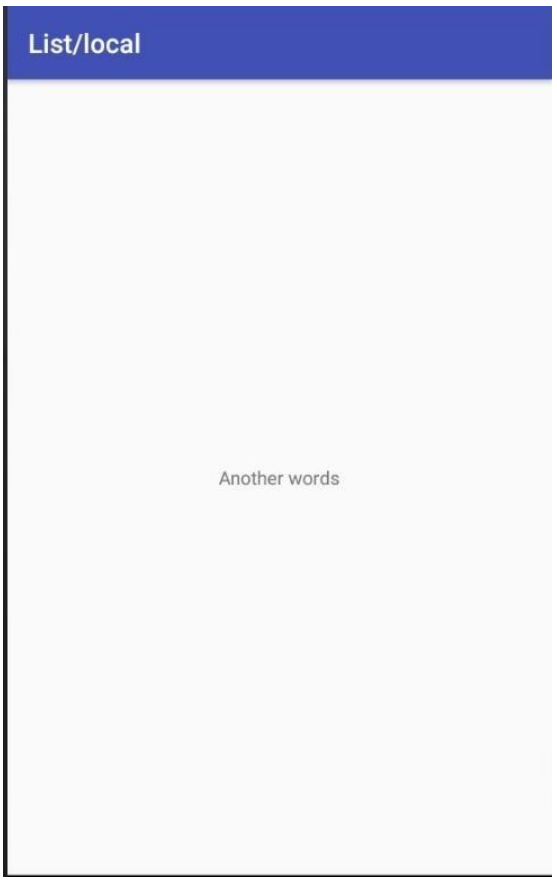


Рисунок 11 –Пример русского языка

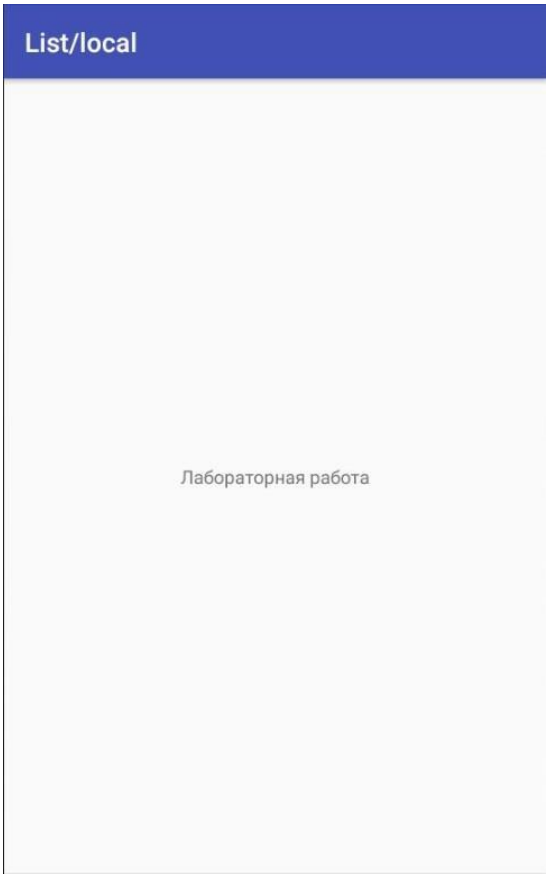


Рисунок 12 – Пример английского языка

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.

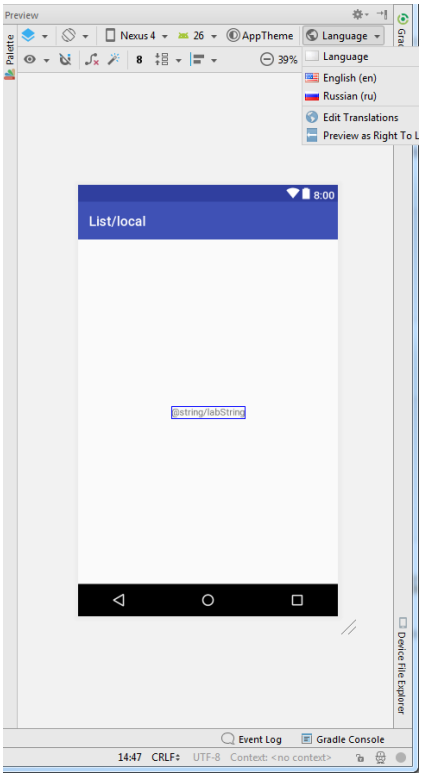


Рисунок 13 –Предпросмотр локализации

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.

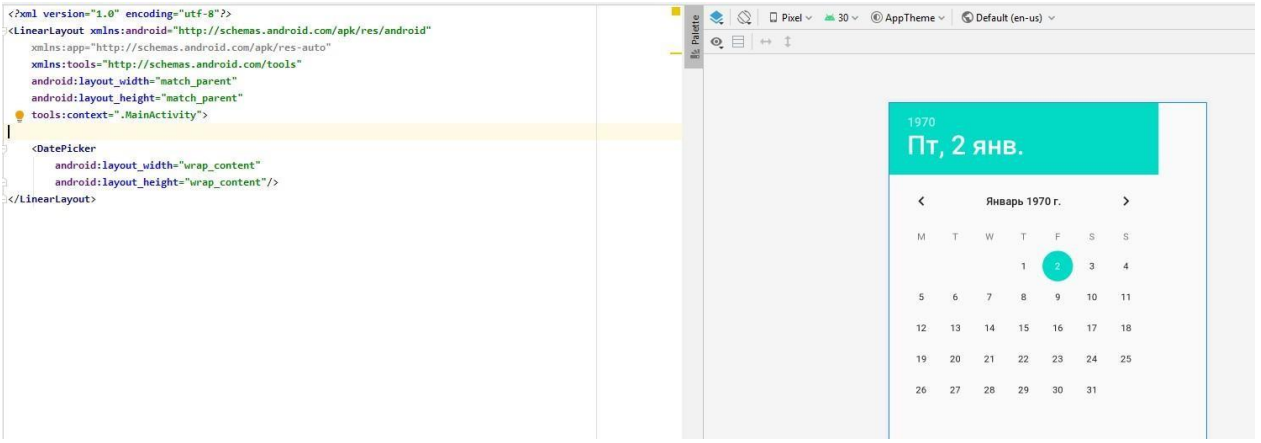


Рисунок 27 – Использование DatePicker

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.

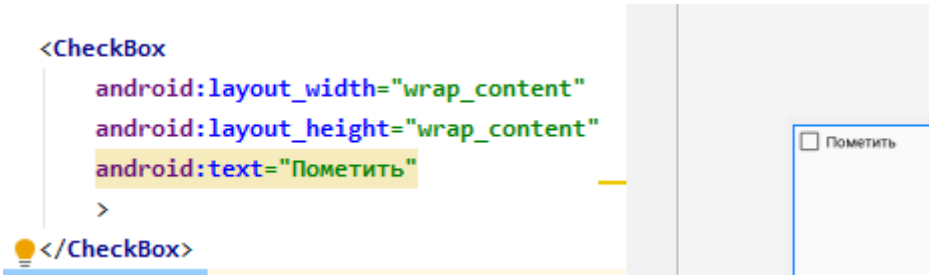


Рисунок 29 – Простейший пример Checkbox

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

Лабораторная работа 8. Ресурсы. Медиа-элементы.

Цель работы: изучить способы отображения и манипулирования внешними файлами.

Формируемые компетенции: ОПК-2, ОПК-6, ПК-2

Теоретическая часть:

Разумеется, не всё нужное можно нарисовать и отобразить штатными средствами AndroidStudio. Поэтому, в ней есть средства для отображения внешних файлов. Начнём с простых изображений. Чтобы отобразить внешнюю картинку в проекте есть два способа, в зависимости от того, является ли он частью проекта или нет.

Если Вы хотите использовать заранее подготовленные изображения, можно смело использовать папку «Res». Как Вы помните, все файлы в данной папке распределены по категориям. Например, всё, что Вы нарисовали в AndroidStudio, располагается в папке drawable. Общие строковые значения хранятся в папке values и тд. Для внешних файлов же следует использовать папку, которую Вы сами создадите (кроме png-картинок). Она должна называться raw:

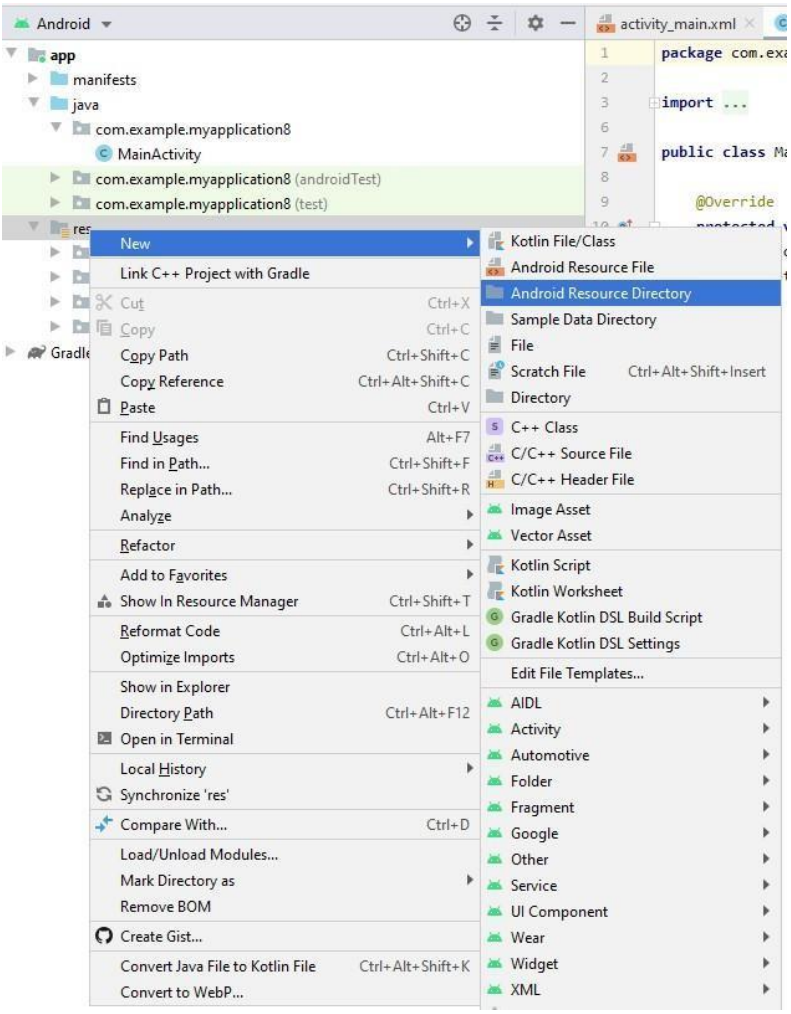


Рисунок 1 – Добавление директории

Назвать папку можно как угодно, но раз уж она хранит в себе raw-файлы, то пусть так и называется.

