



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

## **Практическое занятие № 1/2ч.**

### **Разработка мобильных компонент анализа безопасности информационно-аналитических систем**

(наименование дисциплины (модуля) в соответствии с учебным планом)

Уровень бакалавриат

(бакалавриат, магистратура, специалитет)

Форма обучения очная

(очная, очно-заочная, заочная)

Направление(-я) подготовки 10.05.04 Информационно-аналитические системы безопасности

(код(-ы) и наименование(-я))

Институт комплексной безопасности и специального приборостроения ИКБСП

(полное и краткое наименование)

Кафедра КБ-4 «Прикладные информационные технологии»

(полное и краткое наименование кафедры, реализующей дисциплину (модуль))

Используются в данной редакции с учебного года 2021/22

(учебный год цифрами)

Проверено и согласовано «\_\_\_» 20\_\_ г.

(подпись директора Института/Филиала  
с расшифровкой)

Москва 2021 г.

## ОГЛАВЛЕНИЕ

1 УСТАНОВКА И НАСТРОЙКА ОПЕРАЦИОННОЙ СИСТЕМЫ, JAVA SDK (JDK), СРЕДЫ РАЗРАБОТКИ И ПАКЕТОВ РАЗРАБОТКИ ANDROID SDK.....	3
1.1 Java SDK (JDK) .....	3
1.2 Среда разработки и Android SDK.....	3
2 СОЗДАНИЕ AVD. ПЕРВОЕ ПРИЛОЖЕНИЕ. СТРУКТУРА ANDROID-ПРОЕКТА.....	9
2.1 Структура проекта. ....	9
2.2 Создание приложения .....	10
2.3 Структура проекта .....	14
2.4 Создание эмулятора.....	17
2.5 Запуск приложения.....	20
3 КОМПОНЕНТЫ ЭКРАНА И ИХ СВОЙСТВА. ....	21
3.1 XML.....	23
3.2 Изображение.....	25
4 ВИДЫ LAYOUT. КЛЮЧЕВЫЕ ОТЛИЧИЯ И СВОЙСТВА.....	26
4.1 Задание.....	27
4.1.1 LinearLayout .....	27
4.1.2 TableLayout.....	28
4.1.3 RelativeLayout.....	29
4.1.4 ConstraintLayout .....	30
4.2 Задание.....	32
5 LAYOUT-ФАЙЛ В ACTIVITY. СМЕНА ОРИЕНТАЦИИ ЭКРАНА. ....	34
5.1 Layout файл.....	34
5.2 Задание:.....	34
5.3 Ориентация экрана.....	35
5.4 Задание.....	37
6 ОБРАЩЕНИЕ ИЗ КОДА К ЭЛЕМЕНТАМ ЭКРАНА. ОБРАБОТЧИКИ СОБЫТИЙ. ....	38
7 ОБРАБОТЧИКИ СОБЫТИЙ НА ПРИМЕРЕ BUTTON.....	40

# 1 УСТАНОВКА И НАСТРОЙКА ОПЕРАЦИОННОЙ СИСТЕМЫ, JAVA SDK (JDK), СРЕДЫ РАЗРАБОТКИ И ПАКЕТОВ РАЗРАБОТКИ ANDROID SDK.

## 1.1 Java SDK (JDK)

Разработка приложений производится на языке Java или Kotlin. Требуется скачать и установить набор средств разработки SDK, называемое еще JDK.

Т.к. язык Kotlin не изучается в рамках программы института, разработка будет вестись с использованием языка Java. Для успешного прохождения данного курса требуются начальные знания Java.

Скачать данный набор возможно по адресу:

<https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>.

Выбор версии производится под операционную систему.

После установки рекомендуется перезагрузить компьютер.

## 1.2 Среда разработки и Android SDK

Android Studio является официально поддерживаемой Google IDE для разработки Android-приложений. Основанная на IntelliJ IDEA, Android Studio доступна под лицензией Apache 2.0. Google предоставляет Android Studio для Windows, Mac OS X и Linux. Загрузить данное программное обеспечение возможно с официальной страницы приложения: <https://developer.android.com/studio>. Перед установкой Android Studio требуется убедиться, что операционная система и компьютер соответствует следующим требованиям:

ОС Windows:

- Microsoft Windows 7/8/10 (32-разрядная или 64-разрядная версия);
- 4 ГБ оперативной памяти, рекомендуется 8 ГБ оперативной памяти;
- 2 ГБ свободного места на диске, рекомендуется 4 ГБ (500 МБ для IDE + 1,5 ГБ для SDK Android и образа эмулятора);

- Минимальное разрешение экрана 1280 на 800 пикселей;
- JDK 8;
- Для ускоренного эмулятора: 64-разрядная операционная система и процессор Intel с поддержкой функций Intel VT-x, Intel EM64T (Intel 64) и Execute Disable (XD).

Mac OS:

- Mac OS X 10.10 или выше, вплоть до 10.14;
- 4 ГБ оперативной памяти, рекомендуется 8 ГБ оперативной памяти;
- 2 ГБ свободного места на диске, рекомендуется 4 ГБ (500 МБ для IDE + 1,5

ГБ для Android SDK и образа эмулятора);

- Минимальное разрешение экрана 1280 на 800 пикселей;
- JDK 6.

ОС Linux:

- Стационарный компьютер с GNOME или KDE;

- 64-разрядное распределение, позволяющее запускать 32-разрядные приложения;

- Библиотека GNU C (glibc) версии 2.11 или новее;
- 4 ГБ оперативной памяти, рекомендуется 8 ГБ оперативной памяти;
- 2 ГБ свободного места на диске, рекомендуется 4 ГБ (500 МБ для IDE + 1,5

ГБ для Android SDK и образа эмулятора);

- минимальное разрешение экрана 1280 на 800 пикселей;
- JDK 8;
- для ускоренного эмулятора: процессор Intel с поддержкой функций

Intel VT-x, Intel EM64T (Intel 64) и Execute Disable (XD) или процессор AMD с поддержкой технологии виртуализации AMD (AMD-V).

После установки среды разработки требуется её запустить (рис.1.1):

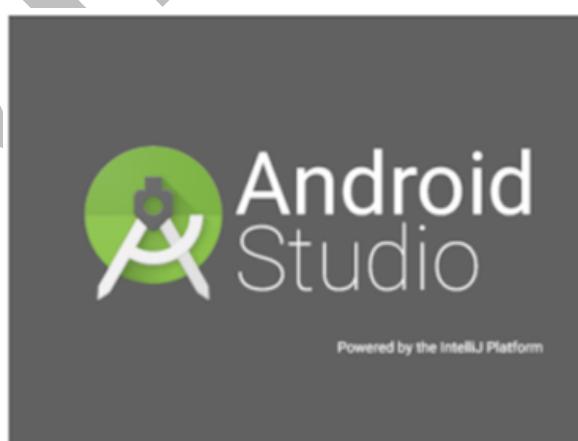


Рисунок 1.1 - Окно приветствия Android Studio

При первом запуске предлагается установить в диалоговых окнах несколько параметров конфигурации (рис.1.2). В первом диалоговом окне основное внимание уделяется импорту настроек из ранее установленной версии Android Studio:

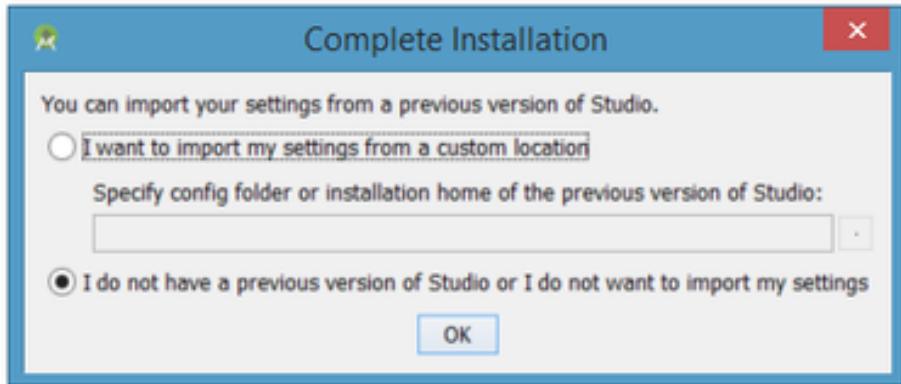


Рисунок 1.2 – Параметры импорта настроек

Возможно принять настройки по умолчанию и нажать на кнопку «OK». После этого Android Studio выведет диалоговое окно «Мастера установки» (рис.1.3).:

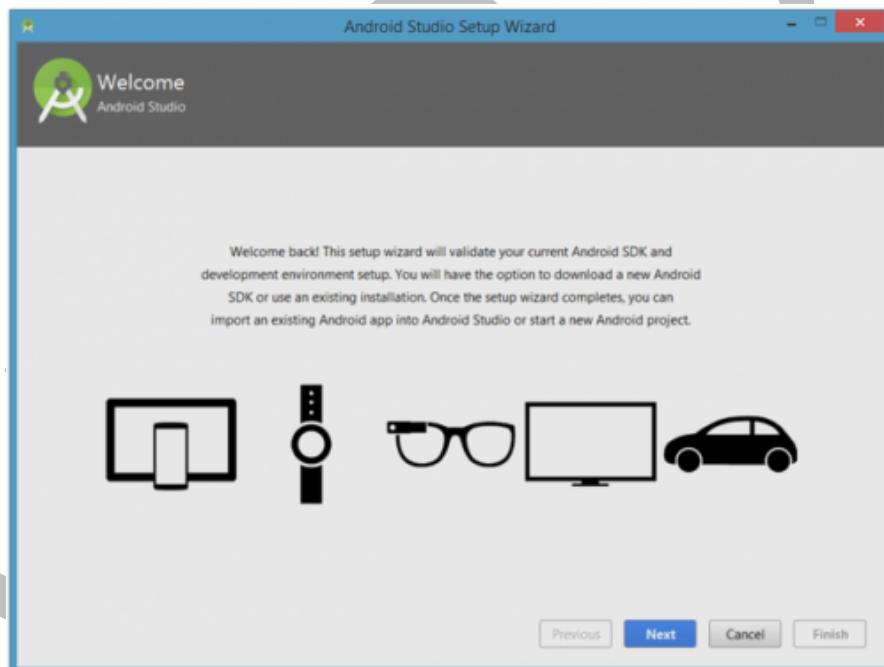


Рисунок 1.3 – Статус настроек Android SDK и среды разработки

После нажатия кнопки «Далее», «Мастер установки» предложит выбрать тип установки компонентов SDK (рис.1.4). На данный момент рекомендуется использовать стандартную конфигурацию:

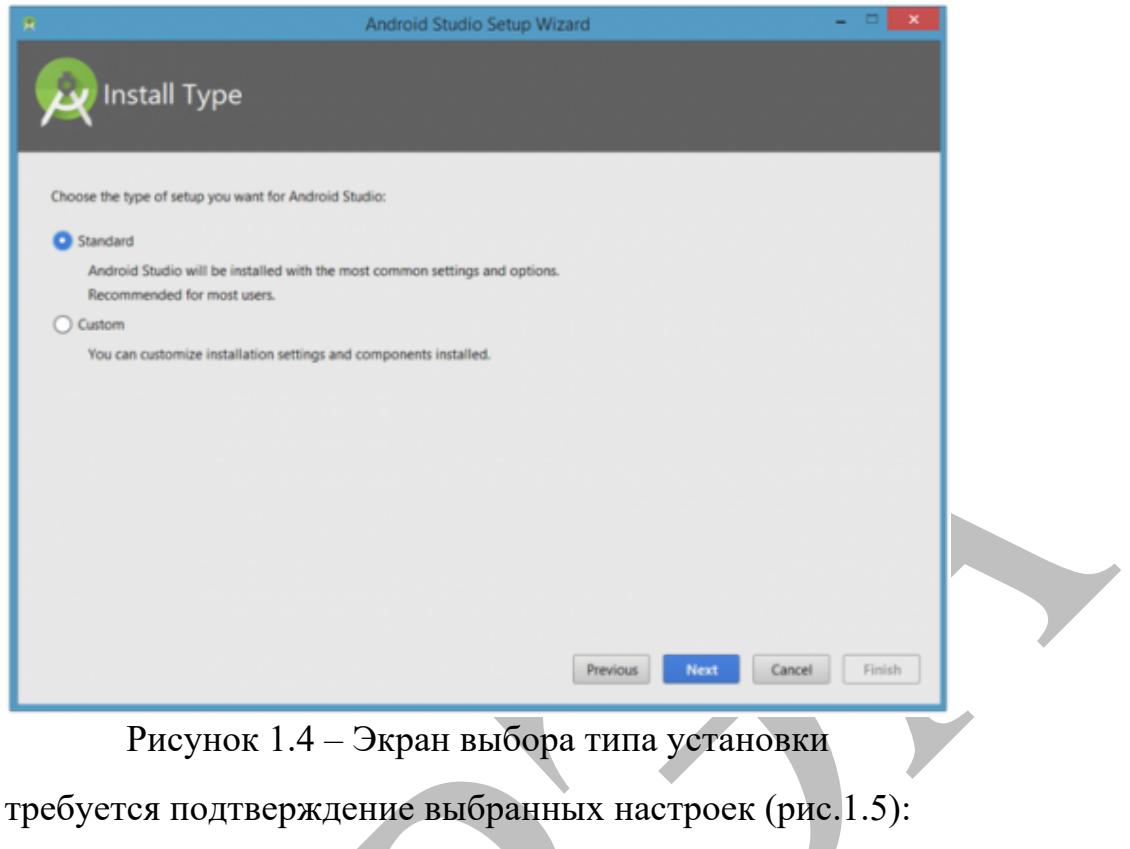


Рисунок 1.4 – Экран выбора типа установки

Далее требуется подтверждение выбранных настроек (рис.1.5):

