

Министерство науки и высшего образования РФ  
Южно-Российский государственный политехнический  
университет (НПИ) имени М.И. Платова

---

Шахтинский автодорожный институт (филиал) ЮРГПУ(НПИ)  
им. М.И. Платова



**В.А. Зуев**

## **РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ**

Методические указания  
к выполнению практических занятий  
и лабораторных работ

*для направления  
09.03.02 Информационные системы и технологии*

Новочеркасск  
ЮРГПУ (НПИ)  
2019

УДК 681.3.06

Рецензент – канд. техн. наук О.А. Наугольников

**В.А. Зуев**

**Разработка мобильных приложений:** методические указания к выполнению практических занятий и лабораторных работ/ Южно-Российский государственный политехнический университет (НПИ) имени М.И. Платова. – Новочеркасск: ЮРГПУ(НПИ), 2019. – 68 с.

Даны рекомендации к выполнению практических занятий и лабораторных работ по дисциплине «Разработка мобильных приложений» для устройств на основе операционной системы Android. Рассмотрена технология создания мобильных приложений в среде разработки Android Studio на языке Java.

Предназначены для направления 09.03.02 «Информационные системы и технологии» для студентов всех форм обучения.

УДК 681.3.06

© Южно-Российский государственный  
политехнический университет (НПИ)  
имени М.И. Платова, 2019

# ТЕМА №1

## ПЛАТФОРМА ANDROID И СРЕДСТВА РАЗРАБОТКИ ПРИЛОЖЕНИЙ

### Практическое занятие

#### Среда разработки приложений для мобильных устройств Android Studio

*Цель занятия:* знакомство с инструментами разработки Android- приложений.

#### *Методические указания*

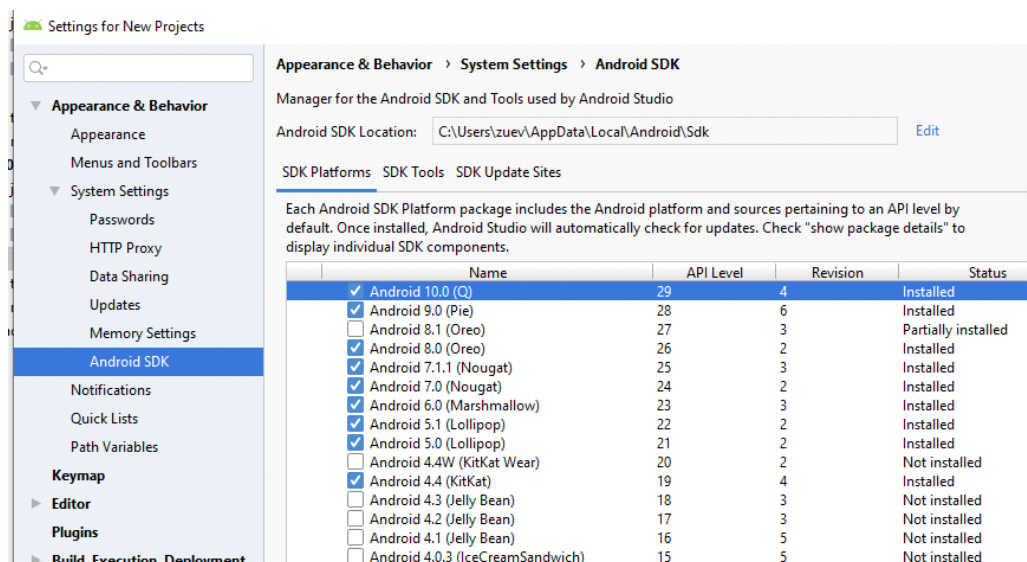
Android - операционная система, которая широко используется в мобильных устройствах: смартфонах и планшетных компьютерах. Это бесплатная операционная система, основанная на Linux с интерфейсом программирования Java. Приложение для Android пишется на языке Java или Kotlin. Наиболее популярной средой разработки в настоящее время является Android Studio [1-3].

Android Studio - среда разработки под Android, основанная на IntelliJ IDEA. Она предоставляет интегрированные инструменты для разработки и отладки. Для отладки приложений используется эмулятор телефона - виртуальная машина, на которой будет запускаться созданное приложение.

Для того, чтобы установить Android Studio на Windows, следует выполнить следующие действия. Загрузите актуальную версию Android Studio с официального сайта <https://android-studio.ru/> и затем запустите сохраненный файл. Установите Android Studio и все необходимые инструменты SDK по инструкциям мастера установки. Отдельные инструменты и другие пакеты SDK сохраняются вне каталога приложений в Android Studio. Например: \Users\\sdk\

При настройке Android SDK следует установить пакеты соответствующего уровня, который зависит от версии OS Android, для которой предполагается разрабатывать

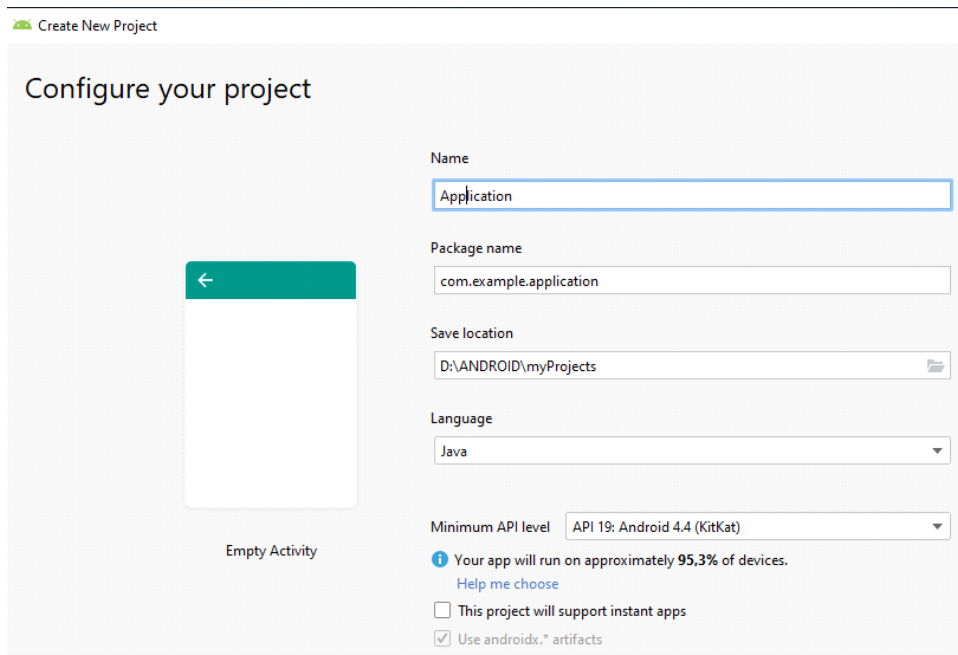
приложения. Следует учитывать, что при установке каждого из них потребуется значительный объем памяти. Например, для API 29 – примерно 260 Мб.



При создании проекта в нем создается по меньшей мере один модуль. В модуле содержится описание на языке xml изображения на экране приложения и код на языке Java. Каждый модуль по сути является приложением, а проект - контейнер для модуля. Если в проекте содержится несколько модулей, то следует указать, какой из них надо запустить.

При выполнении студентом комплекса лабораторных работ целесообразно создать один проект, где каждый из модулей будет соответствовать приложению для очередной темы лабораторной работы.

Рассмотрим этапы создания нового проекта. После запуска Android Studio выбираем **Start a New Android Studio Project**, появится диалоговое окно мастера. Предлагается несколько шаблонов проекта, где выбираем **Empty Activity**. Далее предлагается ввести следующую информацию.



Поле **Name** - имя, которое будет отображаться в заголовке приложения (по умолчанию MyApplication). Поле **Package name** формирует специальный java-пакет на основе вашего имени из предыдущего поля. Поле **Save location** позволяет выбрать место на диске для создаваемого проекта.

После нажатия на кнопку **Next** переходим к следующему окну. Здесь выбираем типы устройств, под которые будем разрабатывать приложение. Выбираем смартфоны. Далее необходимо выбрать внешний вид экрана приложения. Выберем, например, вариант **Blank Activity**. После нажатия кнопки **Finish** формируется проект и создаётся необходимая структура из различных файлов и папок.

В левой части среды разработки на вкладке Android появится иерархический список из папок, которые относятся к проекту. Вкладка Android содержит две основные папки: **app** и **Gradle Scripts**. Первая папка **app** является отдельным модулем для приложения и содержит все необходимые файлы приложения - код, ресурсы картинок и т.п. Вторая папка служит для различных настроек для управления проектом. Раскрываем папку **app**. В ней находятся три папки: **manifest**, **java**, **res**.

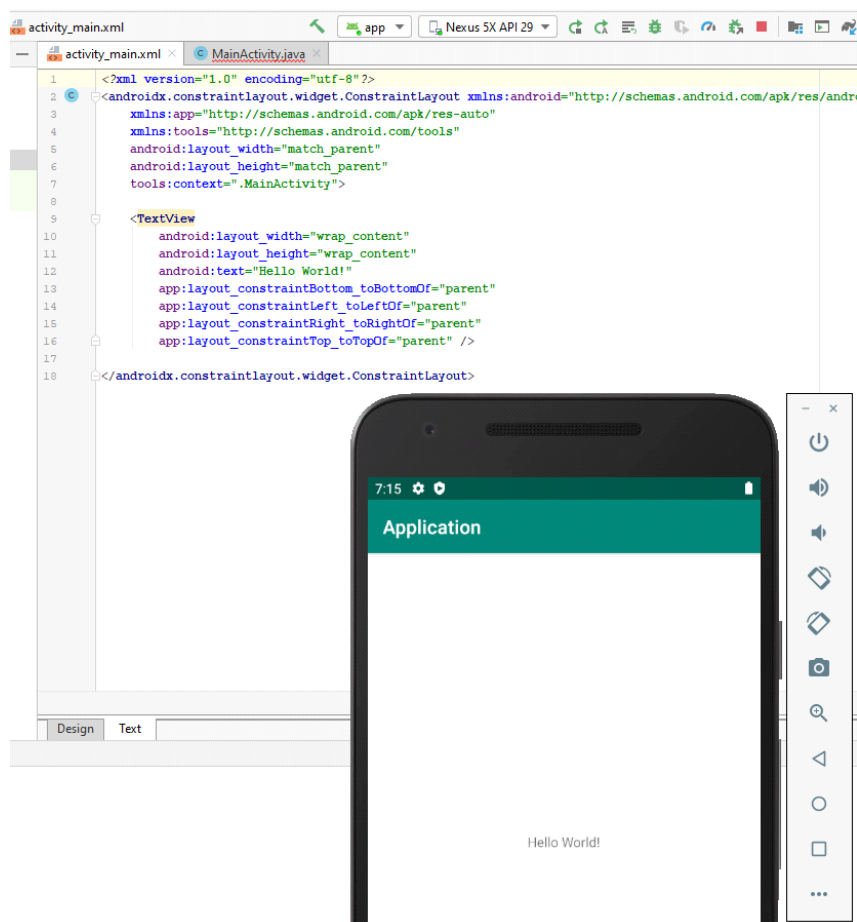
Папка **manifest** содержит единственный файл манифеста **AndroidManifest.xml**. В этом файле должны быть объявлены

все активности, службы, приёмники и контент-провайдеры приложения. Также он должен содержать требуемые приложению разрешения. «AndroidManifest.xml» можно рассматривать, как описание для развертывания Android-приложения.

Папка java содержит три подпапки - рабочую и для тестов. Рабочая папка имеет название пакета и содержит файлы классов. Сейчас там один класс MainActivity.

Папка res содержит файлы ресурсов, разбитых на отдельные подпапки.

Запуск программы осуществляется выбором команды Run в пункте меню **Run**. После этого в эмуляторе на экране появится окно приложения с надписью «Hello World!» и заголовком программы.

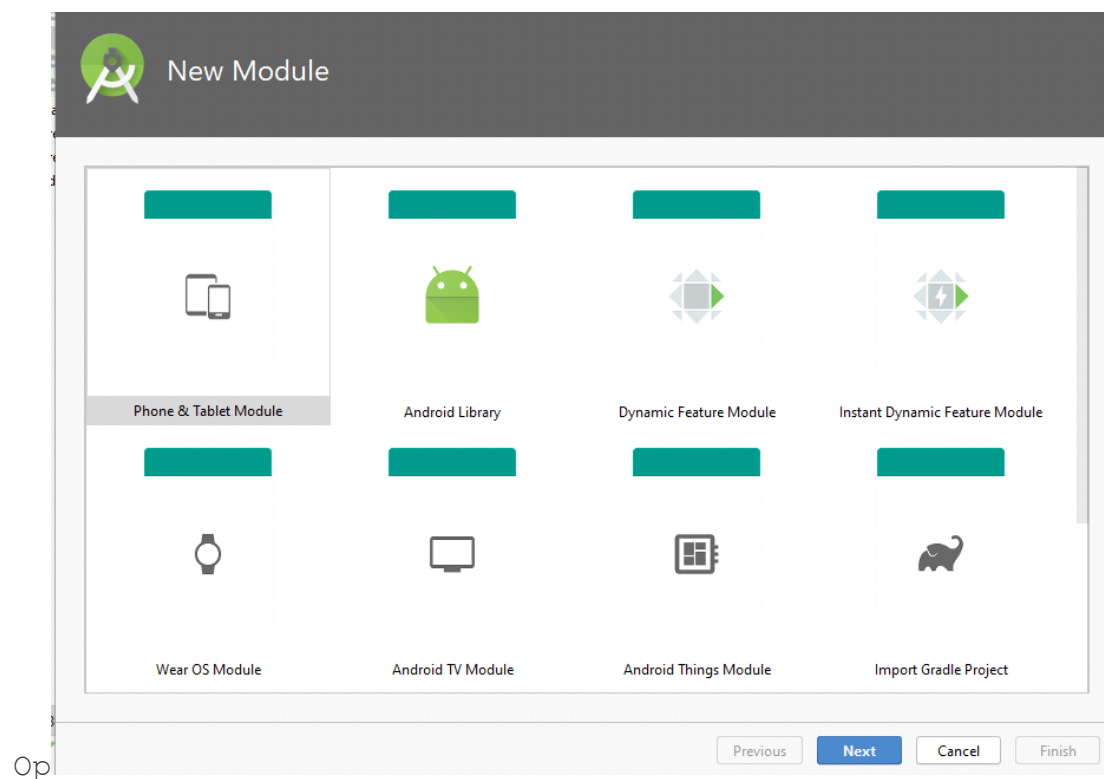


Итак, проект создан. Теперь создадим в проекте свой модуль (приложение) для этого текущего урока. Эта процедура будет частично похожа на создание проекта, но с небольшими

отличиями. Чтобы создать модуль – в меню выбираем **File -> New -> New module**. Тип модуля выбираем **Phone and Tablet Application**

Итак, проект создан. Теперь создадим в проекте свой модуль. Создадим модуль (приложение) для текущего занятия. Эта процедура будет частично похожа на создание проекта, но с небольшими отличиями.

