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 защищенных мобильных
приложений» для студентов направления
10.05.03 «Информационная безопасность автоматизированных систем»
Направленность (профиль)
«Анализ безопасности информационных систем»

Ставрополь

Содержание

ВВЕДЕНИЕ

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

Задачами дисциплины являются:

- ознакомление студентов с видами пакетов прикладных программ и классами программного обеспечения;
- обучение студентов постановке задач, корректному и эффективному использованию инструментальных средств;
- приобретение студентами навыков самостоятельного и последовательного применения пакетов прикладных программ в профессиональной деятельности;
- формирование логического мышления.

Лабораторная работа 1. Ресурсы. Медиа-элементы.

Цель работы: изучить способы отображения и манипулирования внешними файлами.

Формируемые компетенции: ОПК-7

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:

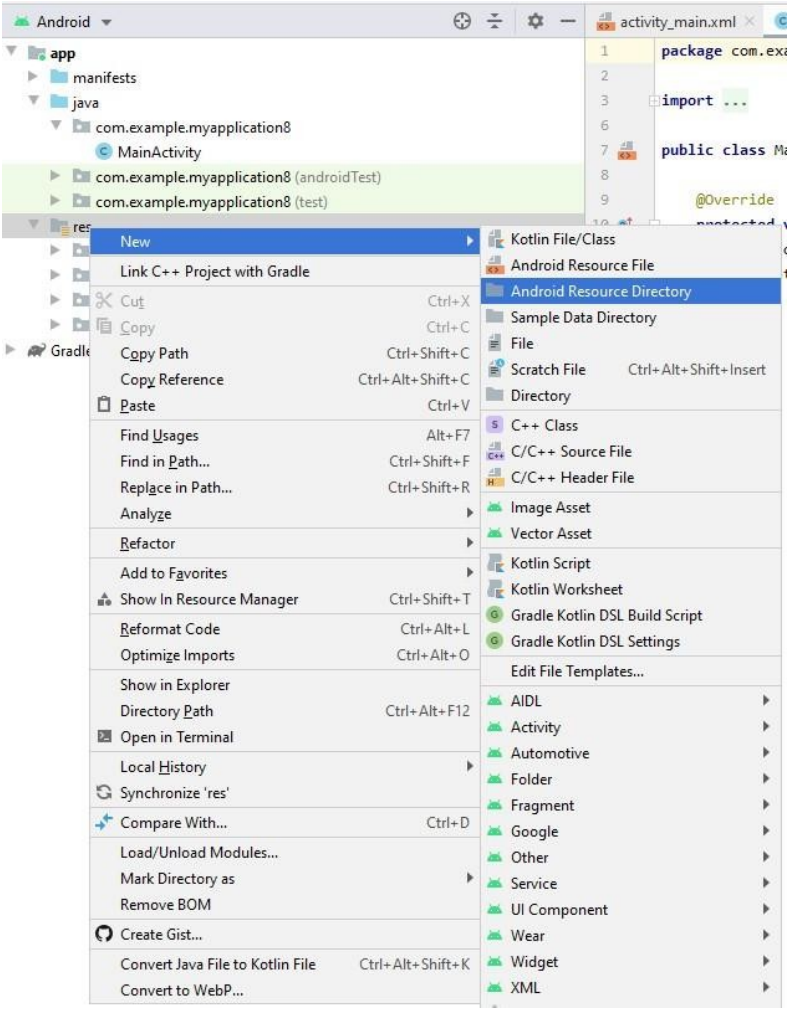


Рисунок 1 – Добавление директории

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

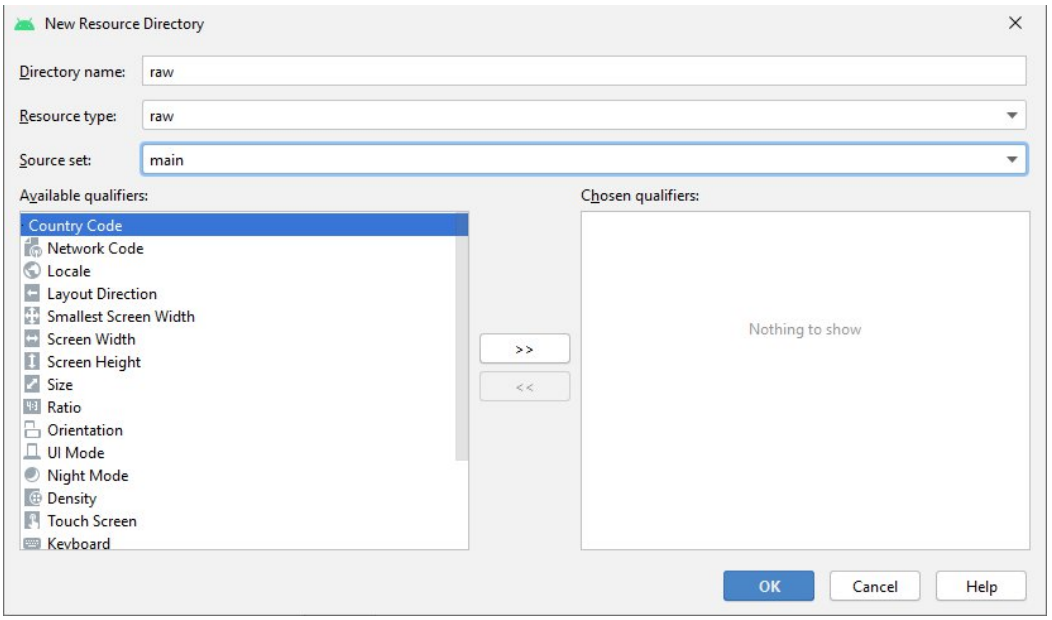


Рисунок 2 – Указание типа и названия директории

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:

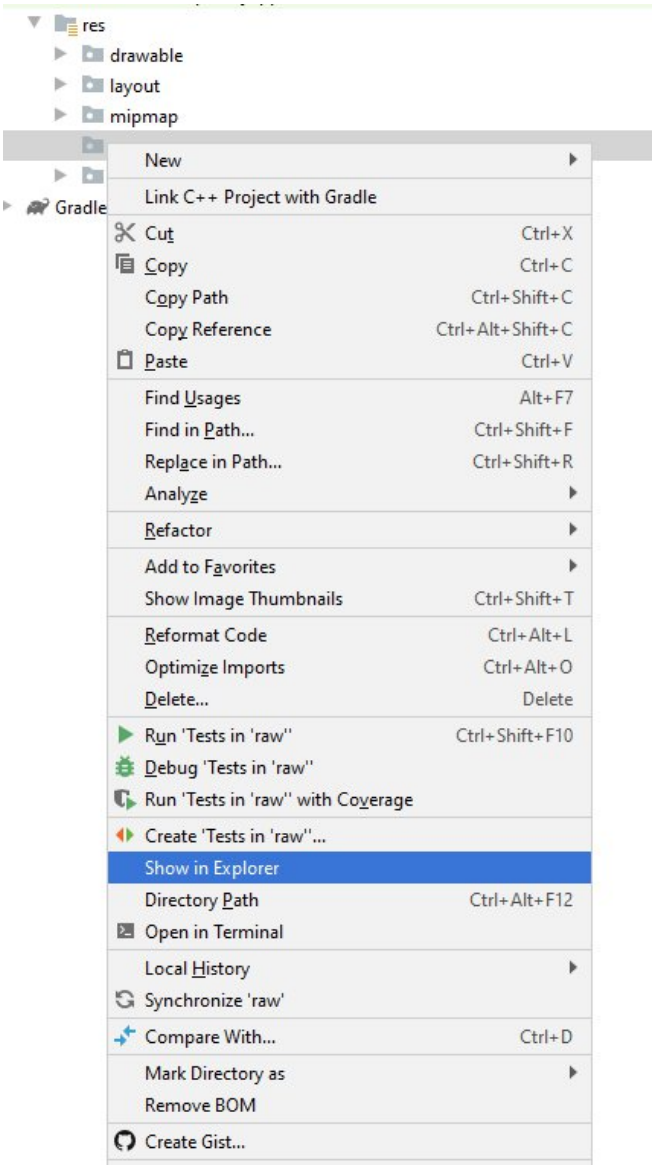


Рисунок 3 – Контекстное меню созданной папки

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 2. 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.

