Южно-Уральский государственный
гуманитарно-педагогический университет

Южно-Уральский научный центр
Российской академии образования (РАО)

Л. С. НОСОВА

РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Учебно-методическое пособие

Челябинск
2021

УДК 681.14(076)
ББК 32.973.26-018.2я7
Н84

Рецензенты:
канд. пед. наук О. А. Дмитриева;
канд. пед. наук О. Н. Иванова

Носова, Людмила Сергеевна
Н84 Разработка мобильных приложений: учебно-методическое пособие / Л.С. Носова; Южно-Уральский государственный гуманитарно-педагогический университет. – [Челябинск] : Южно-Уральский научный центр РАО, 2021. – 113 с. : ил.

ISBN 978-5-907408-59-3

В данном пособии представлены учебно-методические рекомендации к выполнению лабораторных работ по дисциплине «Разработка мобильных приложений». Лабораторные работы описывают возможности создания мобильных приложений для операционной системы Android с помощью программы Android Studio на языке программирования Kotlin и могут использоваться для освоения на практических занятиях, а также в процессе самостоятельной работы студентов и магистрантов.

Учебно-методическое пособие предназначено для студентов и преподавателей высших учебных заведений, а также может быть полезно учителям средних образовательных учреждений для повышения квалификации и формирования ИКТ-компетентности. Учебное пособие соответствует требованиям ФГОС ВО.

УДК 681.14(076)
ББК 32.973.26-018.2я7

ISBN 978-5-907408-59-3

СОДЕРЖАНИЕ

Введение

Дисциплина «Разработка мобильных приложений» относится к модулю части, формируемой участниками образовательных отношений, Блока 1 «Дисциплины/модули» основной профессиональной образовательной программы по направлению подготовки 09.03.02 «Информационные системы и технологии» (уровень образования бакалавр). Дисциплина является дисциплиной по выбору.

Общая трудоемкость дисциплины составляет 2 з.е., 72 час.

Цель изучения дисциплины:

изучение основ программирования на языке Kotlin для разработки мобильных приложений для Android в среде Android Studio с целью решения практических задач.

Задачи дисциплины:

1) сформировать теоретические знаний в области разработки мобильных приложений, в том числе для операционной системы Android;

2) сформировать навыки работы в средах разработки мобильных приложения, в том числе в среде разработки Android Studio;

3) сформировать практические умения в области разработки мобильных приложения с использованием языка программирования Kotlin

Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения образовательной программы представлен в таблице.

Таблица – Планируемые результаты

ПК-7 способность проводить анализ требований к программному обеспечению, выполнять работы по проектированию программного обеспечения	
ПК.7.1 Знать основные модели жизненного цикла программного обеспечения, методы формализации бизнес-процессов, методологии разработки программного обеспечения и технологии программирования, методологии и технологии проектирования и использования баз данных	З. Знать основные методы проектирования базовых и прикладных технологий для Android
ПК.7.2 Уметь собирать и проводить анализ информации, необходимой для разработки программного обеспечения, разрабатывать архитектуру, прототипы и дизайн информационных систем, а также модели баз данных	У. Уметь применять методы проектирования технологий для решения прикладных задач для Android
ПК.7.3 Иметь навыки владения современными методами и средствами проектирования программного обеспечения и баз данных	В. Владеть методами проектирования базовых и прикладных технологий для решения профессиональных задач для Android

С данном пособии собран краткий теоретический материал и лабораторные работы, раскрывающие возможности создания мобильных приложений для операционной системы Android с помощью программы Android Studio на языке программирования Kotlin. Данный материал может использоваться для освоения на практических занятиях, а также в процессе самостоятельной работы студентов инженерных специальностей.

Краткие теоретические сведения

Процесс разработки мобильного приложения

Задачи разработки мобильного приложения сводятся к нескольким этапам:

1. Идея приложения от ее зарождения до постановки конкретной цели.

2. Производство приложения – непосредственно разработка интерфейса и программного кода.

3. Выпуск приложения – размещение его на маркетплейсах.

4. Оперирование и поддержка – самая продолжительная часть жизненного цикла программного продукта.

5. Вывод приложения с рынка программных продуктов из-за отсутствия прибыли или потери популярности, морального устаревания и т.п.

6. Закрытие приложения, т.н. фактическая смерть приложения на рынке программных продуктов.

В настоящее время популярность разработки мобильных приложений возрастает, так как они используются не только для смартфонов, но и в современных автомобилях, носимой электронике и смарт-телевизорах. В связи с этим, расширяется круг пользователей таких приложений, что выдвигает определенные требования к приложениям. Пользователь не должен обладать особыми и/или специальными знаниями для работы с приложением. Приложения должны быть максимально интерактивными и «отзывчивыми». Решающую роль в выборе приложения играют отзывы других пользователей. Следовательно, необходимо создавать для пользователей «привычные» интерфейсы. Для удобства разработчиков создаются Guideline – руководства по разработке пользовательского интерфейса и поведения приложения. Для Android он представлен по ссылке https://developer.android.com/design/index.html.

Мы будем говорить именно о разработке для Android, так по данным статистики 80% мобильных устройств работает под управлением этой операционной системы. И стоимость этих устройств ниже стоимости устройств Apple. Тоже самое касается и стоимости оборудования и программ для разработки приложений под эту платформу. Кроме того, разработчикам доступны исходники операционной системы Android, можно посмотреть особенности работы и настроить приложение. В свою очередь компания Google постоянно разрабатывает обновления и поддерживает процесс создания приложений, сокращая время размещения программ в макетплейсе и минимизируя расходы разработчиков на лицензию.

Однако, при разработке приложения для Android имеются и недостатки. Так как устройств большое количество и все они различаются по размеру диагонали, то требуется разработка приложения различных верстоок с различными интерфейсами. Открытый код порой приводит к вседозволенности для разработчиков, что плохо сказывается на безопасности приложений и повышает уязвимость устройств. Кроме того, на рынке устройств до сих пор можно встретить старые версии операционной системы, что требует от разработчиков необходимость выпуска и поддержки приложений и/или их адаптации. При разработке Android Studio можно посмотреть статистику и выбрать оптимальные версии API Level (рис. 1).

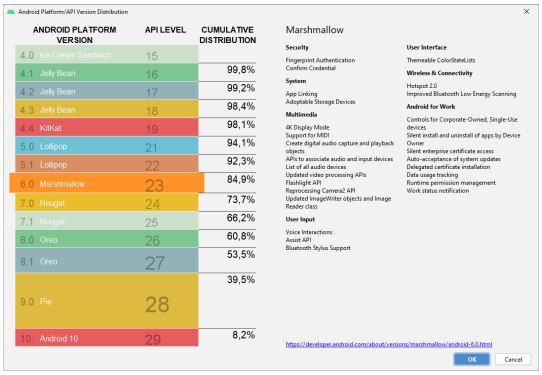


Рисунок 1 – Версии API Level

Архитектура операционной системы Android

Для понимания принципов разработки мобильных приложений необходимо рассмотреть особенности архитектуры операционной системы Android (рис. 2).

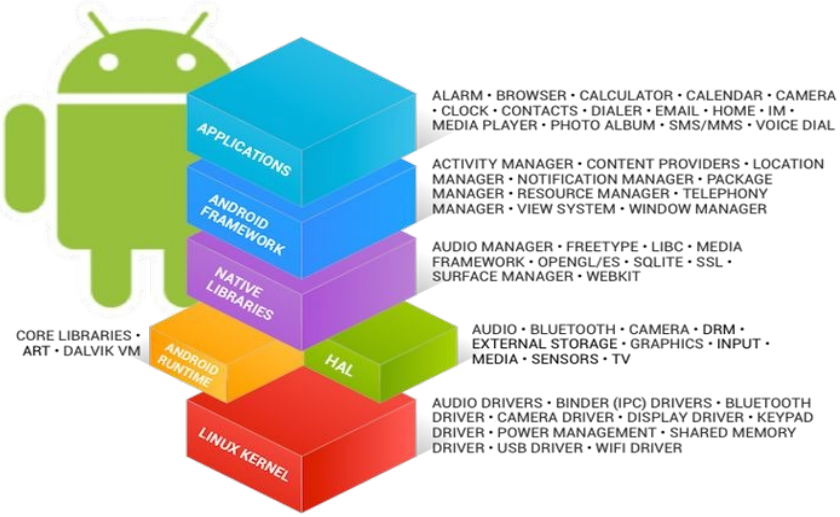


Рисунок 2 – Архитектура операционной системы Android

Система многоуровневая и самый нижний уровень – уровень ядра, далее идет HAL – уровень абстрагирования от аппаратного обеспечения, далее по иерархии располагается уровень нативных библиотек, которые непосредственно работают с оборудованием операционной системы. Над ними находятся непосредственно классы, компоненты, с которыми уже работают приложения под Android. На разных уровнях поддерживаются разные возможности безопасности.

Процесс загрузки операционной системы Android представлен на рис. 3.

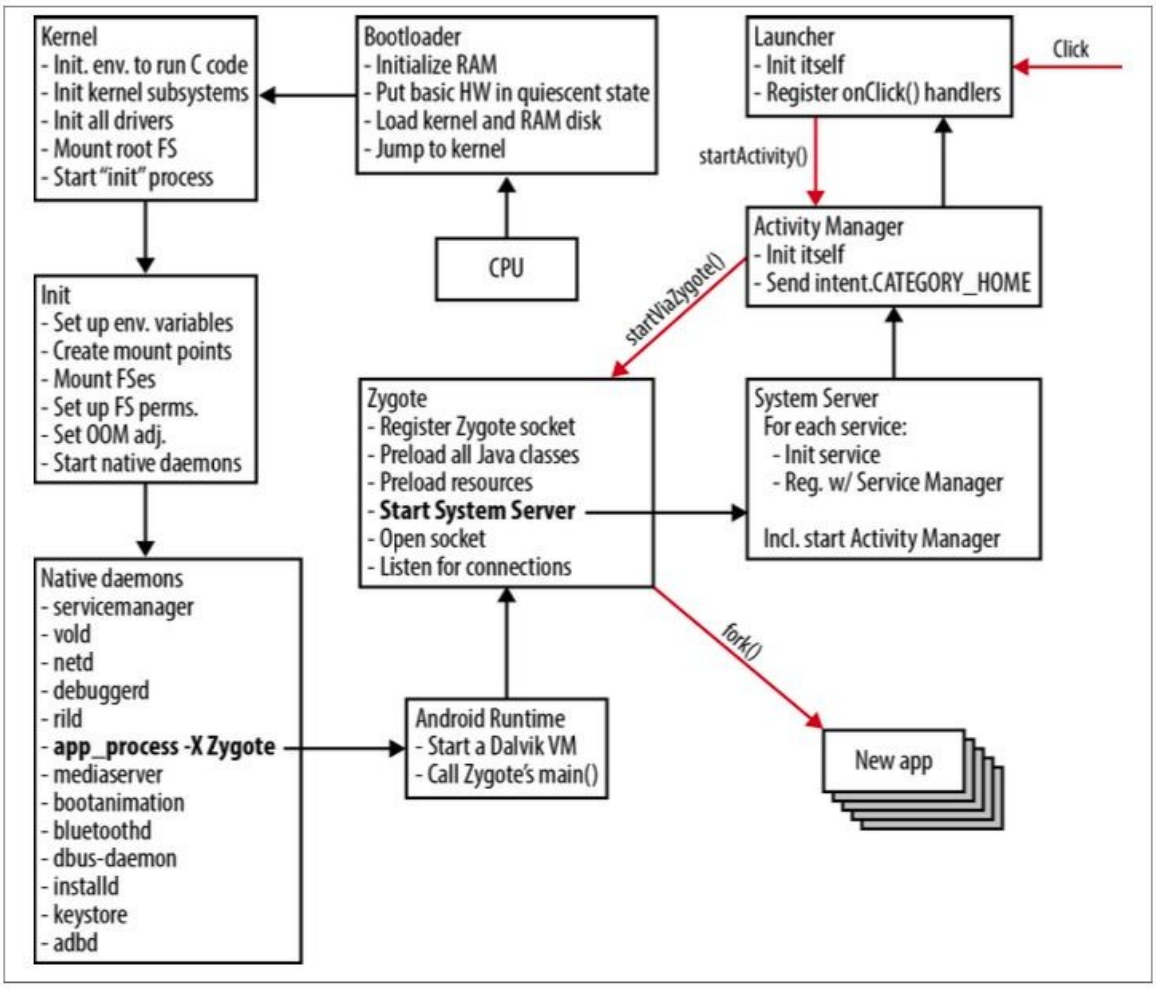


Рисунок 3 – Процесс загрузки операционной системы Android

Сначала запускается процессор CPU, после того как собственно процесс запустился он запускает программу-загрузчик Bootloader. Она загружает основные драйвера основных устройств и собственно код ядра для загрузки ядра операционной системы. После этого в режиме ядра Kernel запускается непосредственно сама операционная система. Далее идет переключение в пространство пользователя и выполняется первый процесс Init, который инициализирует основные нативные библиотеки операционной системы Android. После чего запускаются нативные библиотеки, например, работа сервисами, библиотеки работы с bluetooth-устройствами и все низкоуровневые реализации. Они написаны на нативном языке C++. Отсюда же

запускается процесс Android runtime, который запускает виртуальную машину Java. Все остальные процессы уже работают с этой виртуальной машиной. Далее запускается процесс Zygote – один из самых важных процессов, потому что все процессы, порожденные пользователями, являются дочерними процессами Zygote.Таким образом child-процесс, собственно, и является основным для работы пользовательского приложения.

Если рассмотреть процесс со другой стороны (рис. 3, справа вверху), то после нажатия по приложению на экране лаунчера (Launcher), он вызывает метод на одном из основных сервисных объектов с намерением запустить выбранное приложение. После этого от процесса Zygote отводится дочерний процесс, создается отдельный пользователь для приложения.

Также есть специальные компоненты Activity менеджер и System service. Activity менеджер – это сервис, который управляет жизненным циклом приложений в операционной системе. System service – это специальный компонент, в котором регистрируются все сервисы, запущенные в операционной системе.

Таким образом каждое приложение работает на своем процессе, это обеспечивает изоляцию приложение друг от друга и некую модель безопасности.

Все приложения в системе используют изолированную память и в общем не как напрямую не могут повлиять на работу других приложений и непосредственно обратиться изменить память соседнего приложения.

Файл приложения

Android приложение – это не только код, написанный на java (рис. 4). Android приложение содержит множество других компонентов и ресурсов для работы приложения. Строковые константы, рисунки и другие ресурсы, которые используются в

приложении, называются артефактами. Они загружаются вместе с приложением. В связи с этим процесс запаковки приложения и ресурсов состоит из ряда этапов. Java class и полученные в результате компиляции java-code компилятором javac элементы посредством специального обработчика компилятора dx преобразуется в специальные файлы Byte code для операционной системы. Сюда же попадают специальные конфигурационные файлы, которые находятся в контексте приложения. Там содержится информация об основных компонентах приложения. Здесь же располагаются ресурсы, которые посредством специального инструмента запаковываются в apk-файл. По сути, такой файл является zip-архивом, но не таким jar, потому что содержит довольно много компонентов.

Еще один главный файл – это Android manifest, без него приложение не сможет быть установлено, запущено и выполнено.

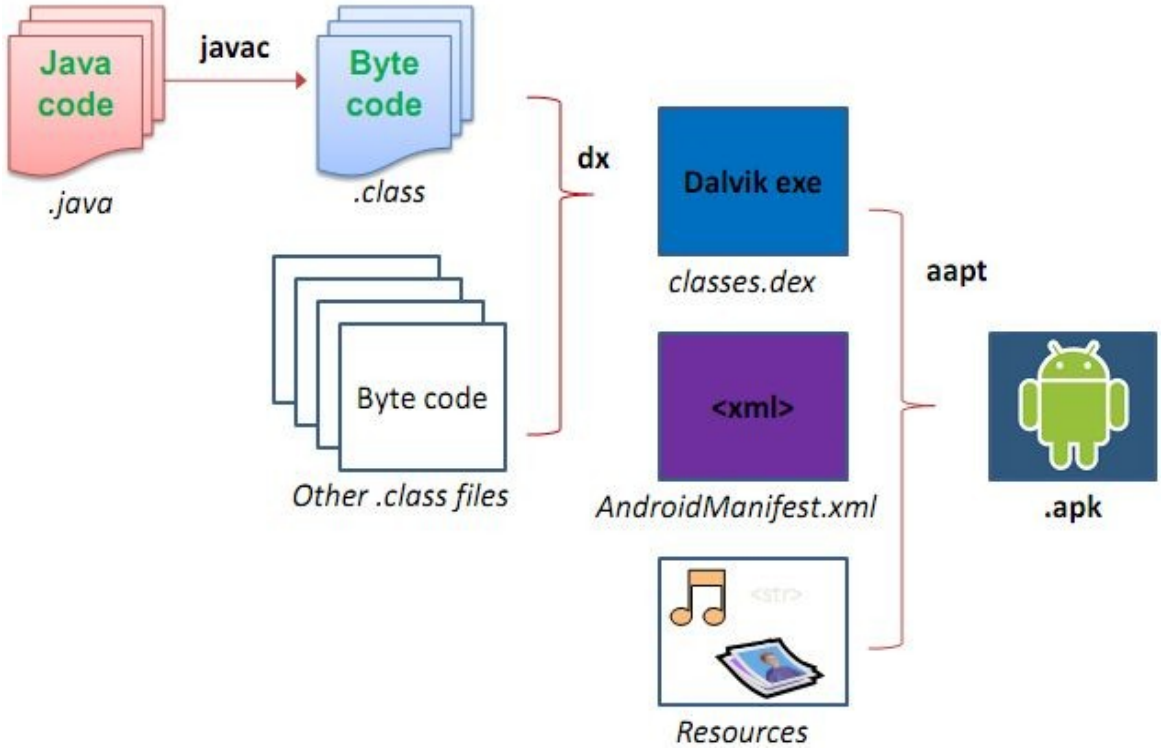


Рисунок 4 – Структура файла приложения

Android manifest

Android manifest – это основной файл конфигурации разрабатываемого приложения. У каждого приложения есть manifest, там содержатся такие данные как название приложения, package-name и все компоненты, которые в приложении используется.

Также есть специальный блок applications – эта информация настройках приложения и android-компонентах, activity, service, receiver и контент-провайдеры и также подробная информация об особенностях запуска приложений, разрешения на них.

Пример файла manifest представлен на рисунке (рис. 5).

```
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:textSize="36sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/android"
        tools:layout_editor_absoluteX="103dp"
        tools:layout_editor_absoluteY="135dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 5 – Пример Android manifest

Android-приложение имеет существенное отличие от обычного java-приложения тем, что с точки зрения операционной системы приложения java имеет метод main, который является единственной точкой входа в приложение.

Когда запускается приложение, то вы всегда начнет исполняться метод main из него далее запускаются другие сервисы, службы и решаются задачи взаимодействия.

В операционной системе Android метод main существует, но он скрыт от разработчика. Таким образом Android выделяет несколько компонентов activity, service, receiver и контент-провайдеры, которые являются и могут являться точками входа в приложение.

Поэтому в файле manifest есть четыре группы таких компонентов, описывающих все точки входа в приложение, как операционная система может запустить или вызвать процесс и, собственно, начать взаимодействие. Следовательно, без Android manifest и описания этих групп система ничего не сможет сделать.

Это существенное отличие от других программ, что привносит достаточно большую сложность вообще процесс разработки приложений ввиду множественных путей запуска.

Сами компоненты разделены по своим обязанностям. Activity – это основной элемент, та часть приложения, которое видит пользователь, с которой можно взаимодействовать. Собственно визуальная часть приложения. Точка входа в приложение.

Service – это специальные компоненты, у которых нет интерфейса, они не взаимодействуют напрямую с пользователем. Они призваны выполнять какую-то фоновую задачу приложения.

Receiver – это специальные компоненты, которые позволяют получать специальные сообщения, которые системы умеет рассылать. Например, при потере сигнала сети операционная система рассылает специальное сообщение и приложение может подписаться на этот тип событий.

Компонент контент-провайдеры – это специальные компоненты, которые являются поставщиками данных для приложения. Например, в телефоне есть контакты, но контакты являются специальными данными, которые хранятся в отдельном хранилище в операционной системе и приложение не имеет прямого доступ к этим данным, обеспечивая определенный уровень

безопасности. Для того чтобы позволять приложением из разных процессов получать и обмениваться данными как раз существует компонент контент-провайдер.

Файл manifest имеет расширение xml. Файл описывает разные объекты, они содержат набор атрибутов (более подробная информация есть на сайте http://developer.android.com).

Язык программирования Kotlin

Kotlin – статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine и разрабатываемый компанией JetBrains. Официальный сайт https://kotlinlang.org/. По одной из версий язык назван в честь острова Котлин в Финском заливе, на котором расположен город Кронштадт.

Kotlin разрабатывается с 2010 г. С мая 2017 года является официальным инструментом для разработки приложений Andriod, с 2019 года назван компанией Google приоритетным языком для разработки.

Примеры кода представлены на рис. 6.