Чтобы создать модуль – в меню выбираем **File -> New -> New module**. Тип модуля выбираем **Phone and Tablet Application** и жмем **Next**.



Далее надо заполнить поля:

**Application/Library name** – непосредственно имя приложения, которое будет отображаться в списке приложений в смартфоне.

**Module name** – это название модуля. Это название будет отображаться слева в списке модулей, там, где сейчас есть app. Предлагается такой шаблон для названия модулей:

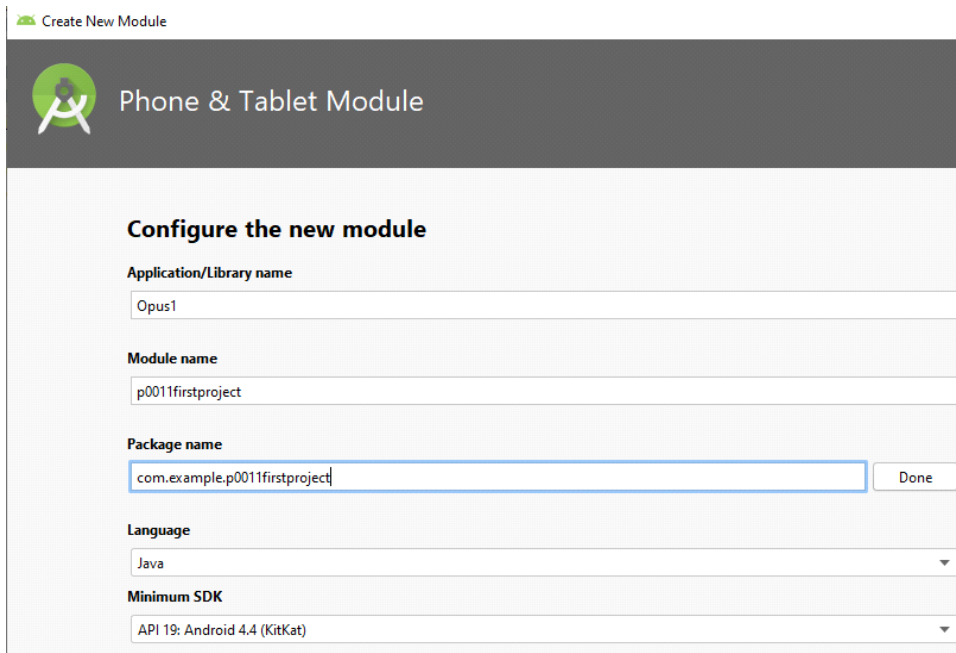
р<номер занятия><номер проекта в этой занятии>.

На номер занятия выделим три цифры, а на номер проекта – одну. Также, будем добавлять название приложения,

например, - firstproject. Все это пишем маленькими буквами и без пробелов. Получится такое имя модуля: p0011firstproject.

**Package name** – имя пакета отредактируем вручную, нажав edit справа. Оставим там ru. siurgtu и добавим точку и имя модуля.

**Minimum SDK** оставляйте без изменений и нажимайте кнопку **Next**.



Create New Module

Phone & Tablet Module

**Configure the new module**

**Application/Library name**  
Opus1

**Module name**  
p0011firstproject

**Package name**  
com.example.p0011firstproject Done

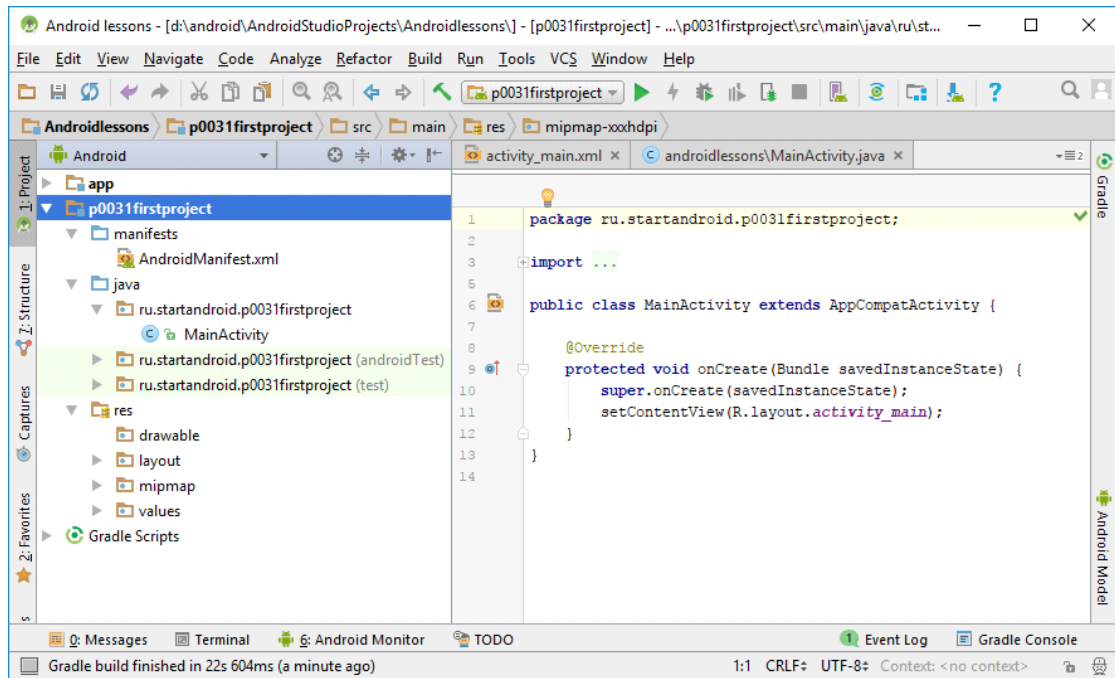
**Language**  
Java

**Minimum SDK**  
API 19: Android 4.4 (KitKat)

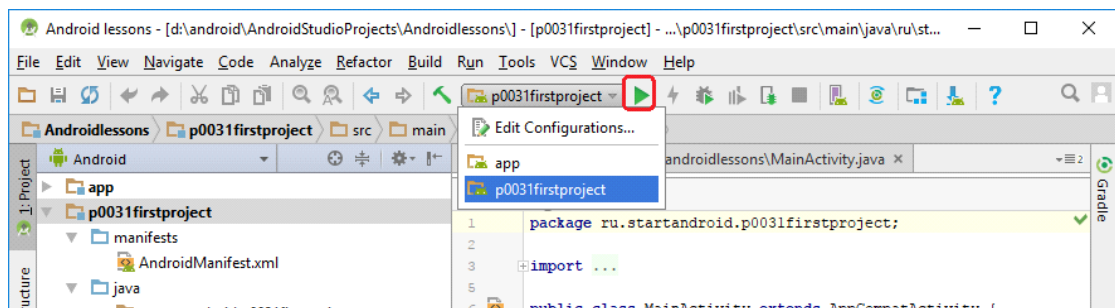
Далее выберите **Empty Activity** и нажимайте **Next**.

В следующей форме ничего менять не нужно; нажимайте кнопку **Finish**. В итоге модуль (Package) будет создан, мы увидим его в списке слева. На скриншоте, который показан ниже, это p0031firstproject - значение, которое было указано в поле Module name.

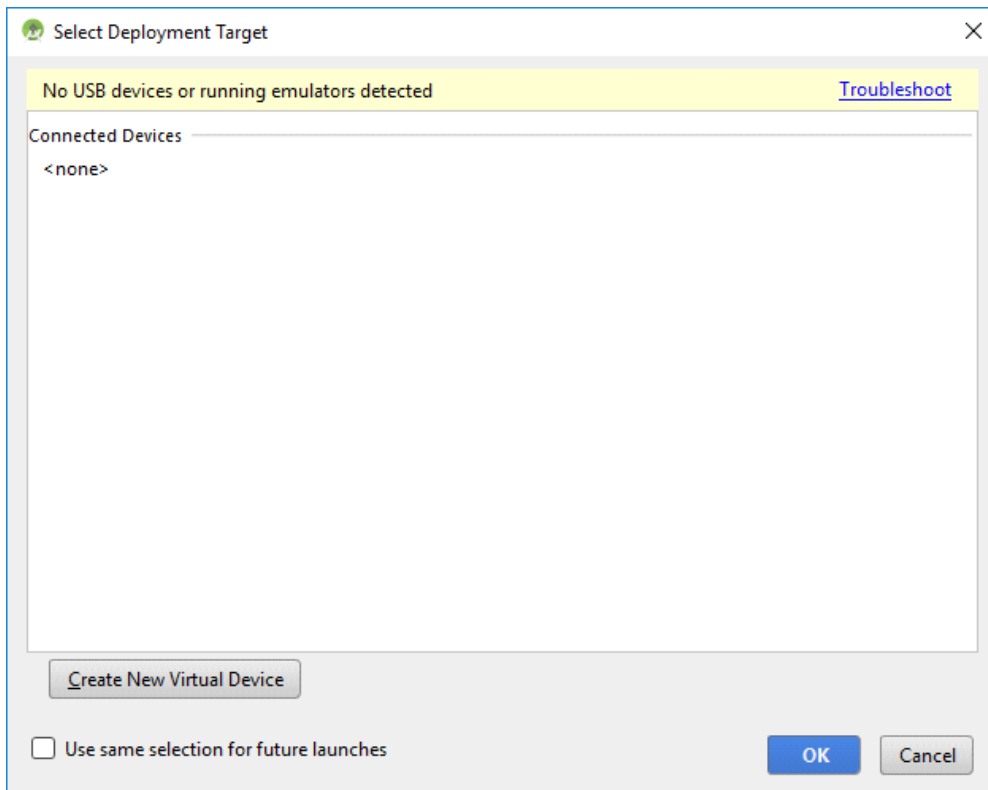




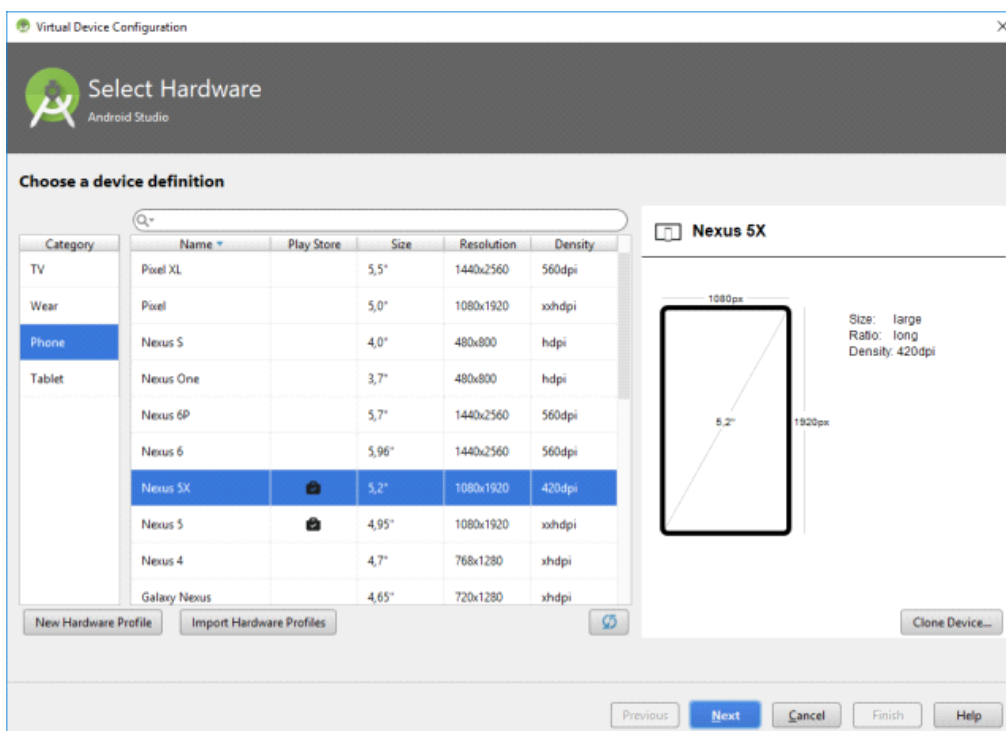
Запустим наше первое приложение! Для этого надо выбрать соответствующий ему модуль в выпадающем списке сверху.



Чтобы запустить приложение, нужно реальное Android-устройство или эмулятор. Если через шнур подключен смартфон или планшет, то приложение будет запущено на нем.