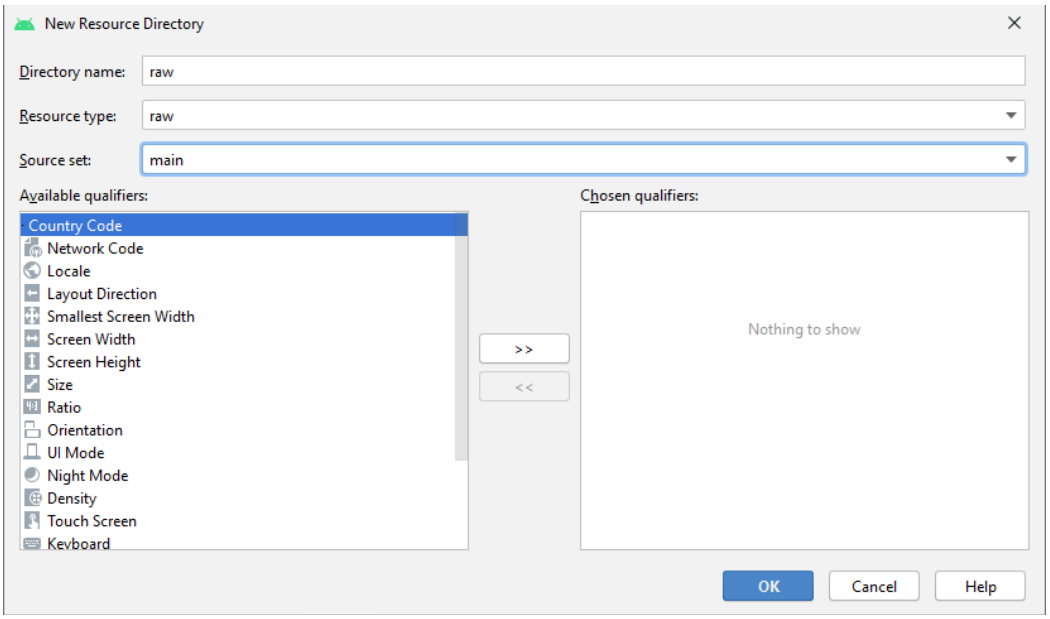


Рисунок 2 – Указание типа и названия директории

Скачаем любую картинку из интернета (приличную). Найдём изображение в формате png, чтобы посмотреть простой способ добавления и поместим её внутрь папки drawable. Чтобы быстро найти эту папку, нажмём правой кнопкой по ней и выберем Show in Explorer:

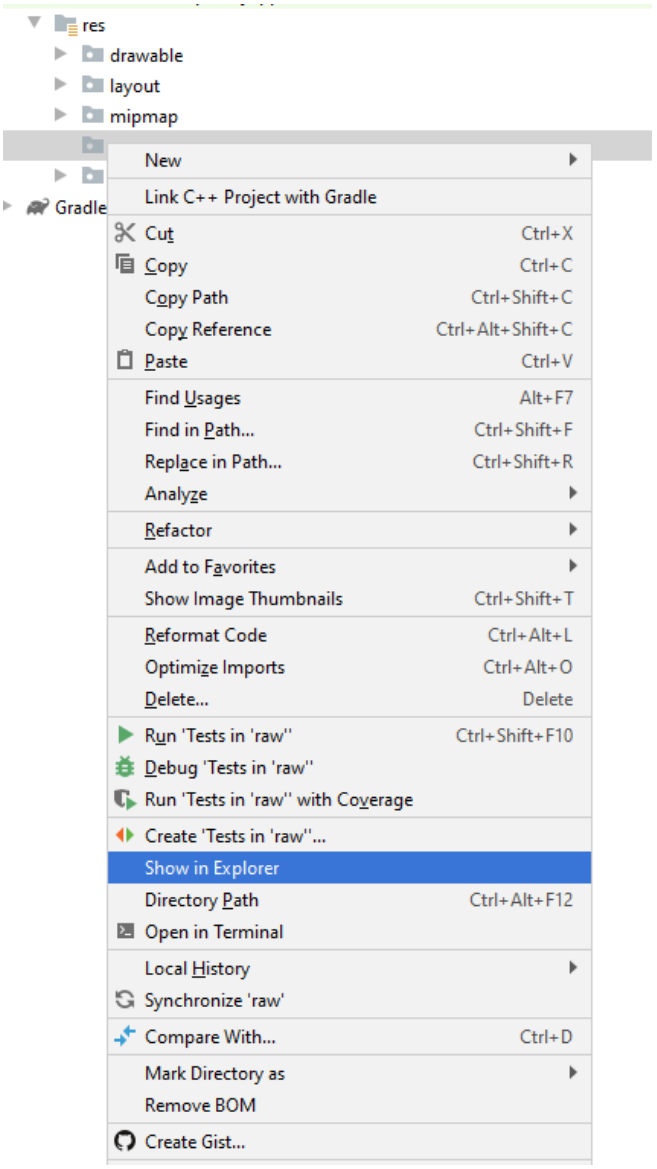


Рисунок 3 – Контекстное меню созданной папки

Переместим скачанное изображение в папку. Для отображения изображений используется знакомый Вам тэг – ImageView. Поместим код ниже в activity_main.xml:

```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"
    tools:context=".MainActivity">

<ImageView android:layout_height="150dp"
    android:layout_width="150dp"
    android:background="#000"
    android:id="@+id/img"    />
</LinearLayout>
```

Рисунок 4 – activity_main.xml

Теперь присвоим нашему ImageView в качестве источника – картинку, скачанную из интернета. Для этого перейдём в MainActivity.java и добавим следующий код:

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ImageView img = (ImageView) findViewById(R.id.img);
    InputStream imageStream = this.getResources().openRawResource(R.raw.logo);
    Bitmap bitmap = BitmapFactory.decodeStream(imageStream);
    img.setImageBitmap(bitmap);
}
```

Рисунок 5 – Заполнение содержимого из файла

Запустив приложение, убедимся, что всё работает:

Рисунок 6 – Результат добавления изображения

Теперь рассмотрим вариант, когда у Вашего приложения есть доступ к устройству пользователя и Вы точно знаете, где лежит нужный Вам файл. Чтобы получить доступ к устройству, необходимо настроить соответствующее разрешение. Вообще, с разрешениями Вы познакомитесь в будущих лабораторных занятиях. Сейчас же мы познакомимся с теми, что понадобятся для выполнения именно этой. Открываем файл AndroidManifest (расположение помним из первой лабораторной) и добавляем следующую строку:

activity_main.xml | MainActivity.java | AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication8">

    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"></uses-permission>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application8"
        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>
    </application>

</manifest>
```

Рисунок 7 – Добавление разрешения на чтение

Данная строка позволяет получить доступ к хранилищу телефона, чтобы можно было прочитать файлы. Итак, предположим, что пользователь сохранил изображение на флешке в папке Lab в корне:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    File f = new File( pathname: Environment.getStorageDirectory().getAbsoluteFile() + "/photo.jpg");
    ImageView mImgView1 = (ImageView)findViewById(R.id.imageView);
    Bitmap bmp = BitmapFactory.decodeFile(f.getAbsolutePath());
    mImgView1.setImageBitmap(bmp);
}
```

Рисунок 8 – Открытие внешнего файла

Запустим и убедимся, что всё работает.

Для добавления видео и аудио принцип не очень отличается. Рассмотрим для начала видеофайл. Скачайте и поместите видеофайл в формате .mp4 в папку raw. Добавьте в activity_main.xml следующий код:

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <Button
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:text="Play"
        android:id="@+id/playButton"
        android:onClick="play"/>
    <Button
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:text="Pause"
        android:id="@+id/pauseButton"
        android:onClick="pause"/>
    <Button
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:text="Stop"
        android:id="@+id/stopButton"
        android:onClick="stop"/>
    <VideoView
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:id="@+id/videoPlayer"/>
</LinearLayout>
```

Рисунок 9 – добавление разметки для управления пллером

А в MainActivity это:

```
public class MainActivity extends AppCompatActivity {

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

        videoPlayer =  (VideoView)findViewById(R.id.videoPlayer);
        Uri myVideoUri= Uri.parse( "android.resource://" + getPackageName() + "/" + R.raw.cats);
        videoPlayer.setVideoURI(myVideoUri);
    }

    public void play(View view){
        videoPlayer.start();
    }
    public void pause(View view){
        videoPlayer.pause();
    }
    public void stop(View view){
        videoPlayer.stopPlayback();
        videoPlayer.resume();
    }

}
```

Рисунок 10 – Добавление кода по управлению плеером

Естественно, кнопки Play и Pause можно убрать, и добавить вместо них встроенный компонент:

```
videoPlayer =  (VideoView)findViewById(R.id.videoPlayer);
Uri myVideoUri= Uri.parse( "android.resource://" + getPackageName() + "/" + R.raw.cats);
videoPlayer.setVideoURI(myVideoUri);
MediaController mediaController = new MediaController( context: this);
videoPlayer.setMediaController(mediaController);
mediaController.setMediaPlayer(videoPlayer);
```

Рисунок 11 – Добавление контроллера для плеера

Естественно, стоит выбирать файлы, форматы которых телефон сможет прочитать. Так, распространённый формат «.mkv» не сможет быть открыт данным плеером, так как на телефоне не установлены требуемые кодеки.