```
fun main() {
  val scope = "world"
  println("Hello, $scope!")
}
```

```
fun sayHello(maybe: String?, neverNull: Int) {
   // use of elvis operator
   val name: String = maybe ?: "stranger"
   println("Hello $name")
}
```

```
  // returns null if foo is null, or bar() returns null, or baz() returns null
  foo ?. bar() ?. baz()
```

Рисунок 6 – Пример кода на Kotlin

Также Kotlin компилируется в JavaScript и в исполняемый код ряда платформ. Язык совмести с языком Java, для разработчиков приложения под android имеется возможность не переписывать приложение, а внедрять описание новых функций на языке Kotlin. Таким образом можно говорить, что Kotlin обладает свойством совместимости, полностью поддерживается средством разработки приложений Android Studio.

Другое свойство языка – это его производительность. Код на Kotlin работает также быстро как на Java, при этом длина кода значительно сокращается.

Однако разработчики отмечают некоторые недостатки Kotlin: меньшая скорость компиляции и работы Android Studio.

Activity и его жизненный цикл

Activity – это белый прямоугольник на экране, обычная пустая Activity собственно не содержит никаких элементов.

В Activity можно размещать свои. Достаточно добавить класс, который является унаследованным от класса activity и объявить его файле Android manifest. Пример объявления представлен на рисунке 7.

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

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >

    <activity
      android:name="ru.mail.sample.MainActivity"
      android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Рисунок 7 – Пример файла Android manifest

Рассмотрим жизненный цикл Activity (рис. 8).

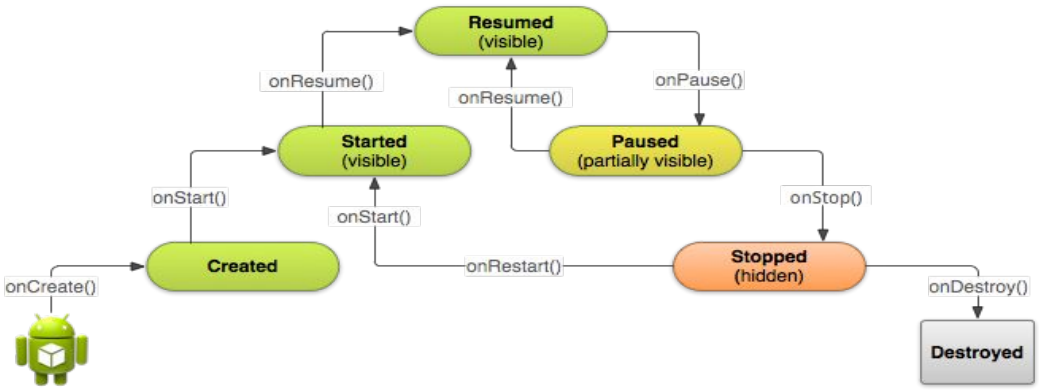


Рисунок 8 – Жизненный цикл Activity

Метод onCreate –вызывается в тот момент, когда activity создано. В этот момент Activity надо проинициализировать, обычно в этом методе устанавливается разметка Activity, решается, где, как будут располагаться элементы и в какой иерархии.

Activity принимает после этого состояние Created. Затем будет вызван метод onStart() – это значит, что Activity запущена и находится на переднем экране и соответственно у неё состояние Started. Дальше вызывается метод onResume(), то есть что Activity продолжает работать. Это конечный этап, после которого Activity как бы живет на экране можно с ней взаимодействовать, компоненты будут реагировать и Activity считается полностью созданной и отображаются все элементы на ней. Если поверх Activity появляется какое-то всплывающее окно, происходит частичное перекрытие Activity, то вызывается метод onPause(), в режиме приостановки: Activity еще не ушла на задний план, но не должна обновлять свое состояние.

Дальше может вызываться метод onStop(), когда происходит полное перекрытие Activity, то есть когда она уходит на задний план, например при закуске другого приложения.

После этого может быть вызван метод onDestroy, когда Activity уничтожается. Уничтожается Activity опять средствами операционной системы. Например, один из таких случаев – это нехватка оперативной памяти.

ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа №1. Начало работы с Android Studio

Цель: изучить особенности установки и настройки Android Studio, эмулятора, подключения устройства для тестирования приложения и создания первого проекта.

Ход работы

Для того чтобы программировать под Android, в настоящее время нужно установить только одну программу – Android Studio. Android Studio – это целый программный комплекс. В него входят следующие компоненты.

1. Компилятор языка. Он необходим для того, чтобы код на языке программирования Kotlin был переведен в последовательность команд для компьютера.

2. Виртуальная машина. Она создает специальную среду выполнения для программы на языке программирования Kotlin.

3. Среда программирования. Проект под Android представляет собой очень сложную систему, поэтому для эффективной работы нужна программа, автоматизирующая процесс разработки мобильного приложения от нажатия кнопок, использования горячих клавиш до выбора пунктов меню.

4. Собственно устройство, на котором можно запускать программы. Если нет устройства, то для отладки используется программа – виртуальное устройство – эмулятор Android.

5. Библиотека классов. Используются для программирования под Android для расширения возможностей программы.

6. Android Data Bridge (ABD) – программа, соединяющая устройство с компьютером, позволяющая им управлять, например, загружать на него программы. Все Android-устройства, в том числе и виртуальные, управляются с компьютера специальной универсальной программой – посредником. ADB.

7. Программа, соединяющая среду программирования с ADB.

8. Индивидуальный драйвер для каждого устройства для установления связи с ADB.

В настоящее время практически все эти компоненты устанавливаются вместе со средой Android Studio.

Задание 1. Установка Android Studio

Устанавливается Android Studio довольно долго, но просто, как написано на официальном сайте «буквально в пару кликов» (рис. 1.1).

1. Перейдите на официальный сайт https://developer.android.com/ studio?hl=ru.

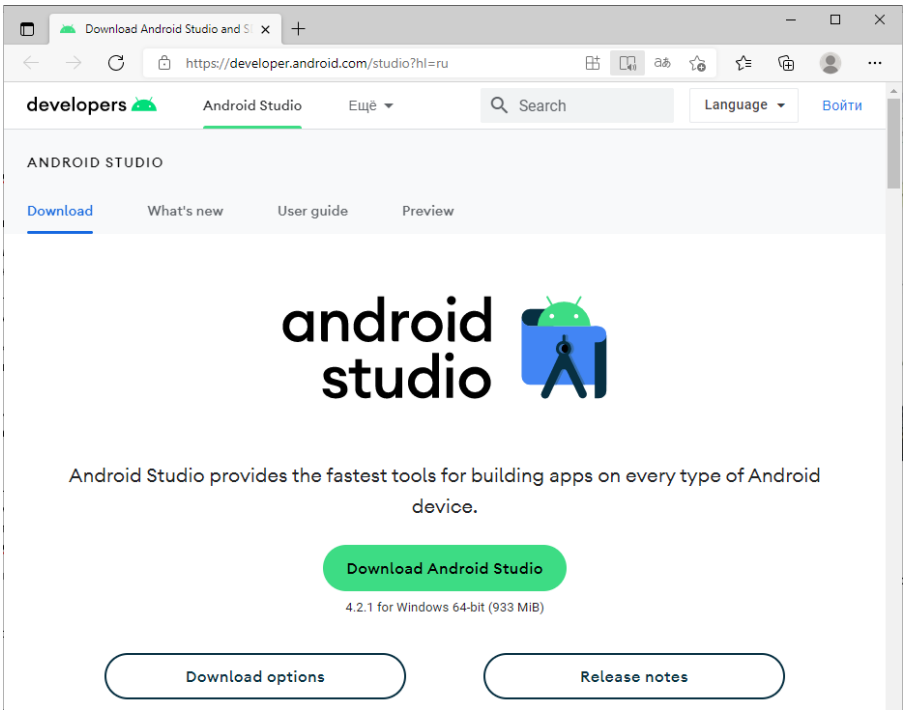


Рисунок 1.1 – Официальный сайт

2. Нажмите на зеленую кнопку 

и начните скачивание. После этого необходимо запустить скаченную программу и пройти до конца мастера установки. Если место на диске позволяет, можно установить эмулятор (рис. 1.2).

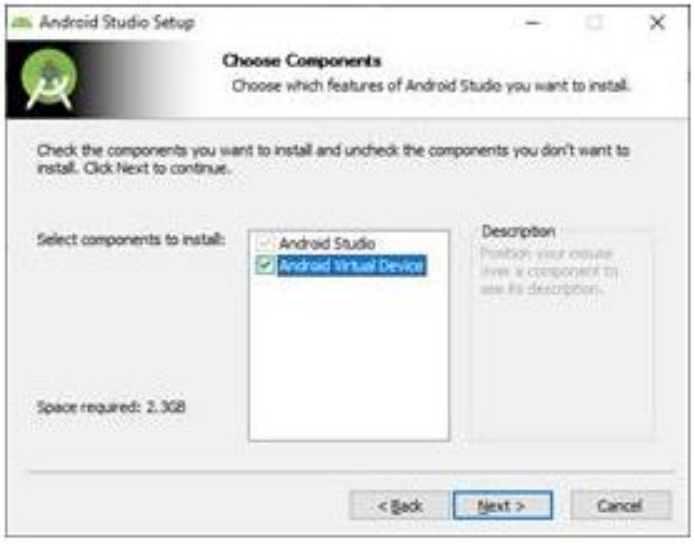


Рисунок 1.2 – Установка эмулятора

Задание 2. Первое приложение

1. Создайте приложение из окна **Welcome**, выбрав **Start a new Andriod project** (рис. 1.3):

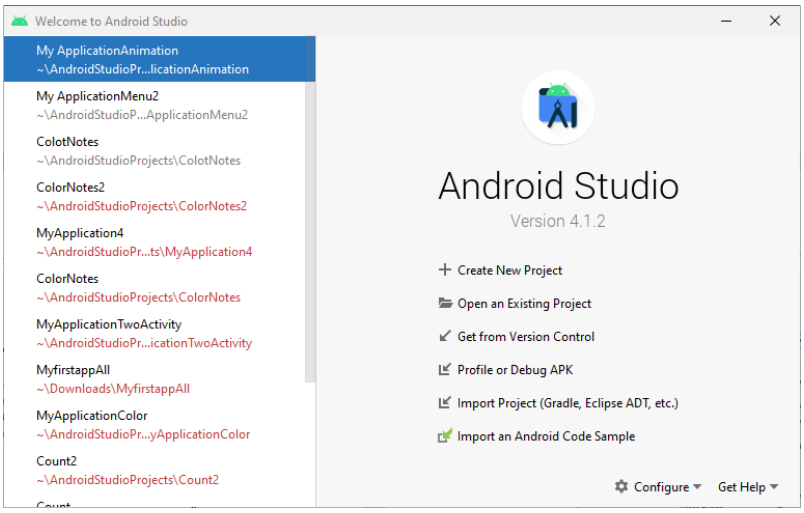


Рисунок 1.3 – Окно Welcome

Или из основного окна через меню (рис. 1.4):

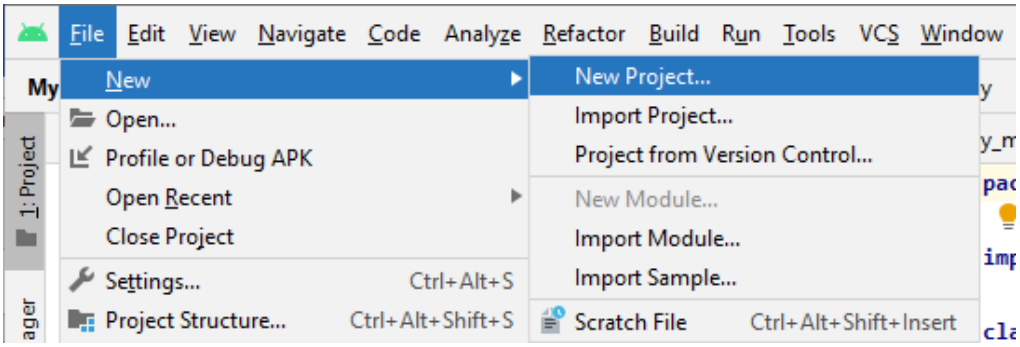


Рисунок 1.4 – Главное меню

2. При запуске матера необходимо ответить на несколько вопросов. Далее предлагается выбрать вид Activity – экрана. По умолчанию предлагается **Empty** (рис 1.5).

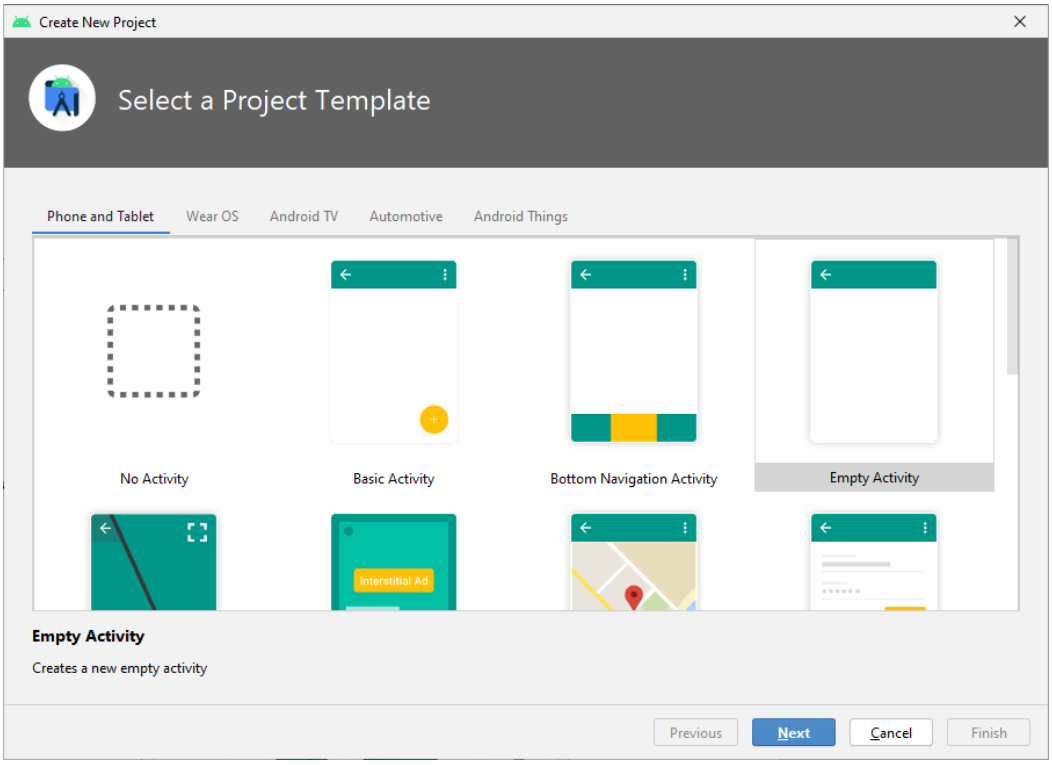


Рисунок 1.5 – Окно выбора Активности

3. Далее, на втором экране, нужно задать название приложения, имя пакета. Именно по пакету, уникальной папке, в котором будет лежать приложение, система различает приложения. Далее указать для каких версий Android будет приложение

и язык программирования, на котором будет вестись разработка. Проверьте, что выбран язык Kotlin, Minimum API Level можно оставить без изменения. Android Studio сама выбирает версию API, чтобы приложение работало практически на всех современных Android-устройствах (рис 1.6).

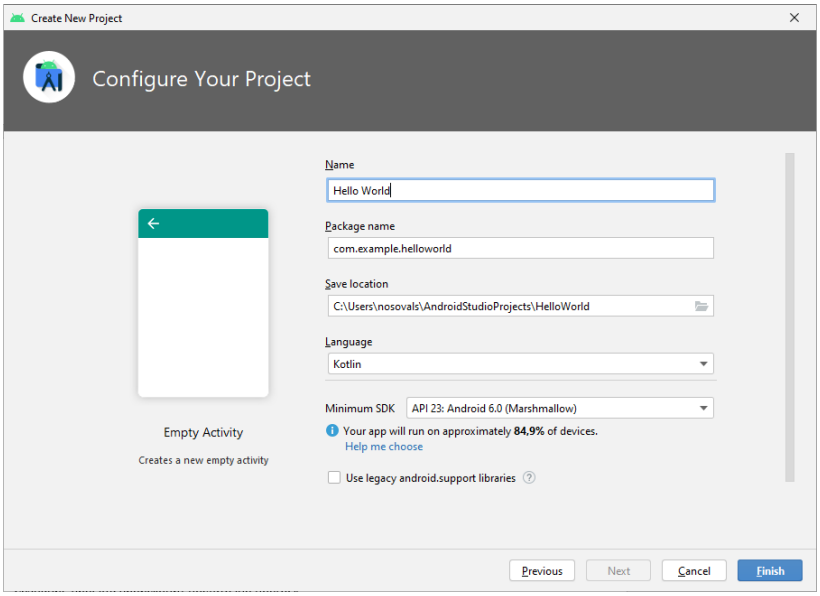


Рисунок 1.6 – Настройки проекта

При выборе минимальной версии API можно перейти по ссылке, указанной в окне **Help me choose**. Можно изучить статистику распространенности версий операционной системы Android (рис 1.7).

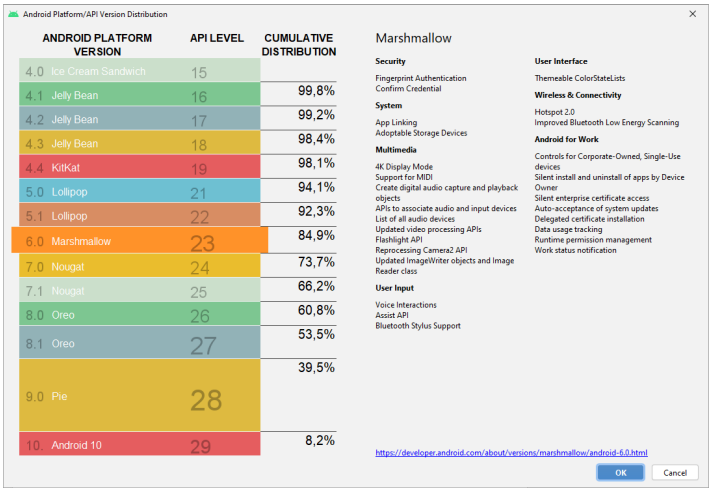


Рисунок 1.7 – Выбор API

Создание проекта происходит достаточно долго. При появлении окна на рис. 1.8 можно считать, что приложение создано и готово к разработке.

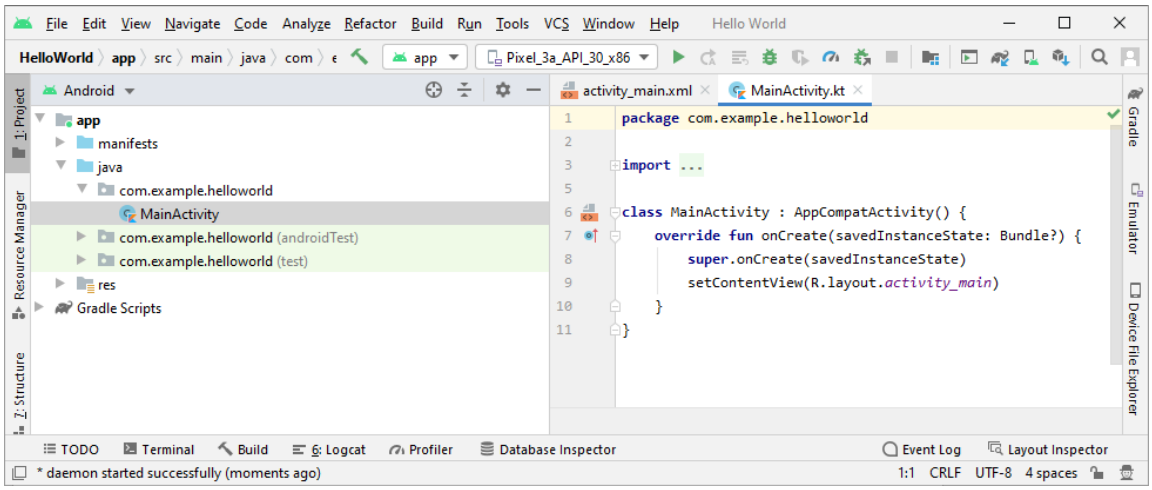


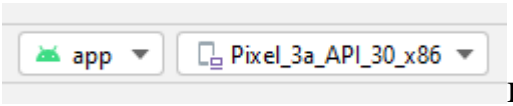
Рисунок 1.8 – Окно проекта

Слева представлено дерево папок. Проект под Android содержит множество файлов, знакомиться будем постепенно. Справа открыт код единственного пока в проекте kt-файла – MainActivity.kt

Внизу – процесс сборки. Зеленые галочки говорят о том, что приложение собралось успешно и готово к запуску.

Проект можно запускать. Однако при первом запуске программы устройство может отсутствовать 

.

Можно выбрать устройство из списка (если устанавливали эмулятор) 

или можно попробовать подключить реальное устройство.

Задание 3*. Подключение устройств

Для того чтобы можно было работать с реальным устройством, должен быть установлен соответствующий USB-драйвер. Обычно никаких дополнительных действий не требуется,

но если после подключения устройство все равно не видимо в окошке выше, то нужно будет установить драйвер.

Установка драйверов реальных устройств – простая, но сильно зависимая от производителя устройства операция. Нужно установить на компьютер USB-драйвер нашего устройства. Хорошо, если он есть на сайте производителя. Попробуйте найти специальный драйвер для своего устройства и установить его.

Ссылки на драйвера и инструкции по установке можно найти на сайте:

https://developer.android.com/studio/run/oem-usb.

На самом устройстве должна быть включена опция «Отладка по USB». Посмотрите в Настройках устройства меню для разработчиков и поставьте флажок «Отладка по USB2.

Тогда нужно пройти по пути:

Настройки ► Система ► О телефоне► Номер сборки

Также может встречаться более короткий путь:

Настройки ► О телефоне ► Номер сборки

И от 5 до 10 раз нажать пальцем по пункту с номером сборки, пока на экране устройства не появится уведомление «Включен режим разработчика».

При первом подключении к компьютеру, на экране устройства появится запрос разрешения на отладку. Необходимо согласиться, нажав «Да». Этот же запрос будет появляться, при попытке подключить устройство к другому компьютеру. После установки драйверов и подтверждения на устройстве в окне запуска должно появиться устройство.

Задание 4. Настройка эмулятора

Программу можно запустить и на эмуляторе, без реального устройства. Однако предварительного его нужно создать (рис. 1.9).

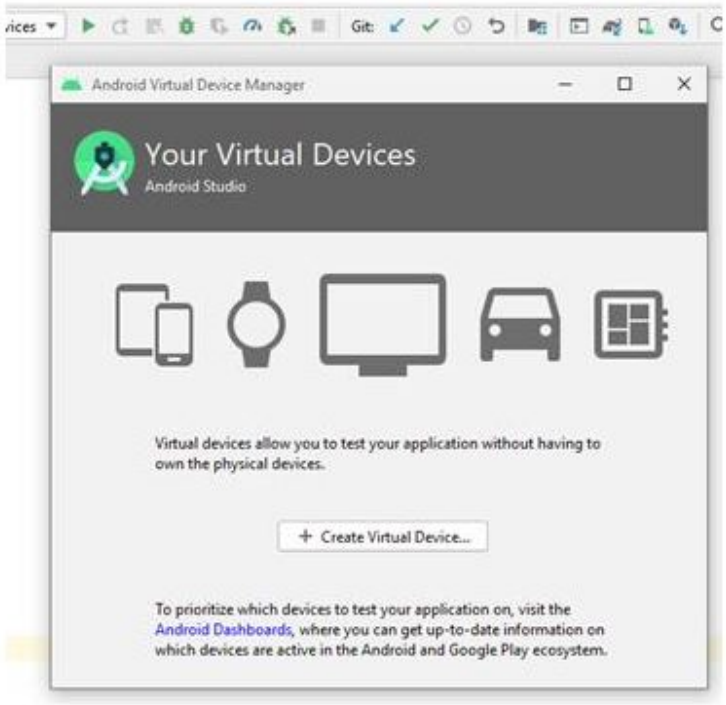


Рисунок 1.9 – Создание виртуального устройства

При установке уже должен быть образ последней версии API – можно выбрать его из Recommended.

Бывает, что стандартные установки не позволяют запуститься эмулятору. В этом случает можно попробовать скачать образ со вкладки x86_Images и увеличить размер памяти (рис. 1.10). При этом образ будет размером около 1,5 Гбайт.

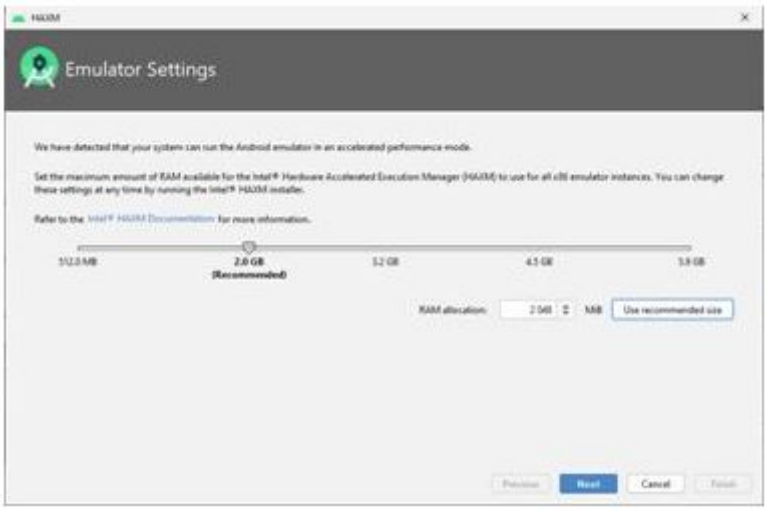


Рисунок 1.10 – Настройки эмулятора

Задание 5. Проект Android

Android-проект – это сложная система, включающая в себя много файлов, разложенных по специальным папкам. Рассмотрим структуру проекта.

Для программиста Android-проект представляет, прежде всего, исходный код на языке программирования Kotlin или Java. На нем пишется то, что делает приложение. Также в состав проекта входят XML-файлы, в которых описывается внешний вид приложения.

Первая папка **res**. Это папка ресурсов – вся графика и разметки, XML-файлы с указаниями Android как расположить интерфейсные элементы на экране, хранятся здесь.

1. Откройте основную разметку **activity_main.xml** (рис. 1.11).

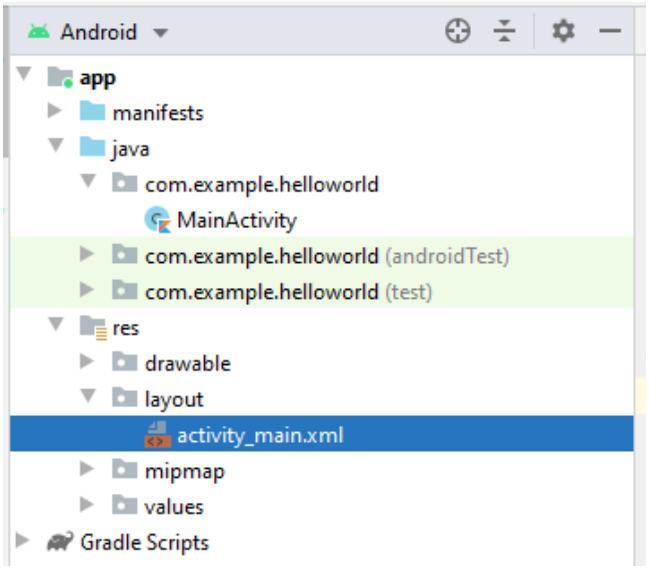


Рисунок 1.11 – Дерево проекта

Вначале открывается графический вид. Обратите внимание на кнопки в правом верхнем углу. С их помощью можно регулировать масштаб просмотра (рис. 1.12).

Рисунок 1.12 – Управление масштабом

Можно редактировать разметку как в визуальном виде, так и кодом, переключая вкладки справа вверху (рис. 1.13).

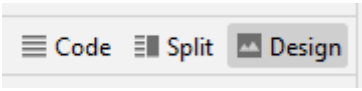


Рисунок 1.13 – Вкладки проекта

2. Измените вид приложения в графическом виде, а потом посмотрите на изменения в коде, которые сгенерирует среда программирования. Выровняйте текст по верхнему краю и измените размер текста по своему усмотрению (рис. 1.14).

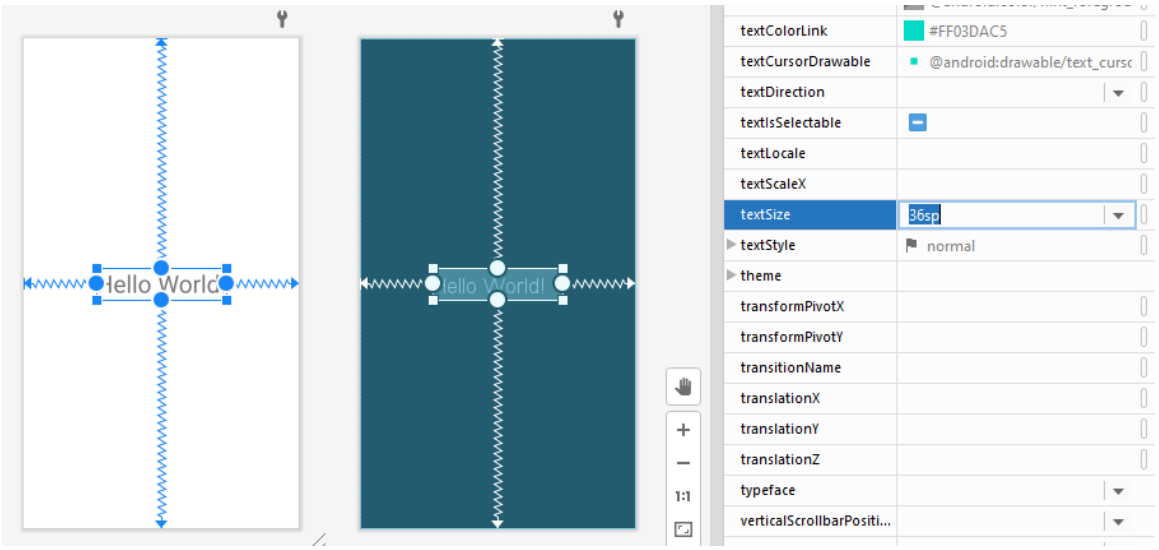


Рисунок 1.14 – Изменение свойств

Вводить значение необходимо в **sp** – единицах, независящих от плотности экрана. Это сделает вид приложения одинаковым на разных устройствах, но с учетом пользовательских настроек шрифтов.

Размер текста практически всегда стоит указывать в **sp**, а размеры элементов в **dp**.

Для выравнивания **TextView** по верхнему краю достаточно удалить нижнюю «пружинку».

3. Переместите картинку **android.png** прямо в проект, в папку **drawable**. При попытке переместить Android Studio предложит использовать папку **drawable-v24**, что означает, что ресурс будет виден только для API-24 и выше, то есть при запуске на устройстве с Android версии 5.x будет ошибка. Можно указать общую для всех версий и устройств и настроек папку **drawable**, тогда будет работать везде (рис. 1.15).

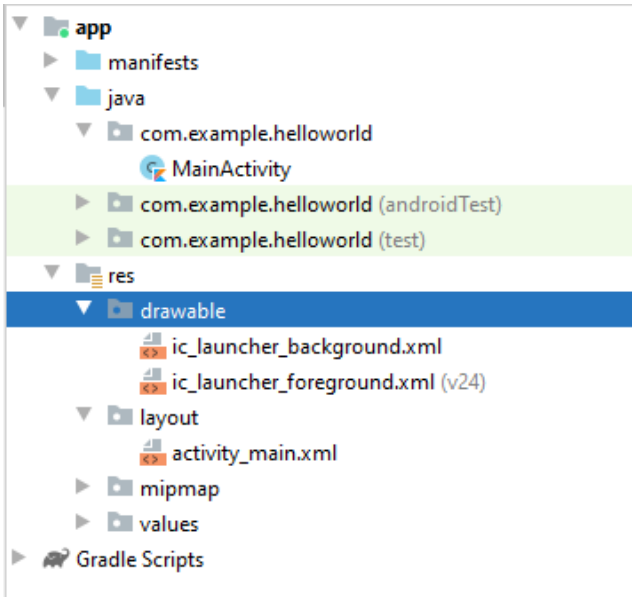


Рисунок 1.15 – Папка res

4. Добавьте (перетащите с палитры) картинку – элемент **ImageView** в центр разметки (рис 1.16).

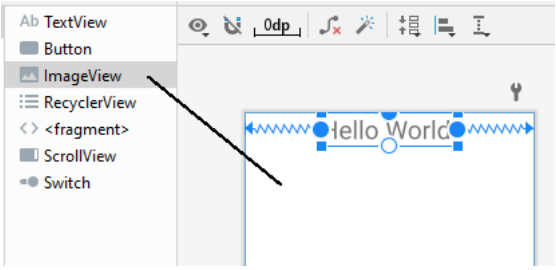


Рисунок 1.16 – Компоненты проекта

И выберете картинку (рис. 1.17).

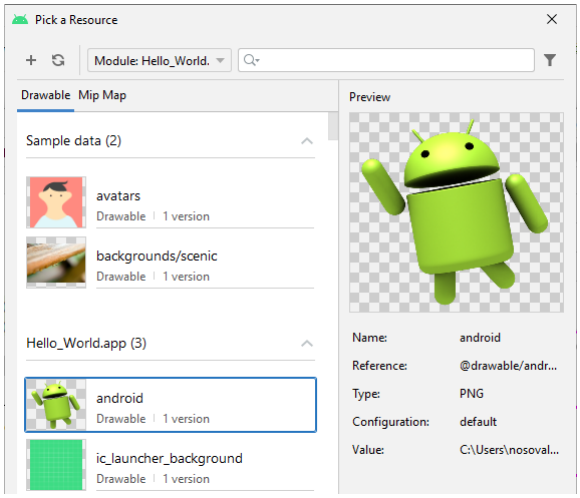


Рисунок 1.17 – Выбор изображения

5. Переключитесь в код (рис. 1.18):