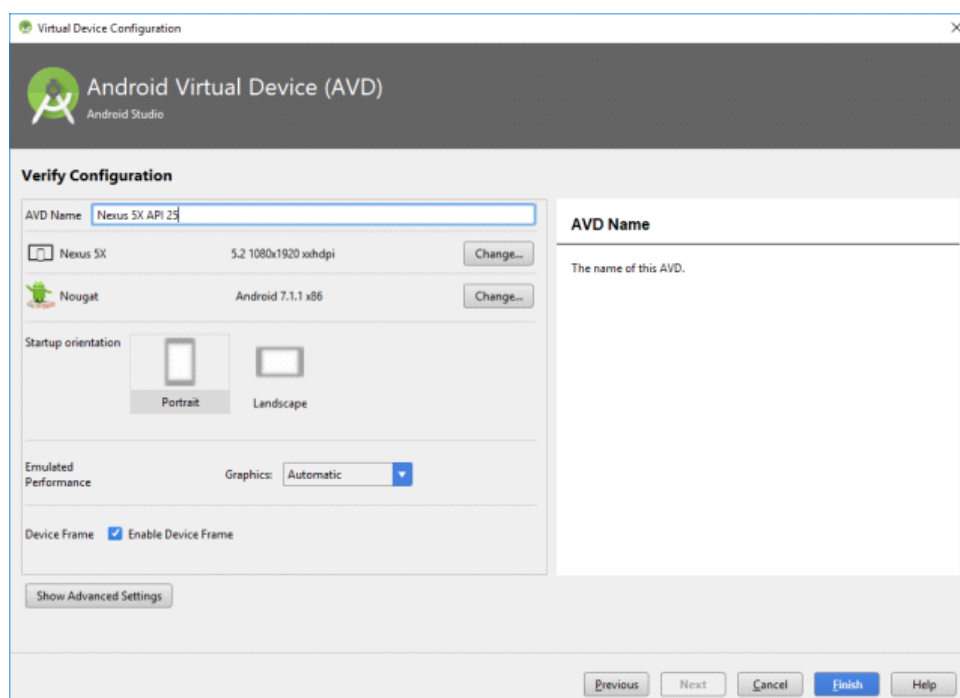


Для выбора эмулятора следует нажать кнопку Create New Virtual Device.

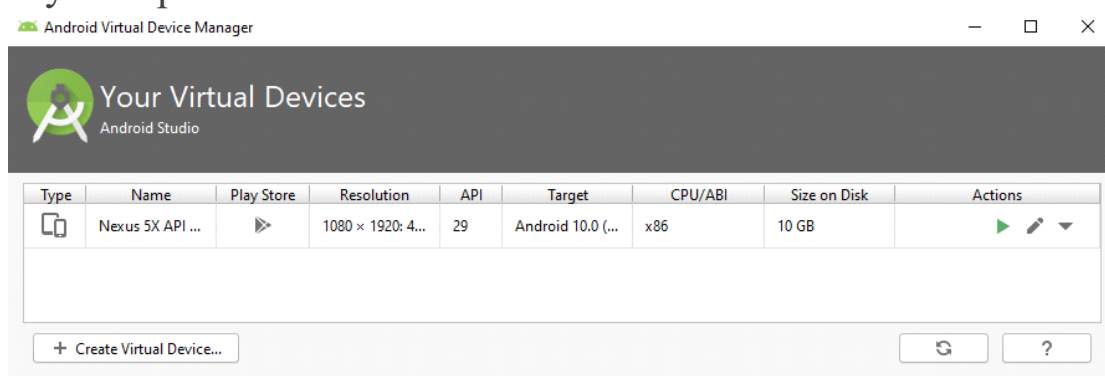


Далее переходите на вкладку x86 Images, где выбирайте

образ, в названии которого нет слова Download. То есть он уже загружен, мы можем его использовать. Затем следует указать название эмулятора и поменять его настройки.



В итоге в списке устройств появляется только что созданный эмулятор. Теперь его можно использовать для запуска приложения.



В результате на экране появится эмулятор выбранного мобильного устройства, в котором будет запущено приложение. Чтобы создать эмулятор телефона, можно выбрать в меню Android Studio пункты **Tools | AVD Manager | Create Virtual Device**.

## Лабораторная работа

### Среда разработки приложений для мобильных устройств Android Studio

*Цель работы:* изучить технологию создания приложений в среде разработки Android Studio

#### *Программа выполнения работы*

1. Изучить методические указания к практическому занятию.
2. Запустить на выполнение Android Studio.
3. Создать в среде Android Studio проект, разработанный на практическом занятии.
4. Создать в соответствующих директориях файлы проекта.
5. Запустить созданное приложение в эмуляторе Android и наблюдать за появлением этого приложения и результатов его работы в окне приложений эмулятора.

#### *Содержание отчета*

1. Название и цель работы.
2. Состав проекта, экранные копии (скриншоты) для созданного приложения в Android Studio.
3. Результаты работы приложения в эмуляторе телефона в виде скриншотов.

#### Контрольные вопросы

1. Какие возможности имеет среда разработки приложений Android Studio?
2. Этапы создания нового проекта?
3. В каких папках хранятся файлы проекта?
4. Что содержится в манифесте проекта?
5. Что нужно сделать для создания модуля?

## ТЕМА №2 ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ В ANDROID

### Практическое занятие

#### Создание пользовательских интерфейсов и использование элементов управления в приложениях под Android

*Цель занятия:* изучить элементы графического интерфейса пользователя, разметку представления на XML, популярные виджеты, обработку событий, научиться создавать приложения с элементами управления.

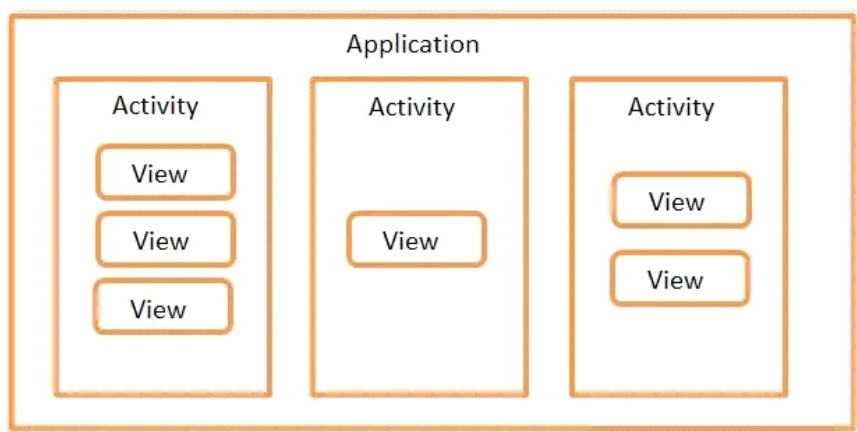
#### *Методические указания*

Рассмотрим компоненты экрана и их свойства.

Если проводить аналогию с Windows, то приложение состоит из окон, называемых Activity. В конкретный момент времени обычно отображается одно Activity и занимает весь экран, а приложение переключается между ними. В качестве примера можно рассмотреть почтовое приложение. В нем одно Activity – список писем, другое – просмотр письма, третье – настройки ящика. При работе вы перемещаетесь по ним.

Макет активности по умолчанию определяет два виджета (widgets): RelativeLayout и TextView. Виджеты представляют собой структурные элементы, из которых составляется пользовательский интерфейс. Виджет может выводить текст или графику, взаимодействовать с пользователем или размещать другие виджеты на экране. Кнопки, текстовые поля, флажки — все это разновидности виджетов. Android SDK включает множество виджетов, которые можно настраивать для получения нужного оформления и поведения. Каждый виджет является экземпляром класса View или одного из его subclasses (например, TextView или Button).

Содержимое Activity формируется из различных компонентов, называемых View.



Необходимо заметить, что View обычно размещаются в ViewGroup. Самый распространенный пример ViewGroup – это Layout. Layout бывает различных типов и отвечает за то, как будут расположены его дочерние View на экране (таблицей, строкой, столбцом ...).

Обычно при работе приложения на экране дисплея находятся элементы управления, которые являются доступными для манипулирования экранными объектами. Виджет – это элемент управления, являющийся объектом класса View, который служит интерфейсом для взаимодействия с пользователем. Условно элементы управления можно разделить на четыре основные категории:

- командные элементы управления, применяемые для выполнения функций;
- элементы выбора, позволяющие выбирать данные или настройки;
- элементы ввода, применяемые для ввода данных;
- элементы отображения, используемые для вывода.

*Командные элементы* управления выполняют некоторые действия. Главным командным элементом является кнопка Button, которая обладает множеством вариантов отображения. Действие выполняется сразу после нажатия на кнопку.

*Элементы выбора* позволяют пользователю выбрать из группы допустимых объектов тот, с которым будет совершено действие. Элементы выбора применяются также для действий по настройке. Распространенными элементами выбора являются флажки и списки. Щелкнув по флажку, пользователь

немедленно увидит появившуюся галочку. Элементы управления типа «список» позволяют осуществлять выбор из конечного множества текстовых строк, каждая из которых представляет команду, объект или признак. Пользователь может выбрать единственную строку текста, нажав на нее.

*Элементы ввода* дают пользователю возможность не только выбирать существующие сведения, но и вводить новую информацию. Самые простые элементы – виджет `TextView`, который отображает текст без возможности его редактирования, и виджет `EditText`, позволяющий редактирование текста (поле ввода). Для отображения графики используется виджет `ImageView`. В эту категорию попадают также такие элементы управления, как счетчики и ползунки.

*Элементы управления отображением* используются для управления визуальным представлением информации на экране. Типичными примерами элементов отображения являются разделители и полосы прокрутки. Сюда же входят разделители страниц, линейки, направляющие, сетки и рамки.

Каждому объекту нужно задать размеры, координаты, цвет, текст и т.д. Android поддерживает способ, основанный на XML-разметке, который напоминает разметку веб-страницы. Можно использовать и визуальный способ перетаскивания объектов с помощью мыши.

Файлы XML-разметки находятся в папке `res/layout` проекта. Папка содержит ресурсы, не связанные с кодом. Кроме разметки, там же содержатся изображения, звуки, строки для локализации и т.д.

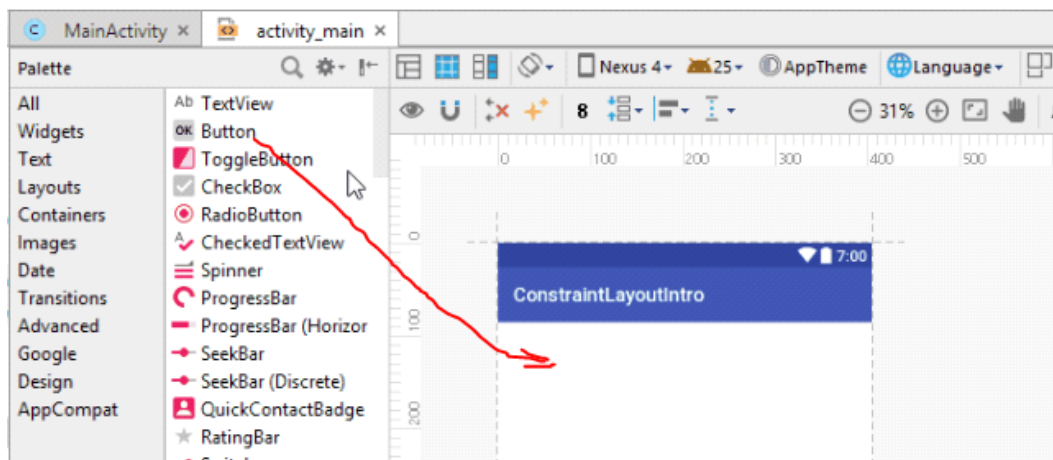
Чтобы размещать на экране различные компоненты (кнопки, поля ввода, чекбоксы и т.п.), необходимо использовать специальный контейнер, в котором будут помещаться компоненты. В Android компоненты называются `View`, а контейнер - `ViewGroup`. Существуют несколько типов `ViewGroup`: `LinearLayout`, `RelativeLayout`, `FrameLayout`, `TableLayout`, `ConstraintLayout` и т.д. Они различаются тем, как они будут упорядочивать компоненты внутри себя. `LinearLayout`, например, выстроит их по горизонтальной или вертикальной линии, `TableLayout` - в виде таблицы.

Рассмотрим контейнер `ConstraintLayout`. Слово `Constraint` переводится как ограничение, принуждение, привязка. Класс `ConstraintLayout` является родительским для классов `ViewGroup` и `TextView`.

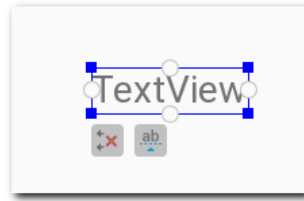
Код на языке `xml`, находящийся в файле `activity_main.xml` для «пустого» приложения, которое выводит в середине экрана текст “Hello Word!”:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Для размещения виджета в нужном месте экрана воспользуемся привязкой. Разместите на экране какой-нибудь виджет, например `TextView`. Сделать это можно перетаскиванием мышью соответствующего компонента из палитры **Palette**



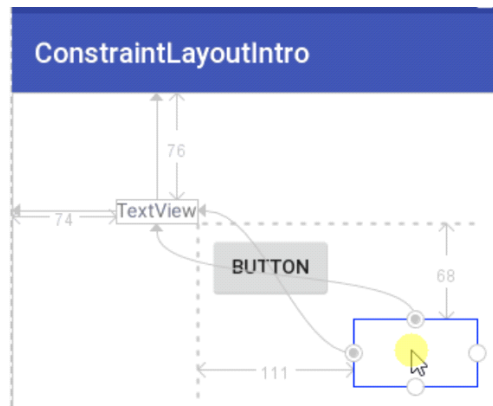




В запущенном приложении виджет окажется не в том месте, где его разместили, а в верхнем левом углу (координаты  $x=0$ ,  $y=0$ ). Чтобы его закрепить в желаемом месте надо добавить привязки (constraints). Они будут задавать положение View на экране относительно каких-либо других элементов или относительно родительского View. Если выделить на экране виджет TextView, то можете видеть 4 круга по его бокам.

Эти круги используются, чтобы создавать привязки. Существует два типа привязок: одни задают положение View по горизонтали, а другие - по вертикали.