Для добавления аудио принцип схож, как и с видео. Однако, для отображения аудиозаписи нам не всегда нужен интерфейс. Для простого

воспроизведения воспользуемся следующим кодом. Изменим те 3 кнопки (Play, Pause, Stop) для работы с аудио:

```
public void play(View view){

    mPlayer.start();
    playButton.setEnabled(false);
    pauseButton.setEnabled(true);
    stopButton.setEnabled(true);
}
public void pause(View view){

    mPlayer.pause();
    playButton.setEnabled(true);
    pauseButton.setEnabled(false);
    stopButton.setEnabled(true);
}
public void stop(View view){
    stopPlay();
}
```

Рисунок 12 – изменённый код обработки кнопок

И добавим инициализацию:

```
MediaPlayer mPlayer;
Button playButton, pauseButton, stopButton;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mPlayer= MediaPlayer.create( context: this, R.raw.cats);
    mPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mp) {
            stopPlay();
        }
    });
    playButton = (Button) findViewById(R.id.playButton);
    pauseButton = (Button) findViewById(R.id.pauseButton);
    stopButton = (Button) findViewById(R.id.stopButton);

    pauseButton.setEnabled(false);
    stopButton.setEnabled(false);
}
```

Рисунок 13 – Инициализация MediaPlayer

Метод stopPlay будет выглядеть следующим образом:

```
private void stopPlay(){
    mPlayer.stop();
    pauseButton.setEnabled(false);
    stopButton.setEnabled(false);
    try {
        mPlayer.prepare();
        mPlayer.seekTo( msec: 0);
        playButton.setEnabled(true);
    }
    catch (Throwable t) {
        Toast.makeText( context: this, t.getMessage(), Toast.LENGTH_SHORT).show();
    }
}
```

Рисунок 14 – Метод для сброса MediaPlayer

Такой подход (фоновая музыка) подходит, если есть необходимость запустить фоновую музыку (например, в главном меню игры или во время её). Если же мы хотим добавить возможность пользователю самостоятельно выбирать с какого момента воспроизводить аудио, мы можем добавить эту информацию в интерфейс. Прежде всего добавим полосу, которая будет показывать, на каком этапе :

```
<SeekBar
    android:id="@+id/controlBar"
    android:layout_width="150dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="32dp" />
```

Рисунок 15 – Добавление полосы отображения текущей позиции аудиофайла

А в MainActivity:

Изменим содержимое Main activity на следующее. Добавляем две переменные (таймер), которые будут обновлять интерфейс при воспроизведении:

```
MediaPlayer mPlayer;
Button playButton, pauseButton, stopButton;

Timer mTimer;
TimerTask schedule;
```

Рисунок 16 – Определение Timer-а и расписания

Меняем метод play:

```
public void play(View view) throws IOException {
    mPlayer.start();

    SeekBar controlBar = (SeekBar) findViewById(R.id.controlBar);
    final int duration = mPlayer.getDuration();

    int amountToUpdate = controlBar.getProgress() == 0?  duration / 100 :  duration / 100 - controlBar.getProgress() / 100;

    controlBar.setMax(amountToUpdate);
    mTimer = new Timer();
    schedule = new TimerTask() {
        @Override
        public void run() {
            if (!(controlBar.getProgress() * 100 >= duration)) {
                int p = controlBar.getProgress();
                p = mPlayer.getCurrentPosition() / 100;
                controlBar.setProgress(p);
            }
        }
    };
    mTimer.schedule(schedule,  delay: 0, amountToUpdate);

    playButton.setEnabled(false);
    pauseButton.setEnabled(true);
    stopButton.setEnabled(true);
}
```

Рисунок 17 – Изменение кнопки воспроизведения

Чтобы получить длительность аудиофайла необходимо вызвать метод mPlayer.getDuration(). amountToUpdate – вспомогательная переменная, которая будет показывать, сколько раз необходимо обновлять интерфейс. Вначале он будет равен duration/100. Когда пользователь нажимает на «Pause», то работу таймера необходимо приостановить. Timer в Java не поддерживает паузу, поэтому, необходимо его просто остановить. Остановку таймера реализуем в методах Pause и stopPlay:

```
private void stopPlay(){
    mPlayer.stop();
    pauseButton.setEnabled(false);
    stopButton.setEnabled(false);
    try {
        mPlayer.prepare();
        mPlayer.seekTo( msec: 0);
        playButton.setEnabled(true);
    }
    catch (Throwable t) {
        Toast.makeText( context: this, t.getMessage(), Toast.LENGTH_SHORT).show();
    }
    SeekBar controlBar = (SeekBar) findViewById(R.id.controlBar);
    controlBar.setProgress(0);
    mTimer.cancel();
}

public void pause(View view){

    mPlayer.pause();
    playButton.setEnabled(true);
    pauseButton.setEnabled(false);
    stopButton.setEnabled(true);
    mTimer.cancel();
}
```

Рисунок 18 – Изменение методов паузы и остановки

Когда пользователь снова нажимает Play после паузы, то нужно продолжить воспроизведение с момента остановки. Следовательно, мы должны правильно посчитать, сколько осталось «тиков» до полной остановки таймера.

Добавим полосу для регулирования громкости. В разметке укажем:

```
<SeekBar
    android:id="@+id/volumeControl"
    android:layout_width="150dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="32dp" />
```

Рисунок 19 – Добавление полосы управления звуком в разметке

А в коде:

```
int maxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
int curValue = audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
SeekBar volumeControl = (SeekBar) findViewById(R.id.volumeControl);
volumeControl.setMax(maxVolume);
volumeControl.setProgress(curValue);
volumeControl.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        audioManager.setStreamVolume(AudioManager.STREAM_MUSIC, progress, flags: 0);
    }
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
});
```

Рисунок 20 – Обработка изменения полосы звука

Похожим образом можно изменить и способ перемещения по controlBar. Но в данной лабораторной работе это рассмотрено не будет.

Естественно, мы не всегда хотим открывать только то, что сами добавили в проект. Пользователю тоже можно предоставить открыть нужный файл. Однако, данный пример будет рассмотрен в лабораторной работе №10.

Задание:

1. Добавьте несколько изображений в папку res. Сделайте так, чтобы при нажатии кнопок, изображение менялось на другое по очереди.
2. Добавьте в папку res видеозапись. Отобразите полосу громкости для видео и MediaController.
3. Добавьте аудиозапись в папку res. При запуске приложения данная аудиозапись должна воспроизводиться фоном и ставиться на паузу, если пользователь воспроизводит видео. При паузе видео, аудиозапись должна воспроизводиться спустя 1.5 секунды. После окончания аудио, оно должно воспроизвестись заново.

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. ImageView.

2. VideoView.

3. Timer.

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 9. Создание меню.

Цель работы: Научиться создавать меню. **Формируемые компетенции:** ОПК-2, ОПК-6, ПК-2

Теоретическая часть

В стандартных бизнес-приложениях всегда есть меню. Меню может быть двух видов:

1. Меню приложения (отдельная кнопка снизу экрана или три точки вверху).
2. Контекстное меню (если нажать и держать на элементе);

Пойдём по порядку. В некоторых телефонах android всё ещё осталась кнопка вызова меню внизу экрана, однако, большая часть перешла на новый тип отображения приложения (меню вынесено в верхнюю строку, где содержится название приложения).

В первую очередь создадим новую папку в директории res. Называться она будет menu. Сделаем это для того, чтобы мы могла использовать не один шаблон меню, а несколько.

Рисунок 1 – Создание новой директории

Теперь добавим разметку для меню. Для этого создадим xml-файл в только что созданной папке и назовём его main_menu.

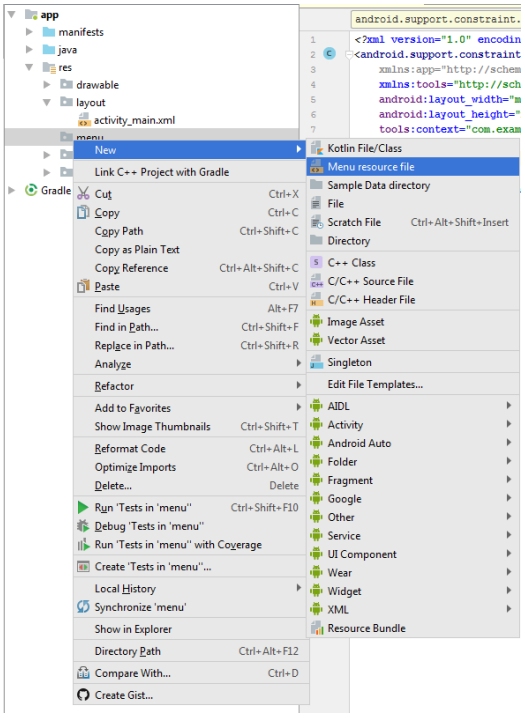


Рисунок 2 – Добавление файла main_menu

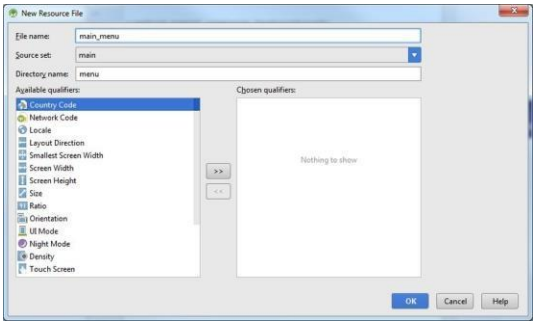


Рисунок 3 – Установка названия

По умолчанию это пустой файл с корневым тегом – menu.