```
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:textSize="36sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/android"
        tools:layout_editor_absoluteX="103dp"
        tools:layout_editor_absoluteY="135dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 1.18 – Фрагмент кода

Проанализируйте настройки компонентов.

Прежде всего, на разметке появились **TextView** и **ImageView**. Мы видим атрибуты размера текста у **TextView** **android:textSize="36sp"** и атрибуты расположения картинки в

центре разметки, привязки к краям «родителя»: **app:layout_constraintLeft_toLeftOf**="parent" и т.д. И еще мы видим установленную на ImageView картинку: **app:srcCompat="@drawable/android**".

6. Обратите внимание на атрибуты размеров. У всех **View**, которые попадают в разметку очень редко, используют размеры в пикселях. Часто используются значения "**wrap_content**" и "**match_parent**", означающие «по размеру содержимого» и «на весь контейнер». Изменим эти значения для картинки.

7. Отредактируйте файл **/res/layout/activity_main.xml:**

```
<ImageView
    android:id="@+id/imageView1"
    android:layout_"match_parent"
    android:layout_"match_parent"
```

8. Вернитесь к графическому виду (рис. 1.19).

Рисунок 1.19 – Графический вид проекта

Картинка растянулась на весь экран (при этом может стать нерезкой, если изначально была небольшого размера). Выберите и переместите в папку **drawable** и используйте любое свое изображение.

Теперь обратите внимание на строку «HelloWorld». Если в XML-коде **TextView** нажать на ней «Alt+Enter», то Android Studio предложит переместить текст в строковые ресурсы (рис. 1.20).

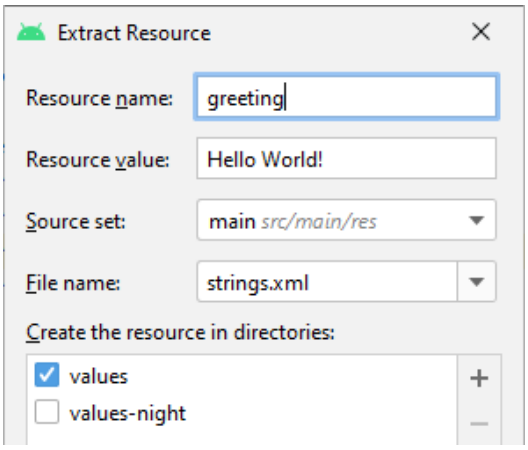


Рисунок 1.20 – Перемещение в строковые ресурсы

XML-текст станет таким: **android:text="@string/greeting"**, что означает «текст будет установлен из текстового ресурса с именем greeting.

Таким образом, текстом **TextView** будет строка, объявленная в файле **strings.xml** в папке **values**, что удобно при работе (рис. 1.21).

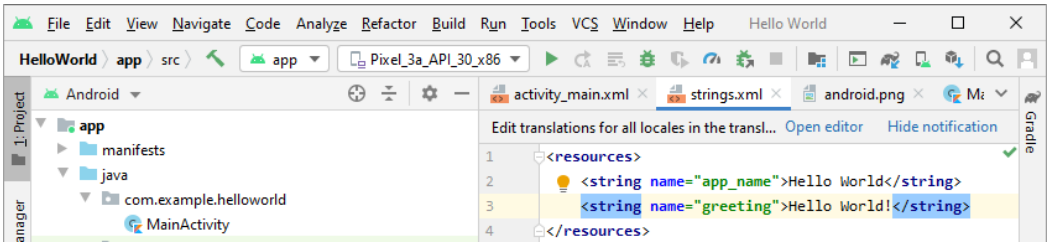


Рисунок 1.21 – Файл **strings.xml**

Для того чтобы изменить интерфейс программы, например, перевести на другой язык, достаточно изменить соответствующий файл. Например, на скриншоте имеется ресурс **app_name**. Если изменить его значение, изменится строка в заголовке приложения и имя самого приложения на экране устройства.

Кроме того, локализацию приложения можно произвести, просто добавив еще одну папку с переводом. Для английского – **values-en**, например. Система сама будет выбирать соответствующую папку в зависимости от настроек смартфоне.

9. Измените значения ресурсов в файле **/res/values/strings.xml**. И посмотрите на результат в разметке.

Остается самое главное. Папка **src** – папка с Kotlin-кодом (рис. 1.22).

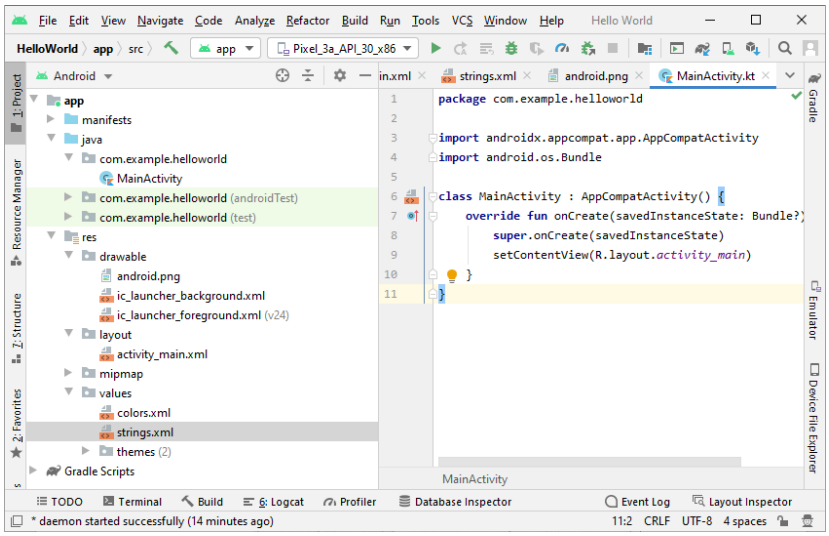


Рисунок 1.22 – Код программы

Программа для Android обычно состоит из одной или нескольких экранов (активностей). Для каждой создается класс, наследник класса **Activity**. На рисунке 1.22 есть функция – **onCreate()**. Она вызывается тогда, когда активность создается. Пока она состоит из единственной команды – установки на активность разметки.

10. Добавьте анимацию к картинке при старте программы. Отредактируйте файл **/src/…/MainActivity.kt** (рис. 1.23).

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
          //Создадим обект анимации - выпадение сверху
        val imageView : ImageView = findViewById(R.id.imageView)
        val translation = TranslateAnimation( fromXDelta: 0f,  toXDelta: 0f,  fromYDelta: -1000f,  toYDelta: 0f)
          // Устанавливаем длительность 5 секунд
        translation.duration = 5000
          // Просчитываем эффект "подпрыгивание"
        translation.interpolator = BounceInterpolator()
          // Запускаем анимацию на размещенном изображении
        imageView.startAnimation(translation)
    }
}
```

Рисунок 1.23 – Код программы

При изменении **MainActivity.kt** надо добавлять библиотеки. Обычно, при использовании разных классов нужно добавлять соответствующие строки «import». В Android Studio можно наживать **«Alt-Enter»** на строках, выделенных красным и выбрать **import**… везде, где требуется. В примере в коде появятся такие строки (рис. 1.24):

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.animation.BounceInterpolator
import android.view.animation.TranslateAnimation
import android.widget.ImageView
```

Рисунок 1.24 – Добавление библиотек

11. Теперь проведите тестирование проекта. Картинка должна «выпрыгивать» сверху (рис. 1.25).

Рисунок 1.25 – Мобильное приложение в эмуляторе

12. Поменяйте картинку на любую свою. Однака перед перемещением в папку **drawable** надо правильно переименовать файл. Файлы картинок и других ресурсов могут состоять только из маленьких латинских букв, цифр и знака подчеркивания и не

начинаться с цифры. После перемещения нужно изменить свойство **app:srcCompat** на новый ресурс. При этом указывается только имя файла без расширения.

Задание для самостоятельной работы: создайте приложение – визитку, содержащую информацию о разработчике, контактную информацию и фото. Анимируйте фото, изменив настройки, указанные в лабораторной работе по своему усмотрению.

Лабораторная работа №2.
Ориентация экрана.
Работа с компонентом «Кнопка»

Цель: изучить особенности настройки приложений со сменой ориентации экрана, изучить особенности работы компонента «Кнопка» в среде разработки Android Studio.

Ход работы

Первые мобильные устройства не имели возможность смены ориентации экрана. Разработчики создавали программы, в которых смена экрана становилась возможной. В последствии такие настройки стали включаться в возможности аппаратных платформ. В настоящее время существует два режима работы экрана: портретный режим и альбомный. Однако при разработке приложений, где возможна смена ориентации экрана, могут возникать проблемы, например, часть экрана помещается за пределы видимости. Рассмотрим особенности работы с такими приложениями.

1. Запустите Android Studio.
2. Создайте новый проект (рис. 2.1):

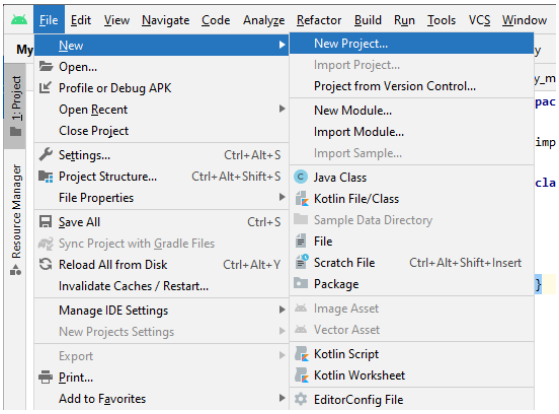


Рисунок 2.1 – Главное меню

3. Выберите тип Activity – **Empty** (рис. 2.2).

Шаблон **Empty Activity** предназначен для обычных телефонов. На картинке над названием шаблона вы видите приблизительный вид приложения с использованием данной заготовки. Для учебных программ в 99% подойдёт этот вариант. Практически все примеры на сайте написаны с помощью данного шаблона.

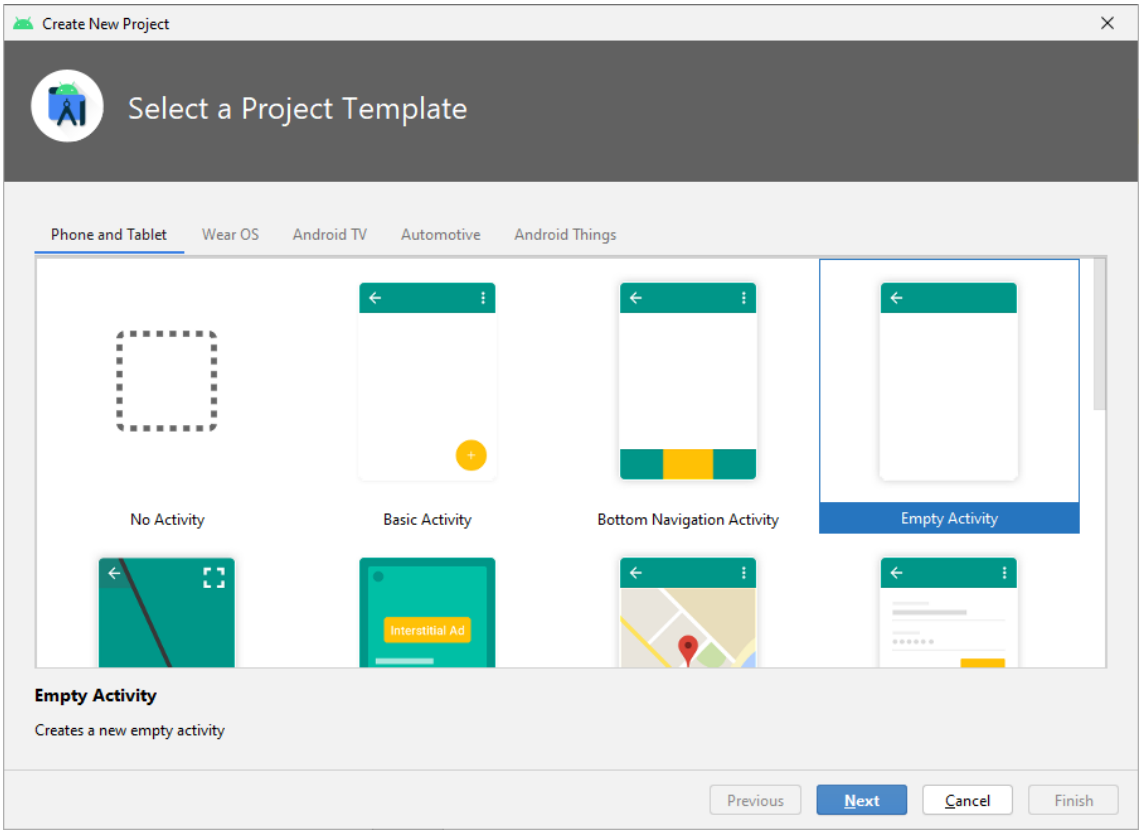


Рисунок 2.2 – Выбор типа Activity

4. Нажмите кнопку «Next».

5. Укажите Name – название проекта, укажите папку – место его расположения на компьютере, язык программирования – Kotlin, версию API (можно оставить по умолчанию) (рис. 2.3).

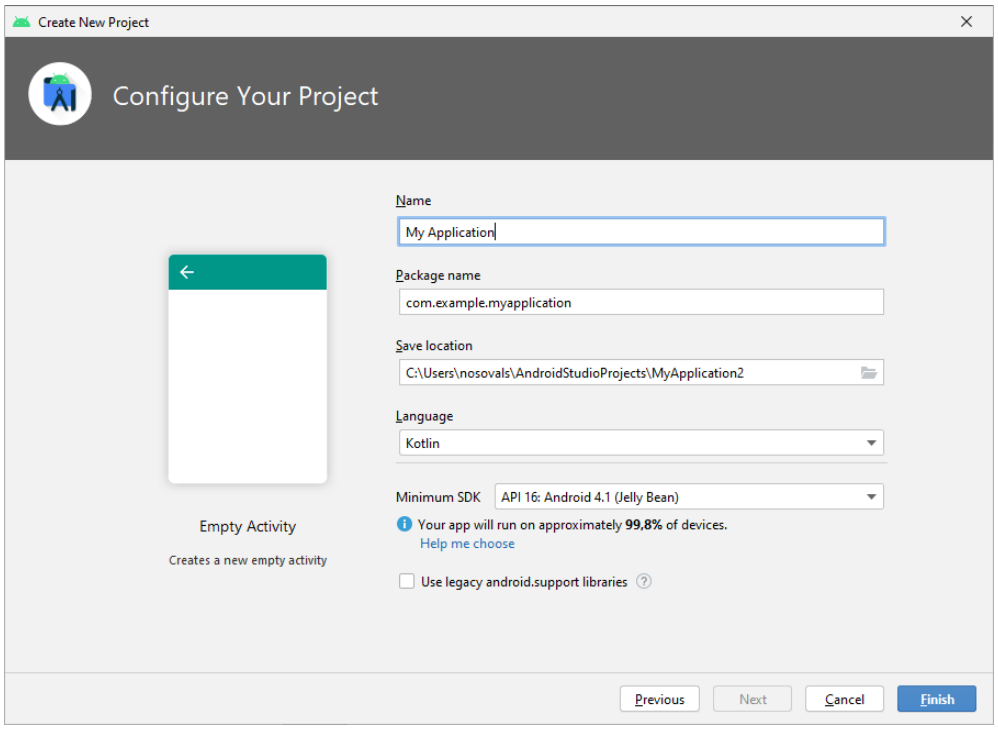


Рисунок 2.3 – Настройки проекта

6. Нажмите кнопку «Finish».

7. Дождитесь окончания процессе сборки проекта, процесс может занять несколько минут.

8. В Android Studio включены скрипты сборки проекта Gradle. Однако их необходимо периодически обновлять. Сделать это можно следующим образом:

9. Выберите в списке элементов проекта узел Gradle Scripts, откройте двойным щелчком мыши файл build.gradle, который относится к модулю (рис. 2.4).

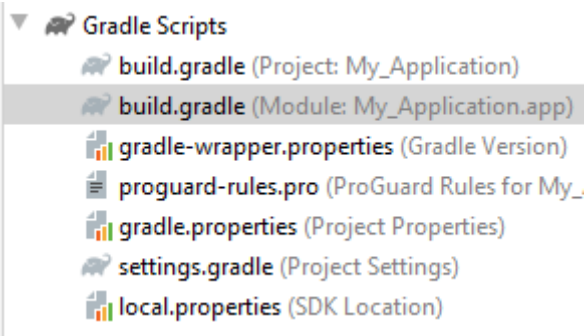


Рисунок 2.4 – Скрипты сборки

10. Если библиотеку необходимо обновить, строка в модуле будет подсвечена (рис. 2.5).

```
implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
implementation 'androidx.core:core-ktx:1.3.2'
implementation 'androidx.appcompat:appcompat:1.2.0'
implementation 'com.google.android.material:material:1.3.0'
implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
testImplementation 'junit:junit:4.+'
androidTestImplementation 'androidx.test.ext:junit:1.1.2'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
```

Рисунок 2.5 – Подключение библиотек

11. При наведении на подсвеченную строку курсом мыши, Android Studio предложит обновить библиотеку (рис. 2.6).