Создадим горизонтальную привязку. Привяжем положение TextView к левому краю его родителя. Родителем TextView этом примере является ConstraintLayout, который занимает весь экран. Поэтому края ConstraintLayout совпадают с краями экрана. Чтобы создать привязку, нажмите мышкой на TextView, чтобы выделить его. Затем зажмите левой кнопкой мыши левый кружок и тащите его к левой границе. TextView также уехал влево. Он привязался к левой границе своего родителя. Не обязательно, чтобы они должны быть вплотную. Мы можем задать отступ. Для этого просто зажмите левой кнопкой мыши TextView, перетащите вправо и отпустите. Обратите внимание на число, которое меняется. Это величина отступа TextView от объекта, к которому он привязан (в нашем случае - от левой границы родителя). Аналогично можно сделать вертикальную привязку, после чего наш виджет окажется в нужном месте.



Можно привязывать не только к границам родителя, но и к другим View. Сделайте привязку кнопки Button к TextView. Для этого разместите виджет кнопки на экране, выделите ее мышью, коснитесь указателем мыши кружка в левой ее части (кружок станет зеленым). Не отпуская левую кнопку мыши, проведите указатель к правому кружку виджета TextView. Когда этот кружок станет зеленым, отпустите кнопку мыши. Теперь виджет Button оказался привязанным к виджету TextView по горизонтали. Аналогично сделайте привязку по вертикали, проведя линию от верхнего кружка виджета Button до нижнего кружка виджета TextView. Перемещая Button, увидим линии со стрелками и в них относительные координаты виджета Button.

Более подробно средства привязки виджетов можно изучить по материалам [1-5]. Следует также самостоятельно изучить стандартные типы разметок: `FrameLayout`, `LinearLayout`, `TableLayout`, `RelativeLayout`.

Рассмотрим процесс разработки простого приложения, содержащего типичный набор виджетов.

*Создадим обработчик событий на примере Button.*

Откройте созданный автоматически проект, содержащий пустую Activity. Когда разметка открыта в графическом представлении, то слева от основной части редактора кода можно увидеть панель инструментов, в которой сгруппированы различные элементы по категориям: **Widgets**, **Texts**, **Layouts** и т.д. В группе **Images** найдите элемент **ImageButton**, перетащите его на форму и отпустите. Далее необходимо выбрать изображение для кнопки.

Во вкладке **Component Tree** выберем элемент **Constraint Layout** (экран). В панели свойств **Properties** отобразятся самые употребительные свойства выбранного компонента. К ним относятся идентификатор, ширина и высота. Выбираем **View all properties** (две стрелочки), чтобы открыть все свойства компонента. Найдите свойство **background**. Щелкните рядом с этим словом во второй колонке. Появится текстовое поле, в которое можно ввести значение вручную, и кнопка с тремя точками, которая запустит диалоговое окно для создания ресурса. Переходим на вкладку **Color** и выбираем нужный цвет.

Далее можно поменять картинку для графической кнопки. Находим подходящее изображение и копируем его в папку `res/drawable` проекта. Затем выделяем элемент **ImageButton** на форме и в панели свойств выбираем свойство `srcCompat`. Снова щелкаем на кнопке с тремя точками и выбираем ресурс в категории **Drawable** - там должен появиться ресурс с именем добавленного ранее файла. Там же в окне свойств находим свойство `onClick`; прописываем имя метода для обработки нажатия на кнопку `onClick`.

Установите курсор мыши внутри текста “`onClick`” у кнопки (переключившись в режим `Text`) и нажмите комбинацию `Alt+Enter`. Во всплывающем окне выбрать вариант `Create “onClick(View)” in “MainActivity”`. В коде класса `MainActivity` появится заготовка для обработки щелчка кнопки. Необходимо набрать следующий текст:

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    private TextView mHelloTextView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mHelloTextView=(TextView) findViewById(R.id.textView);
    }
    public void onClick(View view) {
        mHelloTextView.setText("Hello!");
    }
}
```

```
}  
}
```

Далее необходимо запустить программу в эмуляторе Android, нажать на созданную кнопку и наблюдать за изменением текста на экране.

## **Лабораторная работа**

### **Создание пользовательских интерфейсов и использование элементов управления в приложениях под Android**

*Цель работы:* изучить интерфейс IDE Android Studio и получить навыки составления и отладки простого Android-приложения с использованием базовых элементов графического интерфейса пользователя.

#### *Порядок выполнения работы*

1. Изучить методические указания к практическому занятию.
2. Запустить на выполнение Android Studio.
3. Открыть в среде Android Studio ранее созданный проект и добавить в него новый пакет. Либо можно создать новый проект.
4. Разместить на форме элементы управления Button, ImageButton, CheckBox и TextView, настроить их свойства (параметры).
5. В окне java-кода проекта добавить строки обработки нажатия на кнопку.
6. Запустить созданное приложение в эмуляторе Android и наблюдать за появлением этого приложения и результатов его работы в окне приложений эмулятора.
7. Добавить в проект другие элементы управления, настроить их свойства и проверить работу приложения в эмуляторе Android.

### *Содержание отчета*

- 1 . Название и цель работы.
- 2 . Листинг созданного приложения в Android Studio с использованием командных элементов управления, элементов выбора, элементов ввода и элементов отображения.
- 3 . Результаты работы приложения в эмуляторе телефона.

### *Примерные варианты заданий*

1.	Разработать графический интерфейс приложения «Калькулятор валют»
2.	Разработать графический интерфейс приложения «Калькулятор цены автомобиля с учетом возможных опций»
3.	Разработать графический интерфейс приложения «Калькулятор цены компьютера» с учетом цен на комплектующие
4.	Разработать графический интерфейс приложения «Калькулятор затрат на автомобильные запчасти»
5.	Разработать графический интерфейс приложения «Калькулятор стоимости ремонта автомобиля»
6.	Разработать графический интерфейс приложения «Модель движения материальной точки в поле тяготения»
7.	Разработать графический интерфейс приложения «Калькулятор затрат на ремонт дома»
8.	Разработать графический интерфейс приложения «Калькулятор стоимости оплат за оказанные услуги ЖКХ для квартиры многоэтажного дома»
9.	Разработать графический интерфейс приложения «Калькулятор перевода расстояния пройденного пути мили-километры»
10.	Разработать графический интерфейс приложения «Калькулятор перевода веса килограммы-фунты»
11.	Разработать графический интерфейс приложения «Калькулятор перевода температуры по шкалам Цельсия-Фаренгейта-Кельвина»
12.	Разработать графический интерфейс приложения «Калькулятор расчета стоимости печати фотографий разного размера»
13.	Разработать графический интерфейс приложения «Калькулятор стоимости заказанных блюд в кафе»
14.	Разработать графический интерфейс приложения «Калькулятор »
15.	Разработать графический интерфейс приложения «Калькулятор стоимости домашнего блюда»

### *Контрольные вопросы*

1. Приведите иерархию классов View ViewGroup
2. Какими способами можно сделать разметку – структуру расположения элементов в окне?
3. Назовите назначение XML-атрибутов в файле разметки
4. Приведите стандартные типы разметок.
5. Каково назначение разметки LinearLayout?
6. Назовите основные (базовые) виджеты.
7. Какими классами могут быть представлены текстовые поля?
8. Какой виджет используется для отображения графики?
9. Какими классами представлены кнопки и флажки?
10. Какие компоненты используются для отображения времени?
11. Какие классы используются в текстовых полях с автозаполнением с возможностью редактирования текста?
12. Какой элемент используется для отображения вертикального списка с прокруткой?
13. Для чего предназначен элемент Spinner?
14. Какие возможности элемента GridView?
15. Какие возможности виджета Gallery?

## ТЕМА №3 ОБЗОР БАЗОВОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ ДЛЯ ANDROID

### Практическое занятие

#### Создание пользовательских интерфейсов и использование элементов управления в приложениях под Android

*Цель занятия:* изучить элементы базового языка программирования Java для программирования для ОС Android.

#### *Методические указания*

##### *Создание классов и объектов в Java.*

Язык Java является объектно-ориентированным и в полном объеме использует эти принципы. Даже небольшие учебные проекты часто состоят из многих классов. При этом каждому общедоступному (public) классу соответствует свой файл, имеющий то же имя. Для удобства в Java предусмотрено специальное средство группировки классов, называемое пакетом (package). Пакеты обеспечивают независимые пространства имен (namespaces), а также ограничение доступа к классам. Классы всегда задаются в каком-либо пакете. Чтобы поместить класс в пакет, требуется продекларировать имя пакета в начале файла, в котором объявлен класс, в виде:

```
package имя_пакета;
```

Кроме того, необходимо поместить исходный код класса в папку, соответствующую пакету. Если декларация имени пакета отсутствует, считается, что класс принадлежит пакету с именем default. При создании проекта в среде NetBeans помещение класса в пакет происходит автоматически.

Класс – это описание того, как будет устроен объект, являющийся экземпляром данного класса, а также какие методы объект может вызывать. Классы в Java задаются следующим образом. Сначала пишется зарезервированное слово class, затем имя класса, после чего в фигурных скобках

пишется реализация класса – задаются его поля (глобальные переменные) и методы.

Объектные переменные – такие переменные, которые имеют объектный тип. В Java объектные переменные – это не сами объекты, а только ссылки на них. То есть все объектные типы являются ссылочными.

Объявление объектной переменной осуществляется так же, как и для других типов переменных. Сначала пишется тип, а затем через пробел имя объявляемой переменной. Например, если мы задаем переменную `obj1` типа `Circle`, "окружность", ее задание осуществляется так:

```
Circle obj1;
```

Связывание объектной переменной с объектом осуществляется путем присваивания. В правой части присваивания можно указать либо функцию, возвращающую ссылку на объект (адрес объекта), либо имя другой объектной переменной. Если объектной переменной не присвоено ссылки, в ней хранится значение `null`.

Объектные переменные можно сравнивать на равенство, в том числе на равенство `null`. При этом сравниваются не сами объекты, а их адреса, хранящиеся в объектных переменных.

Создается объект с помощью вызова специальной подпрограммы, задаваемой в классе и называемой конструктором. Конструктор возвращает ссылку на созданный объект. Имя конструктора в Java всегда совпадает с именем класса, экземпляр которого создается. Перед именем конструктора во время вызова ставится оператор `new` – "новый", означающий, что создается новый объект. Например, вызов `obj1=new Circle();` означает, что создается новый объект типа `Circle`, "окружность", и ссылка на него (адрес объекта) записывается в переменную `obj1`. Переменная `obj1` до этого уже должна быть объявлена. Оператор `new` отвечает за динамическое выделение памяти под создаваемый объект. Часто совмещают задание объектной переменной и назначение ей объекта. В нашем случае оно будет выглядеть как

```
Circle obj1=new Circle();
```

У конструктора, как и у любой подпрограммы, может



быть список параметров. Они нужны для того, чтобы задать начальное состояние объекта при его создании. Например, мы хотим, чтобы у создаваемой окружности можно было при вызове конструктора задать координаты  $x$ ,  $y$  ее центра и радиус  $r$ . Тогда при написании класса `Circle` можно предусмотреть конструктор, в котором первым параметром задается координата  $x$ , вторым –  $y$ , третьим – радиус окружности  $r$ . Тогда задание переменной `obj1` может выглядеть так:

```
Circle obj1=new Circle(130,120,50);
```

Оно означает, что создается объект-окружность, имеющий центр в точке с координатами  $x=130$ ,  $y=120$ , и у которой радиус  $r=50$ . Если разработчики класса не создали ни одного конструктора, в реализации класса автоматически создается конструктор по умолчанию, имеющий пустой список параметров. И его можно вызывать в программе так, как мы это первоначально делали для класса `Circle`.

Принято имена объектных типов писать с заглавной буквы, а имена объектных переменных – с маленькой.

В Java принято называть объектные переменные так же, как их типы, но начинать имя со строчной буквы.

Следующие строки программного кода создают два независимых объекта с одинаковыми начальными параметрами:

```
Circle circle1=new Circle(130,120,50);  
Circle circle2=new Circle(130,120,50);
```

С помощью объектных переменных осуществляется доступ к полям данных или методам объекта: сначала указывается имя переменной, затем точка, после чего пишется имя поля данных или метода. Например, если имя объектной переменной `circle1`, а имя целочисленного поля данных  $x$ , то присваивание ему нового значения будет выглядеть как

```
circle1.x=5;
```

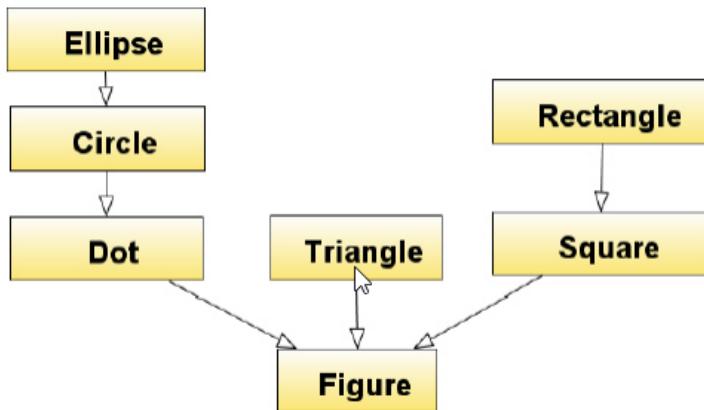
А если имя подпрограммы `show`, у нее нет параметров и она не возвращает никакого значения, то ее вызов будет выглядеть как