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

</menu>
```

Рисунок 4 – Начальное содержимое меню

Чтобы добавить элементы списка к нему используются теги <item>.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:title="Настройки"></item>
    <item android:title="Второй пункт"></item>
    <item android:title="Домой"></item>
</menu>
```

Рисунок 5 – Добавление пунктов меню

Теперь необходимо сказать приложению, чтобы оно использовало данное меню во время нажатия соответствующей кнопки. Для этого в MainActivity.java переопределяем метод onCreateOptionsMenu().

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_menu, menu);
    return super.onCreateOptionsMenu(menu);
}
```

Рисунок 6 – Регистрация меню

Вот и всё, меню готово. Однако, нажатие на пункты ничего не обрабатывают, и поэтому пока что оно бессмысленно. Добавим события на каждый из пунктов.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.preferences:
            Toast.makeText( context: this,  text: "Выбраны настройки", Toast.LENGTH_SHORT).show();
            return true;
        case R.id.second:
            Toast.makeText( context: this,  text: "Выбран второй пункт", Toast.LENGTH_SHORT).show();
            return true;
        case R.id.home:
            Toast.makeText( context: this,  text: "Вы хотите домой", Toast.LENGTH_SHORT).show();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Рисунок 7 – Обработка выбора пунктов главного меню

Если выбран один из пунктов меню, то мы должны вернуть в конечном итоге true, чтобы сказать, что всё выполнено успешно. Если же ни один из пунктов меню не был нажат, то вызываем родительский метод onOptionItemSelected(), который просто закроет меню.

Переходим ко второму пункту. Контекстное меню вызывается при длительном нажатии на элемент экрана. Чтобы обработать это событие,

необходимо перегрузить метод onCreateContextMenu() в файле MainActivity.java.

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {

}
```

Рисунок 8 – Создание контекстного меню

Чтобы не писать эту часть вручную, можно нажать сочетание клавиш ctrl + O, и выбрать из списка требуемый метод. Для быстрой навигации, среда проиндексировала все методы, и чтобы быстро найти нужный метод достаточно вводить первые буквы из названия метода (например, onCreateContextMenu - occm).

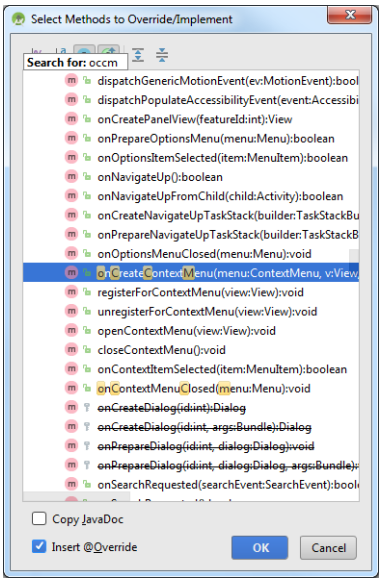


Рисунок 9 – Добавление перегрузки

Самим будет являться объект ContextMenu, который приходит в качестве параметра. Отобразить пункты меню можно двумя способами

1. Если меню статическое, то мы можем просто создать его вручную, создав файл в папке res/menu.

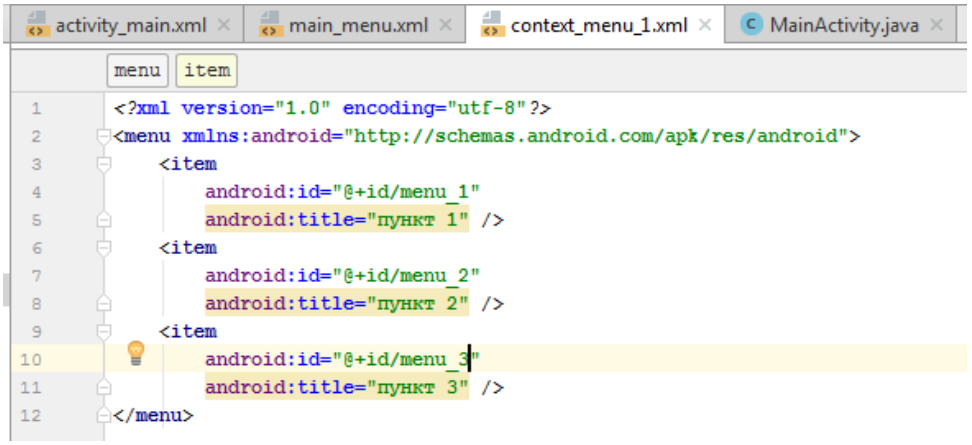


Рисунок 10 – Создание файла контекстного меню

И перегружаем метод для отображения меню.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    findViewById(R.id.text).setOnCreateContextMenuListener(this);

}

@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {
    getMenuInflater().inflate(R.menu.context_menu_1, menu);
}
```

Рисунок 11 – Регистрация контекстного меню

По умолчанию создан TextView в разметке activity_main.xml. Добавим ему идентификатор и навесим контекстное меню. Естественно, методы onCreateContextMenu и onOptionsItemSelected могут быть созданые как и слушатели onClick. Здесь же мы подразумеваем, что все элементы будут обладать одинаковыми характеристиками.

2. Если же меню планируется быть динамическим, то можно пополнять список пунктов вручную.

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {
    menu.add( i: 1,  i1: 1,  i2: 0,  charSequence: "Red");
    menu.add( i: 2,  i1: 2,  i2: 2,  charSequence: "Green");
    menu.add( i: 2,  i1: 3,  i2: 1,  charSequence: "Blue");
}
```

Рисунок 12 – Метод при открытии контекстного меню

Первым параметром передаётся идентификатор группы, которой принадлежит пункт меню (например, чтобы скрыть или отобразить некоторые связанные пункты, их можно пометить в одну группу и скрывать/отображать по нажатию определённой кнопки). Второй параметр – идентификатор пункта в рамках данного меню. Третий – Порядок по списку, то есть самым верхним будет "Red", затем "Blue" и наконец "Green". Последний параметр –это надпись, которая будет отображаться.

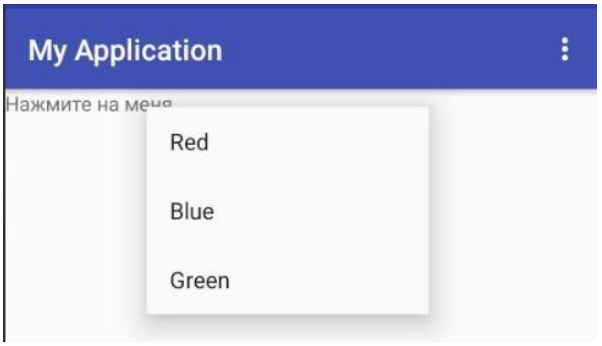


Рисунок 13 – Результат вызова контекстного меню

Обработка происходит в отличном от главного меню методе.

```
@Override
public boolean onContextItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case 1:
            Toast.makeText( context: this,  text: "Выбран пункт 1", Toast.LENGTH_SHORT).show();
            ((TextView)findViewById(R.id.text)).setTextColor(Color.RED);
            return true;
        case 2:
            Toast.makeText( context: this,  text: "Выбран пункт 2", Toast.LENGTH_SHORT).show();
            ((TextView)findViewById(R.id.text)).setTextColor(Color.GREEN);
            return true;
        case 3:
            Toast.makeText( context: this,  text: "Выбран пункт 3", Toast.LENGTH_SHORT).show();
            ((TextView)findViewById(R.id.text)).setTextColor(Color.BLUE);
            return true;
        default:
            return super.onContextItemSelected(item);

    }
}
```

Рисунок 14 – Обработка выбора пункта из контекстного меню

Результат нажатия на "Red" представлен на рисунке ниже.

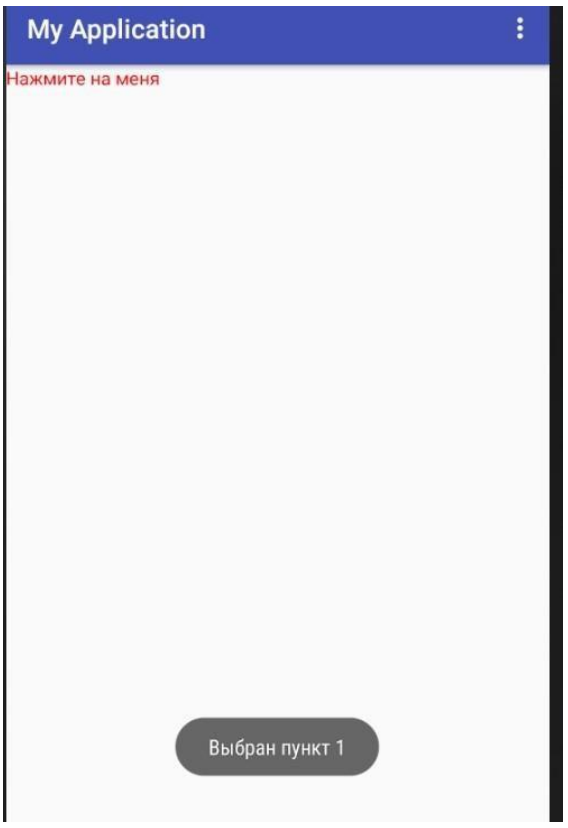


Рисунок 15 – Результат выбора пункта

Задание: для выполнения лабораторной работы необходимо создать два типа меню, согласно варианту.

Варианты:

1. В главном меню 3 пункта: смена типа фигур (круг, овал, кольцо) в ImageView. Контекстное меню на элементах: "вернуть" и "повторить", действие для этого элемента.
2. В главном меню 3 пункта: Изменение размеров круга (маленький, средний, большой). Контекстное меню на элементе: "переместить влево", "переместить вправо".
3. В главном меню 3 пункта: Изменение цвета фона. Контекстное меню: добавить фигуру ("квадрат", "круг").
4. В главном меню 3 пункта: Изменение цвета текста в TextView, где отображается число. Контекстное меню текста: "увеличить на 10" и "уменьшить на 10" число, записанное в TextView.