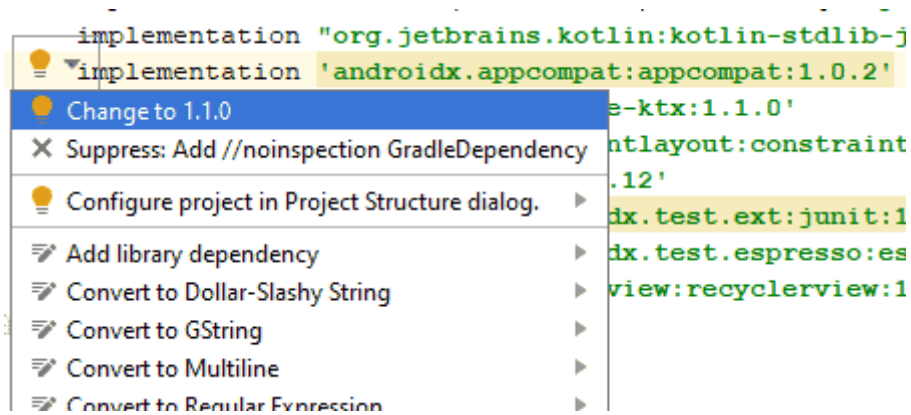


Рисунок 2.6 – Обновление версии библиотеки

12. Необходимо периодически проверять актуальность библиотек и обновлять их.

13. Запустите проект и убедитесь, что обновления и сборка проекта закончились успешно. Для этого нажмите на кнопку «Run» (зеленый треугольник на панели инструментов). На забудьте выбрать эмулятор перед запуском (рис. 2.7).

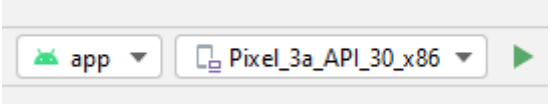


Рисунок 2.7 – Выбор эмулятора

14. Дождитесь на эмуляторе загрузки операционной системы Android (рис. 2.8).

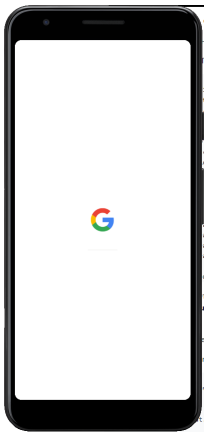


Рисунок 2.8 – Загрузка эмулятора

15. Если проект запущен, на экране будет выбранный ранее шаблон Activity. По умолчанию цвет фона приложений белый. В левом верхнем углу указывается название проекта, введенное при его создании (Name) (рис 2.9).

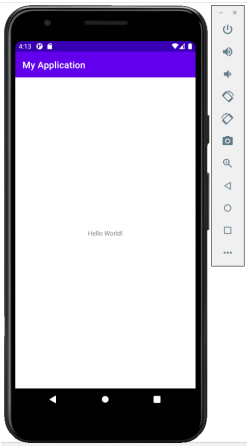


Рисунок 2.9 –Запущенное приложение

16. Закройте эмулятор.

17. Создайте новую папку в папке res. Для этого в контекстном меню выберите команды: **New – Android resource directory** (рис. 2.10).

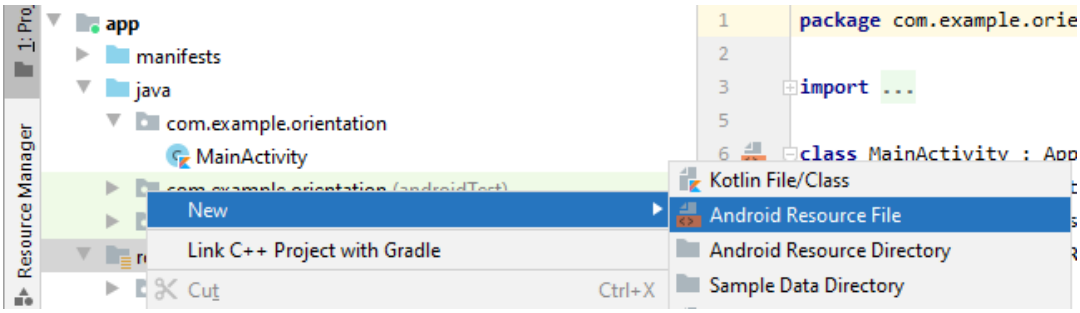


Рисунок 2.10 – Выбор команды

18. Выберите ориентацию (рис. 2.11, 2.12).

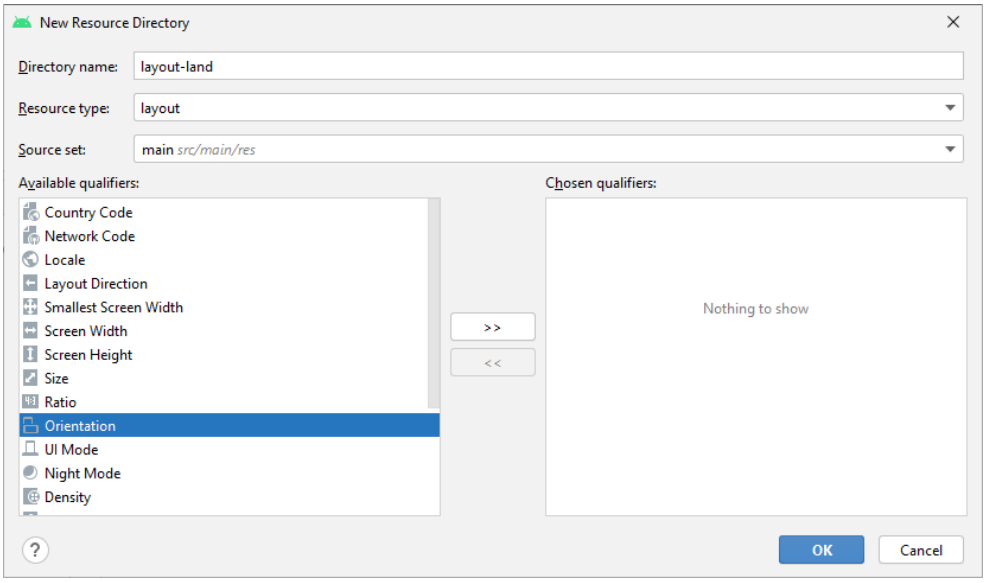


Рисунок 2.11 – Выбор ориентации экрана

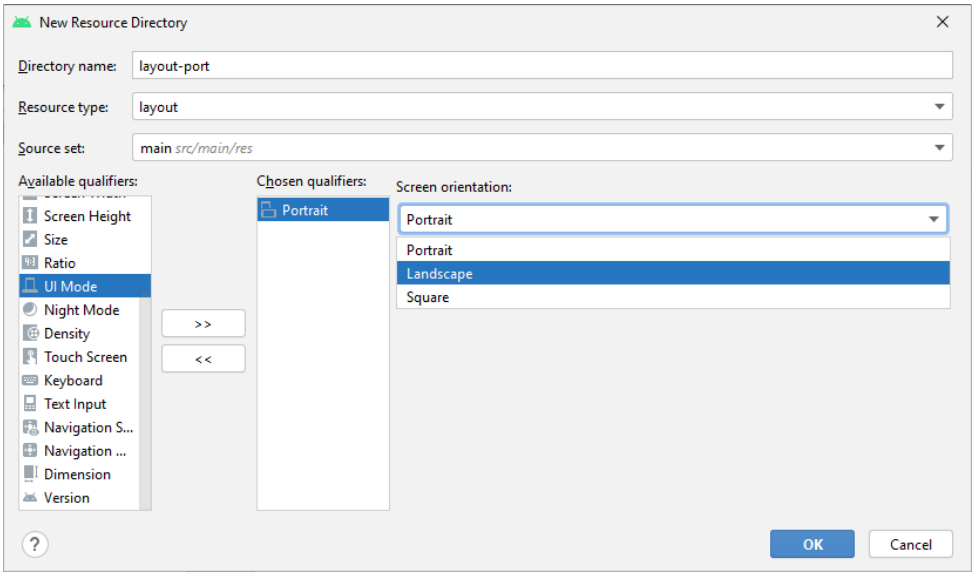


Рисунок 2.12 – Вид ориентации

19. Затем можно открыть файл и преступить к разработке.

20. Однако в Android Studio можно автоматизировать процесс. Для этого перейдите на вкладку **activity_mail.xml.** На нижней панели инструментов выберите кнопку **Orientation for Preview** и в выпадающем списке укажите **Create Landscape Variation** (рис. 2.13).

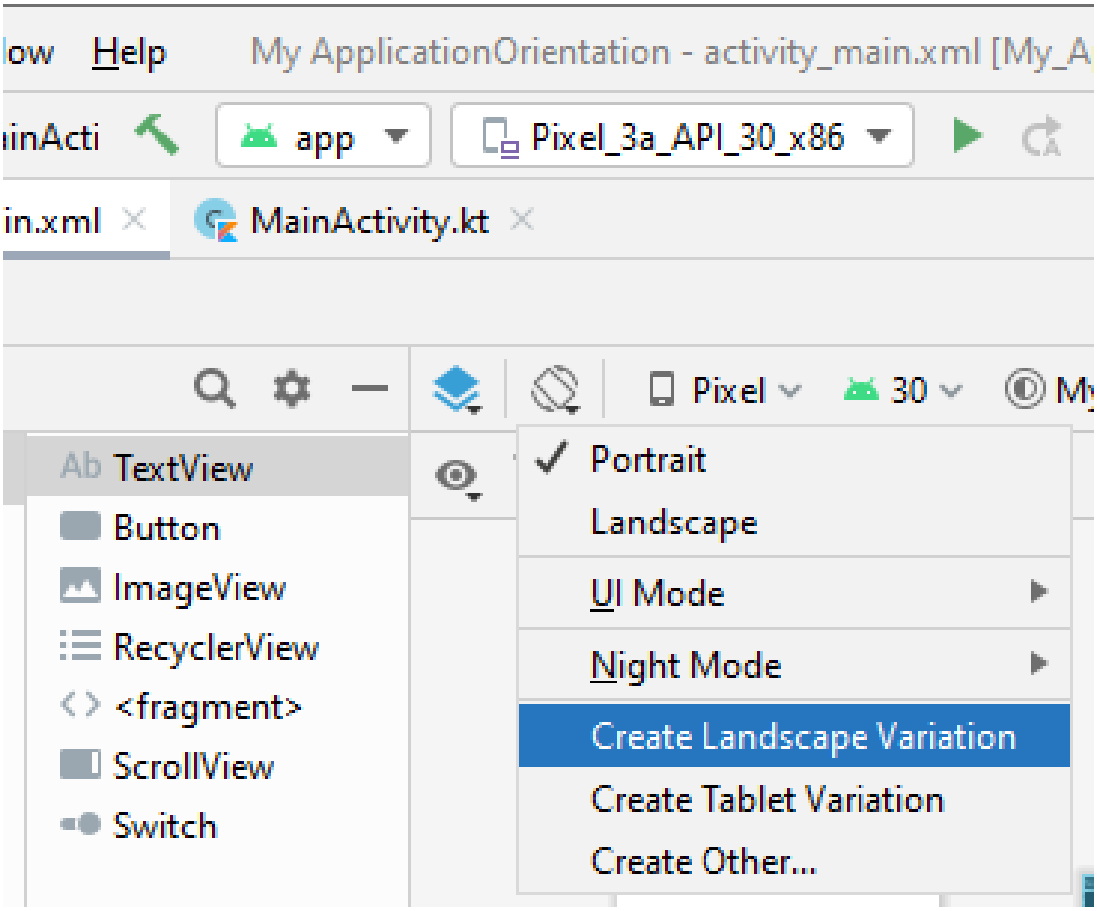


Рисунок 2.13 – Изменение ориентации экрана

21. Убедитесь, что проект находится в режиме Design (в правом верхнем углу) (рис. 2.14).

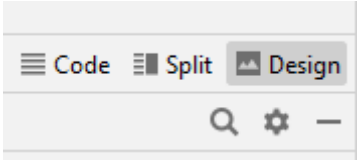


Рисунок 2.14 – Вкладки

22. Перейдите в режим «Code» (рис. 2.15).

```
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <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" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 2.15 – Код программы

23. Все компоненты приложения размещаются внутри шаблона – **Layout**. Сейчас в студии используется код шаблона на основе **ConstraintLayout**.

24. Переключитесь на вкладку **MainActivity.kt** и изучите Kotlin-код (рис. 2.16).

activity_main.xml | build.gradle (:app) | MainActivity.kt

```
package com.example.myapplication

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Рисунок 2.16 – Код программы

Первой строкой указан **Package Name**, введенный при создании проекта. Далее указаны строки импорта необходимых для работы проекта классов (они могут быть сгруппированы, раскройте их).

Далее идет объявление класса **MainActivity**, наследника абстрактного класса **Activity** – **AppCompatActivity**. Это базовый класс.

Внутри класса представлен метод **onCreate()** – он вызывается при создании и отображении разметки активности. Указано ключевое слово **override** (значит метод будет переопределен).

Строка кода **super.onCreate(savedInstanceState)** вызывает метод-конструктор родительского класса, выполняющий необходимые операции для работы активности. Строка кода **setContentView(R.layout.activity_main)** подключает содержимое из файла разметки. В качестве аргумента указывается имя файла без расширения из папки **res/layout**. По умолчанию проект создаёт в нём файл **activity_main.xml**. Можно переименовать файл или создать свой файл, например, с именем **my.xml** и подключить его к своей активности. Тогда код будет выглядеть так: **setContentView(R.layout.my).** При назначении имен необходимо придерживаться стандартов. Если создается еще одна разметка для активности, то рекомендуется использовать префикс **activity_** для имени файла. Например, разметка для второй активности может иметь имя **activity_second.xml**.

25. Измените фон проекта. Для этого вернитесь на вкладку **activity_mail.xml** в режиме **Design**.

26. Выберите в дереве компонентов проекта разметку **ConstraintLayout** (рис. 2.17):

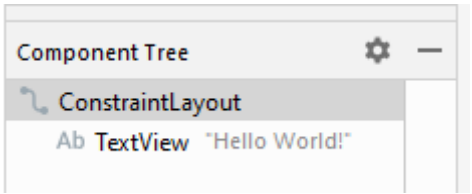


Рисунок 2.17 – Дерево компонентов

27. На вкладке **Attributes** для объекта **ConstraintLayout** в панели свойств отобразятся свойства выбранного компонента.

В прокрутке **All attributes** найдите свойство **background**. И измените цвет фона на вкладке **Color** (рис. 2.18):

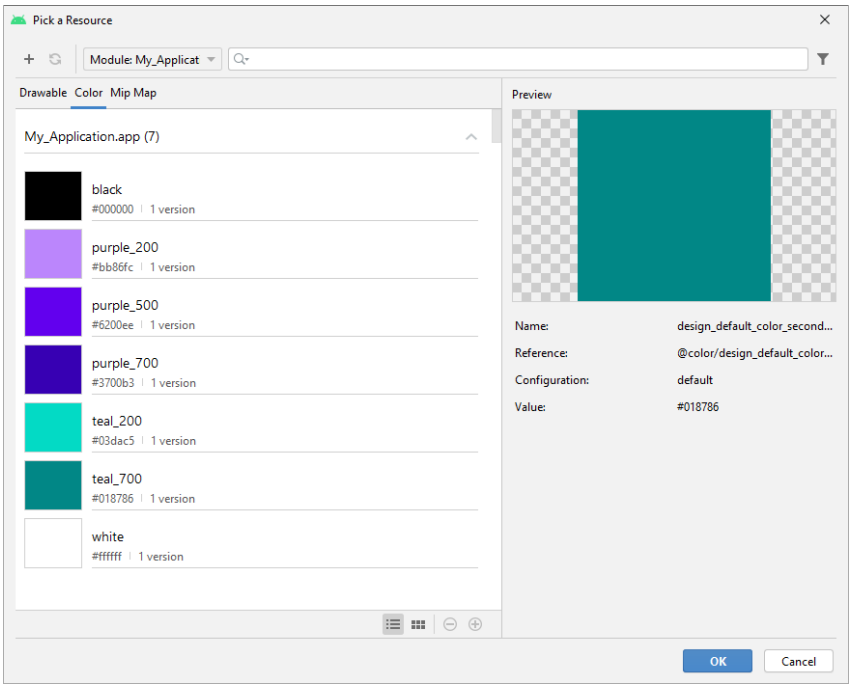


Рисунок 2.18 – Окно изменения цвета

28. Если переключиться в режим Code, то можно увидеть, что в xml-код добавлена строка: android: background=" @color/teal_700".

29. Из палитры компонентов переместите **ImageButton** на **Activity** (рис. 2.19):

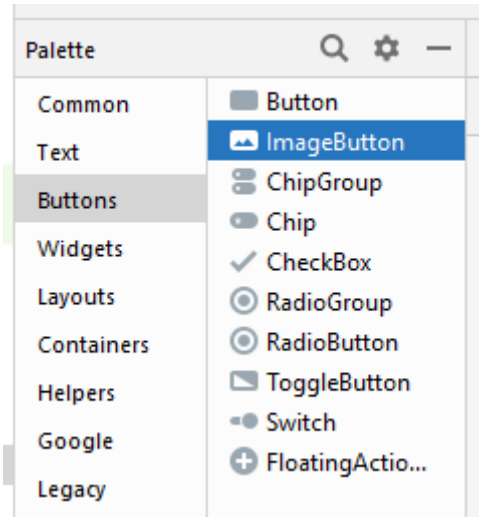


Рисунок 2.19 – Палитра компонентов

30. Andriod Studio предложит выбрать изображение. Укажите любое.

31. Подготовьте изображение для кнопки. Скопируйте изображение в буфер обмена и вставьте изображение с помощью контекстного меню в папке **drawable** в дереве проекта в узле **res** (рис. 2.20).

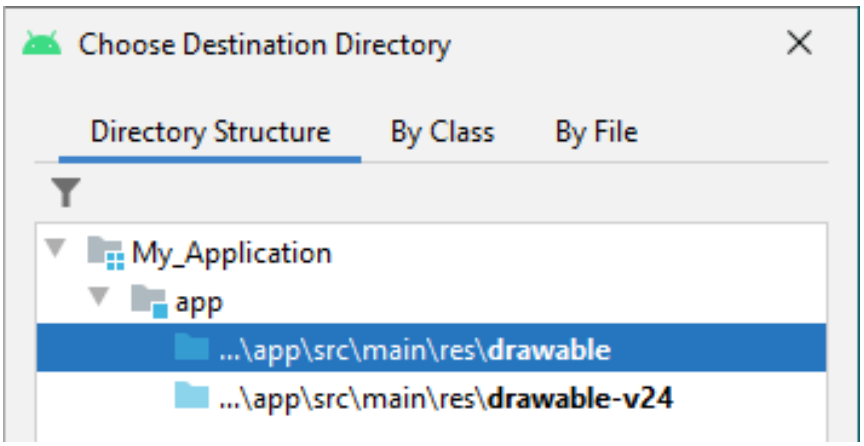


Рисунок 2.20 – Выбор папки

32. Подтвердите операцию копирования файла (рис. 2.21). Помните о правилах именования ресурсов для проекта.

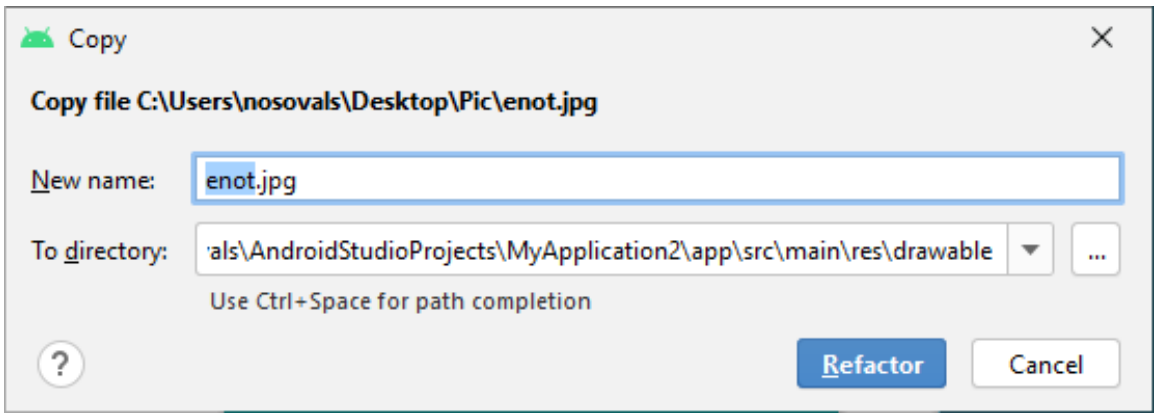


Рисунок 2.21 – Копирование файла изображения

33. Выделите элемент **ImageButton** на форме. В панели свойств выберите свойство **srcCompat**, нажмите на кнопку и выберите в диалоговом окне ресурс в категории **Drawable** (укажите имя добавленного ранее файла) (рис. 2.22).

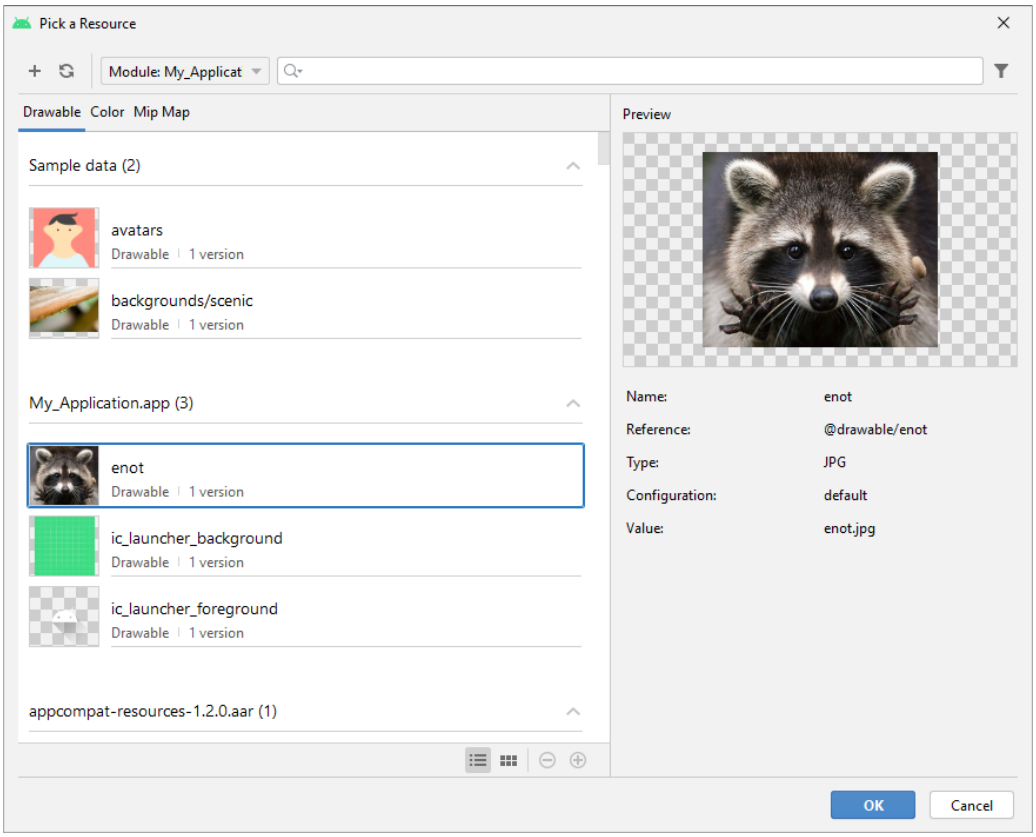


Рисунок 2.22. Выбор изображения

34. Удалите компонент **TextView** с надписью **Hello, World**.

35. Из палитры компонентов снова переместите компонент **TextView** под кнопку и убедитесь, что у него свойство ID не является пустым (например, оно будет **textView**).