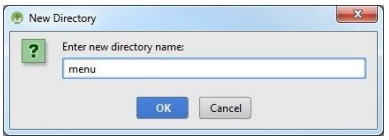


Рисунок 1 – Создание новой директории

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

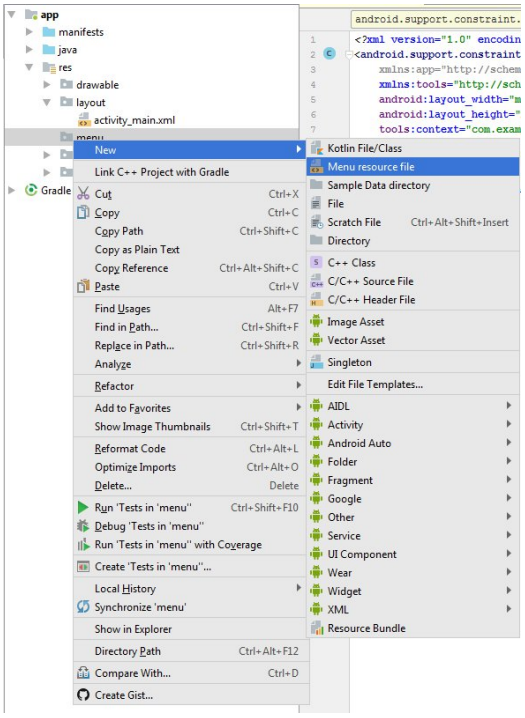


Рисунок 2 – Добавление файла main_menu

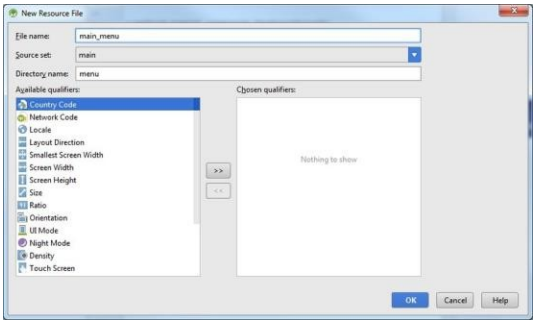


Рисунок 3 – Установка названия

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).

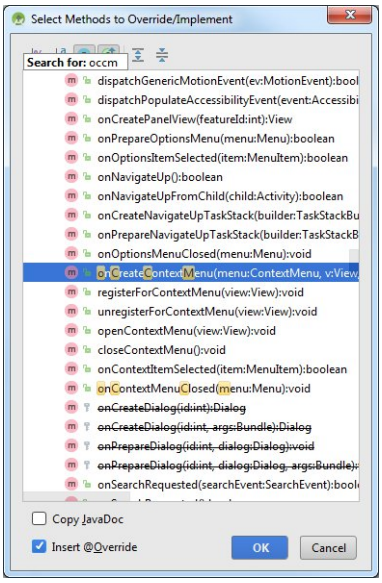


Рисунок 9 – Добавление перегрузки

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.

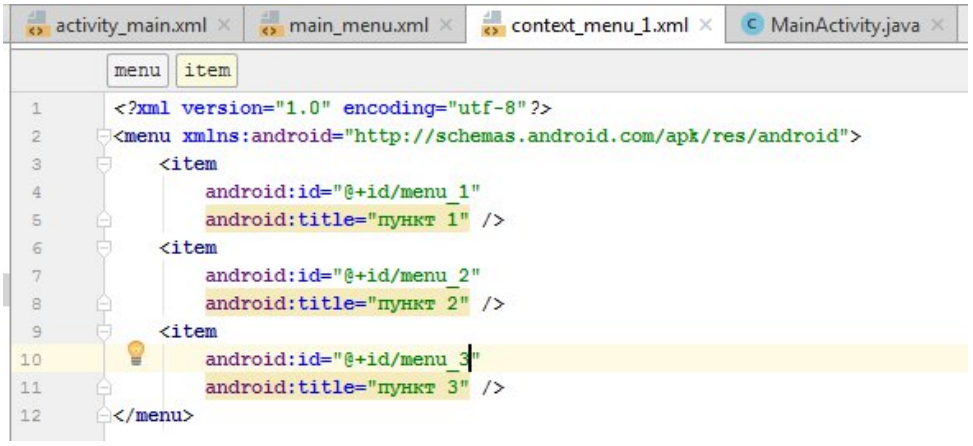


Рисунок 10 – Создание файла контекстного меню

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.