5. В главном меню 3 пункта: Изменение размера текста TextView. Контекстное меню: "переместить ниже", "переместить выше" TextView.

6. В главном меню 2 пункта: Изменение количества прямоугольников ("плюс 1", "минус 1"). Контекстное меню на элементах: "сменить на красный цвет" и "сменить на чёрный цвет".

7. В главном меню 2 пункта: Изменение количества кругов ("плюс 1", "минус 1"). Контекстное меню на элементах: "Увеличить размер" и "уменьшить размер".

8. В главном меню 3 пункта: Изменение размеров игрового поля (3x3, 4x4, 5x5), состоящего из квадратов. Контекстное меню на элементах: "Удалить элемент", "сменить цвет элементу".

9. В главном меню 3 пункта: Смена стиля кнопок (не менее трех разных стилей). Контекстное меню на элементах: "вернуть" и "повторить" для кнопки.

10. В главном меню 3 пункта: Смена изображения в ImageView. Контекстное меню на элементах: "повернуть по часовой стрелке", "повернуть против часовой стрелки".

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. Меню.
2. Обработка выбора пункта меню.
3. Перегрузка.

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 №10. Разрешения

Цель работы: Изучить методы использования аппаратных возможностей устройства.

Формируемые компетенции: ОПК-2, ОПК-6, ПК-2

Теоретическая часть

Итак, мы рассмотрели банальное создание приложений. Переходим к более сложным приложениям. Чаще всего, приложение не ограничивается своими данными и ему приходится получать данные из вне, то есть из самого телефона. Также, приложение может контролировать некоторые функции телефона (например, включение Bluetooth, wi-fi). Чтобы совершить данное действие, приложение запрашивает разрешения на то, чтобы выполнять ту или иную функцию.

Всего существует более 150 разрешений, которые вряд ли понадобятся Вам. Тем не менее, основные разрешения мы рассмотрим в данной лабораторной работе.

В первой лабораторной работе Вы могли использовать файл AndroidManifest, чтобы изменить имя своего приложения на новое. Помимо простого изменения имени, AndroidManifest содержит в себе всю информацию о вашем приложении (название, разрешения, процессы, минимальные уровни API и тд).

Нас интересуют разрешения. Разрешения имеют следующий синтаксис:

```
<uses-permission android:name=""></uses-permission>
```

В кавычках выбирается наименование разрешения из списка доступных. Например, если мы хотим, чтобы приложение могло получить доступ в интернет, то вставляем следующую строчку:

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

    <uses-permission android:name="android.permission.INTERNET"></uses-permission>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Permissions"
        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>
    </application>

</manifest>
```

Рисунок 1 – пример объявления разрешения

Расположение этой строки очень важно. Разрешения прописываются перед свойством приложения.

Наиболее используемые разрешения представлены ниже:

- android.permission.INTERNET;
- android.permission.ACCESS_NETWORK_STATE – разрешает приложению получить информацию о текущем состоянии подключения к интернету (вай-фай или данные);
- android.permission.ACCESS_NOTIFICATION_POLICY – предоставляет приложению возможность отправлять уведомления пользователю;
- android.permission.ACCESS_WIFI_STATE – возможность получить информацию о текущем подключению к вай-фай;
- android.permission.ANSWER_PHONE_CALLS – предоставляет возможность отвечать на входящие звонки, поступающие на телефон;
- android.permission.BATTERY_STATS – предоставляет информацию о батарее;
- android.permission.BLUETOOTH – позволяет включать / выключать bluetooth;

- android.permission.BROADCAST_PACKAGE_REMOVED. В андроид есть возможность отправлять информацию о том, что устанавливается или удаляется какое-либо приложение. Для этого используется Broadcast – система, которая получает и отправляет сообщения такого рода. Данное разрешение позволяет получать такие сообщения в приложении;

- android.permission.CALL_PHONE – позволяет совершать и принимать вызовы без отображения стандартного диалога;

- android.permission.CALL_PRIVILLEGED – позволяет также совершать вызовы по экстренным номерам (112, 911);

- android.permission.CAMERA – разрешает использовать все камеры на устройстве;

- android.permission.CAPTURE_AUDIO_OUTPUT – позволяет перехватывать исходящие аудиопотоки для обработки внутри приложения;

- android.permission.CAPTURE_VIDEO_OUTPUT – то же самое, но для видео;

- android.permission.CHANGE_NETWORK_STATE – позволяет менять состояние сети подключения данных (вай-фай меняется соответственно CHANGE_WIFI_STATE);

- android.permission.CLEAR_APP_CACHE. Приложения автоматически создают кэш самих себя, чтобы не загружать несколько раз одни и те же данные, что позволяет ускорить запуск. Однако, это часто сильно нагружает постоянную память устройства. Данное разрешение позволяет очистить кэш вручную, но не только себя, но и всех приложений на устройстве;

- android.permission.GET_ACCOUNTS – предоставляет доступ ко всем аккаунтам, доступным на устройстве;

- android.permission.MANAGE_DOCUMENTS – предоставляет разрешение изменять документы, находящиеся на устройстве (работает

только в связке с возможностью читать данные с карты памяти или устройства);

- android.permission.NFC – позволяет использовать NFC-модуль;
- android.permission.READ_CALENDAR – просмотр календаря;
- android.permission.READ(_CALL_LOG, _CONTACTS, _EXTERNAL_STORAGE, _HISTORY_BOOKMARK, _SMS, _VOICEMAIL) – просмотр журнала вызовов, контактов, карты памяти, закладок, смс и голосовой почты;
- android.permission.REBOOT – возможность перезагрузить телефон;
- android.permission.RECEIVE_BOOT_COMPLETE – получает информацию, загрузился ли телефон полность после выключения (передаётся через broadcast);
- android.permission.RECEIVE_SMS (.MMS) – получает уведомление, когда приходит смс (ммс);
- android.permission.RECORD_AUDIO – позволяет записывать аудио;
- android.permission.SEND_SMS – позволяет отправлять смс-сообщения;
- android.permission.SET_ORIENTATION – возможность менять ориентацию экрана;
- android.permission.SET_TIME – позволяет устанавливать системное время на устройстве;
- android.permission.SET_WALLPAPER – позволяет устанавливать обои рабочего стола;
- android.permission.USE_FINGERPRINT – использует сканер отпечатка пальцев;
- android.permission.USE_SIP – позволяет получить доступу к SIP-службе;
- android.permission.VIBRATE – доступ к вибрации телефона;

- android.permission.WRITE_(CALL_LOG, CONTACTS, EXTERNAL_STORAGE, GSERVICES, HISTORY_BOOKMARK, SMS, VOICEMAIL) – позволяет менять уже существующие данные на устройстве (контакты, внешние файлы, сервисы гугл, закладки, смс, голосовую почту).

Чтобы создать уведомление используется следующий код:

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

    NotificationCompat.Builder mbuilder = new NotificationCompat.Builder(getApplicationContext(), channelId: "ch-1")
            .setContentTitle("Название")
            .setContentText("Содержимое уведомления")
            .setSmallIcon(R.drawable.ic_launcher_background);
    NotificationManager mgr = (NotificationManager)getApplicationContext().getSystemService(Context.NOTIFICATION_SERVICE);
    mgr.notify( id: 1, mbuilder.build());

}
```

Рисунок 2 – Создание уведомления

Если же версия Вашего андроид-устройства выше 8, то необходимо изменить процесс создания. Необходимо сперва создать канал, куда будут поступать сообщения, а уже только потом его использовать:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    CharSequence name = "name";
    String description = "Description";
    int importance = NotificationManager.IMPORTANCE_DEFAULT;
    NotificationChannel channel = new NotificationChannel( id: "ch-1", name, importance);
    channel.setDescription(description);
    NotificationManager notificationManager = getSystemService(NotificationManager.class);
    notificationManager.createNotificationChannel(channel);
}
```

Рисунок 3 – Регистрация канала

При любом использовании кода в результате будет выведено сообщение:

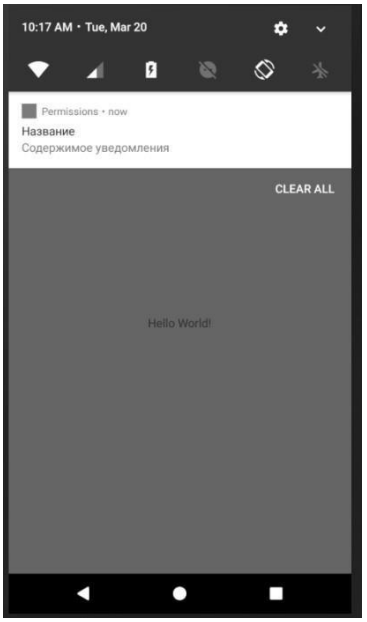


Рисунок 4 – отображение уведомления на устройстве

Чтобы активировать вибрацию телефона используется класс Vibrator. Пример его использования представлен на рисунке 4.

```
Vibrator v = (Vibrator)getSystemService(Context.VIBRATOR_SERVICE);
v.vibrate( milliseconds: 400);
```

Рисунок 5 – Пример генерирования вибрации устройством

Иногда нужно, чтобы вибрация работала постоянно, то есть, не один раз срабатывала, а в течение некоторого промежутка времени. Для этого используется так называемый паттерн. Мы должны создать массив из элементов, где сперва идёт промежуток, спустя который необходимо начать вибрацию, затем попарно пауза и продолжительность вибрации. А второй параметр в методе vibrate() указывает, с какого индекса необходимо начать бесконечный цикл. Например, чтобы вибрация не останавливалась никогда, используйте код на рис. 5.