36. Увеличьте размер текста. Для этого в свойстве **textAppearance** установите значение, например, **AppCompat.Display2**. Измените свойство **text** (рис. 2.23):

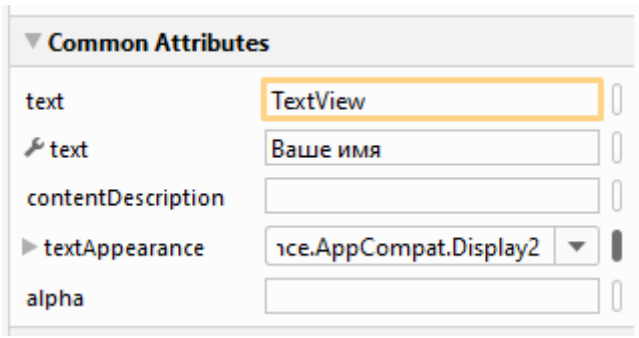


Рисунок 2.23 – Установка свойств

37. Если в дереве компонентов появились красные предупредительные символы, значит не все настройки размещения компонентов относительно разметки установлены. Android Studio может попытаться определить их самостоятельно, при нажатии кнопки **Infer Constraints** (в виде волшебной палочки) (рис. 2.24):

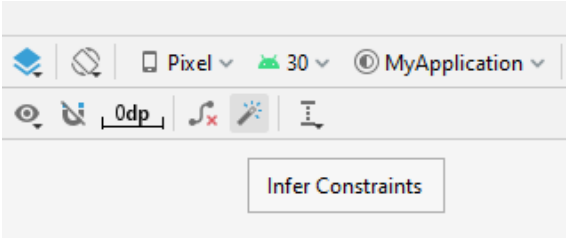


Рисунок 2.24 – Исправление ошибок

38. Проверьте содержимое xml файла (рис. 2.25):

```
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/teal_700"
    tools:context=".MainActivity">

    <ImageButton
        android:id="@+id/imageButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="100dp"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="64dp"
        android:layout_marginEnd="160dp"
        android:layout_marginRight="160dp"
        android:layout_marginBottom="56dp"
        android:contentDescription="@string/racoon_image"
        android:visibility="visible"
        app:layout_constraintBottom_toTopOf="@+id/textView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.054"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/android" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="104dp"
        android:textAppearance="@style/TextAppearance.AppCompat.Display2"
        android:visibility="visible"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageButton"
        tools:text="@string/yourName" />
```

Рисунок 2.25 – Код программы

39. Введите текст, замещающий изображение, в свойстве **android:contentDescription**. Он используется для озвучки системой для слабовидящих людей. Переведите текст в строковые ресурсы. Для этого встаньте на него курсором и нажмите сочетание **Alt+Enter** (рис. 2.26):

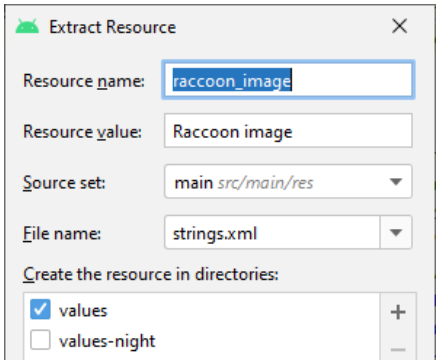


Рисунок 2.26 – Настройка ресурсов

40. Повторите тоже самое для строки **tools:text=«Ваше имя»**.

41. Запустите проект, убедитесь, что цвет фона изменился, изображение отображается на кнопке.

42. Доработайте код, чтобы по нажатию на кнопку изменялся текст. Для этого измените код (рис. 2.27):

```
package com.example.myapplication

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.ImageButton
import android.widget.TextView

//import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {
private lateinit var textView:TextView
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        textView = findViewById(R.id.textView)
        var imageButton : ImageButton  = findViewById(R.id.imageButton)
        imageButton.setOnClickListener { it: View!
            textView.text = "Hello, friend!"
        }
    }
}
```

Рисунок 2.27 – Код программы

43. Проверьте проект.

44. В папке **\app\build\outputs\apk\debug** проекта можно найти готовый APK-файл, который можно устанавливать. При этом важно помнить, что для установки приложений сторонних разработчиков (не через Play Market) в телефоне должно быть разрешение на установку неподписанных приложений.

45. Доработайте проект так, чтобы пользователь вводил свое имя и по нажатию на кнопку программа здоровалась с ним. Для этого переместите с палитры компонентов компонент **Plain Text** над кнопкой (рис. 2.28).

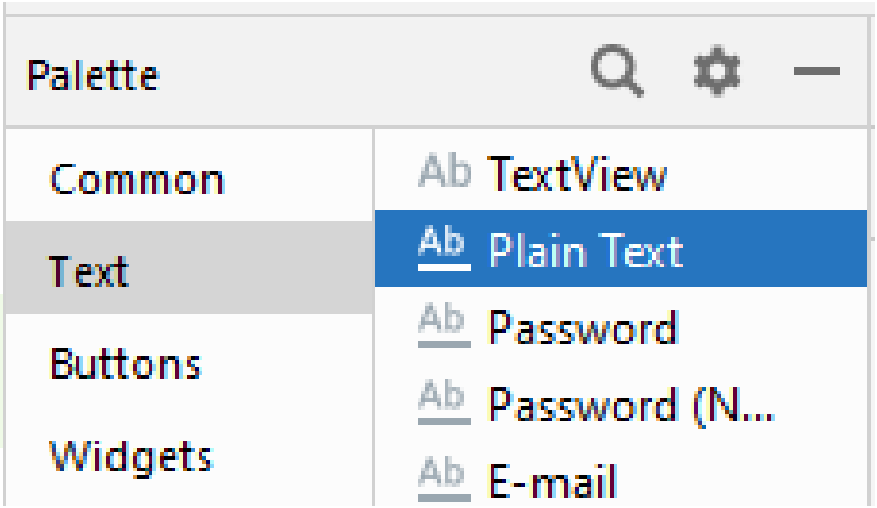


Рисунок 2.28 – Палитра компонентов

46. Исправьте ошибки с помощью кнопки **Infer Constraints** (в виде волшебной палочки), по появлении предупреждающих символов около компонента, доработайте код. В режиме Code укажите свойство **android:hint**, это строка-подсказка, которая исчезает при введении текста.

47. Внесите обработку кода (рис. 2.29):

```
package com.example.myapplication

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.EditText
import android.widget.ImageButton
import android.widget.TextView

class MainActivity : AppCompatActivity() {
    private lateinit var textView: TextView
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        textView = findViewById(R.id.textView)
        var imageButton: ImageButton = findViewById(R.id.imageButton)
        var editText: EditText = findViewById(R.id.editTextTextPersonName)
        imageButton.setOnClickListener { it: View!
            if (editText.text.isEmpty()) {
                textView.text = "Hello, friend!";
            } else {
                textView.text = "Hello, " + editText.text
            }
        }
    }
}
```

Рисунок 2.29 – Код программы

48. Проверьте работу проекта (рис. 2.30):

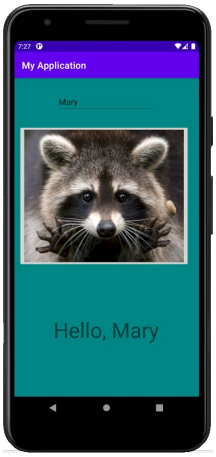


Рисунок 2.30 – Запуск приложения в эмуляторе

49. Сохраните проект.

Задание для самостоятельной работы: создайте приложение, имитирующее общение с пользователем. Узнайте, как зовут пользователя, какое у него настроение и в зависимости от настроения выведите изображение. Доработайте приложение, изменив ориентацию экрана с вертикальной на горизонтальную.

Лабораторная работа №3.
Подсчет количества нажатий по компоненту «Кнопка»

Цель: изучить особенности обработки событий при работе с компонентом «Кнопка».

Ход работы

1. Запустите Android Studio.
2. Создайте новый проект (рис. 3.1):

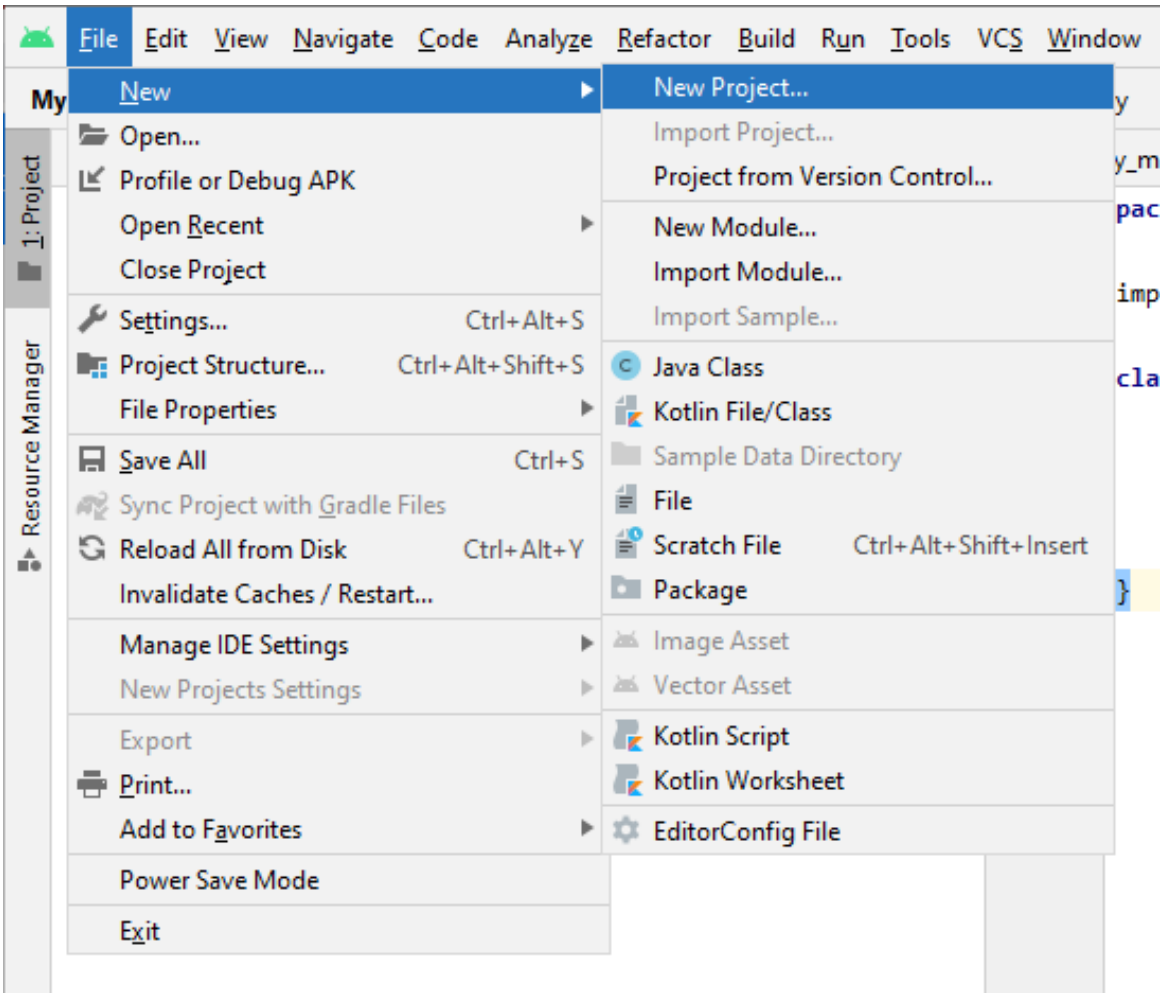


Рисунок 3.1 – Главное меню

3. Выберите тип Activity – **Empty** (рис. 3.2).

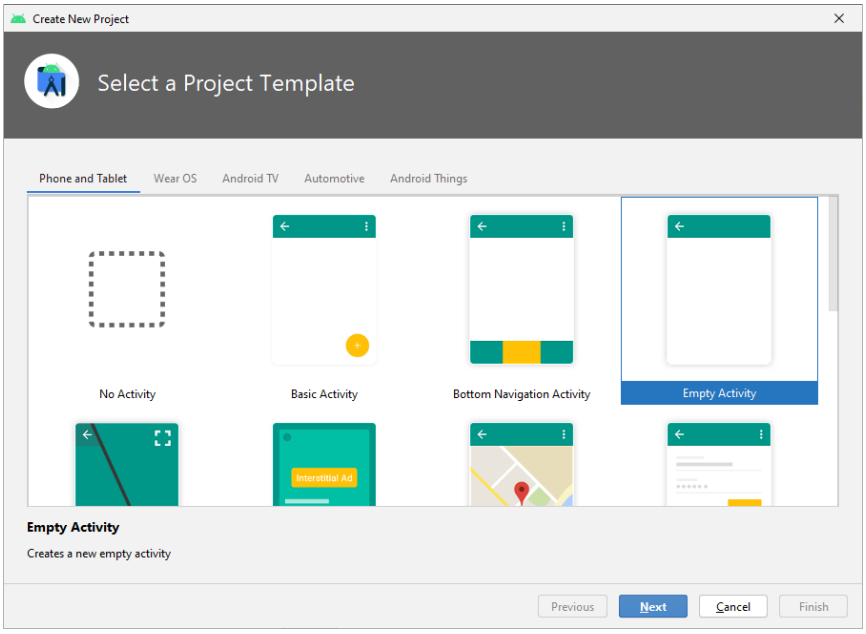


Рисунок 3.2 – Выбор типа Activity

4. Нажмите кнопку «Next».

5. Укажите **Name**, например **Count** (так как мы будем считать количество нажатий по кнопке), укажите папку – место его расположения на компьютере, язык программирования – Kotlin, версию API (можно оставить по умолчанию).

6. Нажмите кнопку «Finish».

7. Добавьте в режиме Design компонент Button (рис. 3.3):

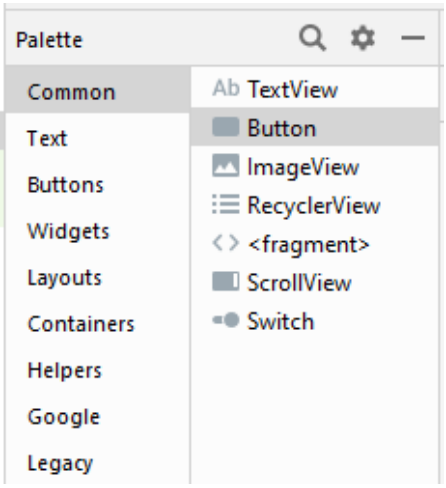


Рисунок 3.3 – Палитра компонентов

8. Измените свойство **text** (рис. 3.4):

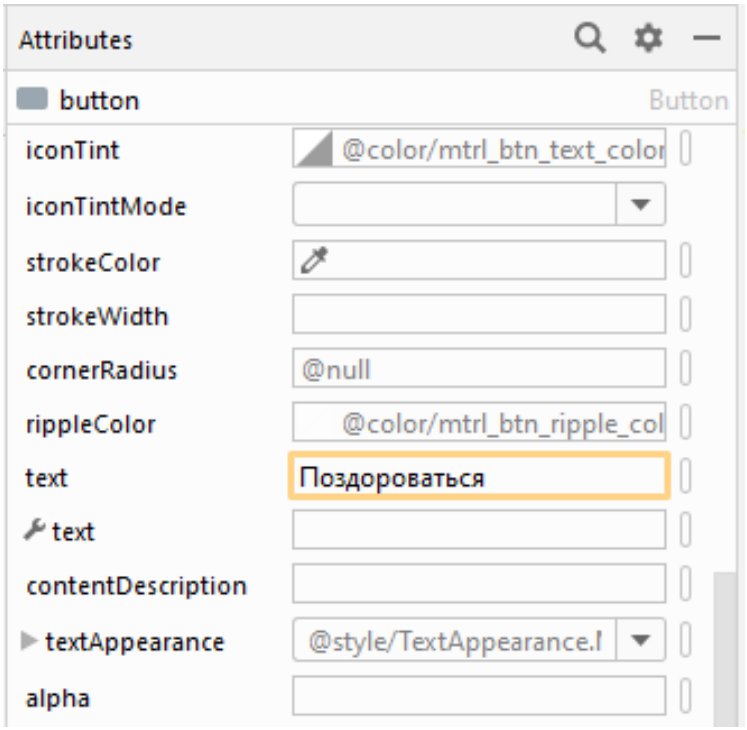


Рисунок 3.4 – Настройка атрибутов компонента

9. Преобразуйте в режиме Code текст в строковые переменные (сочетание клавиш Alt+Enter).

10. Исправьте возможные ошибки при помощи кнопки **Infer Constraints** (в виде волшебной палочки).

11. На форме остался компонент TextView. Удалите текст из TextView (рис. 3.5):

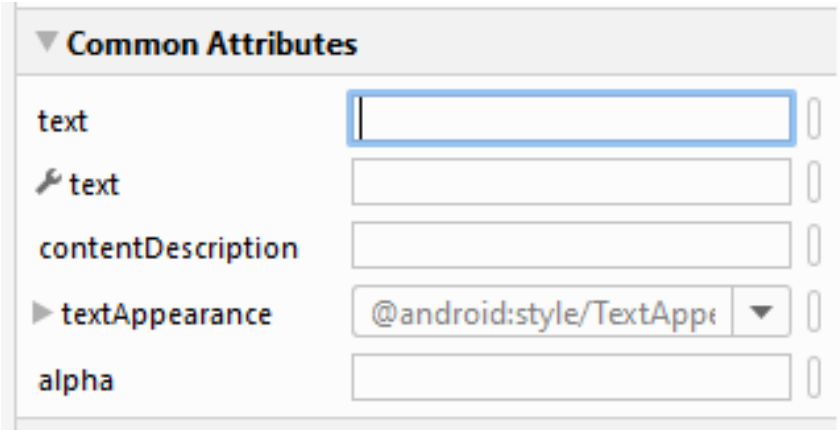


Рисунок 3.5 – Настройка атрибутов компонента

12. Если свойство **id** у него пустое, то необходимо добавить **textView**.

13. Обработайте код. Самостоятельно опишите переменные **button** и **textView** (рис. 3.6).

```
package com.example.count

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)


        var
        var
        button.setOnClickListener { it: View!
            textView.text = "Hello, friend!"
        }
    }
}
```

Рисунок 3.6 – Код программы

14. Запустите приложение, проверьте работу кнопки (рис. 3.7).

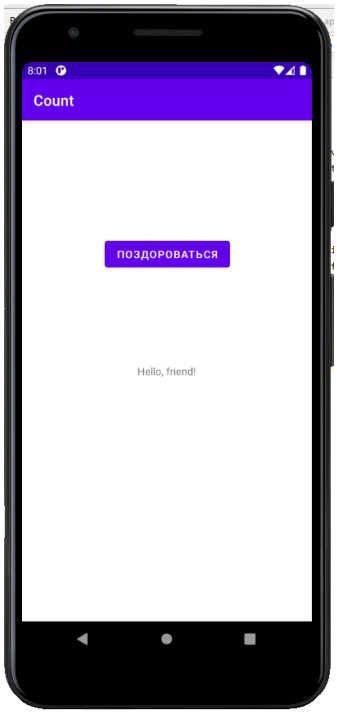


Рисунок 3.7 – Запуск приложения в эмуляторе

15. Добавьте в проект второй компонент **button**. По умолчанию Android Studio именует одинаковый компоненты, добавляя к ним число. Для дальнейшей работы не совсем удобно, поэтому рекомендуется осознанно называть компоненты в соответствии с тем, какую функцию они будут выполнять.

16. Переименуйте второй компонент button (рис. 3.8):

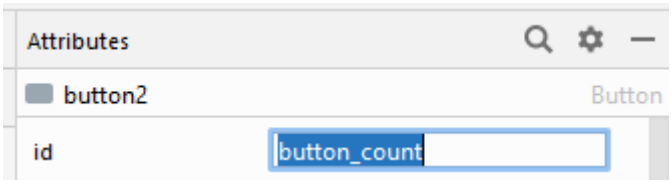


Рисунок 3.8 – Настройка атрибутов компонента

17. Преобразуйте текст в строковые переменные (сочетание клавиш Alt+Enter).

18. Доработайте код, самостоятельно опишите переменную **buttonCounter** (рис. 3.9).

```
package com.example.count

import ...

class MainActivity : AppCompatActivity() {

    private var counter: Int = 0
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)


        var [redacted]
        var [redacted]
        button.setOnClickListener { it: View!
            textView.text = "Hello, friend!"
        }
        var [redacted]
        buttonCounter.setOnClickListener { it: View!
            textView.text = "Я нажал на кнопку ${++counter} раз"
        }
    }
}
```

Рисунок 3.9 – Код программы

Исправление ошибок в коде

В редакторе кода в верхнем правом углу есть прямоугольник. Он может быть в виде зелёной галочки (идеальный код), жёлтым (критических ошибок нет, есть предупреждения, но их лучше исправить) и красным (ошибка в коде, программа не запустится).

Если прямоугольник желтого цвета, но строки, где можно исправить предупреждения выделяются желтым цветом. Android Studio предлагает внести исправления.

19. Доработайте код с помощью подсказок Android Studio таким образом, чтобы в редакторе кода отобразилась зеленая галочка.

20. Запустите проект, проверьте работу кнопки (рис. 3.10).

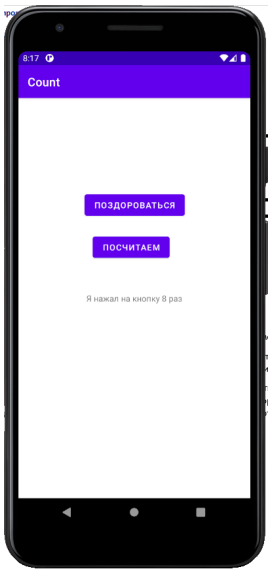


Рисунок 3.10 – Запуск приложения в эмуляторе

Задание для самостоятельной работы: разместите в приложении компонент Image, загрузите в него изображение и обработайте для него событие – подсчет количества нажатий по нему. Измените цвет фона, поменяйте параметры текста (например, шрифт, размер, цвет).

Лабораторная работа №4.
Работа с цветом

Цель: изучить особенности использования цветов при разработке приложений в среде Android Studio.

Ход работы

1. Запустите Android Studio.
2. Создайте новый проект (рис. 4.1):

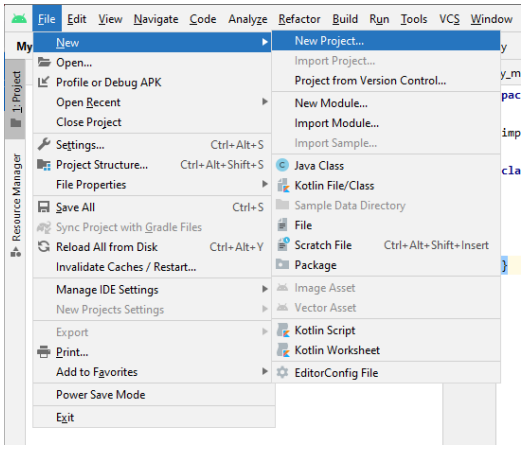


Рисунок 4.1 – Главное меню

3. Выберите тип Activity – **Empty** (рис. 4.2).

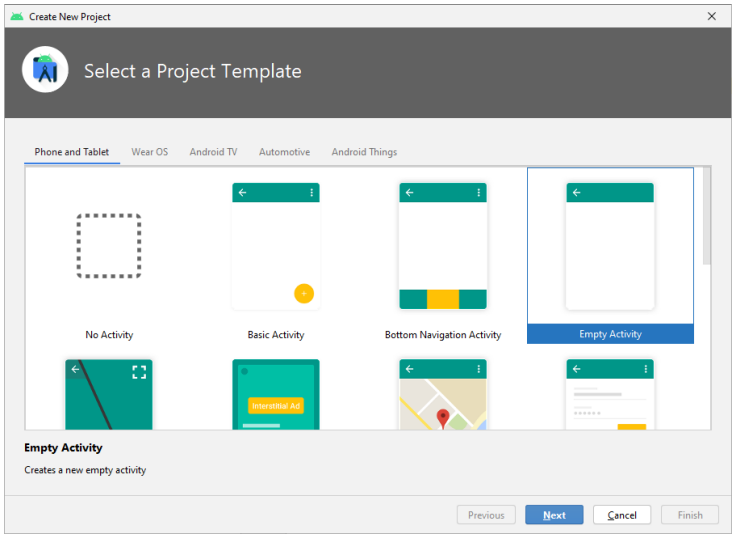


Рисунок 4.2 – Выбор типа Activity

4. Нажмите кнопку «Next».