```
circle1.show();
```

## Наследование классов

Наследование позволяет строить на основе первоначального класса новые, добавляя в классы новые поля данных и методы. В Java все классы являются потомками класса Object. То есть он является базовым для всех классов.

В качестве примера того, как строится иерархия, рассмотрим иерархию фигур, выводимых на экран.



В ней базовым классом является Figure, от которого наследуются Dot - "точка", Triangle - "треугольник" и Square - "квадрат". От Dot наследуется класс Circle - "окружность", а от Circle унаследуем Ellipse - "эллипс". И, наконец, от Square унаследуем Rectangle - "прямоугольник". Отметим, что в иерархии принято рисовать стрелки в направлении от наследника к прародителю. Такое направление называется Generalization - "обобщение", "генерализация". Стрелки символизируют направление в сторону упрощения.

Базовый класс всегда называют именем, которое характеризует все объекты - экземпляры классов-наследников, и которое выражает наиболее общую абстракцию, применимую к таким объектам. В нашем случае это класс Figure. Любая фигура будет иметь поля данных x и y - координаты фигуры на экране. Класс Dot ("точка") является наследником Figure, поэтому он будет иметь поля данных x и y, наследуемые от Figure. То есть в самом классе Dot задавать эти поля не надо. От Dot мы наследуем класс Circle ("окружность"), поэтому в нем также имеется поля x и y, наследуемые от Figure. Но появляется дополнительное поле

данных. У Circle это поле, соответствующее радиусу. Мы назовем его r. Кроме того, для окружности возможна операция изменения радиуса, поэтому в ней может появиться новый метод, обеспечивающий это действие - назовем его setSize ("установить размер"). Класс Ellipse имеет те же поля данных и обеспечивает то же поведение, что и Circle, но в этом классе появляется дополнительное поле данных r2 - длина второй полуоси эллипса, и возможность регулировать значение этого поля.

## **Лабораторная работа**

Цель работы: получить навыки составления и отладки простого Android-приложения с использованием языка Java.

### *Порядок выполнения работы*

1. Изучить методические указания к практическому занятию. Написать код программы согласно варианту заданию.
2. Запустить на выполнение Android Studio.
3. Открыть в среде Android Studio ранее созданный проект и добавить в него новый пакет. Либо можно создать новый проект.
4. Разместить на форме элементы управления.
5. В окне java-кода проекта добавить строки обработки нажатия на кнопку.
6. Запустить созданное приложение в эмуляторе Android и наблюдать за появлением этого приложения и результатов его работы в окне приложений эмулятора.
7. Добавить в проект другие элементы управления, настроить их свойства и проверить работу приложения в эмуляторе Android.

### *Создание классов*

1. Создайте проект Java. Назовите пакет com.example , а главный класс EmployeeTest
2. Создайте пакет com.example.domain, а в нем класс

## Employee.

```
public int empId;  
public String name;  
public String ssn;  
public double salary;
```

### 3. Добавьте конструктор класса:

```
public Employee() {}
```

4. Создайте методы чтения и записи («геттеры» и «сеттеры») для каждого поля. Используйте для этого контекстное меню редактора NetBeans

5. Добавьте в файл класса EmployeeTest импорт класса Employee import com.example.domain.Employee;

6. Добавьте в процедуру main класса EmployeeTest команды создания объекта класса Employee и заполнение его полей

```
Employee emp = new Employee();  
emp.setEmpId(101);  
emp.setName("Jane Smith");  
emp.setSalary(120_345.27);  
emp.setSsn("012-34-5678");
```

7. Добавьте в процедуру main класса EmployeeTest команды отображения данных объекта класса Employee

```
System.out.println("Employee ID: "+emp.getEmpId());  
System.out.println("Employee Name: "+emp.getName());  
System.out.println("Employee Soc Sec #  
"+emp.getSsn());  
System.out.println("Employee salary:  
"+emp.getSalary());
```

### 8. Запустите приложение.

## *Использование иерархии классов*

1. Скопируйте папку с проектом из предыдущей работы и переименуйте проект.

2. Примените инкапсуляцию к классу Employee.

Для этого:

- замените модификаторы доступа полей с public на private;
- замените конструктор без параметров на конструктор с параметрами

```
public Employee(int empId, String name, String ssn,  
double salary) {  
this.empId = empId;  
this.name = name; this.ssn = ssn; this.salary = salary;
```

```
}
```

– уберите все методы записи данных в поля («сеттеры»),  
кроме метода setName

– добавьте метод увеличения зарплаты raiseSalary:

```
public void raiseSalary(double increase){  
if (increase>0){ salary += increase;  
}  
}
```

3.Создайте в том же пакете подкласс класса Employee и назовите его Manager:

```
public class Manager extends Employee { }
```

4. Добавьте в него поле deptName private String deptName;

5. Добавьте конструктор класса с параметрами, который вызывает конструктор родительского класса и иницирует значение поля deptName :

```
public Manager(int empId, String name, String ssn,  
double salary, String deptName) {  
super(empId, name, ssn, salary);  
this.deptName = deptName;  
}
```

6.Добавьте в него метод чтения данных из поля

```
getDeptName: public String getDeptName() {  
return deptName;  
}
```

7. Создайте в том же пакете подкласс класса Employee и назовите его Admin. Запишите в него конструктор с параметрами:

```
public class Admin extends Employee {  
public Admin(int empId, String name, String ssn, double  
salary) {  
super(empId, name, ssn, salary);  
}  
}
```

8. Создайте в том же пакете подкласс класса Employee и назовите его Engineer. Запишите в него конструктор с параметрами:

```
public class Engineer extends Employee {  
public Engineer(int empId, String name, String ssn,  
double salary) {  
super(empId, name, ssn, salary);  
}  
}
```

9. Создайте в том же пакете подкласс класса Manager и назовите его Director:

```
public class Director extends Manager { }
```

10. Добавьте в него поле `budget private double budget;`

11. Добавьте конструктор класса с параметрами, который вызывает конструктор родительского класса и инициализирует значение поля `budget`:

```
public Director(int empId, String name, String ssn,
double salary, String deptName, double budget) {
super(empId, name, ssn, salary, deptName); this.budget
= budget;
}
```

12. Добавьте в него метод чтения данных из поля `budget`:

```
public double getBudget() {
return budget;
}
```

13. Сохраните все классы

14. Добавьте в процедуру `main` класса `EmployeeTest` команды импорта созданных классов

```
import com.example.domain.Admin;
import com.example.domain.Director;
import com.example.domain.Engineer;
import com.example.domain.Manager;
```

15. Запишите в процедуру `main` класса `EmployeeTest` команды создания объектов созданных классов и заполнения их полей `Engineer`

```
eng = new Engineer(101, "Jane Smith", "012-34-5678",
120_345.27);
Manager mgr = new Manager(207, "Barbara Johnson", "054-
12- 2367", 109_501.36, "US Marketing");
Admin adm = new Admin(304, "Bill Munroe", "108-23-
2367", 75_002.34);
Director dir = new Director(12, "Susan Wheeler", "099-
45- 2340", 120_567.36, "Global Marketing",
1_000_000.00);
```

16. Добавьте в класс `EmployeeTest` статический метод отображения данных объекта, представленного по ссылке класса `Employee`, родительского для всех вновь созданных классов

```
private static void printEmployee(Employee emp) {
System.out.println("Employee ID: " + emp.getEmpId());
System.out.println("Employee Name: " + emp.getName());
System.out.println("Employee Soc Sec # " +
emp.getSsn()); System.out.println("Employee salary: " +
emp.getSalary());
}
```

17. Добавьте в класс `EmployeeTest` команды отображения

данных созданных объектов

```
printEmployee(eng); printEmployee(mgr);  
printEmployee(adm); printEmployee(dir);
```

18. Сохраните все классы и запустите приложение.

### *Содержание отчета*

- 1 . Название и цель работы.
- 2 . Листинг созданного приложения в Android Studio с использованием командных элементов управления, элементов выбора, элементов ввода и элементов отображения.
- 3 . Результаты работы приложения в эмуляторе телефона.

### *Примерные варианты заданий*

№	Класс
1.	Сотрудник, 3 поля
2.	Студент, 2 поля
3.	Товар, 3 поля
4.	Собака, 2 поля
5.	Геометрическая фигура, 2 поля
6.	Программное обеспечение, 3 поля
7.	Аппаратное обеспечение, 3 поля
8.	Город, 2 поля
9.	Страна, 2 поля
10.	Книга, 3 поля
11.	Сотрудник, 3 поля – 3 класса наследника
12.	Студент, 2 поля – 2 класса наследника
13.	Товар, 3 поля– 4 класса наследника
14.	Собака, 2 поля– 3 класса наследника
15.	Геометрическая фигура, 3 поля– 3 класса наследника
16.	Программное обеспечение, 4 поля– 2 класса наследника
17.	Аппаратное обеспечение, 3 поля– 3 класса наследника
18.	Город, 2 поля– 2 класса наследника
19.	Страна, 2 поля– 2 класса наследника
20.	Книга, 3 поля – 4 класса наследника

## *Контрольные вопросы*

1. Какие есть встроенные типы данных в Java?
2. Как в Java объявить одномерный массив?
3. Объявите двумерный массив в Java.
4. Какие в Java имеются стандартные функции для работы со строками?
5. Как создать список в Java ?
6. Что такое модификатор доступа в классе?
7. Какие бывают модификаторы доступа в классе?
8. Что такое поле (переменная) класса?
9. Что такое метод класса?
10. Что такое наследование?
11. Что такое иерархия классов?
12. Как наследуются поля и методы классов?

## **ТЕМА №4 УПРАВЛЕНИЕ ДЕЯТЕЛЬНОСТЯМИ, СЛУЖБЫ В ANDROID**

### **Практическое занятие**

**Планирование покадровой анимации, анимирование, анимация шаблонов, видов, использование класса Camera, проверка безопасности, работа со службами, основанными на местоположении, использование HTTP-служб, службы AIDL**

*Цель занятия:* знакомство с возможностями создания Android- приложений с использованием анимации.

### *Методические указания*

Android предоставляет мощные API для анимации элементов пользовательского интерфейса и построения 2D и



3D изображений. Платформа Android предоставляет две системы анимации: анимация свойств и анимация компонентов пользовательского интерфейса (наследников класса View).

Анимация свойств (Property Animation) позволяет определить анимацию для изменения любого свойства объекта, независимо от того изображается оно на экране или нет.

Анимация компонентов пользовательского интерфейса используется для реализации анимации преобразований над наследниками класса View. Для расчета анимации преобразований используется следующая информация: начальная точка, конечная точка, размер, поворот и другие общие аспекты анимации. Анимация преобразований может выполнять серии простых изменений содержимого экземпляра класса View. Например, для текстового поля можно перемещать, вращать, растягивать, сжимать текст, если определено фоновое изображение, оно должно изменяться вместе с текстом. Пакет android.view.animation предоставляет все классы, необходимые для реализации анимации преобразований. Для задания последовательности инструкций анимации преобразований используется XML код. Такие файлы с XML кодом должны располагаться в папке res/anim/ проекта.

Рассмотрим последовательность разработки приложения, использующую анимацию. Создадим новый проект в Android Studio. Далее подготовим несколько дополнительных файлов, которые будут использоваться при анимации. В папке drawable проекта создадим файл sun.xml с таким содержимым:

```
<?xml version="1.0" encoding="utf-8"?>
<shape
  xmlns:android="http://schemas.android.com/apk/res/andro
id"android:dither="true" android:shape="oval" >
  <gradient
    android:      endColor="#ffff6600"
    android:gradientRadius="150"
    android:      startColor="#ffffcc00"
    android:      type="radial"
    android:useLevel="false" />
</size
```

```
        android:height="          150dp"
        android:width="150dp" />
</shape>
```

Здесь для изображения солнца используется овал, а также для цвета применяется градиент - плавное изменение цвета от темно-желтого (startColor) к светло-желтому (endColor).

В той же папке drawable создадим новый файл sky.xml следующего содержания:

```
<?xml version=" 1.0" encoding="utf-8"?>
<shape
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:dither=" true" android:shape="rectangle"
  >
  <gradient
    android:angle="90"
    android:endColor="#ff000033"
    android:startColor="#ffD000ff" />
</shape>
```

Здесь задана фигура в виде прямоугольника (rectangle) с голубым градиентом от нижнего края к верхнему.

В той же папке drawable создадим новый файл grass.xml следующего содержания:

```
<?xml version="1.0" encoding="utf-8"?>
<shape
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:dither="true" android:shape="rectangle" >
  <gradient
    android:angle="90"
    android:endColor="#ffD03300"
    android:startColor="#ffD09900" />
</shape>
```

Здесь задан зеленый прямоугольник с градиентом.

Далее в файл strings.xml в папке res/values добавим следующие строки:

```
<string name="sun">Солнце</string>
<string name="grass">Трава</string>
<string name="sky">Небо</string>
```

Затем откроем разметку главной активности activity main.xml и добавим в неё несколько элементов ImageView.

Должно получиться следующее содержимое файла:

```
<RelativeLayout xmlns:android=
"http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
<ImageView
android:id="@+id/sky"
android:layoutwidth="fill_parent"
android:layout_height="fill_parent"
android:contentDescription="@string/sky"
android:src="@drawable/sky" />
<ImageView
android:id="@+id/sun"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:contentDescription="@string/sun"
android:scaleType="fitCenter"
android:src="@drawable/sun" />
<ImageView
android:id="@+id/grass"
android:layout_width="fill_parent"
android:layout_height="150dp" android:layout
alignParentBottom="true"
android:contentDescription="@string/grass"
android:src="@drawable/grass" />
</RelativeLayout>
```

У всех элементов `ImageView` в атрибуте `android:src` прописаны созданные фигуры, которые теперь можно видеть на экране.

Далее создадим новую папку `res/anim`, в которой будут находиться файлы анимации. В этой папке создадим новый файл `sun rise.xml` со следующим содержимым:

```
<?xml version="1.0" encoding="utf-8"?>
<set
  xmlns:android="http://schemas.android.com/apk/res/a
ndroid" android:duration="5000"
  android:fillAfter="true"
  android:interpolator="@android:anim
/acceleratedecelerateinterpolator"
  android:shareInterpolator="false" >
  <scale
    android:fromXScale="1.0"
    android:fromYScale="1.0"
```