```
Vibrator v = (Vibrator)getSystemService(Context.VIBRATOR_SERVICE);
long[] p = new long[]{0, 1000, 0};
v.vibrate(p,  repeat: 0);
v.cancel();
```

Рисунок 6 – пример бесконечной вибрации

Следующий пример заставляет вибрацию срабатывать в следующем порядке:

1. С задержкой 50 миллисекунд произвести вибрацию длительностью 100 мс,
2. Подождать 1 секунду, вибрировать 300 мс.
3. Подождать 2 с, вибрировать пол секунды.
4. Дальше идёт условно бесконечный цикл, где будет чередоваться 2 и 3 шаг.

```
Vibrator v = (Vibrator)getSystemService(Context.VIBRATOR_SERVICE);
long[] p = new long[]{50, 100, 1000, 300, 2000, 500};
v.vibrate(p,  repeat: 0);
```

Рисунок 7 – второй пример бесконечной вибрации

Чтобы открыть камеру и посмотреть, что она снимает используются следующие компоненты:

```
<SurfaceView
    android:id="@+id/surfaceView"
    android:layout_width="200dp"
    android:layout_height="150dp" />

<Button
    android:id="@+id/button2"
    android:layout_width="100dp"
    android:layout_height="60dp"
    android:layout_below="@+id/surfaceView"
    android:onClick="openCamera"
    android:text="Открыть" />

<Button

    android:layout_width="100dp"
    android:layout_height="60dp"
    android:layout_below="@+id/surfaceView"
    android:layout_toEndOf="@+id/button2"
    android:layout_toRightOf="@+id/button2"
    android:onClick="closeCamera"
    android:text="Закрыть" />
```

Рисунок 8 – activity_main для камеры

```
public void openCamera(View view) {
    camera = Camera.open();
    SurfaceView sf = findViewById(R.id.surfaceView);
    SurfaceHolder sh = sf.getHolder();
    try {
        camera.setPreviewDisplay(sh);
    } catch (IOException e) {
        e.printStackTrace();
    }
    camera.startPreview();
}

public void closeCamera(View view) {
    camera.release();
}
```

Рисунок 9 – main_activity.java для камеры

SurfaceView, по сути, является представлением для рисования. Например, то, чем мы занимались на первой лабораторной можно было добавить в SurfaceView. Но в данном случае мы его используем, чтобы передавать изображение с камеры на наш View.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 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 создать собственное приложение согласно выбранному варианту (вариант выбирается с преподавателем).

Варианты:

1. Планировщик заданий с уведомлением пользователя о событии.
2. Планировщик заданий с уведомлением о событии путём вибрации.
3. Приложение, которое будет выключать/включать bluetooth по расписанию.
4. Приложение, которое будет выключать/включать wi-fi по расписанию.
5. Открытие камеры и сохранение фото на флешке.
6. Приложение для добавления событий в существующий календарь и Вывод всех записей из него
7. Приложение, которое перезагружает устройство в соответствии с графиком.
8. Приложение-помощник для парикмахера.
9. Приложение-помощник для продавца.
10. Приложение-помощник для преподавателя.

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. Резрешения.
2. Уведомления.
3. Камера.

Список литературы, рекомендуемый к использованию по данной теме:

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

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 11. Работа с потоками.

Цель работы: научиться распределять задачи между потоками.

Формируемые компетенции: ОПК-2, ОПК-6, ПК-2

Теоретическая часть:

Каждый процессор может обрабатывать несколько потоков одновременно. Приложения, написанные ранее, выполнялись в одном потоке, который называется главным. Все стандартные вычисления выполняются процессором в главном потоке. Также, этот поток отвечает за обновление пользовательского интерфейса.

Неправильная работа с этим потоком может привести к тому, что приложение станет "не отвечать". Это частая проблема, которую начинающие разработчики не берут во внимание.

Чтобы избежать данной проблемы, существует подход, названный "асинхронное программирование". Данный метод позволяет решать все малозначимые задачи в отдельном потоке, не загружая основной, и не "замораживая" интерфейс.

Естественно, в андроид такая возможность присутствует. Делается это с помощью генерации нового потока Thread.

```
Thread threadExample =  new Thread(new Runnable() {
    @Override
    public void run() {
    }
});
```

Рисунок 1 – Генерация нового потока

В переопределённом методе run() необходимо записать код, который будет выполняться в отдельном потоке.

Рассмотрим простой пример, который загружает изображение из интернета и помещает в ImageView. Для этого выполним следующие шаги:

1. Создадим кнопку, которая будет загружать картинку:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Нажми"
    android:onClick="btnClick"
    android:id="@+id/button" />
```

Рисунок 2 – Создание кнопки

2. Создадим ImageView, который будет отображать изображение:

```
<ImageView
    android:id="@+id/imgView"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_below="@+id/button" />
```

Рисунок 3 – Создание ImageView

3. Чтобы приложение могло воспользоваться интернетом, необходимо добавить соответствующее разрешение в AndroidManifest.xml. Что такое разрешения и как они работают разберёмся в другой лабораторной работе.

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Threads"
    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>
</application>
```

Рисунок 4 – Добавление разрешения

4. Создадим отдельно метод для загрузки изображения и метод для помещения его в ImageView.

Запрос на загрузку файла подразумевает обработку входного потока, который мы получаем с помощью метода getContent(). С помощью decodeStream() мы распознаём поток байт в изображение формата bmp.

```
private Bitmap loadImageFromNetwork(String url){
    try {
        Bitmap bitmap = BitmapFactory.decodeStream((InputStream)new URL(url).getContent());
        return bitmap;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

Рисунок 5 – Метод для загрузки изображения из интернета

Возьмём в качестве примера логотип СКФУ, который доступен для скачивания на официальном сайте по следующему URL-адресу: http://www.ncfu.ru/templates/current/images/logotype.png

```
public void btnClick(View view) {
    Thread threadExample =  new Thread(new Runnable() {
        @Override
        public void run() {
            final Bitmap bmp = loadImageFromNetwork( url: "http://www.ncfu.ru/templates/current/images/logotype.png");
            final ImageView imgView = (ImageView)findViewById(R.id.imgView);
            imgView.post(new Runnable() {
                @Override
                public void run() {
                    imgView.setImageBitmap(bmp);
                }
            });
        }
    });
}
```

Рисунок 6 – Вызов метода загрузки

Определяя переменные константами (final) мы запрещаем возможные изменения объектов в процессе обработки метода.

Метод post() позволяет асинхронно изменить содержимое нашего ImageView. Таким образом, мы создали действие, которое будет выполняться в отдельном потоке (Thread). Чтобы заставить выполнить этот код, требуется вызвать метод start() у threadExample.

```
public void btnClick(View view) {
    Thread threadExample =  new Thread(new Runnable() {
        @Override
        public void run() {
            final Bitmap bmp = downloadImage( url: "http://www.ncfu.ru/templates/current/images/logotype.png");
            final ImageView imgView = (ImageView)findViewById(R.id.imgView);
            imgView.post(new Runnable() {
                @Override
                public void run() {
                    imgView.setImageBitmap(bmp);
                }
            });
        }
    });
    threadExample.start();
}
```

Рисунок 7 – Запуск потока

Второй способ использования параллельного потока более приближен к реалиям современной разработки ПО. AsyncTask – именно этот класс мы будем расширять создания нового потока.

```
private class DownloadTask extends AsyncTask<String, Void, Bitmap> {

    @Override
    protected void onPreExecute() {

    }

    @Override
    protected Bitmap doInBackground(String... strings) {
        return downloadImage(strings[0]);
    }

    @Override
    protected void onPostExecute(Bitmap bmp) {
        imgView.setImageBitmap(bmp);
    }
}
```

Рисунок 8 – Пример создания асинхронного задания

Наследуясь от класса AsyncTask необходимо переопределить следующие методы:

onPreExecute() – метод, который выполняется перед запуском основной задачи. Здесь можно изменить интерфейс, чтобы показать, что скоро начнётся выполняться асинхронный метод (например, запустить прогрессбар).

doInBackground() – в этом методе будет выполняться код в отдельном потоке. Следует отметить, что здесь нельзя трогать интерфейс, иначе, Вы

рискуете снова нарваться на "Приложение не отвечает". Результат выполнения этого метода будет передан в onPostExecute().

onProgressUpdate() – метод, который позволяет вручную устанавливать прогресс выполнения метода doInBackground().

onPostExecute() – после выполнения метод doInBackground() выполнится код, написанный здесь. Здесь можно снова обновить интерфейс.

Если Вам не нужен какой-нибудь метод, то можете просто не переопределять его. Параметры AsyncTask тоже подобраны не случайно. Первый тип – показывает тип параметра, который будет передан в метод doInBackground() посредством execute(). Второй тип – передаваемый тип в onProgressUpdate(). И последний – Тип параметра, переданного по окончании doInBackground и полученного в onPostExecute().

Воспользуемся той же структурой xml-файла и заменим лишь метод onClick().

```
ImageView imgView;
public void onClick(View v) {
    imgView = findViewById(R.id.imgView);
    DownloadTask task = new DownloadTask();
    task.execute("http://www.ncfu.ru/templates/current/images/logotype.png");
}
```

Рисунок 9 – Выполнение задания

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.

Задания: для выполнения лабораторной работы необходимо выполнить следующее:

1. . Рассчитать результат и отобразить его в интерфейсе.
2. . Загрузить несколько любых изображений из интернета с отображением строки состояния (ProgressBar).

Варианты для задания 1:

1. В одномерном массиве, состоящем из n вещественных элементов вычислить:
   a. Сумму отрицательных элементов массива
   b. Произведение элементов массива, расположенных между максимальным и минимальным элементом этого массива.
2. В одномерном массиве, состоящем из n вещественных элементов вычислить:
   a. Сумму положительных элементов массива.
   b. Произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементом этого массива.
3. В одномерном массиве, состоящем из n целых элементов вычислить:
   a. Произведение элементов массива с четными индексами.
   b. Сумму элементов массива, расположенных между первым и последним нулевыми элементами.
4. В одномерном массиве, состоящем из n вещественных элементов вычислить:
   a. Сумму элементов с нечетными индексами.
   b. Сумму элементов массива, расположенных между первым и последним отрицательными элементами.
5. В одномерном массиве, состоящем из n вещественных элементов вычислить:
   a. Максимальный элемент массива.
   b. Сумму элементов массива, расположенных до последнего положительного элемента.
6. В одномерном массиве, состоящем из n вещественных элементов вычислить:
   a. Минимальный элемент массива

b. Сумму элементов массива, расположенных между первым и последним положительными элементами.