5. Укажите необходимые параметры для проекта.

6. Нажмите кнопку «Finish».

7. Дождитесь окончания процессе сборки проекта, процесс может занять несколько минут.

8. Проверьте необходимость обновления скриптов сборки приложения Gradle. Для этого выберите в списке элементов проекта узел Gradle Scripts, откройте двойным щелчком мыши файл build.gradle, который относится к модулю (рис. 4.3).

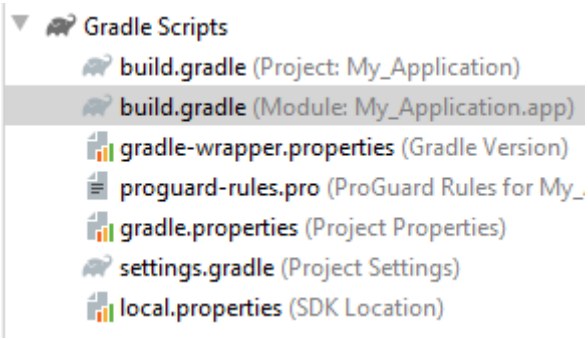


Рисунок 4.3 – Скрипты сборки

9. Если библиотеку необходимо обновить, строка в модуле будет подсвечена.

10. При наведении на подсвеченную строку курсом мыши, Android Studio предложит обновить библиотеку.

11. Запустите проект и убедитесь, что обновления и сборка проекта закончились успешно. Для этого нажмите на кнопку «Run» (зеленый треугольник на панели инструментов). На забудьте выбрать эмулятор перед запуском (рис. 4.4).

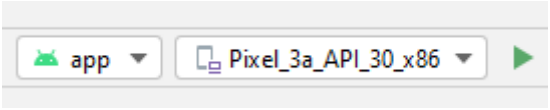


Рисунок 4.4 – Выбор эмулятора

12. Дождитесь на эмуляторе загрузки операционной системы Android. Убедитесь, что эмулятор работает.

13. Закройте эмулятор.

14. Перейдите на вкладку **activity_mail.xml** и убедитесь, что проект находится в режиме Design (в правом верхнем углу).

15. Из палитры компонентов переместите три компонента **Button** на **Activity** в любом порядке разместите их на форме (рис. 4.5):

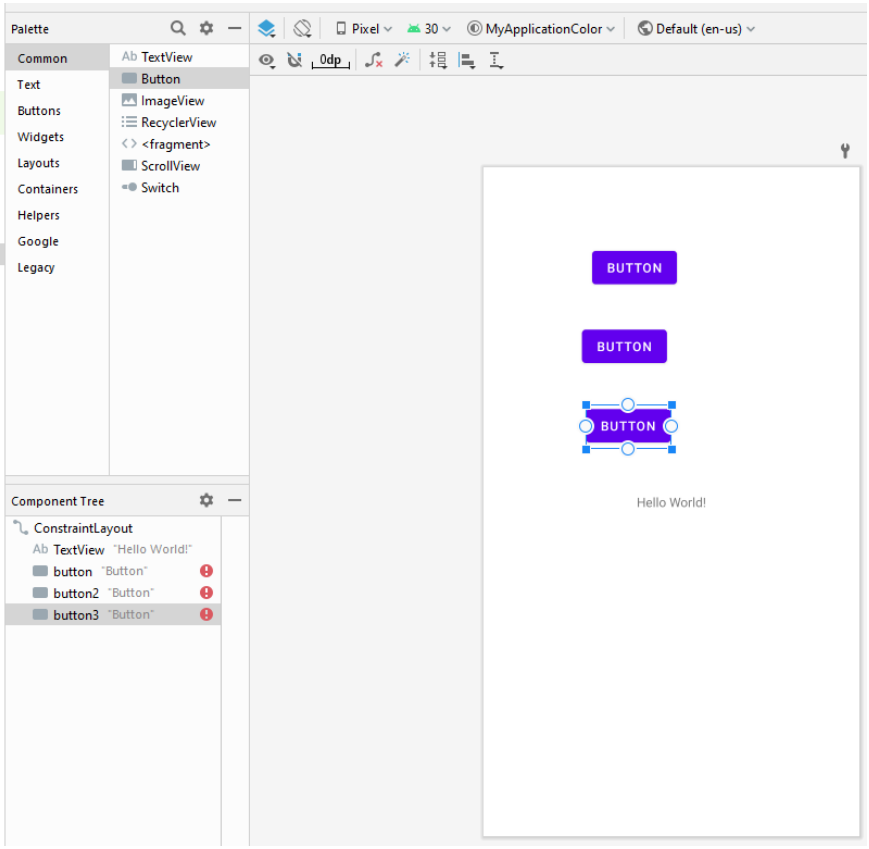


Рисунок 4.5 – Разработка проекта

16. Выделите все три кнопки в окне **Component Tree** с помощью клавиши **Shift** (рис. 4.6).

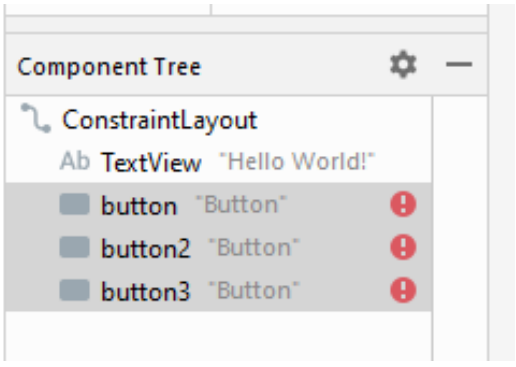


Рисунок 4.6 – Дерево компонентов

17. Используя контекстное меню выровняйте кнопки по центру (**Central**) сначала по горизонтали (**Horizontally**), потом по вертикали (**Vertically**). Проверьте выравнивание кнопок на Activity (рис. 4.7).

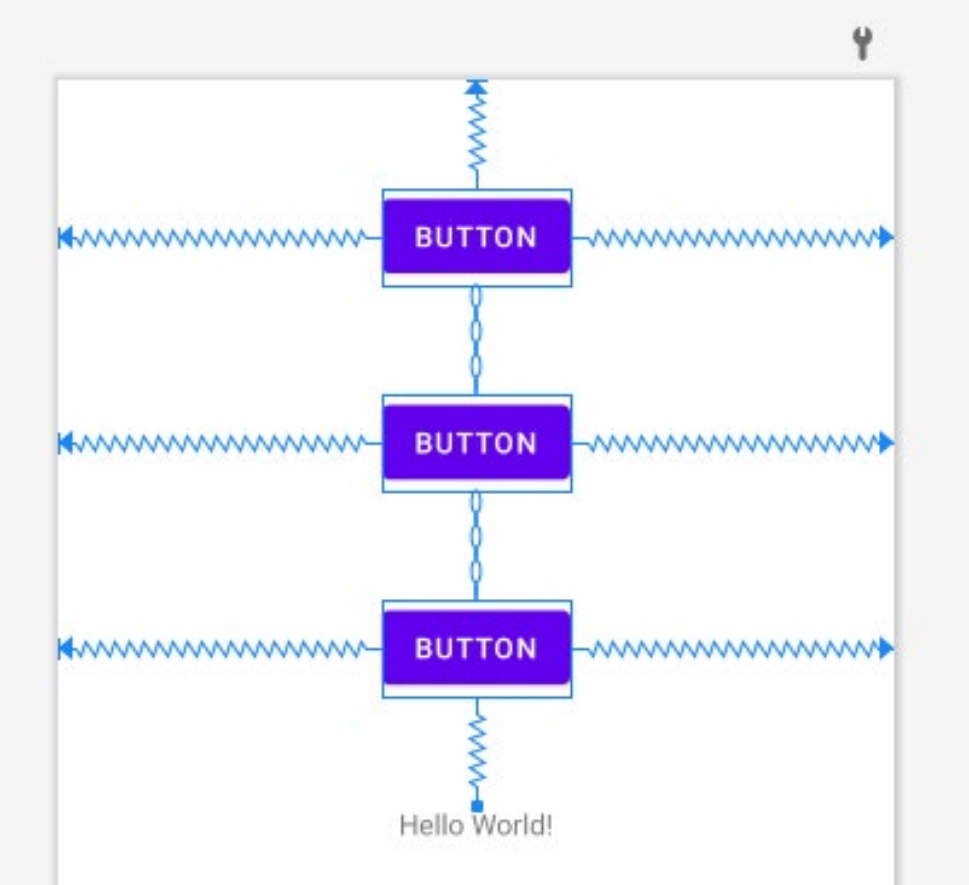


Рисунок 4.7 – Настройка компонентов

18. Изменить положение кнопок можно и с помощью контекстного меню напрямую у кнопок на экране. Начертите указателем мыши ограничивающий прямоугольник вокруг всех кнопок и вызовите контекстное меню щелчком правой кнопки мыши.

19. Для всех трех кнопок измените параметр **id** в окне **Attributes** со стандартных идентификаторов на имена, несущие смысл функции этой кнопки. Например, укажите цвет, в которые нажатие по кнопке будет окрашивать фон. Также измените параметр text (представление кнопки на экране для пользователя) (рис. 4.8).

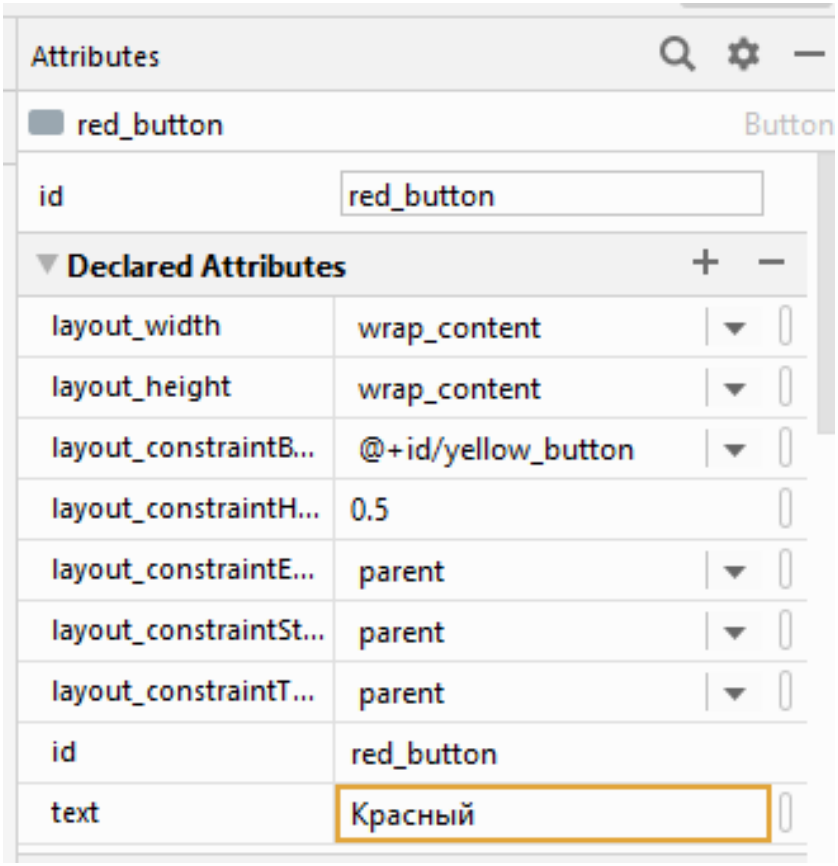


Рисунок 4.8 – Настройка атрибутов компоненты

20. Переключитесь в режим **Code** и переведите все названия кнопок в строковые ресурсы. Для этого встаньте на параметр курсором и нажмите сочетание **Alt+Enter** (рис. 4.9):

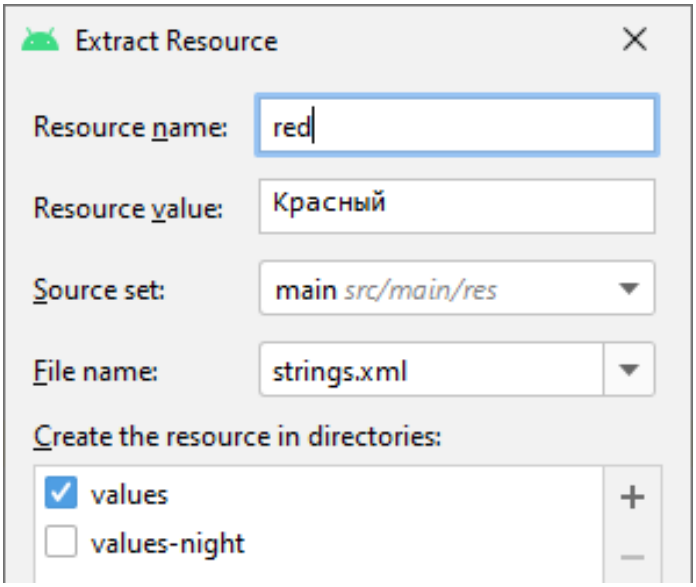


Рисунок 4.9 – Настройка ресурсов

21. Вернитесь в режим **Design**, разместите компонент **TextView** под кнопками и повторите операцию выравнивания любым из указанных ранее способов.

22. Данный компонент будет отображать текущий цвет фона. Так как ранее уже были созданы строковые ресурсы с названиями цветов для кнопок, их можно использовать повторно для текстового компонента. Для этого в режиме **Design** выберите компонент **TextView** и в окне **Attributes** напротив атрибута **text** нажмите кнопку справа. В открывшемся окне ресурсов выберите одну из строк – это будет цвет по умолчанию (рис. 4.10).

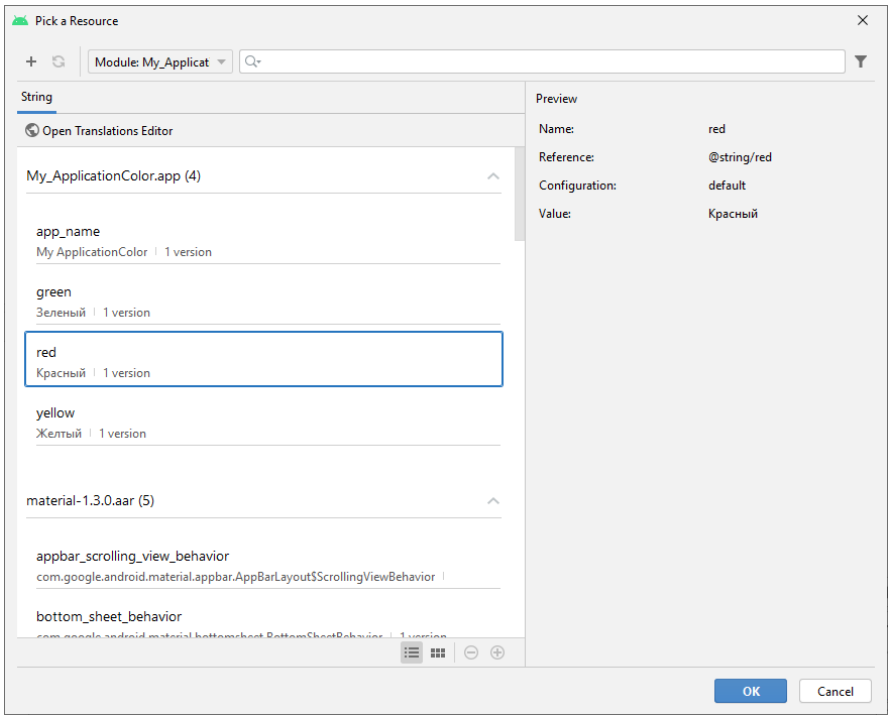


Рисунок 4.10 – Выбор ресурса

23. Создадим палитру цветов для фона. Ресурсы для цветов принято хранить в отдельном файле **res/values/colors.xml** (при этом их можно хранить и в файле **strings.xml**).

24. Откройте файл. В тэге recourses впишите три ресурса необходимых цветов. Например, redColor, yellowColor и greenColor. Напротив строк слева появятся квадраты с примерами цветов. Можно изменить цвет, нажав на квадрат (рис. 4.11).

Рисунок 4.11 – Файл colors.xml

25. Измените фон по умолчанию. Для этого для контейнера **ConstraintLayout** в окне **Attributes** измените свойство **background**. Нажмите кнопку справа, откроется диалоговое окно. В окне укажите раздел **Color** и созданный ресурс **redColor** (рис. 4.12).

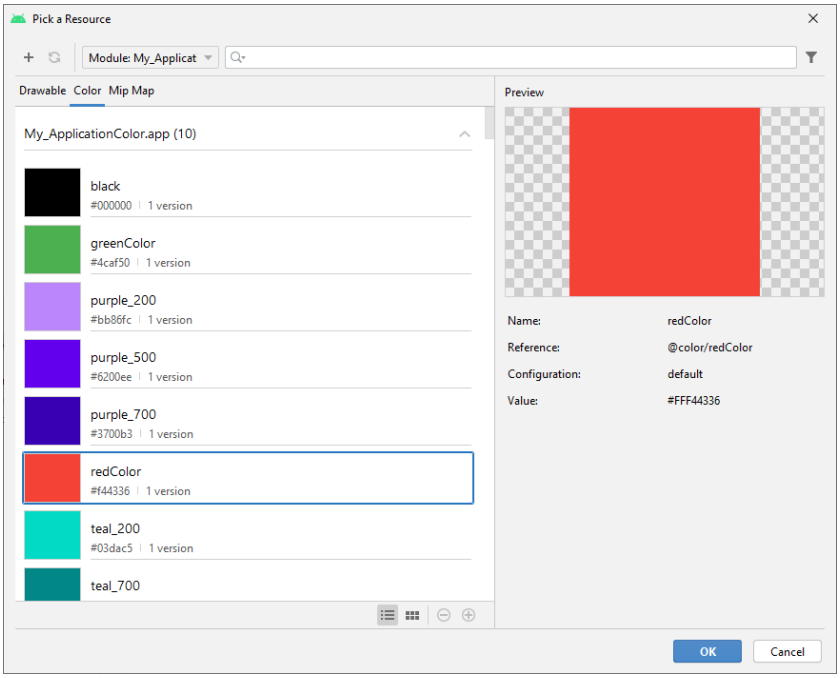


Рисунок 4.12 – Выбор ресурса

26. Измените параметр **id** для **ConstraintLayout** в окне **Attributes** на **root_layout**.

27. Запустите проект, убедитесь, что цвет фона изменился, имеются три кнопки и текстовая строка (рис. 4.13).

Рисунок 4.13 – Запуск приложения в эмуляторе

28. Доработайте код, чтобы по нажатию на кнопку изменялся цвет фона. Для этого измените код (рис. 4.14):

```
class MainActivity : AppCompatActivity() {
    @RequiresApi(Build.VERSION_CODES.M)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val rButton: Button = findViewById(R.id.red_button)
        val yButton:Button = findViewById(R.id.yellow_button)
        val gButton:Button = findViewById(R.id.green_button)
        val textView:TextView = findViewById(R.id.textView)
        val rLayout:ConstraintLayout = findViewById(R.id.root_layout)

        rButton.setOnClickListener { it: View!
            textView.text = "Красный"
            rLayout.setBackgroundColor(resources.getColor(R.color.redColor, theme: null))
        }
```

Рисунок 4.14 – Код программы

29. Самостоятельно создайте две операции для нажатия на оставшиеся две кнопки.

30. Проверьте работу проекта.

31. Код можно доработать, получая название текстовой строки из строковых ресурсов (рис. 4.15):

```
rButton.setOnClickListener { it: View!
    textView.setText(resources.getText(R.string.red))
    rLayout.setBackgroundColor(resources.getColor(R.color.redColor,  theme: null))
}
```

Рисунок 4.15 – Код программы

32. Измените иконку приложения на экране меню. По умолчанию Android Studio для всех приложений использует стандартные пиктограммы – зеленый робот. Хранятся они в папке **res/mipmap**. Папка является виртуальной, реально изображения размещаются в папках **res/mipmap-hdpi**, **res/mipmap-mdpi**, **res/mipmap-xhdpi**, **res/mipmap-xxhdpi**. В каждой из этих папок есть файл с одинаковым именем **ic_launcher.png**. Для поздних версий операционной системы Andriod добавлен файл ic_launcher_round.png с более высоки разрешением. В зависимости от разрешения экрана на устройстве система выбирает наиболее подходящий по размеру изображение и выводит его в качестве значка в заголовке приложения и на домашнем экране (рис. 4.16):

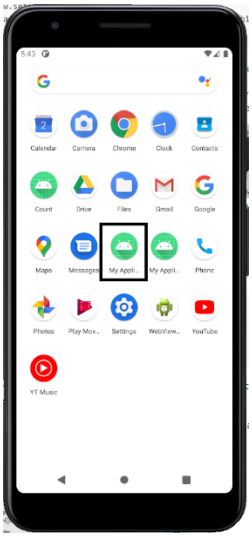


Рисунок 4.16 – Домашний экран в эмуляторе

33. Для изменения изображений можно подготовить весь набор необходимых файлов с различным разрешением или изменить настройки проект в манифесте. За это отвечает параметр **ic** (рис. 4.17):

activity_main.xml | AndroidManifest.xml | colors.xml | MainActivity.kt | ic_launcher.png