```

        android:pivotX="50%"
        android:pivotY="50%"
        android:toXScale="1.5"
        android:toYScale="1.5" />
<translate android:fromYDelta="80%p"
        android:toYDelta="10%p" />
<alpha
        android:fromAlpha="0.3"
        android:toAlpha="1.0" />
</set>

```

В блоке `set` установлены параметры анимации. Например, параметр `android:duration` показывает, что анимация должна совершиться в течение 5 секунд. Параметр `fill After` управляет состоянием анимации - она не должна прыгать в начало. Параметр `android:interpolator` использует системную константу для небольшого ускорения от начала к середине анимации и торможения от середины к концу анимации.

Внутри блока `set` устанавливаются специальные блоки, отвечающие за характер анимации: изменение размеров, позиции и прозрачности. Например, фигура солнца будет увеличиваться от своего начального размера в полтора раза равномерно от своей середины (`scale`). Элемент `translate` двигает солнце по экрану вертикально вверх. Мы отталкиваемся относительно родительского элемента, используя суффикс `"p"`. Солнце начинает движение в позиции 80% от родительского элемента по оси `Y` и заканчивает движение в позиции 10%. При движении также меняется прозрачность солнца от полной прозрачности до полной непрозрачности (`alpha`). Далее в файле `MainActivity.java` записываем программный код приложения:

```

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle; import
android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle
savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activitymain);
// Получаем ссылку на изображение солнца

```

```
ImageView sunImageView =
    (ImageView) findViewById(R.id.sun);
    // Анимация для восхода солнца Animation
sunRiseAnimation =
AnimationUtils.loadAnimation(this, R.anim.sun
rise);
    // Подключаем анимацию к нужному изображению View
sunImageView.startAnimation(sunRiseAnimation);
}
```

## Лабораторная работа

### 2D-анимация, использование служб в приложениях под Android

Цель работы: получить навыки создания Android-приложений с использованием 2D-анимации.

#### *Программа выполнения работы*

1. Изучить методические указания к практическому занятию.
2. Запустить на выполнение Android Studio.
3. Если будет использоваться внешний эмулятор для запуска Android-приложений (например, BlueStacks или Genymotion), загрузить его.
4. Создать в среде Android Studio проект, разработанный на практическом занятии.
5. Создать в соответствующих директориях проекта файлы grass.xml, sky.xml, sun.xml.
6. Добавить в проект графические изображения неподвижных и движущихся изображений.
7. Создать новую папку res/anim и добавить в нее файл sunrise.xml.
8. В окне java-кода проекта добавить строки для получения анимированного изображения.
9. Запустить созданное приложение в эмуляторе Android и наблюдать за появлением этого приложения и результатов его работы в окне приложений эмулятора.
10. Создать новое приложение с применением

анимированного изображения.

### *Содержание отчета*

- 1 . Название и цель работы.
- 2 . Листинг созданного приложения в Android Studio с использованием элементов графики и анимации.
- 3 . Результаты работы приложения в эмуляторе телефона.
- 4 . Выводы по проделанной работе.

### *Примерные варианты заданий*

1.	Создать приложение, которое рисует движение солнца на фоне неба. Снизу изобразить зеленую поляну.
2.	Создать приложение, которое рисует движение ядра, которое выстреливается из пушки под углом $\alpha$ и скоростью $V_0$ . Изобразить стену, в которую может попасть ядро. Траектория движения ядра должна быть построена с учетом силы тяготения Земли.
3.	Создать приложение, которое рисует движение автомобиля по дороге на фоне неба.
4.	Создать приложение, которое рисует движение человека по дороге на фоне домов.
5.	Создать приложение, которое рисует движение автомобиля через перекресток.
6.	Создать приложение, которое рисует движение двух автомобилей через перекресток вдоль одного направления.
7.	Создать приложение, которое рисует движение двух автомобилей через перекресток в перпендикулярном направлении.
8.	Создать приложение, которое рисует движение Колобка по зеленой поляне к домику.
9.	Создать приложение, которое рисует полет самолета на фоне неба с облаком.
10.	Создать приложение, которое выполняет вывод на экран изображения с камеры и сохраняет его в файле
11.	Создать приложение, которое выполняет вывод на экран изображения с камеры и сохраняет в файле выделенную область.
12.	Создать приложение, которое выполняет геолокацию с помощью сервисов Google, используя объект Location
13.	Сделать HTTP-запрос в Android, используя <u>HttpURLConnection</u> .
14.	Создать приложение, которое рисует движение бильярдного

	шара с отражениями от бортов.
15.	Создать приложение, которое рисует движение часовой и секундной стрелок часов.

### *Контрольные вопросы*

1. Каковы этапы жизненного цикла служб?
2. Как создать службу в Android-приложении?
3. Какие существуют способы вызова службы?
4. Как зарегистрировать службу в файле манифеста приложения?
5. Приведите пример создания анимации в XML и в коде.

## **ТЕМА №5 РАБОТА С ФАЙЛАМИ В ANDROID**

### **Практическое занятие**

#### **Работа с файлами и сохранение пользовательских настроек в приложениях под Android**

#### *Методические указания*

ОС Android построена на основе Linux. Этот факт находит свое отражение в работе с файлами. Так, в путях к файлам в качестве разграничителя в Linux использует слеш "/", а не обратный слеш "\" (как в Windows). А все названия файлов и каталогов являются регистрозависимыми, то есть "data" это не то же самое, что и "Data".

Приложение Android сохраняет свои данные в каталоге /data/data/<название\_пакета>/ и, как правило, относительно этого каталога будет идти работа.

Для работы с файлами абстрактный класс `android.content.Context` определяет ряд методов:

- `boolean deleteFile (String name)`: удаляет определенный файл
- `String[] fileList ()`: получает все файлы, которые содержатся в подкаталоге `/files` в каталоге приложения
- `File getCacheDir()`: получает ссылку на подкаталог `cache` в каталоге приложения
- `File getDir(String dirName, int mode)`: получает ссылку на подкаталог в каталоге приложения, если такого подкаталога нет, то он создается
- `File getExternalCacheDir()`: получает ссылку на папку `/cache` внешней файловой системы устройства
- `File getExternalFilesDir(String type)`: получает ссылку на каталог `/files` внешней файловой системы устройства
- `File getFilePath(String filename)`: возвращает абсолютный путь к файлу в файловой системе
- `FileInputStream openFileInput(String filename)`: открывает файл для чтения
- `FileOutputStream openFileOutput (String name, int mode)`: открывает файл для записи

Все файлы, которые создаются и редактируются в приложении, как правило, хранятся в подкаталоге `/files` в каталоге приложения.

Для непосредственного чтения и записи файлов применяются также стандартные классы Java из пакета `java.io`.

Итак, применим функционал чтения-записи файлов в приложении. Пусть у нас будет следующая примитивная разметка `layout`:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/editor"
        android:layout_width="0dp"
```



```

        android:layout_height="0dp"
        android:textSize="18sp"
        android:gravity="start"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toTopOf="@id/save_text"
        app:layout_constraintTop_toTopOf="parent" />
<Button
    android:id="@+id/save_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:onClick="saveText"
    android:text="Сохранить"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintBottom_toTopOf="@id/text"
    app:layout_constraintTop_toBottomOf="@id/editor" />

<TextView
    android:id="@+id/text"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:gravity="start"
    android:textSize="18sp"

    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintBottom_toTopOf="@+id/open_text"
    app:layout_constraintTop_toBottomOf="@+id/save_text"
/>
<Button
    android:id="@+id/open_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="openText"
    android:text="Открыть"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/text" />
</androidx.constraintlayout.widget.ConstraintLayout>

```

Поле `EditText` предназначено для ввода текста, а `TextView` - для вывода ранее сохраненного текста. Для сохранения и восстановления текста добавлены две кнопки.

Теперь в коде Activity пропишем обработчики кнопок с сохранением и чтением файла:

```
package com.example.filesapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {
    private final static String FILE_NAME = "content.txt";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    // сохранение файла
    public void saveText(View view){

        FileOutputStream fos = null;
        try {
            EditText textBox = (EditText)
findViewById(R.id.editor);
            String text = textBox.getText().toString();
            fos = openFileOutput(FILE_NAME, MODE_PRIVATE);
            fos.write(text.getBytes());
            Toast.makeText(this, "Файл сохранен",
Toast.LENGTH_SHORT).show();
        }
        catch(IOException ex) {
            Toast.makeText(this, ex.getMessage(),
Toast.LENGTH_SHORT).show();
        }
        finally{
            try{
                if(fos!=null)
                    fos.close();
            }
            catch(IOException ex){
```

```

        Toast.makeText(this, ex.getMessage(),
Toast.LENGTH_SHORT).show();
    }
}
// открытие файла
public void openText(View view){

    FileInputStream fin = null;
    TextView textView = (TextView)
findViewById(R.id.text);
    try {
        fin = openFileInput(FILE_NAME);
        byte[] bytes = new byte[fin.available()];
        fin.read(bytes);
        String text = new String (bytes);
        textView.setText(text);
    }
    catch(IOException ex) {
        Toast.makeText(this, ex.getMessage(),
Toast.LENGTH_SHORT).show();
    }
    finally{

        try{
            if(fin!=null)
                fin.close();
        }
        catch(IOException ex){

            Toast.makeText(this, ex.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    }
}
}
}

```

При нажатии на кнопку сохранения будет создаваться поток вывода `FileOutputStream fos = openFileOutput(FILE_NAME, MODE_PRIVATE)`

В данном случае введенный текст будет сохраняться в файл "content.txt". При этом будет использоваться режим `MODE_PRIVATE`

Система позволяет создавать файлы с двумя разными режимами:

- `MODE_PRIVATE`: файлы могут быть доступны только владельцу приложения (режим по умолчанию)

- `MODE_APPEND`: данные могут быть добавлены в конец файла

Поэтому в данном случае если файл "content.txt" уже существует, то он будет перезаписан. Если же нам надо было дописать файл, тогда надо было бы использовать режим `MODE_APPEND`:

```
FileOutputStream fos = openFileOutput(FILE_NAME,  
MODE_APPEND);
```

Для чтения файла применяется поток ввода `FileInputStream`:

```
FileInputStream fin = openFileInput(FILE_NAME);
```

Подробнее про использование потоков ввода-вывода можно прочитать в руководстве по Java:

<https://metanit.com/java/tutorial/6.3.php>

В итоге после нажатия кнопки сохранения весь текст будет сохранен в файле

`/data/data/название_пакета/files/content.txt`

## Лабораторная работа

### Работа с файлами в приложениях под Android

*Цель работы:* получить навыки создания Android-приложений с использованием файлов.

#### *Программа выполнения работы*

1. Изучить методические указания к практическому занятию.
2. Запустить на выполнение Android Studio.
3. Создать в среде Android Studio проект, разработанный на практическом занятии.
4. Добавить в проект элементы, обеспечивающие работу с файлами согласно индивидуальному заданию.
5. Запустить созданное приложение в эмуляторе Android и наблюдать за появлением этого приложения и результатов

его работы в окне приложений эмулятора.

### *Содержание отчета*

- 1 . Название и цель работы.
- 2 . Листинг созданного приложения в Android Studio.
- 3 . Результаты работы приложения в эмуляторе телефона.

Использовать варианты задания для лабораторной работы по теме 2, но для сохранения вводимых значений и результата следует применить файлы.

### *Контрольные вопросы*

1. Какие методы для работы с файлами содержит абстрактный класс `android.content.Context` ?
2. Каково назначение предпочтений?
3. Для чего используется предпочтение `CheckBoxPreference`?
4. Для чего используется предпочтение `EditTextPreference`?

## **ТЕМА №6 РАБОТА С БАЗАМИ ДАННЫХ В ANDROID**

### **Практическое занятие**

#### **База данных SQLite и контент-провайдеры**

*Цель занятия:* изучить компонент `GridView` для вывода элементов в виде таблицы; знакомство с возможностями использования СУБД `SQLite` в Android- приложениях.

### *Методические указания*

Создадим простое приложение – справочник контактов,

которое будет хранить имя и email. Вводить данные будем на экране приложения, а для отображения информации используем логи.