7. В одномерном массиве, состоящем из n вещественных элементов вычислить:

a. Номер максимального элемента массива.

b. Произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

8. В одномерном массиве, состоящем из n вещественных элементов вычислить:

a. Номер минимального элемента массива.

b. Сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

9. В одномерном массиве, состоящем из n вещественных элементов вычислить:

a. Максимальный по модулю элемент массива.

b. Сумму элементов массива, расположенных между первым и вторым положительными элементами.

10. В одномерном массиве, состоящем из n вещественных элементов вычислить:

a. Минимальный по модулю элемент массива.

b. Сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

11. В одномерном массиве, состоящем из n вещественных элементов вычислить:

a. Номер минимального по модулю элемента массива.

b. Сумму модулей элементов массива, расположенных после первого отрицательного элемента.

12. В одномерном массиве, состоящем из n вещественных элементов вычислить:

a. Номер максимального по модулю элемента массива.

b. Сумму элементов массива, расположенных после первого положительного элемента.

13. В одномерном массиве, состоящем из n вещественных элементов вычислить:

a. Количество элементов, лежащих в диапазоне от A до B.

b. Сумму элементов массива, расположенных после максимального элемента.

14. В одномерном массиве, состоящем из n вещественных элементов вычислить:

a. Количество элементов массива, равных нулю.

b. Сумму элементов массива, расположенных после минимального элемента.

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. Поток.
2. Task.

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 12. Виды представлений.

Цель работы: Научиться добавлять различные виды активностей.

Формируемые компетенции: ОПК-2, ОПК-6, ПК-2

Теоретическая часть:

При создании новых активностей в мобильном приложении есть несколько вариантов выбора. До этого мы создавали пустые активности, на которых был только один элемент TextView. Если нам требовалось что-то ещё, то мы помещали эти компоненты в разметку вручную. Тем не менее, Android Studio поддерживает возможность использовать несколько шаблонов активностей, которые ускоряют процесс разработки. Рассмотрим каждый из них.

1. Empty Activity – простая пустая активность, которую мы создавали всегда.

2. Basic Activity. Базовая активность, которая содержит несколько компонентов:

   a. Частичное представление content_main. Этот компонент добавлен с помощью тэга <include>. Про него мы говорили в 5 лабораторной работе.

   b. Компонент AppBarLayout. Верхняя панель приложения в большинстве случаев отвечает за навигацию. В ней находится название окна, где сейчас находится пользователь, кнопка меню и кнопка «назад», если с него можно вернуться на предыдущее окно. Про само меню и его пункты мы уже говорили в лабораторной работе 9.

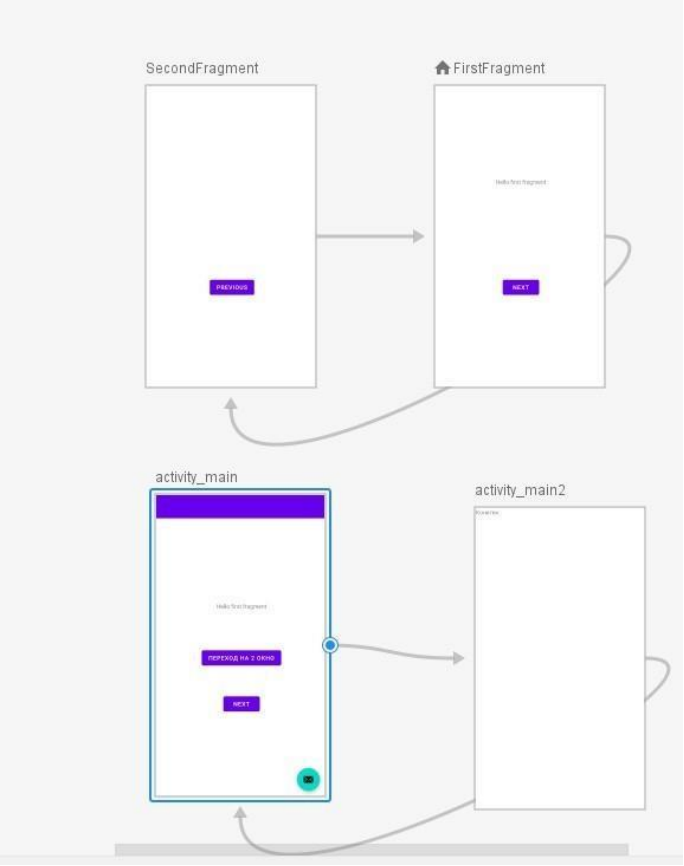
   c. Кнопка перехода на второе окно.

   d. Плавающая кнопка, нажатие на которую происходит некоторое действие, которое мы захотим.

В данной активности присутствует ещё один компонент, который называется «Фрагмент», Фрагмент можно тоже считать частичным представлением. Однако, у него есть свои особенности. Безусловно, у него

тоже есть возможность взаимодействовать с другими объектами и активностями. Но, главная его задача заключается в том, чтобы обеспечить более рациональное распределение компонентов в активности на разных экранах. Например, в портретной ориентации список и TextView из лабораторной работы 7 смотрелись удобно один под другим. Однако, если перевернуть телефон, то справа от списка будет огромная пустота. Чтобы это компенсировать, фрагменты можно располагать как угодно на странице в зависимости от положения телефона. Неплохой пример Вы можете рассмотреть по ссылке.

Также, в данной активности добавили такую вещь, как Navigation Architecture Component. Он позволяет по-другому осуществить переходы между активностями. Более того, этот компонент позволяет визуализировать переходы с помощью специального графа:



Вы можете использовать его, а можете и нет. На процесс работы приложения это никак не влияет. Однако, по мере разрастания Вашего проекта и увеличения числа активностей, можно запутаться. Данный

компонент позволяет визуально рассмотреть, какие зависимости есть между компонентами: из какой активности/фрагмента можно перейти в другое место.

3. . Bottom Navigation Activity. Очень удобная активность. В неё навигационная панель уже располагается внизу. Каждый из элементов меню

– это отдельный фрагмент, который можно увидеть, выбрав определённый пункт из панели. Безусловно, вместо фрагментов можно создать свои активности (в таком случае, правильным подходом будет использование такого же шаблона для каждой из них).

4. . FullscreenActivity – активность, которая позволяет отобразить содержимое в полном окне. При таком подходе контент будет растянут на весь экран, однако, при любом прикосновении можно увидеть снова верхнюю панель приложения и строку состояния. Все настройки, связанные с анимацией полного экрана можно указать в файле .java. На самом деле любую активность можно сделать полноэкранной. Но здесь все настройки уже указаны и нам остаётся только подстроить их под себя.

5. . Login Activity. Данная активность при создании совмещает в себе сразу несколько основных компонентов, отвечающих за реализацию страницы с логином/регистрацией. Он включает в себя:

a. ViewModel – модель, имеющая основные свойства, с которыми мы будем работать. Например, LoginRepository (хранит список всех логинов в приложении), LoginResult (возвращает информацию, успешно или нет пользователь ввёл данные), LoginFormState (хранит информацию об ошибках пользователя, например, неправильная пара логин/пароль).

b. Слушатели на ввод данных через EditText (чтобы подсвечивать ошибки пользователю).

c. Отображение Toast об успешной/безуспешной авторизации.

d. Кнопка логин, запускающая процесс поиска/регистрации пользователя.

6. Navigation Drawer Activity. Одна из самых распространённых активностей. Данная активность включает в себя компонент навигации, то есть боковое меню. NavigationView – тот самый компонент. Скопировав его и поместив его на любую свою страницу Вы сможете открыть его смахнув слева. Обработку выбора пунктов меню Вы уже делали на лабораторной 9.

7. ScrollingActivity – активность, которая поддерживает отображение большого содержимого. Любой компонент в xml можно заставить прокручиваться. Но сделать это можно только поместив его в контейнер ScrollView. Scrolling Activity же позволяет сразу всё содержимое страницы заставлять прокручиваться. Причём шапка активности будет анимированно прятаться.

8. Settings Activity. Когда для Вашего приложения предполагаются некоторые настройки, который пользователь выбирает для себя, их выносят в отдельную активность. Самым простым вариантом создания такой активности это Setting Activity. Изменив содержимое на те компоненты, которые будут использоваться у Вас, можно за несколько минут оформить вполне приличную страницу с настройками.

В связи с существованием такой страницы рассмотрим ещё кое-что. Все настройки, сохранённые пользователем должны быть спрятаны где-то в телефоне так, чтобы при следующем открытии можно было их получить. По умолчанию, при открытии приложения заново всё, что не записалось в БД или файлы (Кэш) автоматически очищается телефоном. Здесь мы не можем допустить подобное. Для решения данной задачи используется встроенный компонент, называемый SharedPreferences.

Данный класс позволяет хранить данные следующих типов:

- Boolean (true/false);
- Float (Число с плавающей точкой);
- Int (Целое число);
- Long(Целое число, но в два раза больше Int)
- String (Строка).

Для записи данных используются методы, начинающиеся с put (как в SQLite), например, putBoolean(“ключ”, false). В качестве ключа должно использоваться уникальное слово/фраза. Ключи не должны повторяться. Чтобы получить значение, можно воспользоваться методом, начинающимся с get (например, getBoolean(“ключ”, false), где вторым параметром передаётся значение по умолчанию. Значение по умолчанию позволяет гарантировать, что, если такого ключа нет в настройках, то мы хоть что-то, но получим).