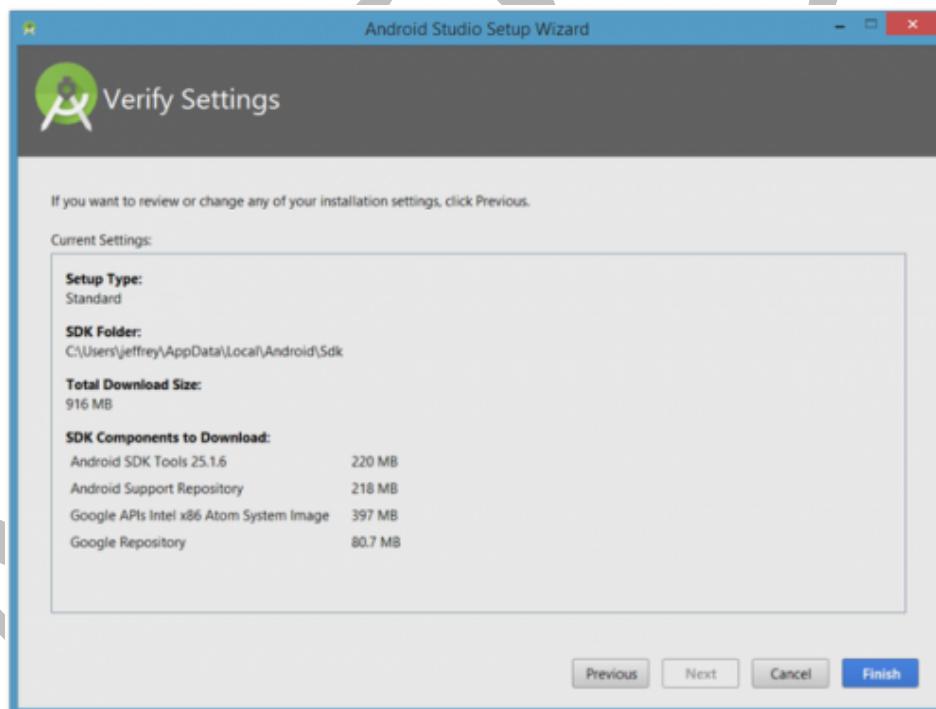


Рисунок 1.5 – Экран настроек среды разработки

«Мастер установки» произведёт загрузку и распаковку необходимых компонентов (рис.1.6). Кнопка «Показать детали» предназначена для отображения подробной информации о загружаемых архивах и их содержимом:

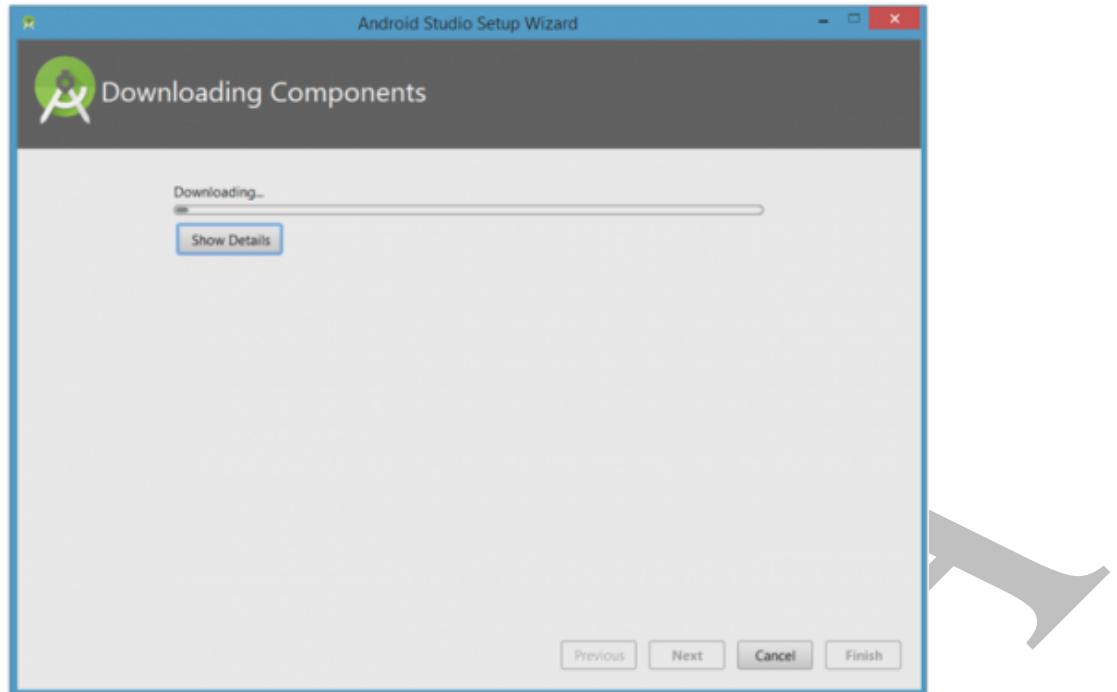


Рисунок 1.6 – Экран загрузки и установки компонентов среды разработки

После нажатия кнопки «Готово» производится завершение работы «Мастера установки» и отображение диалогового окна «Welcome to Android Studio» (рис.1.7):

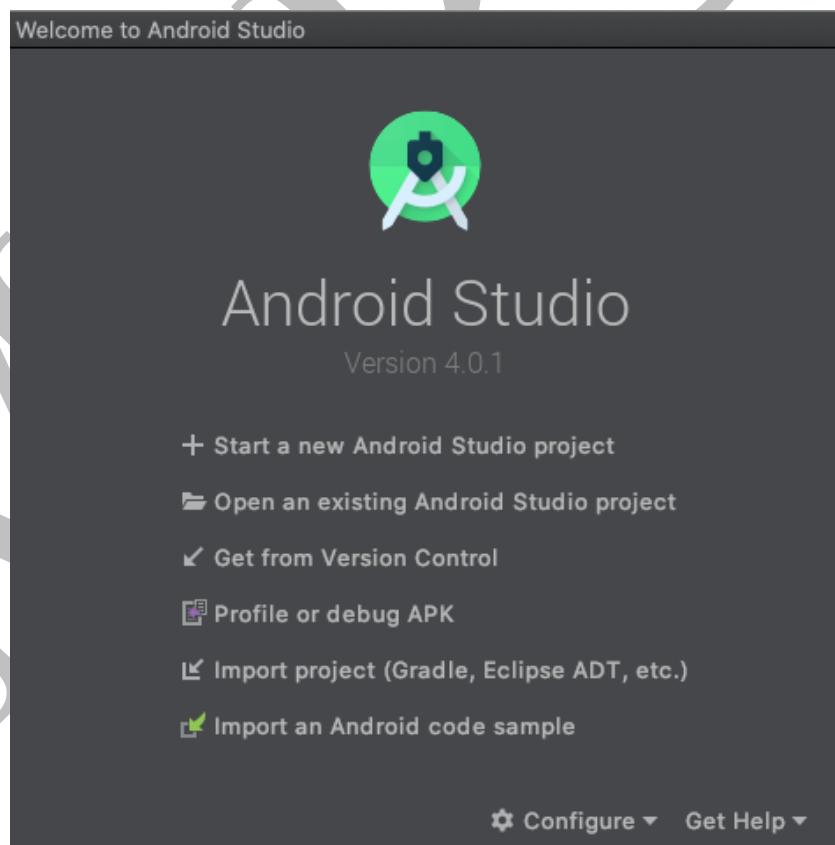


Рисунок 1.7 – Экран приветствия среды разработки

В случае, если компьютер собран не на базе процессора Intel, могут возникнуть сложности с запуском эмуляторов.

Возможные варианты решения проблемы – использовать существующие аналоги эмуляторов устройств или реальное Android-устройство для ускорения разработки.

Особенности среды разработки:

- единая среда, в которой возможно разрабатывать приложения для всех Android-устройств;
- возможность создания приложений под Android TV и Android Wear;
- «мастера» для создания общих макетов и компонентов Android, работающие на основе шаблонов;
- функциональный редактор макетов, который позволяет перетаскивать компоненты пользовательского интерфейса и включает в себя возможность предварительного просмотра макетов на нескольких экранах;
- рефакторинг для Android и быстрые исправления;
- поддержка разработки на основе Gradle;
- инструменты Lint для повышения производительности, юзабилити, устранения проблем связанных с совместимостью версий и другие;
- интеграция с ProGuard и возможность подписки на приложения;
- быстрый и многофункциональный эмулятор;
- instant Run для внесения изменений в запущенное приложение без создания нового файла APK (Application PacKage Zip);
- Встроенная поддержка облачной платформы Google для интеграции с Google Cloud Messaging и App Engine;
- C++ и NDK;
- Расширение возможностей Android Studio с помощью плагинов.

## 2 СОЗДАНИЕ AVD. ПЕРВОЕ ПРИЛОЖЕНИЕ. СТРУКТУРА ANDROID-ПРОЕКТА.

Для запуска и тестирований приложений требуется наличие эмулятора. Стандартный эмулятор, поставляемый вместе с Android SDK, называется Android Virtual Device (AVD). AVD – эмулятор Android-смартфона, на который возможно устанавливать созданные приложения, производить запуск и отладку.

### 2.1 Структура проекта.

Для создания приложение в Android Studio требуется создать проект, внутри которого создаётся модуль. В данном модуле непосредственно производится работа с кодом и внешним видом приложения. Результатом компиляции и сборки данного модуля является приложение. Поэтому модуль является приложением, а проект контейнером для модуля.

Самая простая структура проекта изображена на рисунке 2.1. При запуске проекта запускается модуль, результатом которого является Android-приложение, которое создано в данном модуле. Один проект является одним Android-приложением (один модуль).

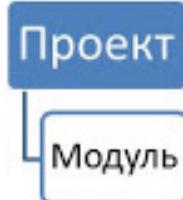


Рисунок 2.1 – Структура проекта

В одном проекте возможно существование нескольких модулей, а также имеется возможность создания нескольких проектов в одном рабочем окружении.



Рисунок 2.2 – Структура проекта с несколькими модулями

На рисунке 2.2 в первом проекте созданы два модуля, а во втором проекте – три модуля. При запуске какого-либо проекта необходимо указать, какой именно модуль требуется запустить. Каждый модуль является отдельным Android-приложением. Т.е. в данном случае: один проект содержит несколько Android-приложений (несколько модулей).

Для прохождения лабораторных и практических занятий требуется создавать один проект на одно занятие, в нем создавать модули для каждого задания.

## 2.2 Создание приложения

Требуется запустить Android Studio (рисунок 2.3):