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

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

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

</manifest>
```

Рисунок 4.17 – Код программы

34. В Android Studio имеется возможность работы с набором предопределённых значков и генератор собственных значков. Вызовите контекстной меню в папке **drawable** или **mipmap** и выберите пункт меню **New**, затем **Image Asset**. В диалоговом можно указать источник иконки – файл на компьютере, вариант из клипарта или набор символов. Можно задать форму значка, цвет фона и другие параметры (рис. 4.18).

Кроме значков для различных разрешений, генератор создаст дополнительный файл с суффиксом **-web**, который будет скопирован в папку **main**. Этот файл используется для Play Market, приложение размещается в магазине приложений.

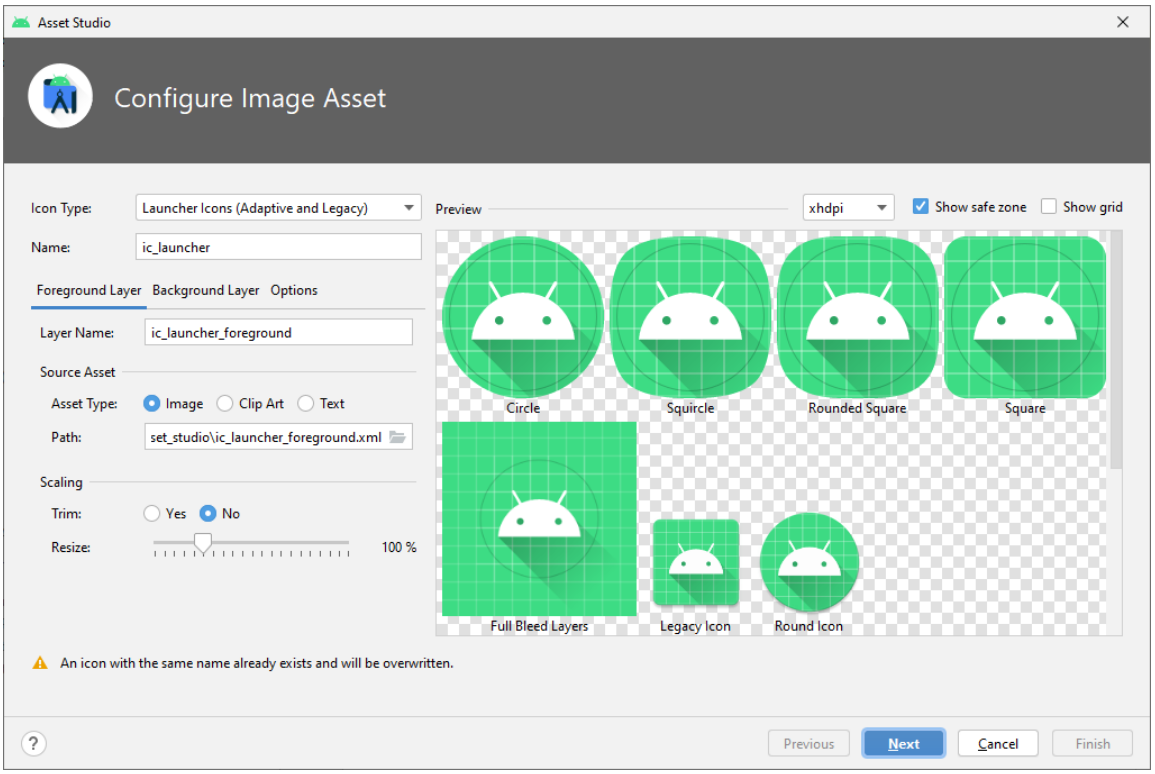


Рисунок 4.18 – Окно Image Asset

35. Измените иконки проекта. Убедитесь, что на экране запуска приложения изображение поменялось.

36. Сохраните проект.

Задание для самостоятельной работы: создайте приложение, изменяющее по нажатию на соответствующие кнопки размер шрифта надписи, размещенной на экране. Для приложение создайте иконку, настройте ее и убедитесь, что икона приложения изменилась на домашнем экране эмулятора.

Лабораторная работа №5.
Работа с Activity

Цель: изучить особенности работы с Activity в среде разработки Android Studio.

Ход работы

1. Запустите Android Studio.
2. Создайте новый проект, в котором по нажатию кнопки будет открываться окно со случайным числом (рис. 5.1):

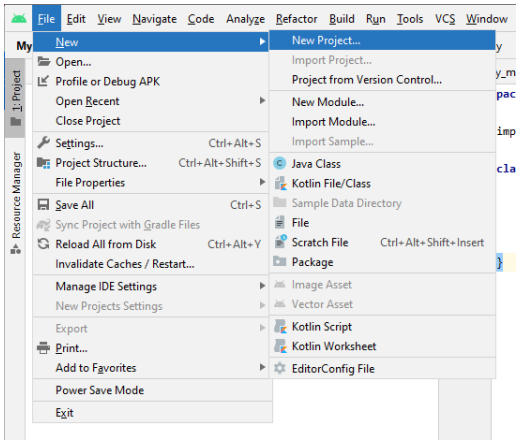


Рисунок 5.1 – Главное меню

3. Выберите тип Activity – **Empty** (рис. 5.2).

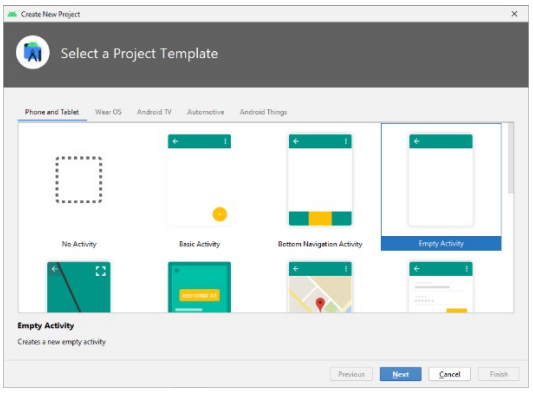


Рисунок 5.2 – Выбор типа Activity

4. Нажмите кнопку «Next».
5. Укажите необходимые параметры для проекта.

6. Нажмите кнопку «Finish».

7. Дождитесь окончания процессе сборки проекта, процесс может занять несколько минут.

8. Проверьте необходимость обновления скриптов сборки приложения Gradle. Если библиотеку необходимо обновить, строка в модуле будет подсвечена. При наведении на подсвеченную строку курсом мыши, Android Studio предложит обновить библиотеку.

9. Запустите проект и убедитесь, что обновления и сборка проекта закончились успешно. Для этого нажмите на кнопку «Run» (зеленый треугольник на панели инструментов). Не забудьте выбрать эмулятор перед запуском.

10. Дождитесь на эмуляторе загрузки операционной системы Android. Убедитесь, что эмулятор работает.

11. Закройте эмулятор.

12. Перейдите на вкладку **activity_mail.xml** и убедитесь, что проект находится в режиме Design (в правом верхнем углу).

13. Из палитры компонентов переместите компонент **Button** на **Activity.**

14. Выделите все кнопку и текстовую строку в окне **Component Tree** с помощью клавиши **Shift**.

15. Используя контекстное меню выровняйте компоненты по своему усмотрению. Проверьте выравнивание кнопок на Activity. Например как на рис. 5.3.

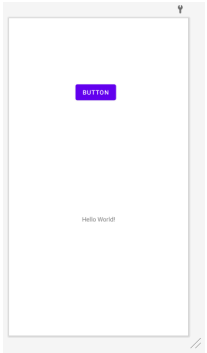


Рисунок 5.3 – Пример разработки приложения

16. Для всех компонентов измените параметр **id** в окне **Attributes** со стандартных идентификаторов на имена, несущие смысл. Например, **randomButton** для кнопки и **textView** для текстовой строки.

17. Измените параметр **text** для кнопки на **Random** и для текстовой строки.

18. Переключитесь в режим **Code** и переведите все названия в строковые ресурсы. Для этого встаньте на параметр курсором и нажмите сочетание **Alt+Enter**.

19. Добавьте в проект еще одну **Activity**. Для этого в дереве проекта в узле **app** – **java** – **com.example[имя проекта]** в контекстном меню выберите **New – Activity – Empty Activity** (рис. 5.4).

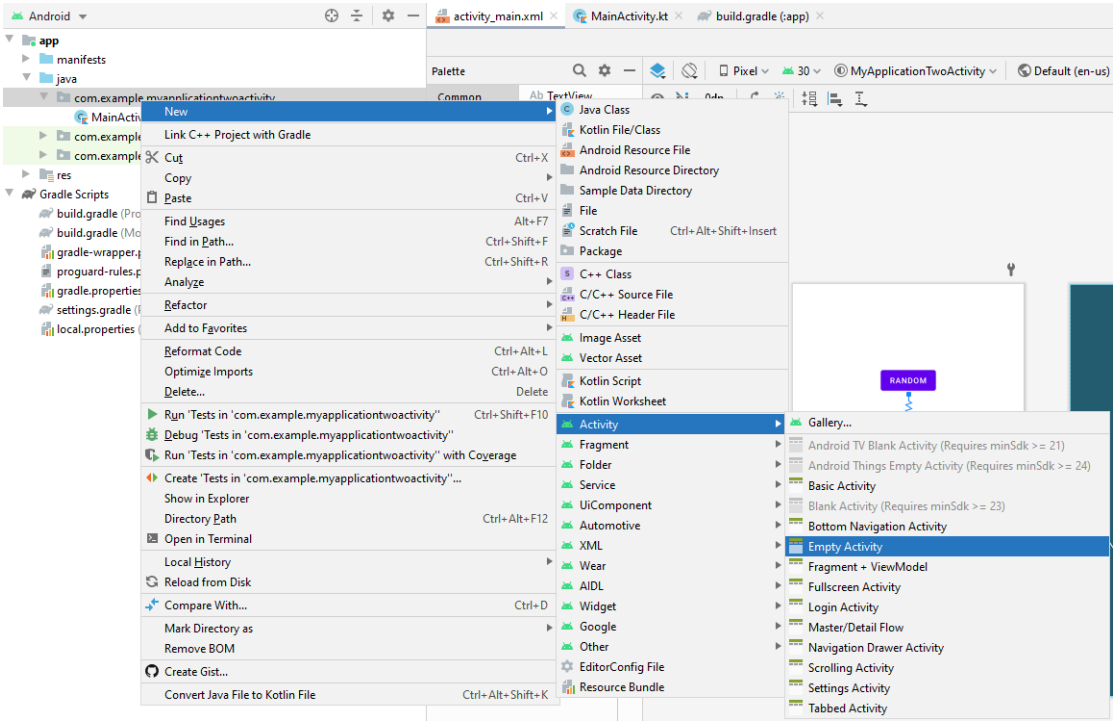


Рисунок 5.4 – Добавление Activity

20. Проверьте имя Activity (его можно изменить при необходимости), имя пакета – оно должно совпадать с ранее указанным, обратите внимание на выбранный язык программирования – Kotlin (рис. 5.5):

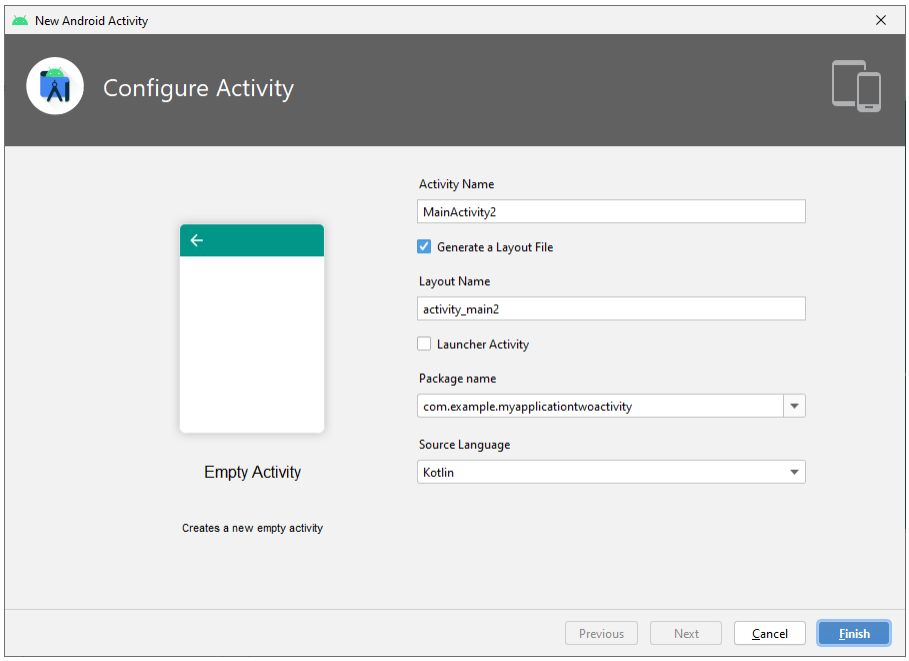


Рисунок 5.5 – Конфигурация Activity

21. Дождитесь завершения процесса сборки проекта.

22. Откройте файл манифеста проекта и убедитесь, что в коде присутствует описание второй Activity (рис. 5.6):

activity_main.xml | activity_main2.xml | MainActivity2.kt | AndroidManifest.xml

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My ApplicationTwoActivity"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplicationTwoActivity">
        <activity android:name=".MainActivity2"></activity>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Рисунок 5.6 – Код программы

23. Обратите внимание, что для второй **Activity** создались собственные файлы **MainActivity2.kt** и **activity_main2.xml**.

24. Оформите внешний вид второго экрана для отображения случайного числа. Расположите заголовок – компонент **TextView**, измените его параметр **text** на заголовок «**Это случайное число между 0 и %1d**», параметр **id** на **textView_label**, измените по своему усмотрению цвет и размер шрифта.

25. Перейдите в режим Code и поместите text в строковые ресурсы. Для этого встаньте на параметр курсором и нажмите сочетание **Alt+Enter**. Например как на рис 5.7.

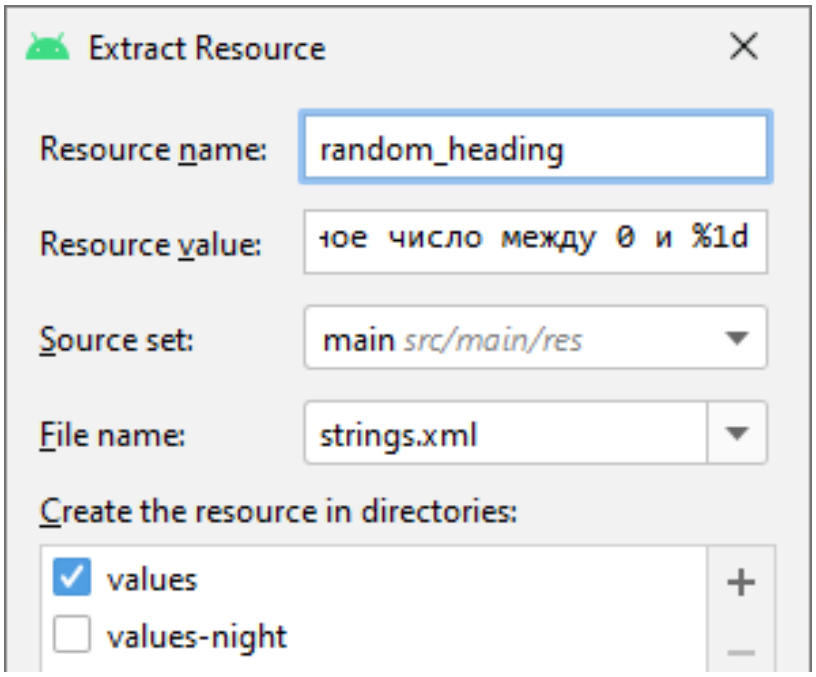


Рисунок 5.7 – Настройки ресурсов

26. Добавьте на экран второй компонент **TextView** для отображения случайного числа под заголовком.

27. Измените его параметры: **id** – **textView_random**, **text** – **R**, по своему усмотрению измените цвет, размер и начертание компонента (рис. 5.8).

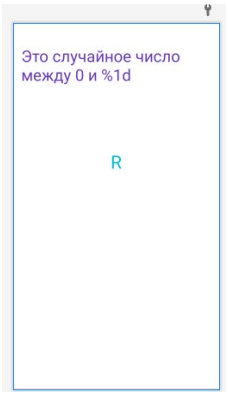


Рисунок 5.8 – Внешний вид приложения

28. Измените цвет фона для второго **Activity**, создав новый ресурс в файле **colors.xml** (как в предыдущей лабораторной работе).

29. Доработайте код, чтобы по нажатию на кнопку на первом экране открывалась вторая **Activity**.

Для обмена информацией между **Activity** используется объект **Intent**. Он абстрактно представляет собой намерение выполнить какое-либо действие. Как только **Intent** отправляется, его получает операционная система Android, и считывает информацию в нем. Для открытия второго экрана необходимо создать и отправить объект **Intent** с указанием **Activity**, которое нужно открыть. Затем вызвать метод **startActivity()** с передачей объекта **Intent** (рис. 5.9).

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

    }

        fun randomMe(view:View) {
            val randomIntent = Intent( packageContext: this, MainActivity2::class.java)
            startActivity(randomIntent)

        }
}
```

Рисунок 5.9 – Код программы

30. Для связывания операции и кнопки в файле **activity_main.xml** для кнопки необходимо указать соответствующую операцию в параметре **android:onClick** (рис. 5.10):

```
<Button
    android:id="@+id/randomButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="147dp"
    android:text="Random"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:onClick="randomMe"/>
```

Рисунок 5.10 – Фрагмент кода

31. Проверьте работу проекта.

32. Доработайте код для второго Activity таким образом, чтобы случайно задавался конец диапазона и из диапазона формировалось случайное число (рис. 5.11):

```
class MainActivity2 : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main2)
        showRandomNumber ()
    }
    fun showRandomNumber () {
        val random : Int  = Random.nextInt( from: 0, until: 1000)
        var randomInt : Int  = Random.nextInt( from: 0,random)
        val textViewRandom: TextView = findViewById(R.id.textView_random)
        val textViewLabel:TextView = findViewById(R.id.textView_label)

        textViewRandom.text = Integer.toString(randomInt)
        textViewLabel.text = getString(R.string.random_heading, random)

    }

}
```

Рисунок 5.11 – Код программы

33. Проверьте работу проекта (рис. 5.12).

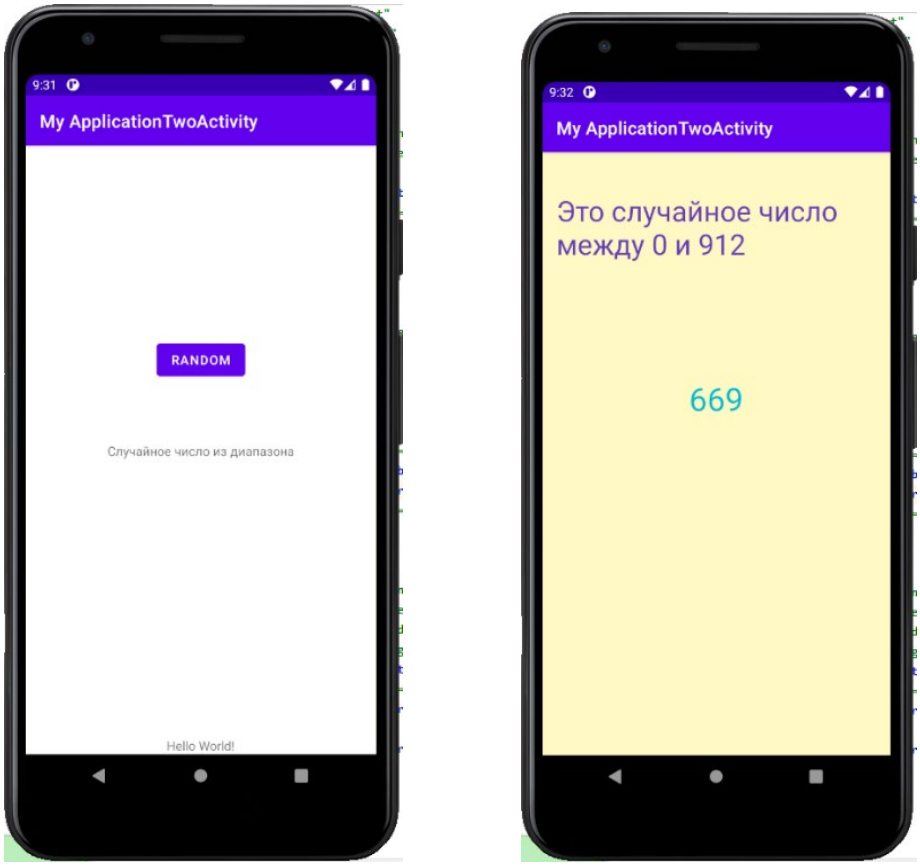


Рисунок 5.12 – Запущенное на эмуляторе приложение

34. Сохраните проект.

Задание для самостоятельной работы: разработайте приложение, угадывающее загаданное пользователем число из указанного диапазона на одной Activity и предлагающее пользователю угадать случайно число из диапазона на второй Activity.

Лабораторная работа №6.
Работа с Activity. Продолжение

Цель: изучить особенности работы с Activity в среде разработки Android Studio.

Ход работы

1. Запустите Android Studio.
2. Создайте новый проект для работы с блокнотом.
3. Выберите тип Activity – **Empty**.
4. Нажмите кнопку «Next».
5. Укажите необходимые параметры для проекта.
6. Нажмите кнопку «Finish».
7. Дождитесь окончания процессе сборки проекта, процесс может занять несколько минут.
8. Проверьте необходимость обновления скриптов сборки приложения Gradle. Для этого выберите в списке элементов проекта узел Gradle Scripts, откройте двойным щелчком мыши файл build.gradle, который относится к модулю. Если библиотеку необходимо обновить, строка в модуле будет подсвечена. При наведении на подсвеченную строку курсом мыши, Android Studio предложит обновить библиотеку.
9. Запустите проект и убедитесь, что обновления и сборка проекта закончились успешно. Для этого нажмите на кнопку «Run» (зеленый треугольник на панели инструментов). На забудьте выбрать эмулятор перед запуском.
10. Дождитесь на эмуляторе загрузки операционной системы Android. Убедитесь, что эмулятор работает.
11. Закройте эмулятор.

12. Для работы в проекте необходимо создать три **Activity** с именами **Activity1, Activity2, Activity3**. Переименуйте первую **Activity.** Для этого вызовите контекстное меню в дереве проекта по имени, выберите пункт **Refactor** – **Rename** (рис. 6.1).

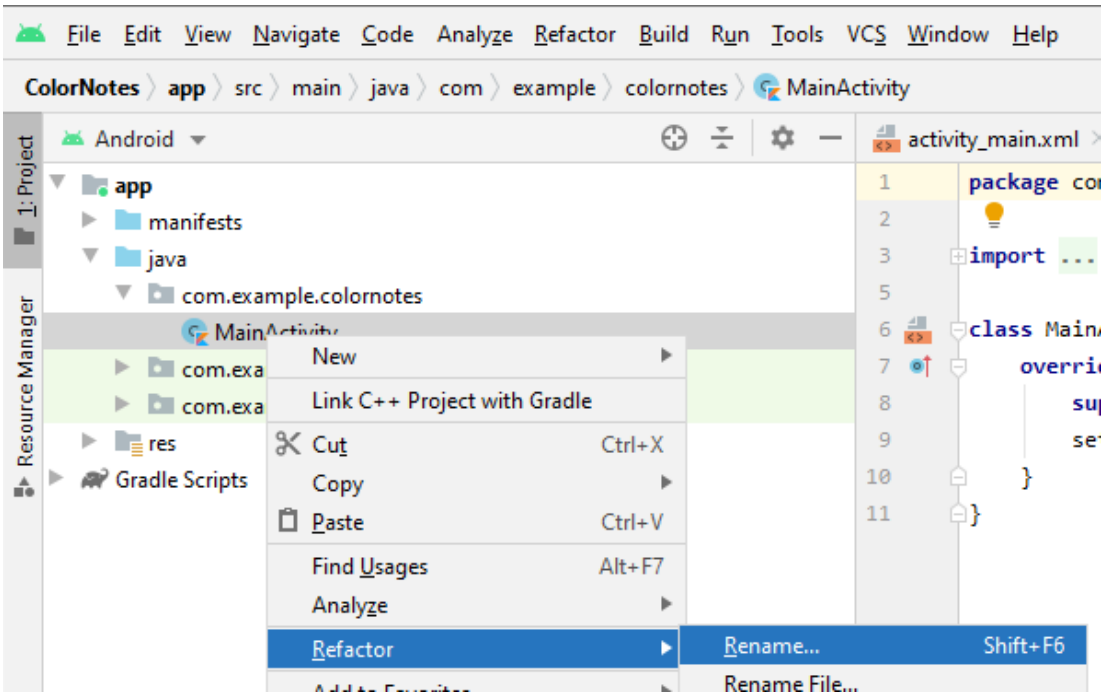


Рисунок 6.1 – Контекстное меню

13. Таким же образом переименуйте **activity_mail.xml** на **activity1.xml**

14. Отредактируйте первую Activity1. Измените у нее цвет фона любым способом: в режиме **Design** или прописав код в режиме **Code**.

15. Расположите на ней компонент **Plain Text** на всю область, внизу расположите компонент **Button**. Компонент **Text View** можно удалить.

16. Измените **id** компонента **Plain Text** на **text**, **id Button** на **next**.

17. Введите свойство **hint** (подсказка) для компонента **Plain Text**, например, «Напишите здесь заметку».

18. Введите свойство **text** (заголовок) для компонента **Button**, например, «Другая заметка».

19. Добавьте для компонента **Plain Text** в режиме **Code** параметр android:background="@null". Данные параметр убирает подчеркивание текста.