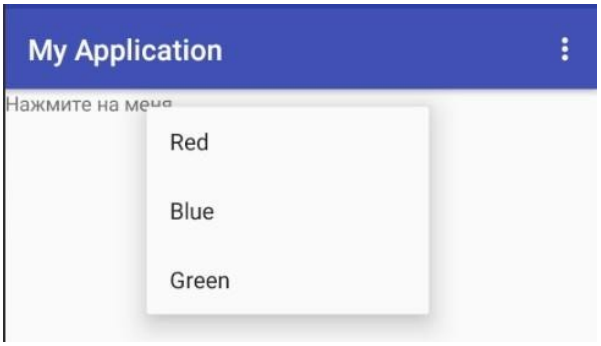


Рисунок 13 – Результат вызова контекстного меню

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.

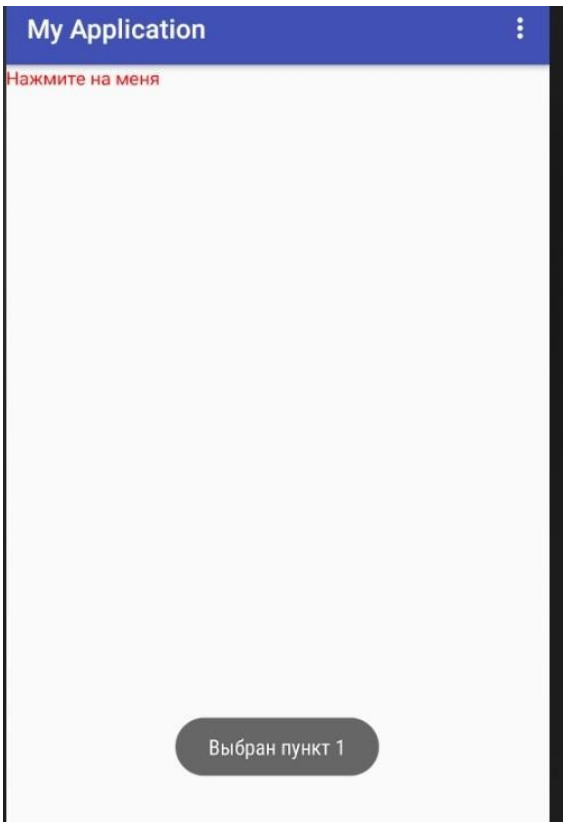


Рисунок 15 – Результат выбора пункта

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 3. 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 4. 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 5. 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.

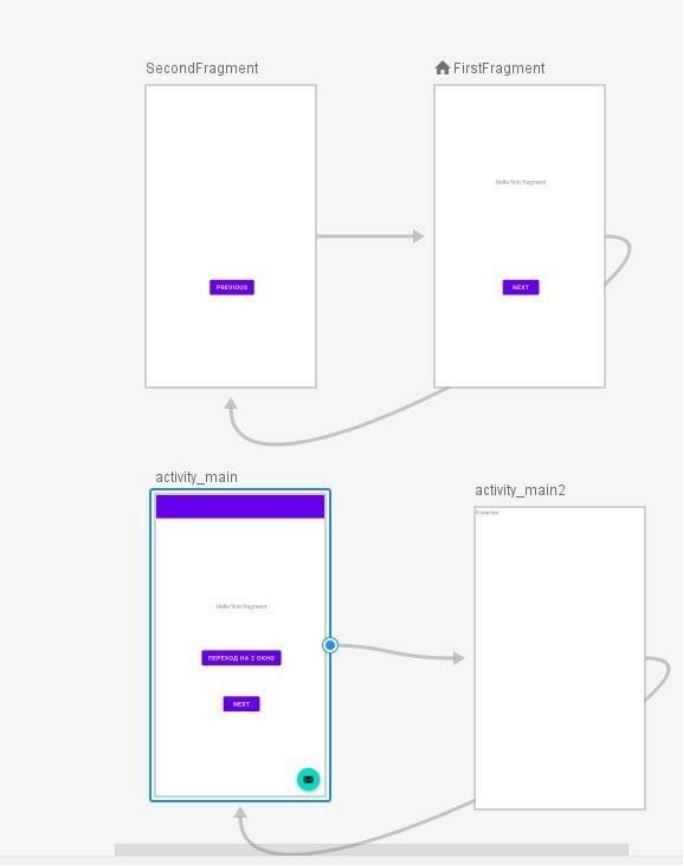
   c. Кнопка перехода на второе окно.

   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 6. 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