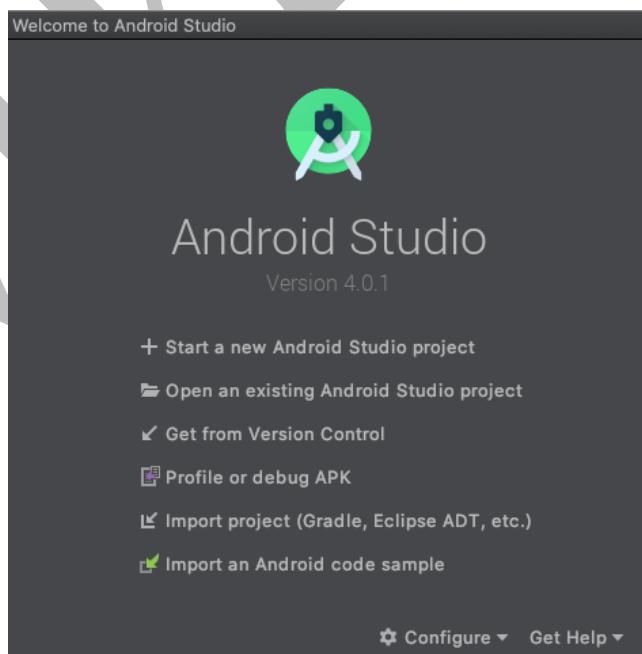


Рисунок 2.3 – Окно приветствия Android Studio

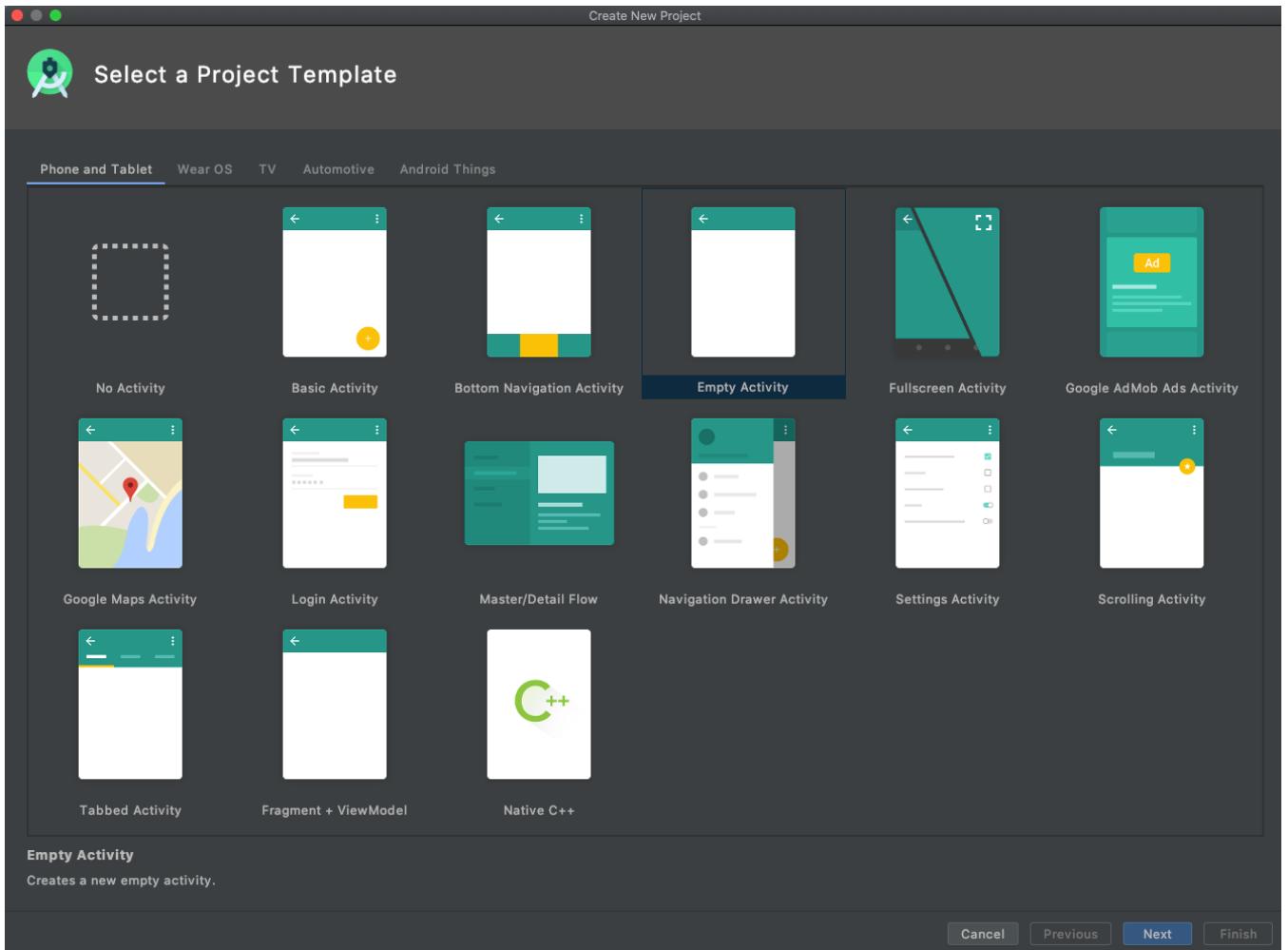


Рисунок 2.4 – Экран выбора макета приложения

На рис. 2.4 приведен экран выбора макета приложения. Некоторые из данных макетов будут рассмотрены в дальнейших занятиях. В занятии рассматривается Empty Activity.

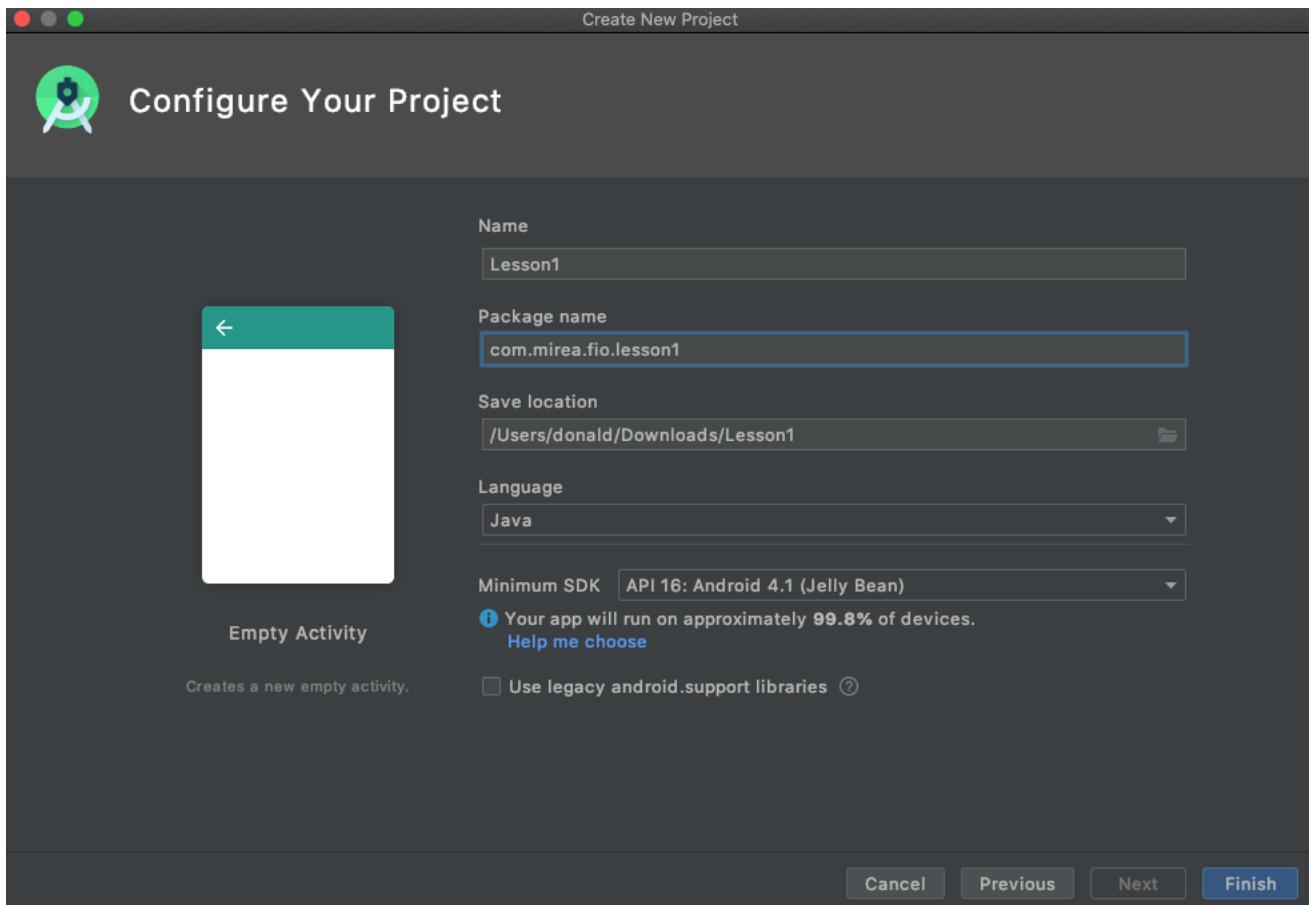


Рисунок 2.5 – Экран конфигурации проекта

После выбора требуемого макета требуется установить следующие настройки (рисунок 2.5):

1. *Name* – имя проекта, отображаемое в списке проектов при открытии Android Studio – «practice1».
2. *Package name* – полное название проекта (согласно правилам именования пакетов в Java). Имя пакета должно быть уникальным среди всех пакетов, установленных в системе Android. Требуется установить значение, с учетом имени приложения и домена. Домен может быть любой, не обязательно реально существующий. Требуется установить: *ru.mirea.«фамилия».practice1*
3. *Save location* – папка на компьютере, где находятся все файлы проекта.
4. *Language* – это язык программирования, Java или Kotlin, например.
5. *Minimum API level* указывает минимально поддерживаемую версию Android. Ниже информация о том, какой процент приложений ее поддерживает.
6. Кнопка *Finish* предназначена для создания проекта.

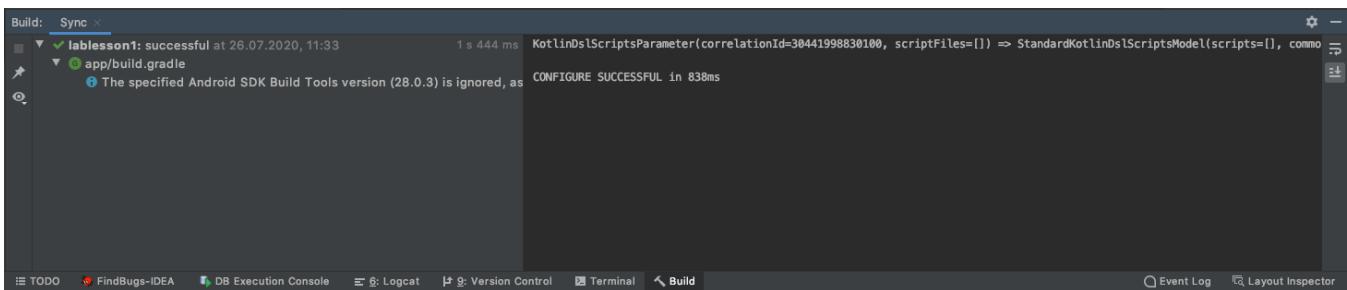


Рисунок 2.6 – Статус сборки проекта

Проект создан. В случае если в нижнем правом углу отображается прогресс бар, это означает, что еще выполняются действия по сборке проекта.

Далее требуется создать в проекте новый модуль, либо переименовать модуль app. Процедура частично похожа на создание проекта.

Для каждого занятия требуется создавать внутри нескольких модулей в проекте.

Для создания модуля в проекте требуется – в меню выбираем *File -> New -> New module*

Тип модуля *Phone and Tablet Application* (рис.2.7)

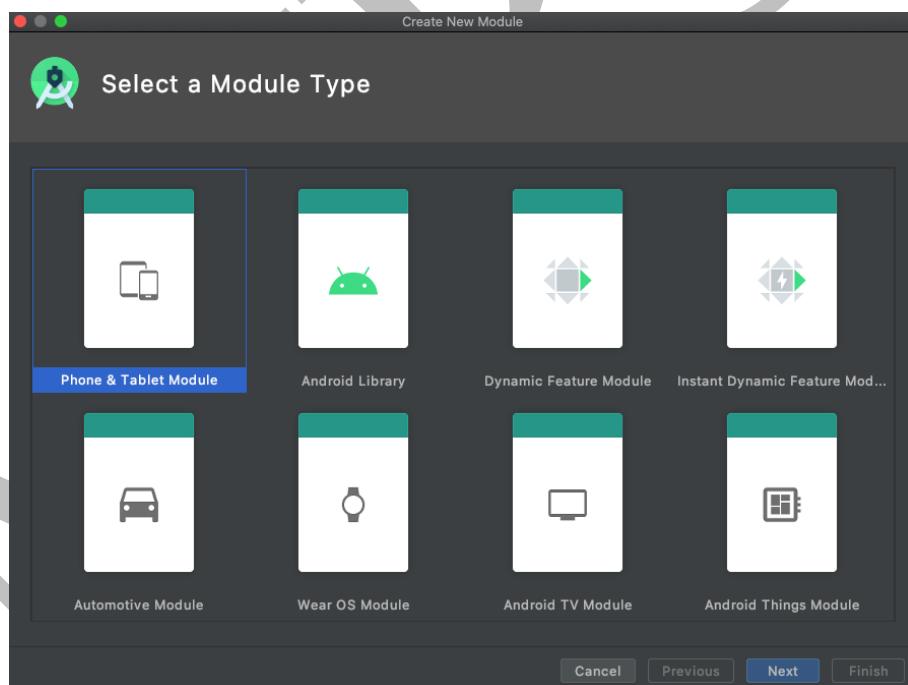


Рисунок 2.7 – Экран выбора типа модуля

*Application/Library name* – непосредственно имя приложения, которое будет отображаться в списке приложений в смартфоне. Требуется указать: Layout Type.

*Module name* – это название модуля, т.е. это название будет отображаться слева в списке модулей, там, где сейчас есть app.

*Package name* – имя пакета редактируется вручную, нажав edit справа. Он не требует внесение правок.

*Minimum SDK* не требует внесение правок.

*Empty activity* – без создания дополнительных элементов.

Далее все действия производятся аналогично созданию проекта.

### 2.3 Структура проекта

В левой панели Android studio размещена структура проекта (рис.2.8).

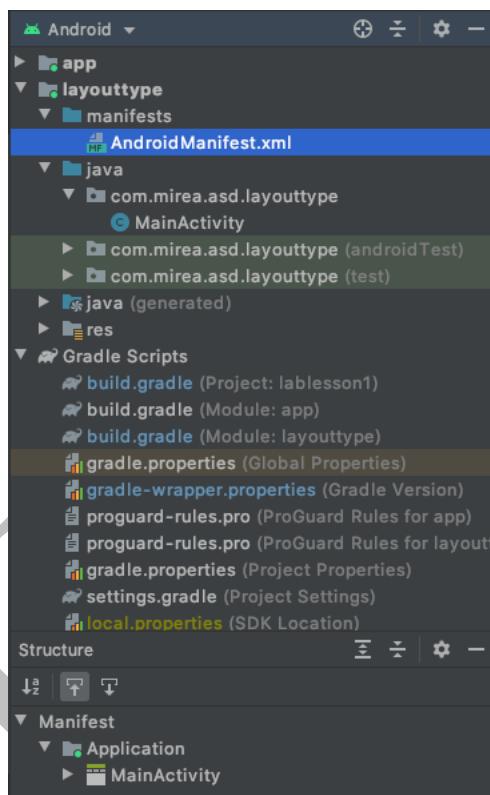


Рисунок 2.8 – Структура проекта

В *manifests* сгруппированы манифесты модулей. *AndroidManifest* — одна из ключевых частей приложения. Манифест описывает, из каких «частей» состоит приложение. Он представляет собой XML-файл, в котором описаны разрешения приложения, «экраны», сервисы, метаданные и т. д.

Далее в дереве размещен исходный код на *Java*, разделенный на две части — непосредственно код, и код для тестов. Также присутствует папка ресурсы (директория *res*) — в ней размещены изображения, текст, аудиофайлы и т.д.

В реалии структура Android-проекта выглядит иначе. Android Studio специально группирует файлы так, чтобы нам с ними было удобно работать.