Например, на странице настроек добавим следующий код:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.settings_activity);
    if (savedInstanceState == null) {
        getSupportFragmentManager() FragmentManager
                .beginTransaction() FragmentTransaction
                .replace(R.id.settings, new SettingsFragment())
                .commit();
    }
    ActionBar actionBar = getSupportActionBar();
    if (actionBar != null) {
        actionBar.setDisplayHomeAsUpEnabled(true);
    }

    SharedPreferences pref = getApplicationContext().getSharedPreferences( name: "MyPref",  mode: 0);
    SharedPreferences.Editor editor = pref.edit();
    editor.putInt("UniqueKey", 145632);
    editor.commit();

}
```

А в MainActivity:

```
public void getValue(View v){
    SharedPreferences pref = getApplicationContext().getSharedPreferences( name: "MyPref",  mode: 0);
    Toast.makeText( context: this,  text: pref.getInt( key: "UniqueKey",  defValue: 1) + "", Toast.LENGTH_SHORT).show();
}
```

Добавив кнопку на главной странице и навесив на неё это событие, мы сможем увидеть, как значение сохраняется:

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Получить значение"
        android:onClick="getValue"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Открыть настройки"
        android:onClick="openSecond"/>
</LinearLayout>
```

Если закрыть и открыть приложение заново, то данные сохранятся.

9. Tabbed Activity. По своему внешнему виду напоминает Bottom Navigation Activity. Отличается тем, что содержимое вкладок может быть только разметка или фрагмент, а в BTN – это отдельные активности.

10. Google AdMob Activity – активность для настройки рекламного отображения. Не будет использовать в данном курсе.

11. Google Maps Activity – активность для отображения содержимого из Google Maps.

В лабораторной работе 5 мы узнали, какими способами можно добавлять новые активности. Естественно, с каждой из них работает такой же подход. Шаблон лишь позволяет ускорить этот процесс, создавая все необходимые файлы.

Задания:

1. Создать приложение с минимум 3 видами окон.
2. Все последующие лабораторные работы будут нацелены на создание одного полноценного игрового приложения.
3. В соответствии с выбранным вариантом создайте шаблоны активностей, и настройте переходы между ними.

Варианты для игры:

1. Крестики нолики:

a. Главный экран;
b. Экран настроек;
c. Экран игры.

2. Реверси
    a. Главный экран;
    b. Экран игры;
    c. Экран добавления игроков;
    d. Экран с победителями.
3. Шашки
    a. Главный экран;
    b. Экран игры;
    c. Экран добавления игроков;
    d. Экран с победителями.
4. Таблички
5. Морской бой
    a. Главный экран;
    b. Экран добавления кораблей;
    c. Экран игры;
    d. Экран добавления игроков;
    e. Экран с победителями.
6. Гонки
    a. Главный экран;
    b. Экран игры;
    c. Экран с рекордами.
7. Поймай круг
    a. Главный экран;
    b. Экран игры;
    c. Экран с рекордами.
8. Угадай карту
    a. Главный экран;

b. Экран игры;
c. Экран с рекордами.

9. Больше-меньше
    a. Главный экран;
    b. Экран игры;
    c. Экран с рекордами.
10. Сапёр
    a. Главный экран;
    b. Экран игры;
    c. Экран с настройками;
    d. Экран с рекордами.
11. Волк ловит яйца
    a. Главный экран;
    b. Экран игры;
    c. Экран с настройками;
    d. Экран с рекордами.
12. Плитки фортепиано
    a. Главный экран;
    b. Экран игры;
    c. Экран с настройками;
    d. Экран с рекордами.
13. Судоку
    a. Главный экран;
    b. Экран игры;
    c. Экран с настройками;
    d. Экран с рекордами.
14. 10 отличий.
    a. Главный экран;
    b. Экран игры;
    c. Экран с рекордами.

15. Пятнашки
    a. Главный экран;
    b. Экран игры;
    c. Экран с настройками;
    d. Экран с рекордами.
16. Змейка
    a. Главный экран;
    b. Экран игры;
    c. Экран с настройками;
    d. Экран с рекордами.

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

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 13. Обработка жестов.

Цель работы: Научиться добавлять различные виды активностей.

Формируемые компетенции: ОПК-2, ОПК-6, ПК-2

Теоретическая часть:

Обработка жестов является одним из важнейших компонентов при построении мобильных приложений. Однако, это правильно использовать в играх, а не в бизнес-приложениях, которые создаются с помощью Android Studio. Тем не менее, будет полезно узнать, как же можно отслеживать движения пальцем по экрану.

Жесты обрабатываются с помощью слушателей. Подобным образов мы делали обработку нажатия кнопки на лабораторной работе 3. Как Вы помните, мы могли использовать слушателя двумя способами: создав объект класса слушателя или использовать анонимно.

При использовании жестов лучшим вариантом будет создать класс обработчик. Это позволит один раз правильно настроить класс и затем просто использовать то, как мы обработаем движения.

В папке с Java создадим новый класс и назовём его OnSwipeTouchListener:

```
public class OnSwipeTouchListener implements View.OnTouchListener {

}
```

Так же, как и с кликом мы должны использовать перегруженный метод. Здесь он называется onTouch:

```
public class OnSwipeTouchListener implements View.OnTouchListener {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        return false;
    }
}
```

Touch – это простое прикосновение к экрану. Событие Click можно смело заменить на Touch и это будет работать. Однако, наша задача

заключается в том, чтобы обработать не само касание, а некоторое движение пользователя по экрану. Для этого используется уже другой обработчик. Внутри нашего класса создадим вложенный класс (как в лабораторной работе 1), который будет называться GestureListener:

```
public class OnSwipeTouchListener implements View.OnTouchListener {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        return false;
    }

    class GestureListener extends GestureDetector.SimpleOnGestureListener{

    }

}
```

Данный класс будет обрабатывать движения. Сам процесс движения заключается в следующем:

1. Пользователь ставит палец на экран и срабатывает событие onTouch;
2. Пользователь двигает палец по экрану в какую-нибудь сторону;
3. При движении мы можем получить основную информацию, связанную с движением:
    a. Где находился до начала движения;
    b. Где стал находиться после начала движения;
    c. Скорость, с которой двинулся по оси X;
    d. Скорость, с которой двинулся по оси Y.

Добавим использование этих подходов. Во-первых, добавим перегрузку метода onDown (пункт 1):

```
class GestureListener extends GestureDetector.SimpleOnGestureListener{
    @Override
    public boolean onDown(MotionEvent e) {
        return super.onDown(e);
    }
}
```

Следующим шагом определим движение. Перегружаем метод onFling:

```
@Override
public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
    return super.onFling(e1, e2, velocityX, velocityY);
}
```

4 параметра, передаваемые в данный метод – та самая информация из пункта 3 (a, b, c, d).

Напишем простой метод, который будет смотреть, куда двинулся палец. Начнём с начала:

```
float diffX = e2.getX() - e1.getX();
float diffY = e2.getY() - e1.getY();
if (Math.abs(diffX) > Math.abs(diffY)) {

}
```

Решим для начала, по какой оси двигался палец: Если больше по X, то влево или вправо, в противном случае – вверх/вниз.

Самый простой вариант будет работать так: если хотя бы на пиксель сдвинулось прикосновение, то в ту сторону и сработало. Если diffX > 0, то по X смещение ушло в правую сторону, иначе – влево:

```
float diffY = e2.getY() - e1.getY();
if (Math.abs(diffX) > Math.abs(diffY)) {
    if (diffX > 0) {

    } else {

    }
}
```

Добавим два пустых метода, которые мы позже переопределим:

```
public void onSwipeLeft() {
}

public void onSwipeRight() {
}
```

Они должны находиться в классе OnSwipeTouchListener. Поместим их вызовы внутри if/else.

Теперь создадим простую реализацию данного класса. В первом слушателе добавим конструктор:

```
@Override
public boolean onTouch(View v, MotionEvent event) {
    return false;
}

public OnSwipeTouchListener(Context ctx) {
    GestureDetector gestureDetector = new GestureDetector(ctx, new GestureListener());
}
```

Вынесем переменную gestureDetector в качестве свойства:

```
public class OnSwipeTouchListener implements View.OnTouchListener {
    GestureDetector gestureDetector;
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        return false;
    }

    public OnSwipeTouchListener(Context ctx) {
        gestureDetector = new GestureDetector(ctx, new GestureListener());
    }
```

И используем его в методе onTouch:

```
GestureDetector gestureDetector;
@Override
public boolean onTouch(View v, MotionEvent event) {
    return gestureDetector.onTouchEvent(event);
}
```

Теперь идём к нашей активности. Создадим простой квадрат по центру экрана:

```
<ImageView
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:background="@color/black"
    android:id="@+id/imageView"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

В методу onCreate навесим нашего слушателя на движения пальца по imageView:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ImageView imageView = findViewById(R.id.imageView);
    imageView.setOnTouchListener(new OnSwipeTouchListener(getApplicationContext()) {
        public void onSwipeRight() {
            Toast.makeText(getApplicationContext(), text: "right", Toast.LENGTH_SHORT).show();
        }
        public void onSwipeLeft() {
            Toast.makeText(getApplicationContext(), text: "left", Toast.LENGTH_SHORT).show();
        }

    });
}
```

Запустим приложение. Начнём движение с квадрата, двинем немного мышку (палец) влево и отпустим. Должно появиться сообщение, “left”. Проделав то же самое, но вправо – увидим, что появилось сообщение “right”. Подобным образом можем добавить и движения вверх и вниз.

Естественно, можно добавить ограничения. Например, чтобы свайп срабатывал только тогда, когда палец «пройдёт» некоторое количество пикселей. Это позволит избежать случайных движений. Можно отслеживать и скорость движения.

Координаты движения представлены в объектах MotionEvent.

Задание: Согласно заданию, выбранному на предыдущей лабораторной работе реализовать механизм работы обработки движений.

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

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