Файл main.xml, где содержится описание экрана, содержащего пару полей для ввода, а также кнопки добавления записи, вывода существующих записей и очистки таблицы.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical">
<LinearLayout
android:id="@+id/linearLayout1"
android:layout_width="match_parent"
android:layout_height="wrap_content">
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Name"
android:layout_marginLeft="5dp"
android:layout_marginRight="5dp">
</TextView>
<EditText
android:id="@+id/etName"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_weight="1">
<requestFocus>
</requestFocus>
</EditText>
</LinearLayout>
<LinearLayout
android:id="@+id/linearLayout3"
android:layout_width="match_parent"
android:layout_height="wrap_content">
<TextView
android:id="@+id/textView2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Email"
android:layout_marginLeft="5dp"
android:layout_marginRight="5dp">
</TextView>
<EditText
android:id="@+id/etEmail"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```

android:layout_weight="1">
</EditText>
</LinearLayout>
<LinearLayout
android:id="@+id/linearLayout2"
android:layout_width="match_parent"
android:layout_height="wrap_content">
<Button
android:id="@+id/btnAdd"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Add">
</Button>
<Button
android:id="@+id/btnRead"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Read">
</Button>
<Button
android:id="@+id/btnClear"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Clear">
</Button>
</LinearLayout>
</LinearLayout>

```

**Файл MainActivity.java** отредактируем для получения следующего:

```

import android.app.Activity;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends Activity implements
OnClickListener {

final String LOG_TAG = "myLogs";

Button btnAdd, btnRead, btnClear;

```

```

EditText etName, etEmail;

DBHelper dbHelper;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    btnAdd = (Button) findViewById(R.id.btnAdd);
    btnAdd.setOnClickListener(this);

    btnRead = (Button) findViewById(R.id.btnRead);
    btnRead.setOnClickListener(this);

    btnClear = (Button) findViewById(R.id.btnClear);
    btnClear.setOnClickListener(this);

    etName = (EditText) findViewById(R.id.etName);
    etEmail = (EditText) findViewById(R.id.etEmail);

    // создаем объект для создания и управления версиями БД
    dbHelper = new DBHelper(this);
}

@Override
public void onClick(View v) {

    // создаем объект для данных
    ContentValues cv = new ContentValues();

    // получаем данные из полей ввода
    String name = etName.getText().toString();
    String email = etEmail.getText().toString();

    // подключаемся к БД
    SQLiteDatabase db = dbHelper.getWritableDatabase();

    switch (v.getId()) {
        case R.id.btnAdd:
            Log.d(LOG_TAG, "--- Insert in mytable: ---");
            // подготовим данные для вставки в виде пар: наименование
            // столбца - значение

            cv.put("name", name);
            cv.put("email", email);
            // вставляем запись и получаем ее ID
            long rowID = db.insert("mytable", null, cv);

```



```

Log.d(LOG_TAG, "row inserted, ID = " + rowID);
break;
case R.id.btnRead:
Log.d(LOG_TAG, "--- Rows in mytable: ---");
// делаем запрос всех данных из таблицы mytable, получаем
Cursor
Cursor c = db.query("mytable", null, null, null, null, null,
null);

// ставим позицию курсора на первую строку выборки
// если в выборке нет строк, вернется false
if (c.moveToFirst()) {

// определяем номера столбцов по имени в выборке
int idColIndex = c.getColumnIndex("id");
int nameColIndex = c.getColumnIndex("name");
int emailColIndex = c.getColumnIndex("email");

do {
// получаем значения по номерам столбцов и пишем все в лог
Log.d(LOG_TAG,
"ID = " + c.getInt(idColIndex) +
", name = " + c.getString(nameColIndex) +
", email = " + c.getString(emailColIndex));
// переход на следующую строку
// а если следующей нет (текущая - последняя), то false -
выходим из цикла
} while (c.moveToNext());
} else
Log.d(LOG_TAG, "0 rows");
c.close();
break;
case R.id.btnClear:
Log.d(LOG_TAG, "--- Clear mytable: ---");
// удаляем все записи
int clearCount = db.delete("mytable", null, null);
Log.d(LOG_TAG, "deleted rows count = " + clearCount);
break;
}
// закрываем подключение к БД
dbHelper.close();
}

class DBHelper extends SQLiteOpenHelper {

public DBHelper(Context context) {
// конструктор суперкласса
super(context, "myDB", null, 1);
}

@Override

```

```

public void onCreate(SQLiteDatabase db) {
Log.d(LOG_TAG, "--- onCreate database ---");
// создаем таблицу с полями
db.execSQL("create table mytable ("
+ "id integer primary key autoincrement,"
+ "name text,"
+ "email text" + ");");
}

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

}
}
}

```

В приведенном коде для работы с БД использовано два класса:

- DBHelper, наследующий SQLiteOpenHelper. В его конструкторе вызываем конструктор супер-класса и указываем имя и версию БД. Метод getWritableDatabase выполняет подключение к базе данных и возвращает объект SQLiteDatabase для работы с ней. Метод close закрывает подключение к БД. В случае, когда БД отсутствует или устарела, класс предоставляет нам самим реализовать создание или обновление в методах onCreate и onUpgrade.

- SQLiteDatabase . который содержит методы для работы с данными(вставка, обновление, удаление и чтение).

В методе onCreate Activity определяем объекты, присваиваем обработчики и создаем объект dbHelper класса DBHelper для управления БД. Сам класс будет описан ниже.

Далее смотрим метод Activity – onClick, в котором обрабатываем нажатия на кнопки.

Класс ContentValues используется для указания *полей* таблицы и *значений*, которые мы в эти поля будем вставлять. Создан объект cv этого класса.

Далее записываем в переменные значения из полей ввода. Затем, с помощью метода getWritableDatabase подключаемся к БД и получаем объект SQLiteDatabase. Он позволит нам работать с БД. Мы будем использовать его методы insert – вставка записи, query – чтение, delete – удаление. У них много

разных параметров на вход, но пока используем самый минимум.

Далее проверяется, какая кнопка была нажата:

`btnAdd` – добавление записи в таблицу *mytable*. Заполняем объект `cv` парами: имя поля и значение. В указанные поля будут вставлены соответствующие значения. Далее заполняем поля *name* и *email*. Поле первичного ключа *id* заполнится автоматически (`primary key autoincrement`). Вызываем метод `insert`, где передаем ему имя таблицы и объект `cv` с вставляемыми значениями. Вторым аргументом метода используется, при вставке в таблицу пустой строки. Нам это сейчас не нужно, поэтому передаем `null`. Метод `insert` возвращает ID вставленной строки, мы его сохраняем `rowID` и выводим в лог.

`btnRead` – чтение всех записей из таблицы *mytable*. Для чтения используется метод `query`. На вход ему подается имя таблицы, список запрашиваемых полей, условия выборки, группировка, сортировка. Так как нужны все данные во всех полях без сортировок и группировок, то используем везде `null`. Указываем только имя таблицы. Метод возвращает нам объект класса `Cursor`. Его можно рассматривать как таблицу с данными. Метод `moveToFirst` – делает первую запись в `Cursor` активной и заодно проверяет, есть ли вообще записи в нем (т.е. выдалось ли что-либо в методе `query`). Далее мы получаем порядковые номера столбцов в `Cursor` по их именам с помощью метода `getColumnIndex`. Эти номера потом используем для чтения данных в методах `getInt` и `getString` и выводим данные в лог. С помощью метода `moveToNext` мы перебираем все строки в `Cursor` пока не добиремся до последней. Если же записей не было, то выводим в лог соответствующее сообщение – *0 rows*. В конце закрываем курсор (освобождаем занимаемые им ресурсы) методом `close`, т.к. далее мы его нигде не используем.

`btnClear` – очистка таблицы. Метод `delete` удаляет записи. На вход передаем имя таблицы и `null` в качестве условий для удаления, что значит удалится все. Метод возвращает количество удаленных записей.

После этого закрываем соединение с БД методом `close`.

Класс `DBHelper` является вложенным в `MainActivity` и описан в конце кода. Этот класс должен наследовать класс `SQLiteOpenHelper`.

В конструкторе вызываем конструктор суперкласса и передаем ему:

`context`-контекст;

`mydb`- название базы данных;

`null`– объект для работы с курсорами (пока не нужен, поэтому `null`);

`1` – версия базы данных.

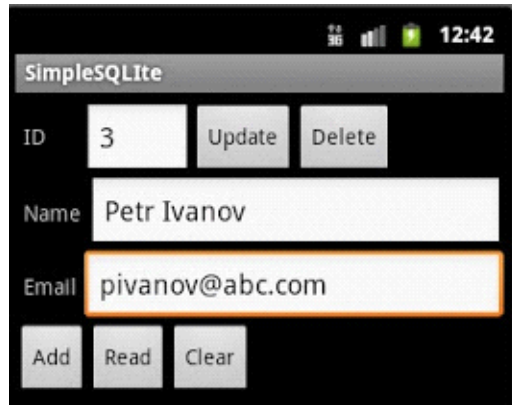
В методе `onCreate` этого класса использован метод `execSQL` объекта `SQLiteDatabase` для выполнения SQL-запроса, который создает таблицу. Этот метод вызывается, если БД не существует и ее надо создавать. В данном запросе создается таблица *mytable* с полями *id*, *name* и *email*.

Метод `onUpgrade` пока не заполняем, т.к. используем одну версию БД и менять ее не планируем.

Все сохраним и запустим приложение. Будем работать с БД и смотреть логи, которые покажут, какие методы выполняются, и что в них происходит.

Замечание. В примерах для упрощения все операции с базой данных выполняются в основном потоке. Но в реальных условиях следует использовать для работы с БД отдельный поток, чтобы ваше приложение не тормозило визуально.

Рассмотрим применение методов `update` и `delete` с указанием условия. Поменяем экран, добавим поле для ввода ID и кнопки для обновления и удаления. Примерный вид экрана показан ниже:



Для этого изменим файл main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical">
  <LinearLayout
    android:id="@+id/linearLayout4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp">
    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="ID"
      android:layout_marginLeft="5dp"
      android:layout_marginRight="25dp">
    </TextView>
    <EditText
      android:id="@+id/etID"
      android:layout_width="70dp"
      android:layout_height="wrap_content"
      android:layout_marginTop="2dp">
    </EditText>
    <Button
      android:id="@+id/btnUpd"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Update">
    </Button>
    <Button
      android:id="@+id/btnDel"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Delete">
```

```

        </Button>
    </LinearLayout>
    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Name"
            android:layout_marginLeft="5dp"
            android:layout_marginRight="5dp">
    </TextView>
    <EditText
        android:id="@+id/etName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1">
        <requestFocus>
    </requestFocus>
    </EditText>
</LinearLayout>
<LinearLayout
    android:id="@+id/linearLayout3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Email"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp">
    </TextView>
    <EditText
        android:id="@+id/etEmail"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1">
    </EditText>
</LinearLayout>
<LinearLayout
    android:id="@+id/linearLayout2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <Button
        android:id="@+id/btnAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Add">
    </Button>

```

```

        <Button
            android:id="@+id/btnRead"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Read">
    </Button>
    <Button
        android:id="@+id/btnClear"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Clear">
    </Button>
</LinearLayout>
</LinearLayout>

```

По нажатию кнопки Update мы будем читать содержимое полей Name и Email, и обновлять запись в таблице, для которой id = значению из поля ID. По нажатию кнопки Delete будем удалять запись из таблицы по id = значению из поля ID.

Подредактируем MainActivity.java. Добавим описание и определение новых экранных элементов, присвоение обработчиков для кнопок.

```

final String LOG_TAG = "myLogs";

Button btnAdd, btnRead, btnClear, btnUpd, btnDel;
EditText etName, etEmail, etID;

...

public void onCreate(Bundle savedInstanceState) {

    ...

    btnClear = (Button) findViewById(R.id.btnClear);
    btnClear.setOnClickListener(this);

    btnUpd = (Button) findViewById(R.id.btnUpd);
    btnUpd.setOnClickListener(this);

    btnDel = (Button) findViewById(R.id.btnDel);
    btnDel.setOnClickListener(this);

    etName = (EditText) findViewById(R.id.etName);
    etEmail = (EditText) findViewById(R.id.etEmail);
    etID = (EditText) findViewById(R.id.etID);

```

```

// создаем объект для создания и управления версиями БД
dbHelper = new DBHelper(this);
}

```

**Обращайте внимание на выделенные строки. Теперь дополним реализацию onClick:**

```

public void onClick(View v) {

    // создаем объект для данных
    ContentValues cv = new ContentValues();

    // получаем данные из полей ввода
    String name = etName.getText().toString();
    String email = etEmail.getText().toString();
    String id = etID.getText().toString();

    // подключаемся к БД
    SQLiteDatabase db = dbHelper.getWritableDatabase();

    switch (v.getId()) {
    case R.id.btnAdd:
        ...
    case R.id.btnRead:
        ...
    case R.id.btnClear:
        ...
    case R.id.btnUpd:
        if (id.equalsIgnoreCase("")) {
            break;
        }
        Log.d(LOG_TAG, "--- Update mytable: ---");
        // подготовим значения для обновления
        cv.put("name", name);
        cv.put("email", email);
        // обновляем по id
        int updCount = db.update("mytable", cv, "id = ?",
            new String[] { id });
        Log.d(LOG_TAG, "updated rows count = " + updCount);
        break;
    case R.id.btnDel:
        if (id.equalsIgnoreCase("")) {
            break;
        }
        Log.d(LOG_TAG, "--- Delete from mytable: ---");
        // удаляем по id
        int delCount = db.delete("mytable", "id = " + id,
null);
        Log.d(LOG_TAG, "deleted rows count = " + delCount);
        break;
    }
}

```



```

    }
    // закрываем подключение к БД
    dbHelper.close();
}

```

Добавим переменную `id`, пишем в нее значение поля `etID`.

В операторе `switch` добавляем две новые ветки:

`btnUpd` – обновление записи в `mytable`. Проверяем, что значение `id` не пустое, заполняем `cv` данными для апдейта и обновляем запись. Для этого используется метод `update`. На вход ему подается имя таблицы, заполненный `ContentValues` с значениями для обновления, строка условия (`Where`) и массив аргументов для строки условия. В строке условия я использовал знак «?». При запросе к БД вместо этого знака будет подставлено значение из массива аргументов, в нашем случае это – значение переменной `id`. Если знаков «?» в строке условия несколько, то им будут сопоставлены значения из массива по порядку. Метод `update` возвращает нам количество обновленных записей, которое мы выводим в лог.

`btnDel` – удаление записи из `mytable`. Проверяем, что `id` не пустое и вызываем метод `delete`. На вход передаем имя таблицы, строку условия и массив аргументов для условия. Метод `delete` возвращает кол-во удаленных строк, которое мы выводим в лог.

Обратите внимание, что условия для `update` и для `delete` одинаковые, а именно *`id = значение из поля etID`*. Но реализованы они немного по-разному. Для `update` использован символ «?» в строке условия и массив аргументов. А для `delete` вставлено значение сразу в строку условия. Таким образом, здесь показаны способы формирования условия.