Чтобы увидеть реальное дерево файлов, требуется выбрать вкладку «Project» (рис.2.9):

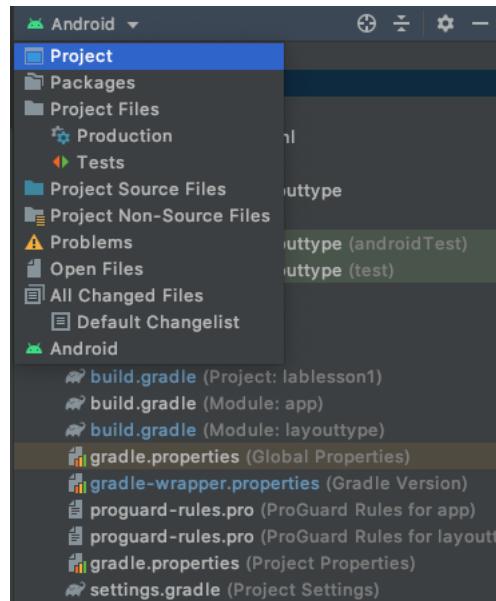


Рисунок 2.9 - Меню проекта

В Android Studio используется система сборки Gradle. В дереве проекта размещены несколько файлов с расширением «.gradle». Данные файлы описывают, каким образом производится сборка проекта. Файл с меткой **Project** отвечает за весь проект, остальные— за отдельные модули. В приложении может быть несколько модулей, и в таком случае у каждого из них будет свой «.gradle» файл.

Далее рассмотрен файл, отвечающий за сборку модуля:

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.2"

    defaultConfig {
        applicationId "com.mirea.asd.layouttype"
        minSdkVersion 15
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])

    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test:runner:1.2.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
}
```

Строчка *apply* является обозначением того, что модуль является приложением.

В случае, когда модуль является библиотекой, требуется установить «com.android.library».

Далее идет блок *android*:

– *compileSdkVersion* обозначает версию, под которую собирается приложение. Данный параметр в большинстве случаев должен быть равен последней доступной версии SDK. Версия инструментов для сборки. Аналогично, в большинстве случаев должна быть последней доступной.

– *buildToolsVersion* – версия инструментов для сборки. Аналогично, в большинстве случаев должна быть последней доступной.

– *applicationId* – имя пакета приложения. Должно быть уникальным во всей экосистеме. Если ранее было использовано данное имя пакета, то не будет возможности загрузить это приложение в Google Play, или установить на устройство, на котором уже установлено приложение с таким же именем пакета.

– *versionCode* – целочисленная обозначение версии приложения. При выгрузке обновления, например, в Google Play, должна обязательно инкрементироваться.

Далее расположен блок *dependencies*, в котором перечислены библиотеки, от которых зависит приложение. В данном блоке перечисляются на отдельных строчках зависимости от библиотек с версиями. При обновлении проекта, gradle производит загрузку всех перечисленных зависимостей. Так же возможно использование Jar-файлов, которые требуется разместить в директории *libs*:

## 2.4 Создание эмулятора

Для тестирования приложение возможно использование эмулятора Android устройства, встроенного в Android Studio. Требуется перейти в настройки эмулятора (рис.2.10). На рис.2.11 приведён экран менеджера виртуальных устройств, в котором отображаются созданные ранее виртуальные устройства и имеется возможность создать новое устройство. После нажатия кнопки создания устройства отобразится экран, приведенный на рис.2.12.



Рисунок 2.10 – Экран Android Studio

The screenshot shows the 'Android Virtual Device Manager' window. It has a header bar with the title 'Android Virtual Device Manager'. Below it is a section titled 'Your Virtual Devices' with the Android Studio logo. A table lists five virtual devices with columns for Type, Name, Play Store, Resolution, API, Target, CPU/ABI, Size on Disk, and Actions. At the bottom of the table is a red box highlighting the 'Create Virtual Device...' button, which is a grey button with white text. There are also buttons for '?', a refresh icon, and a close icon.

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
Emulator	Nexus 5X API 29 ...		1080 × 1920: 42...	29	Android 10.0 (Go...)	x86	6,6 GB	
Emulator	Pixel 2 API 26		1080 × 1920: 42...	26	Android 8.0 (Goo...)	x86	12 GB	
Emulator	Pixel 2 API 22		1080 × 1920: 42...	22	Android 5.1 (Goo...)	x86	2,5 GB	
Emulator	Pixel 2 API 26 2		1080 × 1920: 42...	26	Android 8.0 (Goo...)	x86	9,0 GB	
Emulator	Pixel 2 API 28		1080 × 1920: 42...	28	Android 9.0 (Goo...)	x86	2,9 GB	

Рисунок 2.11 – Экран менеджера виртуальный устройств

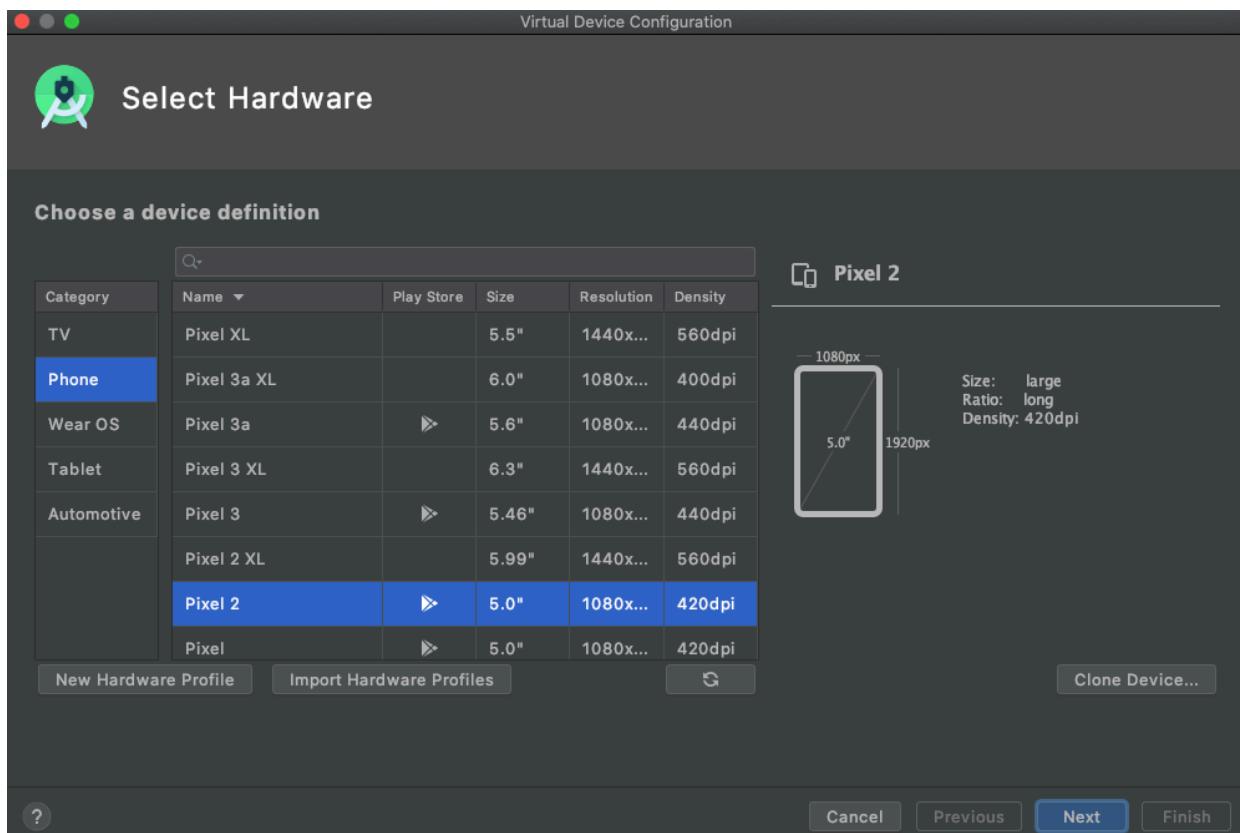


Рисунок 2.12 – Экран создания виртуального устройства

Далее производится переход на вкладку x86 Images или Recommended, в которых размещен образ без метки Download (рис.2.12).

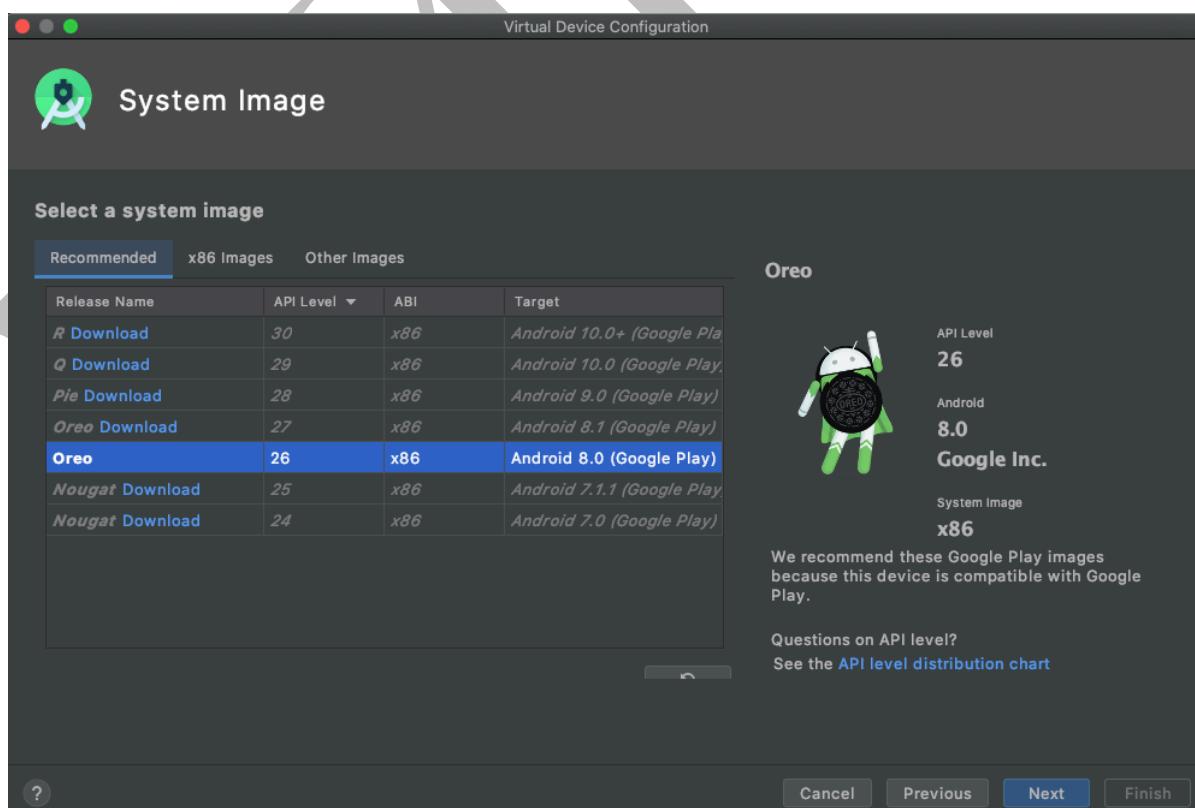


Рисунок 2.13 – Выбор образа системы

В данном случае на эмулятор будет установлен Android версии 8.0. Если требуется другая версия, то производится выбор требуемого образа.

Далее указывается название эмулятора с возможностью изменения конфигурации устройства.

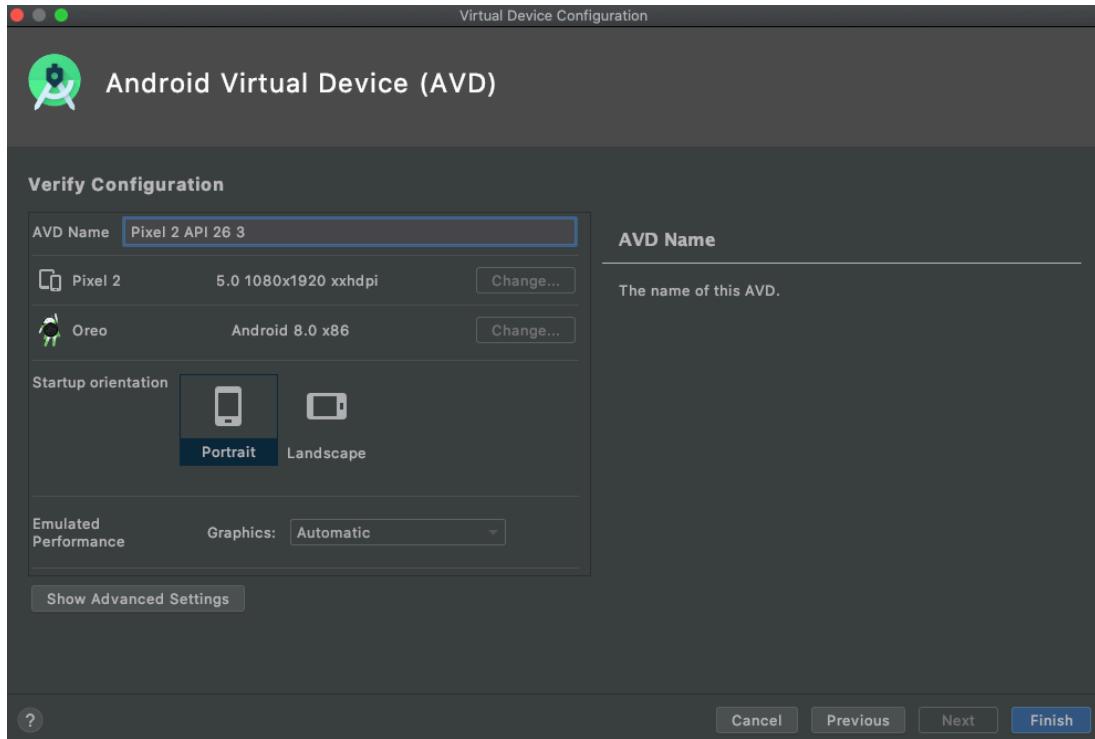


Рисунок 2.14 – Экран конфигурирования эмулятора

После создания эмулятора требуется запустить созданное устройство. Через

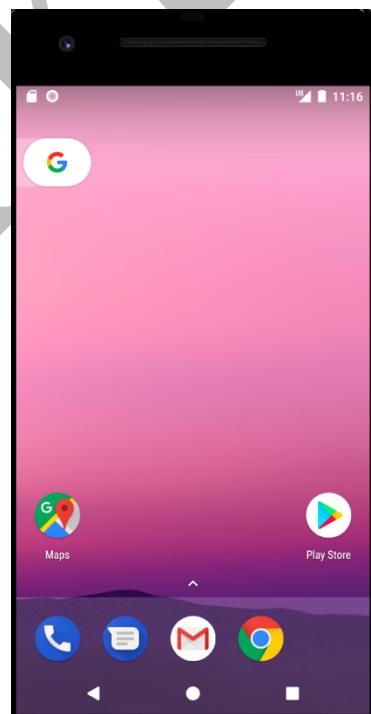


Рисунок 2.15 – Внешний вид эмулятора

## 2.5 Запуск приложения

Для запуска созданного ранее приложения требуется:

- выбрать соответствующий модуль в выпадающем списке;
- выбрать запущенный эмулятор (либо без него, тогда отобразится экран менеджера устройств);
- нажать кнопку *Run*



Рисунок 2.16 - Панель управления Android Studio



Рисунок 2.17 – Приложение Hello World

После успешной сборки приложения, приложение должно отобразиться на экране эмулятора (рис.2.17).

Если эмулятор в списке подключённых устройств отсутствует, то требуется его перезапустить.

### 3 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 возможно рассмотреть почтовое приложение. В нем одно activity – список писем, другое – просмотр письма, третье – настройки ящика. При работе производится перемещение по ним.

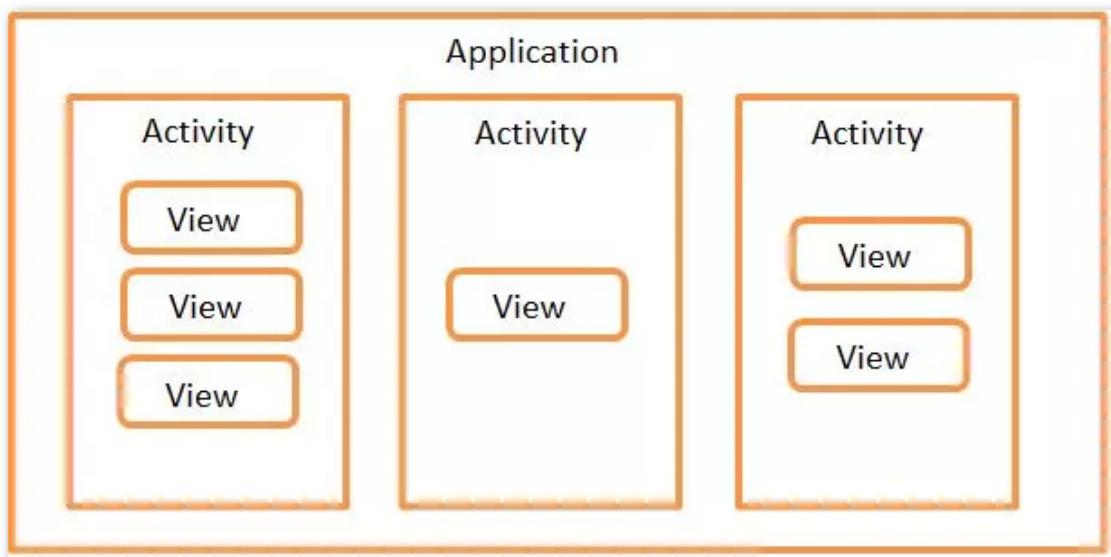


Рисунок 3.1 – Структура View элементов

Содержимое activity формируется из различных компонентов, называемых view. Самые распространенные view – это кнопка, поле ввода, чекбокс и т.д.

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 и т.д.).

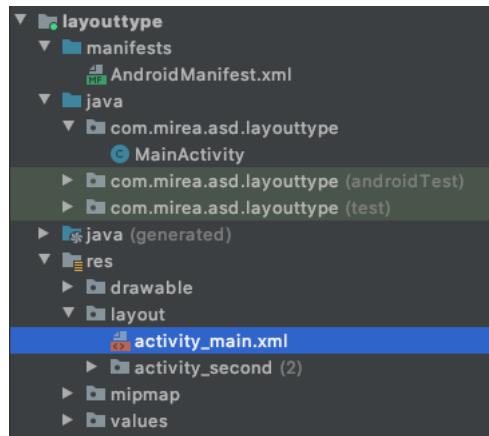


Рисунок 3.2 – Расположение layout

В созданном модуле *layouttype* требуется перейти в *res>layout>activity\_main.xml*. Это layout-файл. В нем определяется набор и расположение view компонентов, которые отображаются на экране. При запуске приложения, activity читает данный файл и отображает то, что там установлено. Открытие файла производится двойным кликом (рис.3.3).

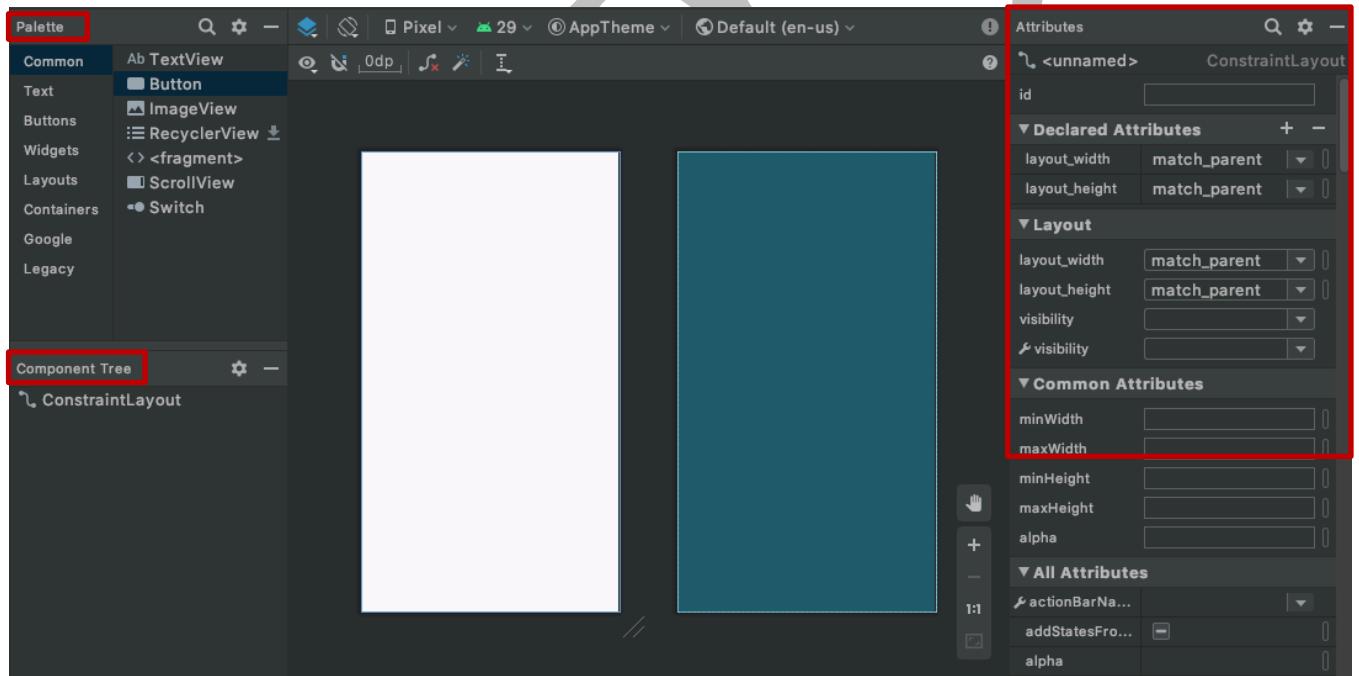


Рисунок 3.3 – Экран редактирования layout файла.

Компоновка (также используются термины разметка или макет) хранится в виде xml-файла в папке */res/layout*. Это сделано для того, чтобы отделить код от дизайна, как это принято во многих технологиях (html и css). Кроме основной компоновки для всего экрана, существуют дочерние элементы компоновки для группы элементов. По сути, компоновка – это некий визуальный шаблон для

пользовательского интерфейса приложения, который позволяет управлять элементами управления, их свойствами и расположением. При обращении к элементам управления через java-код необходимо присваивать элементам уникальный идентификатор через атрибут `android:id`. Сам идентификатор назначается через выражение `@+id/your_value`. После этого возможно обращаться к элементу через код при помощи метода `findViewById(r.id.your_value)`.

На рис.3.3 изображены два экрана. Обычный белый/черный и синий. Это один и тот же экран, но онображен в двух разных режимах:

- *Design* – отображаются View компоненты так, как они выглядят на экране;
- *Blueprint* – отображаются только контуры View компонентов.

Панель *Palette* – список всех View компонентов, которые возможно добавлять на экран: кнопки, поля ввода, чекбоксы, прогрессбары и прочее.

Панель *Component Tree* – отображает иерархию View компонентов экрана. Сейчас, например, корневой элемент – это ConstraintLayout. А в него вложен TextView.

Панель *Atributes* – отображаются свойства выбранного компонента. С помощью свойств имеется возможность настраивать внешний вид, расположение и содержимое View компонента.

### 3.1 XML

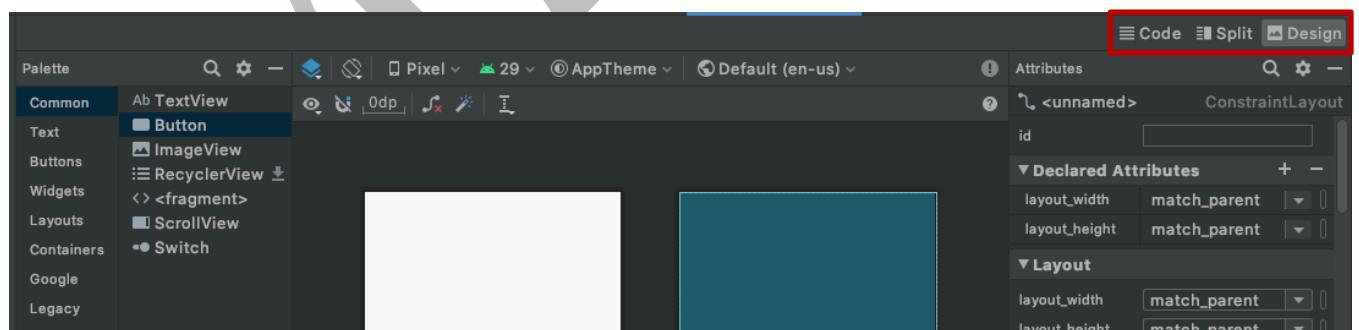


Рисунок 3.4 - Экран редактирования разметки

Вкладка *Code* (рядом с *design*) – xml-описание всех view layout-файла. Названия xml-элементов – это классы view-элементов, xml-атрибуты – это параметры view-элементов, т.е. все те параметры, что изменяются через вкладку

attributes. Также возможно вносить изменения в xml и изменения будут отображаться во вкладке design.

### Добавьте textView на экран и внесите изменения текста в textView

Каждый объект view и viewgroup поддерживают свои собственные атрибуты xml. Некоторые атрибуты характерны только для объекта view (например, объект textView поддерживает атрибут textSize), однако эти атрибуты также наследуются любыми объектами view, которые могут наследовать этот класс. Некоторые атрибуты являются общими для всех объектов view, поскольку они наследуются от корневого класса view (такие как атрибут id). Любые другие атрибуты рассматриваются как «параметры макета». Такие атрибуты описывают определенные ориентации макета для объекта view, которые заданы родительским объектом viewgroup такого объекта.

```
<TextView  
    android:id="@+id/textView3"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="TextView"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

Далее рассматриваются данные атрибуты:

- слово android в названии каждого атрибута – это namespace, обозначающий принадлежность к элементам android;
- id – это целочисленный идентификатор, который служит для обозначения уникальности объекта view в иерархии. Во время компиляции приложения этот идентификатор используется как целое число, однако идентификатор обычно назначается в файле xml макета в виде строки в атрибуте id. Этот атрибут xml является общим для всех объектов view (определенных классом view), который вы будете использовать довольно часто. Символ @ в начале строки указывает на то, что обработчику xml следует выполнить синтаксический анализ остальной части идентификатора, выполнить ее синтаксический анализ и определить ее в качестве ресурса идентификатора. Символ плюса (+) обозначает, что это имя нового ресурса, который необходимо создать и добавить к ресурсам (в файле r.java)

- `layout_width` (ширина элемента) и `layout_height` (высота элемента) могут задаваться в абсолютных значениях, а могут быть следующими: `match_parent` (максимально возможная ширина или высота в пределах родителя) и `wrap_content` (ширина или высота определяется по содержимому элемента).

Обратите внимание, что у view-элемента может не быть `id` (`android:id`). Например, для `TextView` он обычно не нужен, т.к. элементы чаще всего статичны, и к ним редко обращаются при работе приложения. При работе с `EditText` – работа производится с содержимым текстового поля, `button` – требует обработки нажатия.