Все сохраним и запустим. Добавим пару записей, потом нажмем `Read`. В логе можно увидеть подобную запись:

```

ID = 1, name = Ivan Petrov, email = ipetrov @abc.com
ID = 2, name = Anton Sidorov, email = asidorov @def.com

```

Теперь попробуем обновить запись с `ID=1`. Для этого вводим 1 в поле `ID` и новые данные в поля `Name` и `Email`:

Полный код `MainActivity.java`:

```

import android.app.Activity;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends Activity implements
OnClickListener {

    final String LOG_TAG = "myLogs";

    Button btnAdd, btnRead, btnClear, btnUpd, btnDel;
    EditText etName, etEmail, etID;

    DBHelper dbHelper;

    /** Called when the activity is first created. */
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        btnAdd = (Button) findViewById(R.id.btnAdd);
        btnAdd.setOnClickListener(this);

        btnRead = (Button) findViewById(R.id.btnRead);
        btnRead.setOnClickListener(this);

        btnClear = (Button) findViewById(R.id.btnClear);
        btnClear.setOnClickListener(this);

        btnUpd = (Button) findViewById(R.id.btnUpd);
        btnUpd.setOnClickListener(this);

        btnDel = (Button) findViewById(R.id.btnDel);
        btnDel.setOnClickListener(this);

        etName = (EditText) findViewById(R.id.etName);
        etEmail = (EditText) findViewById(R.id.etEmail);
        etID = (EditText) findViewById(R.id.etID);

        // создаем объект для создания и управления версиями БД
        dbHelper = new DBHelper(this);
    }

```

```

public void onClick(View v) {

    // создаем объект для данных
    ContentValues cv = new ContentValues();

    // получаем данные из полей ввода
    String name = etName.getText().toString();
    String email = etEmail.getText().toString();
    String id = etID.getText().toString();

    // подключаемся к БД
    SQLiteDatabase db = dbHelper.getWritableDatabase();

    switch (v.getId()) {
    case R.id.btnAdd:
        Log.d(LOG_TAG, "--- Insert in mytable: ---");
        // подготовим данные для вставки в виде пар:
наименование столбца -
        // значение
        cv.put("name", name);
        cv.put("email", email);
        // вставляем запись и получаем ее ID
        long rowID = db.insert("mytable", null, cv);
        Log.d(LOG_TAG, "row inserted, ID = " + rowID);
        break;
    case R.id.btnRead:
        Log.d(LOG_TAG, "--- Rows in mytable: ---");
        // делаем запрос всех данных из таблицы mytable,
получаем Cursor
        Cursor c = db.query("mytable", null, null, null, null,
null, null);

        // ставим позицию курсора на первую строку выборки
        // если в выборке нет строк, вернется false
        if (c.moveToFirst()) {

            // определяем номера столбцов по имени в выборке
            int idColIndex = c.getColumnIndex("id");
            int nameColIndex = c.getColumnIndex("name");
            int emailColIndex = c.getColumnIndex("email");

            do {
                // получаем значения по номерам столбцов и пишем
все в лог
                Log.d(LOG_TAG,
                    "ID = " + c.getInt(idColIndex) + ", name = "
                    + c.getString(nameColIndex) + ", email = "
                    + c.getString(emailColIndex));
                // переход на следующую строку
                // а если следующей нет (текущая - последняя), то

```

```

false -
    // выходим из цикла
    } while (c.moveToNext());
    } else
        Log.d(LOG_TAG, "0 rows");
    c.close();
    break;
case R.id.btnClear:
    Log.d(LOG_TAG, "--- Clear mytable: ---");
    // удаляем все записи
    int clearCount = db.delete("mytable", null, null);
    Log.d(LOG_TAG, "deleted rows count = " + clearCount);
    break;
case R.id.btnUpd:
    if (id.equalsIgnoreCase("")) {
        break;
    }
    Log.d(LOG_TAG, "--- Update mytable: ---");
    // подготовим значения для обновления
    cv.put("name", name);
    cv.put("email", email);
    // обновляем по id
    int updCount = db.update("mytable", cv, "id = ?",
        new String[] { id });
    Log.d(LOG_TAG, "updated rows count = " + updCount);
    break;
case R.id.btnDel:
    if (id.equalsIgnoreCase("")) {
        break;
    }
    Log.d(LOG_TAG, "--- Delete from mytable: ---");
    // удаляем по id
    int delCount = db.delete("mytable", "id = " + id,
null);
    Log.d(LOG_TAG, "deleted rows count = " + delCount);
    break;
    }
    // закрываем подключение к БД
    dbHelper.close();
}

class DBHelper extends SQLiteOpenHelper {

    public DBHelper(Context context) {
        // конструктор суперкласса
        super(context, "myDB", null, 1);
    }

    public void onCreate(SQLiteDatabase db) {
        Log.d(LOG_TAG, "--- onCreate database ---");
        // создаем таблицу с полями

```

```

        db.execSQL("create table mytable ("
            + "id integer primary key autoincrement,"
            + "name text,"
            + "email text" + ");");
    }

    public void onUpgrade(SQLiteDatabase db, int oldVersion,
int newVersion) {

        }
    }
}

```

Рассмотрим компонент GridView, который выводит элементы в виде таблицы с использованием адаптера.

Сделаем простой пример, где рассмотрим атрибуты этого компонента. В файл main.xml поместим GridView

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <GridView
        android:id="@+id/gvMain"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    </GridView>
</LinearLayout>

```

Создадим файл rect.xml в любой папке res/drawable

```

<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="#99000099"></solid>
</shape>

```

Это просто прямоугольник, залитый синим цветом, который используется как фон. Создадим свой layout для адаптера – item.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout

```

```

xmlns:android="http://schemas.android.com/apk/res/androi
d"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@drawable/rect"
android:orientation="vertical">
<TextView
    android:id="@+id/tvText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:minHeight="40dp"
    android:textSize="20sp"
    android:text="">
</TextView>
</LinearLayout>

```

### Код MainActivity.java:

```

import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.GridView;

public class MainActivity extends Activity {

    String[] data = {"a", "b", "c", "d", "e", "f", "g", "h",
    "i", "j", "k"};

    GridView gvMain;
    ArrayAdapter<String> adapter;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        adapter = new ArrayAdapter<String>(this,
R.layout.item, R.id.tvText, data);
        gvMain = (GridView) findViewById(R.id.gvMain);
        gvMain.setAdapter(adapter);
        adjustGridView();
    }

    private void adjustGridView() {
    }
}

```

В приведенном выше коде определяем GridView и

создаем адаптер. В качестве layout для адаптера используем созданный item.xml, а tvText – это элемент, в который адаптер будет вставлять текст. Метод adjustGridView пока пустой, в нем будем кодить настройки Grid-a.

Посмотрим, какие для GridView есть атрибуты:

- numColumns – кол-во столбцов в сетке. Если его не задавать, то столбец будет по умолчанию один. Поменяем это свойство - укажем, например 3. Сделаем это в пустом пока что методе adjustGridView

```
private void adjustGridView() {  
    gvMain.setNumColumns(3);  
}
```

Сохраним и запустим. В итоге получилось три столбца. Это свойство также может иметь значение AUTO\_FIT.

- columnWidth (ширина столбца). Если ширина столбца явно указана, то кол-во столбцов рассчитывается исходя из ширины, доступной GridView, и ширины столбцов. Иначе, кол-во столбцов считается равным 2

Укажем кол-во столбцов = AUTO\_FIT, а ширину столбцов задавать пока не будем.

```
private void adjustGridView() {  
    gvMain.setNumColumns(GridView.AUTO_FIT);  
}
```

Теперь укажем явно ширину столбцов, пусть будет 50.

```
gvMain.setNumColumns(GridView.AUTO_FIT);  
gvMain.setColumnWidth(50);  
}
```

Видно, что в экран влезло 6 столбцов. Вы можете поизменять параметр ширины столбцов и убедиться, что их кол-во будет меняться.

Свойства horizontalSpacing и verticalSpacing - это горизонтальный и вертикальный отступы между ячейками. Пусть будет 5.

```
private void adjustGridView() {  
    gvMain.setNumColumns(GridView.AUTO_FIT);  
    gvMain.setColumnWidth(80);  
    gvMain.setVerticalSpacing(5);  
}
```

```
    gvMain.setHorizontalSpacing(5);  
}
```

Запустим приложение и увидим, что между ячейками появилось расстояние величиной 5.

Параметр `stretchMode` определяет, как будет использовано свободное пространство, если оно есть. Используется в случае, когда вы указываете ширину столбца и кол-во ставите в режим `AUTO_FIT`. Изменим наш метод, добавим туда настройку `stretch`-параметра.

```
private void adjustGridView() {  
    gvMain.setNumColumns(GridView.AUTO_FIT);  
    vMain.setColumnWidth(80);  
    gvMain.setVerticalSpacing(5);  
    gvMain.setHorizontalSpacing(5);  
    gvMain.setStretchMode(GridView.NO_STRETCH);  
}
```

Параметр `stretchMode` может принимать 4 значения:

`NO_STRETCH` – свободное пространство не используется. Столбцы выровнены по левому краю. Все свободное пространство справа;

`STRETCH_COLUMN_WIDTH` – свободное пространство используется столбцами, это режим по умолчанию; столбцы растянуты по ширине. Она уже может не соответствовать той, что указана в `setColumnWidth`;

`STRETCH_SPACING` – свободное пространство равномерно распределяется между столбцами. Ширина столбцов неизменна. Увеличены интервалы между ними;

`STRETCH_SPACING_UNIFORM` – свободное пространство равномерно распределяется не только между столбцами, но и справа и слева. Ширина столбцов неизменна. Увеличены интервалы между ними и с боков.

Разумеется, все эти параметры можно задавать не только программно, но и через атрибуты в `layout`-файлах. Вместо `ArrayAdapter` можно использовать любой другой. Можно прикрутить обработчик `setOnItemClickListener` и получать позицию или `id` нажатого элемента. Все как в обычных списках.



## Лабораторная работа

### База данных SQLite и контент-провайдеры в приложениях под Android

*Цель работы:* получить навыки создания Android-приложений, использующих базу данных SQLite.

#### *Программа выполнения работы*

1. Изучить методические указания к практическому занятию.
2. Запустить на выполнение Android Studio.
3. Создать в среде Android Studio проект, разработанный на практическом занятии. Модифицировать его с учетом индивидуального задания.
4. Создать в соответствующих директориях файлы проекта.
5. Запустить созданное приложение в эмуляторе Android, сделать скриншоты экрана с результатами.

#### *Содержание отчета*

1. Название и цель работы.
2. Листинг созданного приложения в Android Studio.
3. Результаты работы приложения в эмуляторе телефона.

В разработанных приложениях интерфейс пользователя должен использовать таблицы GridView для представления данных на экране, а также другие необходимы виджеты. Для хранения данных и запросов использовать СУБД SQLite.

#### *Примерные варианты заданий*

1.	Разработать приложение «Домашняя библиотека»
2.	Разработать приложение «Домашняя бухгалтерия»
3.	Разработать приложение «Участники спортивных соревнований»
4.	Разработать приложение «Результаты аттестации студенческой группы»

5.	Разработать приложение «Информационная система салона-магазина по продаже мобильных устройств»
6.	Разработать приложение «Информационная система учета услуг салона красоты»
7.	Разработать приложение «Информационная система учета услуг в фитнес-центре»
8.	Разработать приложение «Информационная система учета продаж в продовольственном магазине»
9.	Разработать приложение «Информационная система учета работ в автомобильной мастерской»
10.	Разработать приложение «Информационная система магазина по продаже компьютеров и их комплектующих»
11.	Разработать приложение «Информационная система фестиваля художественной самодеятельности в университете»
12.	Разработать приложение «Информационная система подведения итогов спортивных соревнований по легкой атлетике»
13.	Разработать приложение «Информационная система агентства по продаже недвижимости»
14.	Разработать приложение «Информационная система учета потребления воды и электроэнергии в многоквартирном доме»
15.	Разработать приложение «Информационная система подведения итогов спартакиады университета»

### *Контрольные вопросы*

1. Каковы основные возможности базы данных SQLite?
2. Какие абстрактные методы содержатся в классе SQLiteOpenHelper?
3. Какие методы для управления базой данных содержатся в классе SQLiteDatabase?
4. Назначение контент-провайдера?
5. Как используется объект ContentProvider: напрямую или косвенно?
6. Этапы создания контент-провайдера?
7. Приведите пример запроса на добавление записи к базе данных SQLite.
8. Приведите пример запроса на чтение записи в базе данных SQLite.

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Хашими С., Коматинени С., Маклин Д. Разработка приложений для Android. – Санкт-Петербург: Питер, 2011. -736 с.

2. Харди Б., Филлипс Б., Стюарт К., Марсикано К. Android. Программирование для профессионалов. 2-е изд. - Санкт-Петербург: Питер, 2016. - 640 с

3. Меднике З., Дорнин Л., Мик Б., Накамура. -Москва Программирование под Android. 2-е изд. - СПб.: Питер, 2013. - 560 с.

4. Дейтел ГГ, Дейтел Х., Уолд А. Android для разработчиков. 3-е изд. - Санкт-Петербург: Питер, 2016. - 512 с.

5. Start Android - учебник по Android для начинающих и продвинутых [Электронный ресурс] Режим доступа: <https://startandroid.ru/ru/>

6. Введение в разработку приложений для ОС Android [Электронный ресурс] - Национальный Открытый Университет «ИНТУИТ» 2014г. Режим доступа: <https://www.intuit.ru/studies/courses/12643/1191/info>

7. Разработка приложений для смартфонов на ОС Android [Электронный ресурс] - Национальный Открытый Университет «ИНТУИТ». Режим доступа: <https://www.intuit.ru/studies/courses/12786/1219/info>

8. Сайт Александра Климова. Изучаем Android. [Электронный ресурс] Режим доступа: <http://developer.alexanderklimov.ru/android/>

Учебно-методическое издание

Зуев Владимир Алексеевич

## **РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ**

Методические указания  
к выполнению практических занятий и лабораторных работ

---

Отв. за выпуск Кузнецова И.И.

Подписано в печать 01.11.2019г.

Формат 60x84<sup>1</sup>/<sub>16</sub>. Бумага офсетная. Ризография.

Усл.-печ.л. 3,95 . Уч.-изд. л.4,25. Тираж 50 экз.

---

Южно-Российский государственный политехнический  
университет (НПИ) имени М.И. Платова

Адрес ун-та: 346428, г. Новочеркасск, ул. Просвещения, 132

Отпечатано в Шахтинском автодорожном институте (филиале)  
ЮРГПУ (НПИ) им. М.И. Платова