### 3.2 Изображение

В проект можно добавить изображение несколькими способами:

- копирование в проект в папку `res/drawable` изображения. Стоит учитывать, что файл изображения будет добавляться в приложение, тем самым увеличивая его размер. Кроме того, большие изображения отрицательно влияют на производительность. Поэтому следует использовать небольшие и оптимизированные (сжатые) графические файлы.
  - выделить правой кнопкой мыши директорию `res` далее правой кнопкой мыши вызвать контекстное меню `New>Image Asset>выбор изображения`. Изображение будет сохранено в папке `/res/drawable`.

## 4 ВИДЫ LAYOUT. КЛЮЧЕВЫЕ ОТЛИЧИЯ И СВОЙСТВА

Для возможности размещения на экране различных компонентов (кнопки, поля ввода, чекбоксы и т.п.), необходимо использовать специальный контейнер. Именно в него будут размещаться компоненты. В Android компоненты называются View, а контейнер – ViewGroup. Существуют несколько типов ViewGroup: LinearLayout, RelativeLayout, FrameLayout, TableLayout, ConstraintLayout и т.д. (рис.4.1) Главное различие заключается в том, как они будут упорядочивать компоненты внутри себя.

Далее рассматриваются основные виды контейнеров:

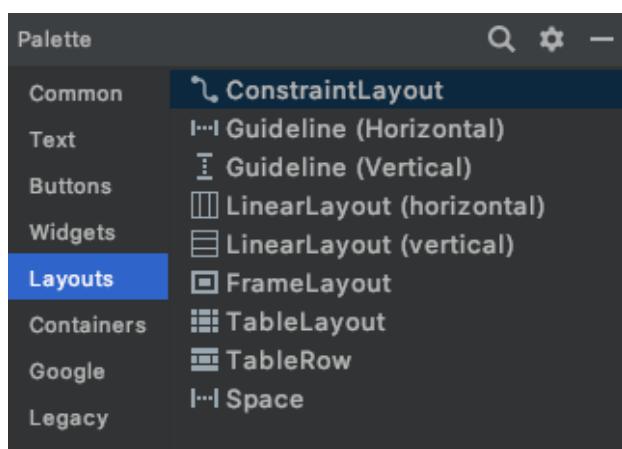


Рисунок 4.1 – Панель контейнеров

- LinearLayout – отображает View-элементы в виде одной строки (если он Horizontal) или одного столбца (если он Vertical).
- TableLayout – отображает элементы в виде таблицы, по строкам и столбцам.
- RelativeLayout – для каждого элемента настраивается его положение относительно других элементов.
- FrameLayout – самый простой тип разметки. Все дочерние элементы прикрепляются к верхнему левому углу экрана. В разметке нельзя определить различное местоположение для дочернего объекта. Последующие дочерние объекты View будут просто рисоваться поверх предыдущих компонентов, частично или полностью затеняя их.
- ConstraintLayout – размещает элементы view путем привязки к дочерним элементам или самому себе.

## 4.1 Задание

Создайте для каждого типа контейнера свой Layout

### 4.1.1 LinearLayout

Требуется выделить папку res/layout и создать новый Layout resource file с именем linear\_layout.xml (рис.4.2).

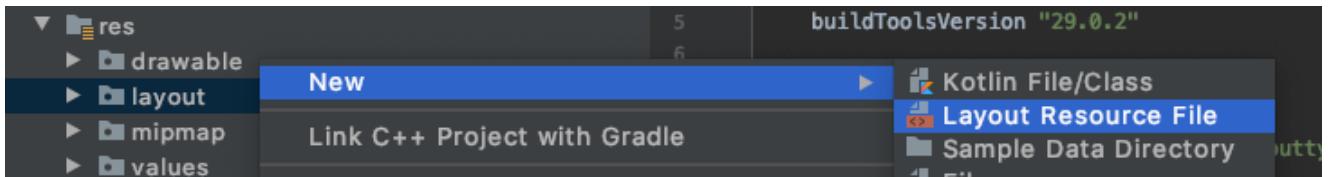


Рисунок 4.2 - Создание макета

В Root element указать *LinearLayout*. Компонент имеет свойство *Orientation*, которое определяет, как будут расположены дочерние элементы – горизонтальной или вертикальной линией (рис.4.3).



Рисунок 4.3 - Создание макета

Разместите *button* и изменяйте ориентацию LL.

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

GroupView возможно вкладывать друг в друга. Вложите в один LinearLayout два других. Удалите в main.xml все элементы кроме корневого LinearLayout. Ориентацию корневого LinearLayout укажем вертикальную и добавим в него два новых горизонтальных LinearLayout. В списке элементов слева они находятся в разделе layouts. Требуется перетаскивать элементы из списка не только на экран, но и на конкретный элемент на вкладке outline. Обратите внимание на параметры ширины и высоты экрана.

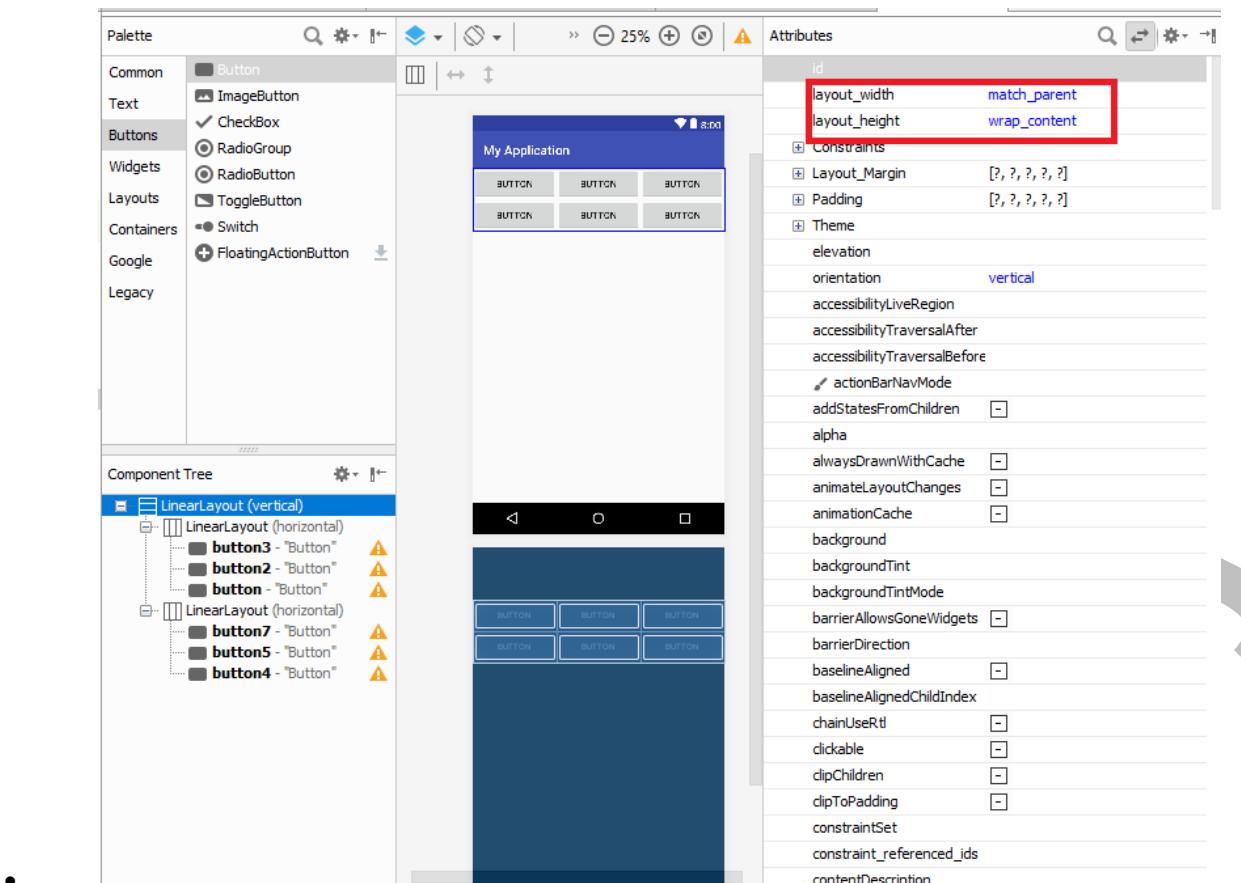


Рисунок 4.2 – Задание LinearLayout

Создайте экран приведённый на рис.4.2

#### 4.1.2 TableLayout

*TableLayout(tl)* состоит из строк *tablerow* (tr). Каждая tr в свою очередь содержит view-элементы, формирующие столбцы. Т.е. Количество view в tr – это количество столбцов. Но количество столбцов в таблице должно быть равным для всех строк. Поэтому, если в разных tr разное количество view-элементов (столбцов), то общее количество определяется по tr с максимальным количеством.

Требуется создать layout-файл *table\_layout.xml* с корневым элементом *TableLayout* и добавить в корневой *TableLayout* три *TableRow*-строки (из раздела *layouts* слева) и в каждую строку добавить по две кнопки. Результат: таблица имеет три строки и два столбца. Добавьте различные элементы view (*tl* может содержать не только *tr*, но и обычные *view*).

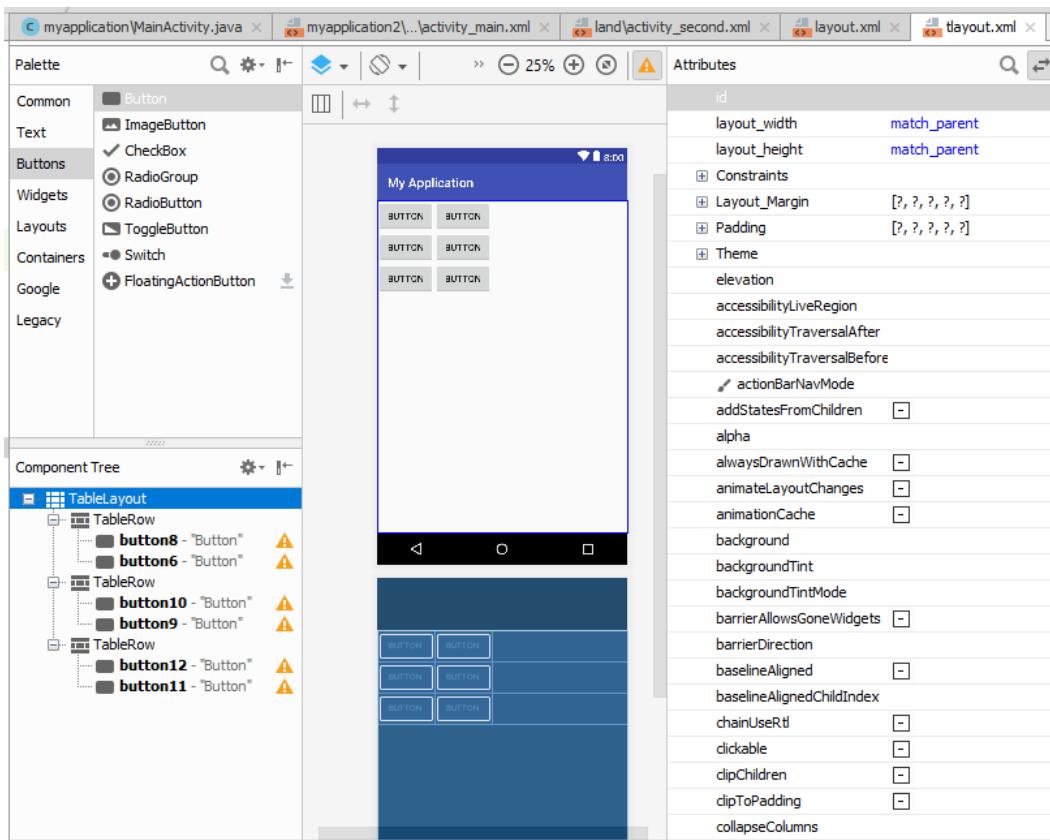


Рисунок 4.3 – Задание TableLayout

Создайте экран приведённый на рис.4.3

#### 4.1.3 RelativeLayout

Требуется создать layout-файл **relative\_layout.xml** с корневым элементом **RelativeLayout**

В данном виде layout каждый view-элемент может быть расположен определенным образом относительно указанного view-элемента.

Виды отношений:

- слева, справа, сверху, снизу указанного элемента (`layout_toLeftOf`, `layout_toRightOf`, `layout_above`, `layout_below`)
- выравненным по левому, правому, верхнему, нижнему краю указанного элемента (`layout_alignLeft`, `layout_alignRight`, `layout_alignTop`, `layout_alignBottom`)
- выравненным по левому, правому, верхнему, нижнему краю родителя (`layout_alignParentLeft`, `layout_alignParentRight`, `layout_alignParentTop`, `layout_alignParentBottom`)

- выравненным по центру вертикально, по центру горизонтально, по центру вертикально и горизонтально относительно родителя (layout\_centerVertical, layout\_centerHorizontal, layout\_centerInParent)

#### 4.1.4 ConstraintLayout

Android Studio по умолчанию предлагает использовать ConstraintLayout при создании разметки экрана. Управление компонентами внутри данного контейнера отличается от предыдущих контейнеров взаимодействия. В режиме Design возможно перемещать все компоненты в визуальном редакторе. Требуется переключиться в режим Blueprint (рис.4.4).

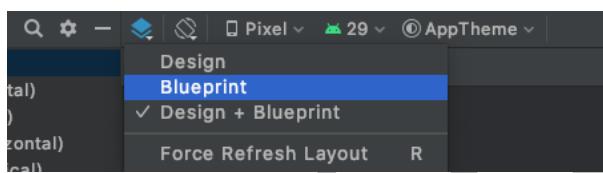


Рисунок 4.4 – Меню выбора типа экранов

Добавьте TextView элемент на экран. Для этого требуется перетащить компонент мышкой из Palette на экран. После этого элемент появится на экране и в Component Tree.

После запуска приложения (SHIFT+F10) созданный элемент отобразится вверху слева. Если открыть текстовое представление экрана (вкладка Code), то элемент TextView будет подчеркнут красной линией. При наведении на него курсором мыши, будет выведена ошибка:

*«This view is not constrained, it only has designtime positions, so it will jump to (0,0) unless you add constraints. »*

Этим сообщением среда разработки сообщает, что View не привязано. Его текущее положение на экране актуально только для разработки (т.е. только в среде разработки). При работе приложения, это положение будет проигнорировано и View будет расположено в точке (0,0), т.е. влево-вверх.

Для устранения данной ошибки необходимо добавить привязки (constraints). Они будут задавать положение View на экране относительно каких-либо других элементов или относительно родительского View. Для добавления привязки для TextView требуется выделить на экране TextView, после чего отобразится 4 круга по его бокам (рис.4.5). Эти круги используются, чтобы создавать привязки.

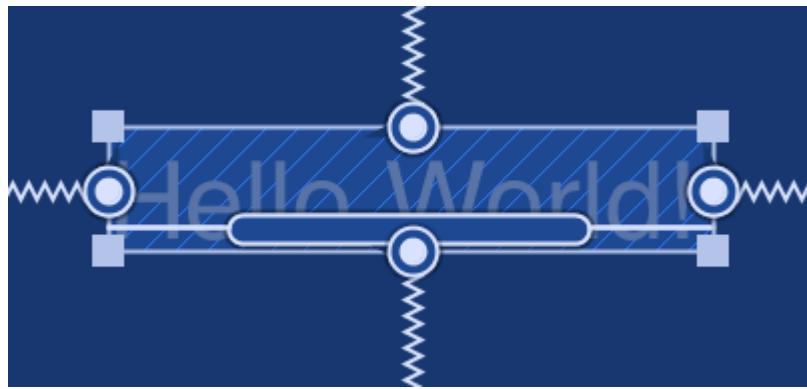


Рисунок 4.5 - Компонент View в контейнере ConstraintLayout

Существует два типа привязок: одни задают положение View по горизонтали, а другие - по вертикали.

Родителем TextView является ConstraintLayout, который в данном случае занимает весь экран. Поэтому края ConstraintLayout совпадают с краями экрана. Чтобы создать привязку, требуется нажать мышкой на TextView, чтобы выделить его. Затем зажать левой кнопкой мыши левый кружок и тащите его к левой границе. TextView также уехал влево. Он привязался к левой границе своего родителя. Возможно задать отступ. Для этого 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).

#### Произведите запуск приложения.

Ранее TextView находился влево-вверху, а теперь он только вверху. Влево он не уехал, т.к. создана горизонтальная привязка и TextView теперь знает, что по горизонтали он должен располагаться с определенным отступом от левого края.

Для создания вертикальной привязки используется верхний кружок и перетаскивается к верхней границе. TextView привязывается по вертикали к верхней границе родителя. После этого возможно перетащить TextView куда нужно, чтобы настроить горизонтальный и вертикальный отступы. При перетаскивании отображаются значения отступов.

Теперь TextView привязан и по горизонтали, и по вертикали (рис.4.5). Т.е. он точно знает, где он должен находиться на экране во время работы приложения.

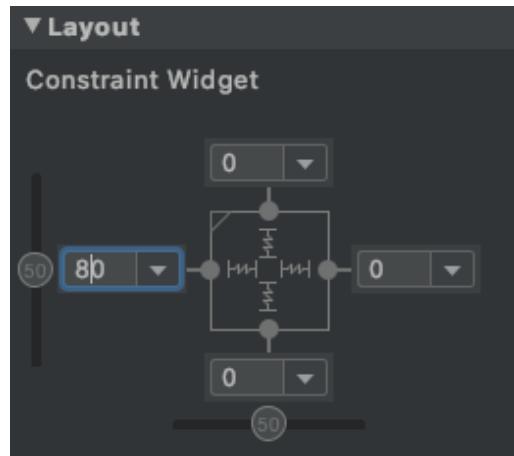


Рисунок 4.5 – Виджет контейнера с отображением оступов

Произведите запуск приложения.

TextView никуда не сместился, а находится там, где его настроили с помощью привязок.

Добавьте еще одно View, например, кнопку – Button. Возможно привязывать не только к границам родителя, но и к другим View. Привяжите кнопку к TextView.

Были рассмотрены примеры, когда View было привязано по каждой оси с одной стороны. Т.е. только слева или справа по горизонтали, и сверху или снизу по вертикали. Также имеется возможность привязать View с обоих сторон по каждой оси. Привязки уравняют друг друга, и View будет находиться ровно посередине между тем, к чему он привязан слева, и тем, к чему он привязан справа. Т.е. в случае View находится посередине между левой и правой границами его родителя.

## 4.2 Задание

Разместить на текущую разметку (res>layout>activity\_main.xml) из меню palette следующие элементы и изучить их свойства (область attributes)

- Text>textview, plaintext (edittext) и т.д.;
- Buttons>button, imagebutton, checkbox и т.д.;
- Widgets>imageview (установить изображение) и т.д..

Создать аналогичный экран (рис.4.3) для отображения информации о контакте.

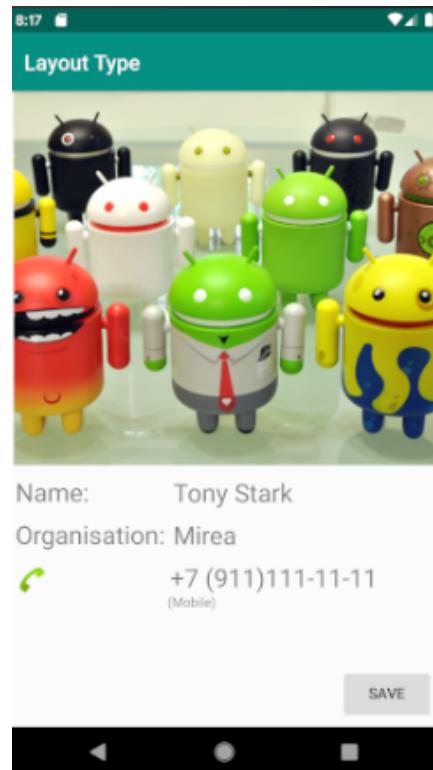


Рисунок 4.6 – Экран задания №1

## 5 LAYOUT-ФАЙЛ В ACTIVITY. СМЕНА ОРИЕНТАЦИИ ЭКРАНА.

### 5.1 Layout файл

При разработке, каждому activity сопоставляется одноименный java-класс (наследник класса android.app.activity). При запуске приложения, когда система должна показать activity и в дальнейшем работать с ним, она будет вызывать методы этого класса. И от того, что в этих методах указано, зависит поведение activity.

Далее представлен файл MainActivity:

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

Метод `onCreate` – вызывается, когда приложение создает и отображает activity. В начале вызывается метод родительского класса через ключевое слово `super`, выполняющий необходимые процедуры по созданию activity. Метод `setContentView(int)` – устанавливает содержимое activity из layout-файла. В качестве аргумента указывается не путь к layout-файлу (`res/layout/activity_main.xml`), а константу, которая является `id` файла. Эта константа генерируется автоматически в файле `r.java`. В этом классе хранятся сгенерированные `id` для всех ресурсов проекта (из папки `res/*`), чтобы была возможность обращения к ним. Имена данных `id`-констант совпадают с именами файлов ресурсов (без расширений). Файл `res/layout/activity_main.xml` создан средой разработки вместе с activity. Его название запрашивалось на том же экране, где и название activity.

### 5.2 Задание:

Создать layout-файл `activity_second.xml`.

Для этого требуется выделить папку `res/layout` в модуле и нажать на ней правую кнопку мыши.

`NEW > XML > LAYOUT XML FILE > LAYOUT RESOURCE FILE.`

В папке `layout` появится новый файл `activity_second.xml`. Этот новый layout-файл должен сразу открыться для редактирования. Далее необходимо добавить на экран элемент `PlainText` из списка слева и через `attributes` произвести изменения его

текста на: «new life for mirea activity» и 6 кнопок button. Требуется всё сохранить (ctrl+s).

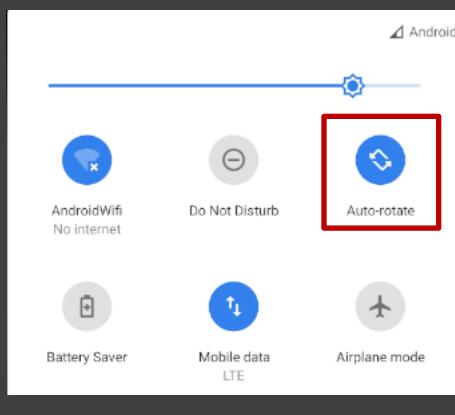
При создании нового layout-файла *activity\_second*, среда добавит в *r.java* новую константу для данного файла - *R.layout.activity\_second*. Теперь в коде возможно через данную константу указать на этот новый layout-файл. Для того, чтобы activity использовало новый файл *activity\_second.xml*, а не *activity\_main.xml*, который был изначально, требуется открыть *MainActivity.java* и изменить аргумент метода *setContentView*:

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_second);  
    }  
}
```

### 5.3 Ориентация экрана

Всего существует два режима: портретный и альбомный. На большинстве телефонов используется по умолчанию портретный режим. Далее рассматривается случай, когда в приложении имеется одно текстовое поле и шесть кнопок. На рисунке 5.2 приведён внешний вид вертикальной ориентации экрана. Но после поворота устройства на 90 градусов (для эмулятора требуется нажать комбинацию клавиш Ctrl+F11) пятая кнопка видна частично, а шестая оказалась за пределами видимости.

В случае, если ориентация экрана изменена, но приложение не выполнило поворот, требуется проверить настройки эмулятора.



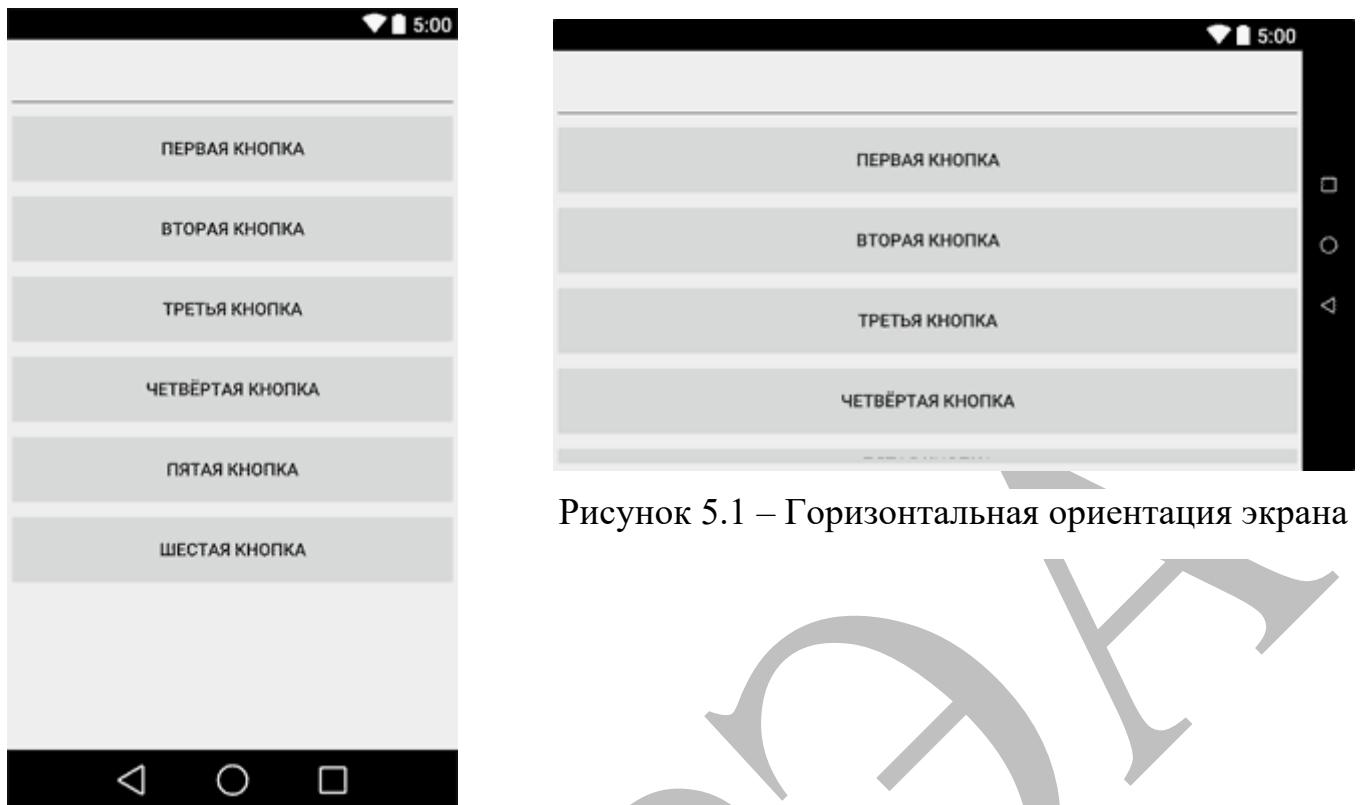


Рисунок 5.1 – Горизонтальная ориентация экрана

Рисунок 5.2 – Портретная  
ориентация экрана

Чтобы избежать данной проблемы, необходимо по-другому скомпоновать кнопки. Например, расположить их не подряд друг за другом, а разбить на пары. Для решения данной задачи, будет использоваться контейнер *TableLayout*. С его помощью возможно разбить кнопки на две колонки и поместить их в три ряда.

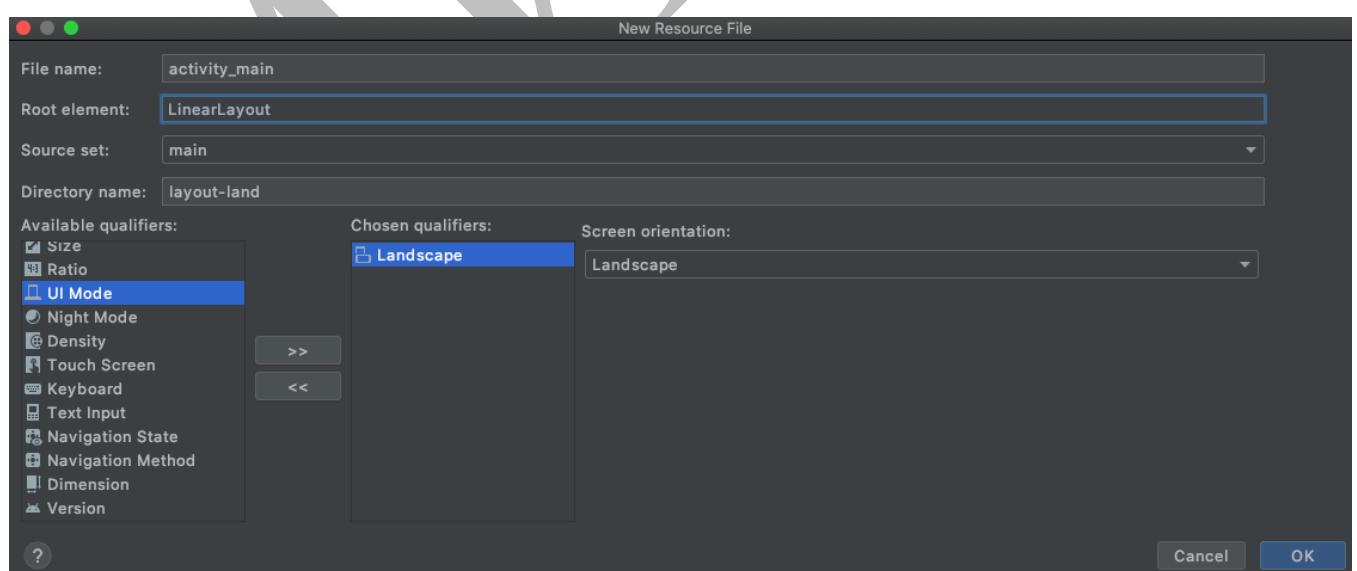


Рисунок 5.3 – Экран создания горизонтальной разметки

Для данной операции понадобится выделить папку *res*, вызвать из него контекстное меню и последовательно выбрать команды *New | Android resource File* (рис.5.3). В диалоговом окне из выпадающего списка Resource type требуется выбрать layout. В списке Available qualifiers расположен элемент Orientation, который переносится в правую часть Chosen qualifiers: с помощью кнопки с двумя стрелками. По умолчанию появится имя папки layout-port в первой строке Directory Name. Так как требуется альбомный вариант в выпадающем списке Screen orientation выбирается *Landscape*.

В структуре модуля (вкладка android). Размещено два файла *activity\_main.xml*: обычный и land.

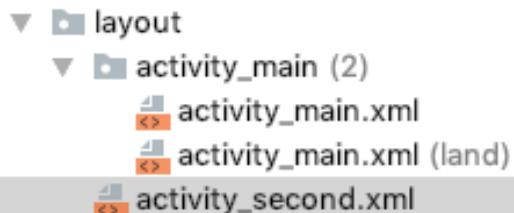


Рисунок 5.4 – Каталог файлов разметки

#### 5.4 Задание

Требуется открыть файл *activity\_second.xml (land)* и изменить расположение кнопок так, чтобы все из них были отображены

## 6 ОБРАЩЕНИЕ ИЗ КОДА К ЭЛЕМЕНТАМ ЭКРАНА. ОБРАБОТЧИКИ СОБЫТИЙ.

*Activity* является классом, который по сути представляет отдельный экран (страницу) приложения или его визуальный интерфейс. Отдельные *activity*, которые уже непосредственно используются в приложении, являются наследниками этого класса. Приложение может иметь одну *activity*, а может и несколько. Каждая отдельная *activity* задает отдельное окно для отображения.

Для обращения к элементу экрана из кода требуется его *ID*. Идентификатор указывается в layout-файлах. Для *ID* существует четкий формат - `@+id/name`, где `+` означает, что это новый ресурс и он должен добавиться в R.java класс, если он там еще не существует.

Класс R содержит идентификаторы для всех ресурсов, расположенных в каталоге res. Для каждого типа ресурсов в классе R создается внутренний класс (например, для всех графических ресурсов из каталога *res/drawable* создается класс *R.drawable*) и для каждого ресурса данного типа присваивается идентификатор. По этому идентификатору впоследствии возможно извлечь ресурс в файле кода.

При обновлении ресурсов во время компиляции этот файл также обновляется.

В файле *activity\_main.xml* для *TextView* требуется указать `id = @+id/textView`, сохранить (Ctrl+S) и скомпилировать приложение. Чтобы обратиться к *TextView* программно требуется метод *findViewById*. Данный метод по ID возвращает View.

Требуется открыть *MainActivity.java* и после строки с вызовом метода *setContentView* указывается (внутри метода *onCreate*):

```
TextView myTextView = (TextView) findViewById(R.id.textView);
```

Если View подчеркнуто красным, то скорей всего этот класс не добавлен в секцию import. Нажмите CTRL+SHIFT+O для автоматического обновления импорта.

Теперь *myTextView* имеет тип *TextView*, а результат метода *findViewById* преобразуется из *View* в *TextView*. После успешной инициализации возможно применять к *myTextView* методы класса *TextView*. Для примера используется метод *setText*:

```
myTextView.setText("New text in MIREA");
```

После сохранения и запуска (CTRL+F11) текст должен измениться.

Добавьте на экран кнопку (Button) id = @+id/button, текст остаётся по умолчанию. Требуется сохранить проект- CTRL+SHIFT+S (если не сохранить, то в R.java не появится ID). Получаем доступ к элементу программно:

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

Следующим методом возможно изменить ее текст:

```
button.setText("MireaButton");
```

После запуска приложения текст на кнопке изменится.

Требуется добавить CheckBox, id = @+id/check\_box. По умолчанию отметка не установлена. Для того, чтобы установить метку программно используется метод setChecked, который меняет параметр Checked.

```
CheckBox checkBox = findViewById(R.id.checkBox);  
checkBox.setChecked(true);
```

Метод *findViewById* используется, чтобы по ID получить объект, соответствующий какому-либо View-элементу (Button, TextView, CheckBox) и далее вызываются необходимые методы объектов (setText, setEnabled, setChecked).

## 7 ОБРАБОТЧИКИ СОБЫТИЙ НА ПРИМЕРЕ BUTTON.

Сама кнопка обрабатывать нажатия не умеет, ей требуется обработчик (его также называют слушателем - listener), который присваивается с помощью метода setOnClickListener. Когда на кнопку нажимают, обработчик реагирует и выполняет код из метода onClick (рис.7.1).

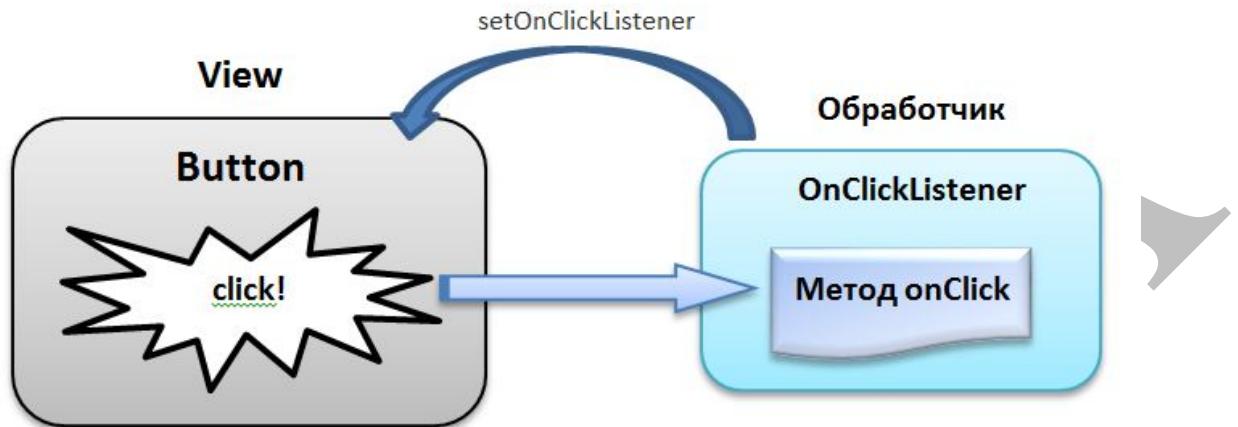


Рисунок 7.1 - Принцип обработки событий

Соответственно для реализации необходимо выполнить следующие шаги:

- создаётся обработчик;
- заполняется метод onClick;
- присваивается обработчик нажатий кнопке.

Создайте новый модуль. В меню File>New>New Module>Phone & Tablet Module>Empty Activity. Название проекта ClickButtons.

Требуется открыть разметку activity\_main.xml и добавить TextView (`android:id="@+id/tvOut"`) с текстом и две кнопки: OK (`android:id="@+id/btnOk"`) и Cancel (`android:id="@+id/btnCancel"`). Внешний вид экрана представлен на рисунке 7.2.

Требуется, чтобы по нажатию кнопки менялось содержимое TextView. По нажатию кнопки OK – выводится текст: «Нажата кнопка OK», по нажатию Cancel – «Нажата кнопка Cancel».

Данный экран связан с классом MainActivity.java. Описание объектов требуется вынести за пределы метода onCreate. Это сделано для того, чтобы было

возможно обращение к ним из любого метода класса (глобальная переменная). В onCreate этим объектам присваиваются значения с помощью метода *findViewById*.



Рисунок 7.2 – Экран приложения

В итоге должен получиться следующий код:

```
10 public class MainActivity extends AppCompatActivity {
11     private TextView tvOut;
12     private Button buttonOk;
13     private Button buttonCancel;
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_main);
19         tvOut = (TextView) findViewById(R.id.tvOut);
20         buttonOk = (Button) findViewById(R.id.btnOk);
21         buttonCancel = (Button) findViewById(R.id.btnCancel);
22
23     }
24 }
```

Требуется выполнить обновление секции import (CTRL+SHIFT+O). Объекты tvOut, btnOk и btnCancel соответствуют View-элементам экрана и возможно с ними работать. Требуется реагировать на нажатие кнопки. Для решения этой задачи у кнопки существует метод *setOnClickListener* (*View.OnClickListener l*). Аргументом

метода является объект с интерфейсом *View.OnClickListener*. Именно этому объекту кнопка делегирует обработку нажатия. Требуется создать данный объект (в *onCreate*):

```
23     View.OnClickListener oclBtnOk = new View.OnClickListener() {  
24         @Override  
25         public void onClick(View v) {  
26             textView.setText("Нажата кнопка OK");  
27         }  
28     };
```

Для добавления зависимостей в класс требуется навести курсор на подчёркнутое слово и добавить недостающие элементы (*Alt+Enter*) *View.OnClickListener*, т.к. метод кнопки *setOnClickListener* принимает на вход именно его.

Таким образом, создаётся объект *oclBtnOk*, который реализует интерфейс *View.OnClickListener*. Именно этот метод будет вызван при нажатии кнопки. По нажатию кнопки требуется выводить текст: «Нажата кнопка OK» в *TextView* (*tvOut*). Реализуется в методе *onClick*:

```
23     View.OnClickListener oclBtnOk = new View.OnClickListener() {  
24         @Override  
25         public void onClick(View v) {  
26             textView.setText("Нажата кнопка OK");  
27         }  
28     };
```

Осталось передать обработчик нажатия кнопке с помощью метода *setOnClickListener*.

```
30     // присвоим обработчик кнопке OK (btnOk)  
31     buttonOk.setOnClickListener(oclBtnOk);
```

В результате должен получится следующий код:

```
public class MainActivity extends AppCompatActivity {
    private TextView textView;
    private Button buttonOk;
    private Button buttonCancel;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // найдем View-элементы
        textView = (TextView) findViewById(R.id.tvOut);
        buttonOk = (Button) findViewById(R.id.btnOk);
        buttonCancel = (Button) findViewById(R.id.btnCancel);

        View.OnClickListener oclBtnOk = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                textView.setText("Нажата кнопка OK");
            }
        };

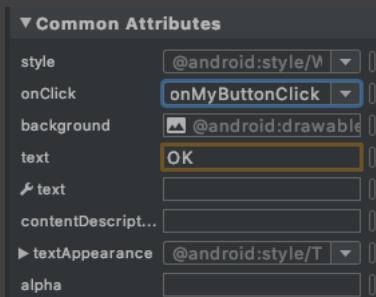
        // присвоим обработчик кнопке OK (btnOk)
        buttonOk.setOnClickListener(oclBtnOk);
    }
}
```

### Задание:

Создайте обработчик события для кнопки Cancel.

Также существует альтернативный способ обработки событий нажатий - атрибут onClick в xml (на панели свойств отображается как On Click):

android:onClick="onMyButtonClick"



В классе требуется создать метод:

```
public void onMyButtonClick(View view)
{
    // выводим сообщение
    Toast.makeText(this, "Ещё один способ!", Toast.LENGTH_SHORT).show();
}
```