20. Переключитесь в режим **Code** и переведите все названия в строковые ресурсы. Для этого встаньте на параметр курсором и нажмите сочетание **Alt+Enter** (рис. 6.2).

```
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/purple_200"
    tools:context=".Activity1">

    <EditText
        android:id="@+id/text"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:background="@null"
        android:ems="10"
        android:hint="@string/yournote"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:autofillHints="@string/notes" />

    <Button
        android:id="@+id/next"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="110dp"
        android:layout_marginLeft="110dp"
        android:layout_marginBottom="46dp"
        android:text="@string/nextTip"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 6.2 – Код программы

21. Исправьте все ошибки, запустите приложение (рис. 6.3).

Рисунок 6.3 – Запущенное на эмуляторе приложение

22. Добавьте в проект еще одну **Activity**. Для этого в дереве проекта в узле **app** – **java** – **com.example[имя проекта]** в контекстном меню выберите **New – Activity – Empty Activity** (рис. 6.4).

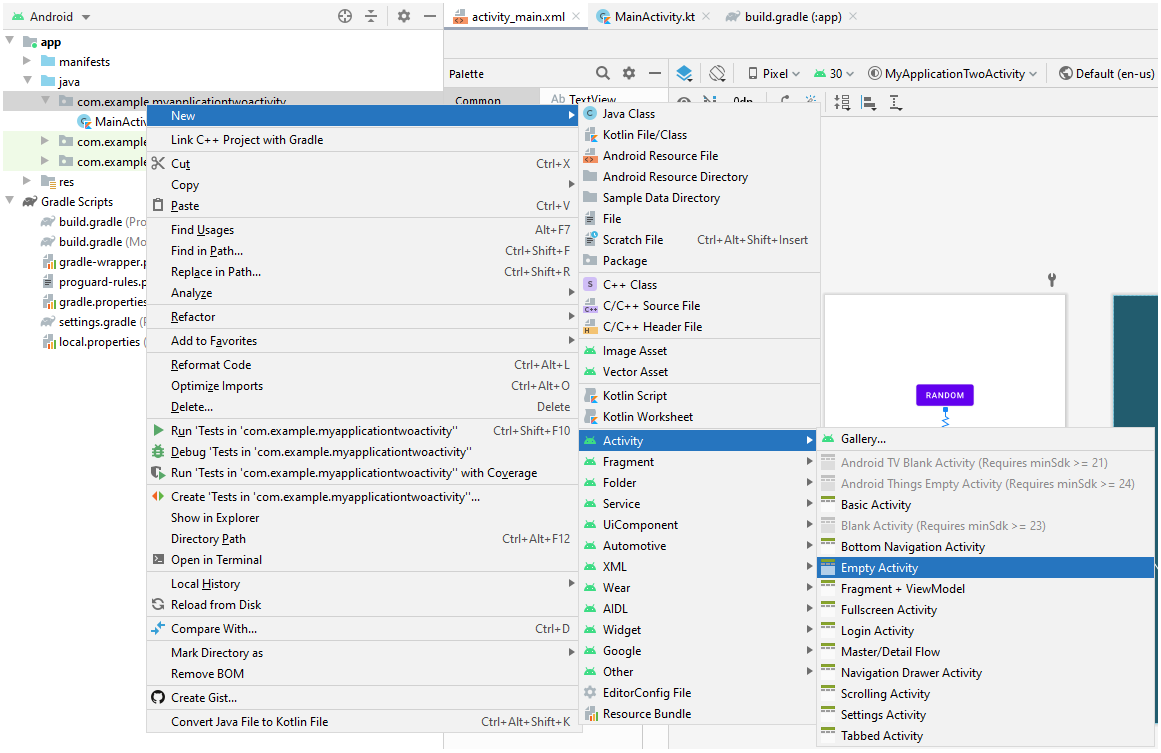


Рисунок 6.4 – Добавление Activity

23. Проверьте имя Activity – Activity2, имя Layout – activity2. Имя пакета должно совпадать с ранее указанным, обратите внимание на выбранный язык программирования – Kotlin (рис. 6.5):

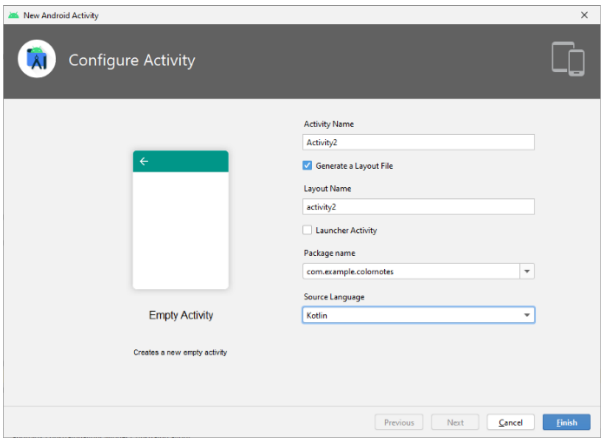


Рисунок 6.5 – Конфигурация Activity

24. Добавьте еще одну Activity с именем Activity3, имя Layout – activity3.

25. Изучите содержание манифеста (рис. 6.6):

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ColorNotes"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ColorNotes">
        <activity android:name=".Activity3"></activity>
        <activity android:name=".Activity2" />
        <activity android:name=".Activity1">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Рисунок 6.6 – Код программы

26. **Activity2** и **Activity3** должны иметь такие же компоненты как и **Activity1**, отличие должно быть в цвете фона. Можно изменить настройки вручную, однако проще скопировать xml-файл из **Activity1**.

27. Добавьте в код каждого **Layout** имя обработки события **onClick**: android:onClick="nextSheet".

28. Создайте в файле **Activity1.kt** обработчик события для кнопки. Здесь запуск второй Activity осуществляется при помощи intent («намерений») (рис. 6.7).

```
class Activity1 : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity1)
        val next : Button = findViewById(R.id.next)
        next.setOnClickListener(){ it: View!
            val intent = Intent( packageContext: this, Activity2::class.java)
            startActivity(intent)
        }
    }
}
```

Рисунок 6.7 – Код программы

29. По аналогии создайте в файле **Activity2.kt** обработчик события для кнопки, которая вызывает **Activity3**.

30. Для **Activity3** вызывать **Activity1** не стоит, так как все вызовы будут храниться в памяти. Если несколько раз нажимать кнопки перехода, то это приведет к нехватке ресурсов, даже несмотря на то, что операционная система Android будет их удалять. В связи с этим, код для **Activity3** будет иным (рис. 6.8):

```
class Activity3 : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity3)
        val next : Button = findViewById(R.id.next)
        next.setOnClickListener(){ it: View!
            Toast.makeText(applicationContext, text: "Это последняя страница", LENGTH_SHORT).show()
        }
    }
}
```

Рисунок 6.8 – Код программы

31. Переведите подсказку в строковые ресурсы. Для этого встаньте на параметр курсором и нажмите сочетание **Alt+Enter.**

32. Проверьте работу приложения: передвигайтесь между Activity кнопкой «Другая заметка» и обратно кнопкой «Назад» на эмуляторе. Введите текст в заметки. Запустите приложение повторно, убедитесь, что заметки при перелистывании сохраняются, а при перезагрузке не сохраняются.

33. Для обработки сохранения заметок в приложении, необходимо вспомнить жизненный цикл Activity. При создании вызывается процедура onCreate(), когда активность выходит на первый план – onResume(), когда приостанавливается – onPause(). С помощью этих процедур можно сохранить данные о настройках и восстанавливать их при необходимости.

34. Отредактируйте код для Activity1 (рис. 6.9):

```
class Activity1 : AppCompatActivity() {

    val text: EditText = findViewById(R.id.text)

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity1)
        val next : Button = findViewById(R.id.next)
        next.setOnClickListener(){ it: View!
            val intent = Intent( packageContext: this, Activity2::class.java)
            startActivity(intent)
        }
    }
    override fun onPause() {
        super.onPause()
        val prefs : SharedPreferences.Editor! = getPreferences(Context.MODE_PRIVATE).edit()

        prefs.putString("note1",text.editableText.toString())
        prefs.apply()
    }

    override fun onResume() {
        super.onResume()
        val saveState : String? =
            getPreferences(Context.MODE_PRIVATE).getString( key: "note1", defValue: null)
        if (saveState !=null){
            text.setText(saveState)
        }
    }

}
```

Рисунок 6.9 – Код программы

35. В этом коде в первой функции открывается редактор изменений, записываются данные из текстовой области и затем внесенные изменения подтверждаются. Во второй функции происходит чтение данных по «ключу». Осуществляется проверка наличия сохраненной ранее заметки и при ее наличии данные восстанавливаются.

36. Если далее действовать по аналогии, то для Activity2 и Activity3 код будет практически повторяться, что не является оптимальным с точки зрения программирования. Оптимизируем код для Activity1. Все действия можно делать из одной Activity, при этом менять цвет фона и ведя нумерацию листов для заметок. Для этого создадим два массива: цветов фона и номеров листов. Цвета задаются в формате **0x…..** Их можно скопировать из ранее выбранных фонов (рис. 6.10).

```
class Activity1 : AppCompatActivity() {
    var colors = arrayOf(0xC5CAE9, 0xB2EBF2, 0xFFF9C4)
    var sheetNumber = 0
    val text: EditText = findViewById(R.id.text)

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity1)
        sheetNumber = getIntent().getIntExtra( name: "sheetNumber", defaultValue: 0)
        val next : Button = findViewById(R.id.next)
        next.setOnClickListener(){ it: View!
            if (sheetNumber < colors.size - 1){
                val intent = Intent( packageContext: this, this::class.java)
                intent.putExtra( name: "sheetNumber",  value: sheetNumber + 1)
                startActivity(intent)
            } else {
                Toast.makeText(applicationContext, getString(R.string.text3), Toast.LENGTH_SHORT).show()
            }
        }
    }
```

Рисунок 6.10 – Код программы

37. Для корректной работы функции onResume() необходимо задать **id** для **Layout** в xml-файле: android:id = "@+id/sheet" (рис. 6.11).

Рисунок 6.11 – Код программы

38. Удалите файлы (kt и xml) для двух других Activity.

39. Измените код файла манифеста проекта, убрав из него записи об удаленных Activity.

40. Измените пиктограмму проекта (можно воспользоваться файлом в папке Lab7 на портале).

41. Добавьте еще несколько листов для заметок, для этого можно просто увеличить количество элементов в массиве **colors**.

42. Проверьте работу проекта.

43. Сохраните проект.

44. * Помимо сохранения заметок, их можно удалять, используя у редактора настроек функцию **clear()** – она очищает все **SharedPreferences**. Также можно использовать функцию remove(), которая принимает «ключ» и очищает данные по этому «ключу». Кроме того, сохранять большие объемы информации в **SharedPreferences** не стоит, лучше в этом случае использовать файлы и/или базы данных.

Задание для самостоятельной работы: доработайте приложение «Блокнот» с возможностью изменения размера шрифта на странице заметки.

Лабораторная работа №7.
Работа с Menu

Цель: изучить особенности разработки приложений с компонентом Menu.

Ход работы

1. Запустите Android Studio.
2. Создайте новый проект для демонстрации работы с меню:
3. Выберите тип Activity – **Empty**.
4. Нажмите кнопку «Next».
5. Укажите необходимые параметры для проекта.
6. Нажмите кнопку «Finish».
7. Дождитесь окончания процессе сборки проекта, процесс может занять несколько минут.
8. Проверьте необходимость обновления скриптов сборки приложения Gradle. Для этого выберите в списке элементов проекта узел Gradle Scripts, откройте двойным щелчком мыши файл build.gradle, который относится к модулю. Если библиотеку необходимо обновить, строка в модуле будет подсвечена. При наведении на подсвеченную строку курсом мыши, Android Studio предложит обновить библиотеку.
9. Запустите проект и убедитесь, что обновления и сборка проекта закончились успешно. Для этого нажмите на кнопку «Run» (зеленый треугольник на панели инструментов). На забудьте выбрать эмулятор перед запуском.
10. Дождитесь на эмуляторе загрузки операционной системы Android. Убедитесь, что эмулятор работает.
11. Закройте эмулятор.

12. Создайте в файле **res/values/strings.xml** текстовые строки будущего меню (рис. 7.1):

```
Edit translations for all locales in the translations editor.
1  <resources>
2      <string name="app_name">My ApplicationMenu</string>
3      <string name="action_settings">Settings</string>
4      <string name="action1">Первый</string>
5      <string name="action2">Второй</string>
6      <string name="action3">Третий</string>
7  </resources>
```

Рисунок 7.1 – Код программы

13. Создайте новую папку **menu** в папке **res** с помощью контекстного меню: **res, | New | Directory** (рис. 7.2):

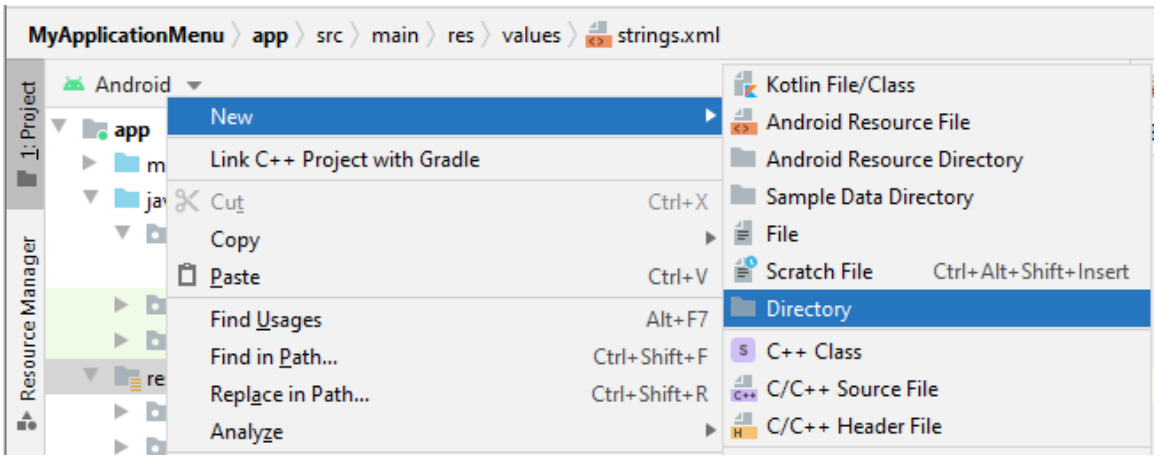


Рисунок 7.2 – Добавление папки

14. Создайте в созданной папке файл **menu_main.xml** – имя указывает, что меню относится к основной активности **MainActivity**. Для этого в контекстном меню папки **menu** выберите: **New | Menu Resource File**. Если в приложении несколько экранов, то у каждой активности необходимо отдельное меню со своими настройками.

15. Откройте файл **menu_main.xml** и добавьте в полученный шаблон код (рис. 7.3):

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never"/>
</menu>
```

Рисунок 7.3 – Код программы

16. Перейдите в файл **MainActivity.** Добавьте новый метод **onCreateOptionsMenu().** Данный метод отвечает за появление меню у активности. Начните вводить заголовок, код система сформирует автоматически (рис. 7.4).

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    return super.onCreateOptionsMenu(menu)
}
```

Рисунок 7.4 – Код программы

17. Добавьте код для отображения пунктов меню на экране (рис. 7.5):

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    val inflater = menuInflater
    inflater.inflate(R.menu.menu_main, menu)
    return super.onCreateOptionsMenu(menu)
}
```

Рисунок 7.5 – Код программы

18. В методе inflate() указан ресурс меню (R.menu.menu_main) и объект класса Menu.

По-английски "inflate" переводится как надувать, т.е. по замыслу разработчиков Android, мы как бы надуваем данными

объект, например, меню. Так запускаются данные из XML-файла в объект MenuInflater.

19. Запустите проект, проверьте наличие меню (рис. 7.6).

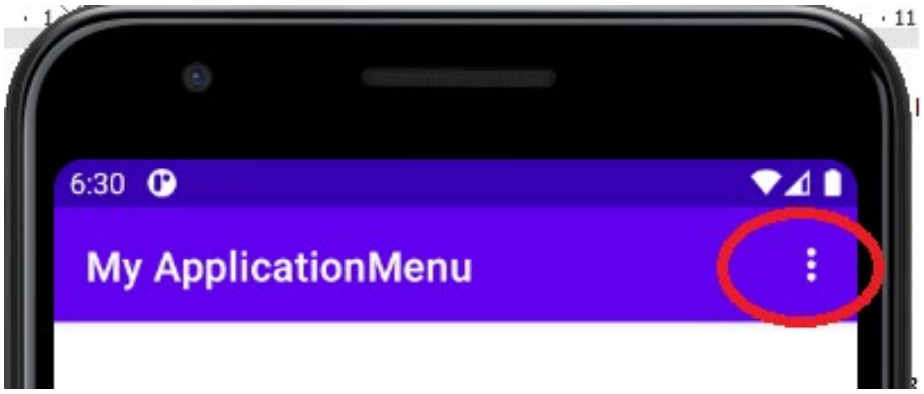


Рисунок 7.6 – Фрагмент приложения с меню

20. Используя элемент **item,** который отвечает за отдельный пункт меню, добавьте ещё три пункта по такому же принципу, меняя только идентификатор и текст для меню (рис. 7.7):

```
    <item
        android:id="@+id/action1"
        android:orderInCategory="100"
        android:title="@string/action1"
        app:showAsAction="never"/>

    <item
        android:id="@+id/action2"
        android:orderInCategory="100"
        android:title="@string/action2"
        app:showAsAction="never"/>

    <item
        android:id="@+id/action3"
        android:orderInCategory="100"
        android:title="@string/action3"
        app:showAsAction="never"/>
</menu>
```

Рисунок 7.7 – Код программы

21. Запустите проект, проверьте наличие новых пунктов меню (рис 7.8).

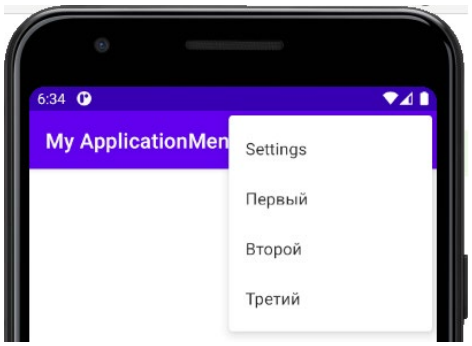


Рисунок 7.8 – Фрагмент программы с меню

22. Параметр **orderInCategory** позволяет задать свой порядок вывода пунктов меню. Если создали пять пунктов меню, но пока не определились с порядком их вывода на экране, чтобы не перемещать постоянно целые блоки кода для пунктов меню в нужном порядке, можно воспользоваться данным параметром. Атрибут **app:showAsAction** определяет поведение меню в **ActionBar**. Значение **never** означает, что элемент меню не должен выводиться в заголовке, а только в всплывающем меню, т.е. находиться за тремя точками. Если установлено значение **always**, то пункт **Settings** сразу появится в заголовке вашего приложения. Также доступны значения **ifRooms**, **withText** и **collapseActionView**. Попробуйте самостоятельно. Например, **ifRoom** выводит пункт меню, если позволяет место. Если пунктов будет много, то они будут только мешаться. Как правило, в таком варианте выводят очень короткое слово или значок для частых операций, чтобы избежать лишнего щелчка на три точки.

23. Пункты меню есть, необходимо создать обработку выбора пунктов меню. Для обработки нажатий пунктов меню служит метод **onOptionsItemSelected()**. Добавьте его в **MainActivity**.

Начните вводить заголовок, код система сформирует автоматически (рис. 7.9).

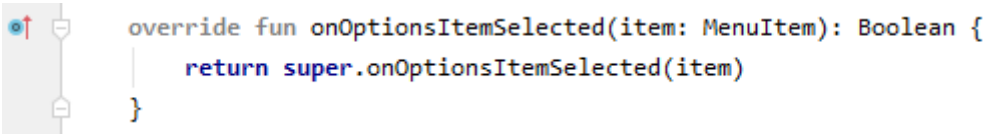


Рисунок 7.9 – Код программы

24. Параметр **item** отвечает за пункт меню. Необходимо получить идентификатор меню через метод **getItemId()** и указать для него код. Так как обычно меню состоит из нескольких пунктов, то удобно использовать конструкцию **when**. Для вывода информации будем использовать текстовую метку. Добавьте на экран активности компонент **TextView** или можно использовать имеющийся **TextView** с надписью "Hello World!", только присвойте ему идентификатор id, например **textView** (рис. 7.10).

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    val textView : TextView = findViewById(R.id.textView)
    when (item.itemId) {
        R.id.action1 -> {
            textView.text = "Вы выбрали пункт 1!"
            return true
        }
        R.id.action2 -> {
            textView.text = "Вы выбрали пункт 2!"
            return true
        }
        R.id.action3 -> {
            textView.text = "Вы выбрали пункт 3!"
            return true
        }
    }
    return super.onOptionsItemSelected(item)
}
```

Рисунок 7.10 – Код программы

25. Запустите приложение, проверьте работу каждого пункта меню.

26. Внешний вид пунктов меню можно изменить на вид с переключателями. Для этого нужно добавить элемент **group** с атрибутом **android:checkableBehavior="single"** (рис. 7.11):

```
    <group android:checkableBehavior="single">
    <item
        android:id="@+id/action1"
        android:orderInCategory="100"
        android:title="@string/action1"
        app:showAsAction="never"/>

    <item
        android:id="@+id/action2"
        android:orderInCategory="100"
        android:title="@string/action2"
        app:showAsAction="never"/>

    <item
        android:id="@+id/action3"
        android:orderInCategory="100"
        android:title="@string/action3"
        app:showAsAction="never"/>
    </group>
</menu>
```

Рисунок 7.11 – Код программы

27. Запустите приложение, проверьте внешний вид пунктов меню.

28. Самостоятельно измените названия пунктов меню и обработайте изменение фона для Activity при выборе пунктов меню или демонстрацию всплывающих подсказок (функция Toast).

29. В Android Studio имеется возможность разработки меню в режиме **Design**. Для этого выберите файл **menu_mail.xml** и откройте его режиме **Design**.

30. Проверьте работу проекта.

31. Сохраните проект.

Задание для самостоятельной работы: выполнить п. 22 и п. 28 лабораторной работы.

Лабораторная работа №8.
Работа с Menu. Продолжение

Цель: изучить особенности разработки приложений с компонентом Menu.

Ход работы

1. Запустите Android Studio.
2. Создайте новый проект для работы с меню:
3. Выберите тип Activity – **Empty**.
4. Нажмите кнопку «Next».
5. Укажите необходимые параметры для проекта.
6. Нажмите кнопку «Finish».
7. Дождитесь окончания процессе сборки проекта, процесс может занять несколько минут.
8. Проверьте необходимость обновления скриптов сборки приложения Gradle. Для этого выберите в списке элементов проекта узел Gradle Scripts, откройте двойным щелчком мыши файл build.gradle, который относится к модулю. Если библиотеку необходимо обновить, строка в модуле будет подсвечена. При наведении на подсвеченную строку курсом мыши, Android Studio предложит обновить библиотеку.
9. Запустите проект и убедитесь, что обновления и сборка проекта закончились успешно. Для этого нажмите на кнопку «Run» (зеленый треугольник на панели инструментов). На забудьте выбрать эмулятор перед запуском.
10. Дождитесь на эмуляторе загрузки операционной системы Android. Убедитесь, что эмулятор работает.
11. Закройте эмулятор.
12. В операционной системе Android используется обычное меню и контекстное меню. Оно вызывается долгим нажа-

тием на объект (long-press в течение двух или трех секунд). Отличается контекстное меню тем, что в нем не поддерживаются значки и быстрые клавиши. При работе с Android Studio необходимо помнить, что контекстное меню применяется к **View**, а обычное к **Activity**, поэтому у каждого компонента может быть свое контекстное меню. Начиная с версии Android 3.0 появилось всплывающее меню (PopupMenu). Всплывающее меню реализуется в виде всплывающего модального окна. Контекстное меню считается устаревшим. С версии Android 4.0 введены новые функции для работы с **PopupMenu**.

13. Работа с контекстным меню осуществляется следующими методами: **registerForContextMenu()** – закрепляет меню за компонентом, **onCreateContextMenu()** – создает меню, **add()** – добавляет пункты меню, **onContextItemSelected() –** обрабатывает выбор пункта меню. Рассмотрим особенности работы с **PopupMenu.**

14. Создайте пункты всплывающего меню. Это можно сделать в xml-файле. Создайте папку **menu** в папке **res** и добавьте в нее файл **popupmenu.xml** (см. действия в предыдущей лабораторной работе). Обратите внимание на свойства пунктов меню (в пунктах меню можно добавлять иконки **icon**, делать их недоступными **enabled** и отображать как checkButton – свойства **checkable** и **check**) (рис. 8.1).

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:id="@+id/menugroup1" >
        <item
            android:id="@+id/menu1"
            android:icon="@mipmap/ic_launcher"
            android:title="Popup menu item 1"/>
        <item
            android:id="@+id/menu2"
            android:title="Popup menu item 2"/>
        <item
            android:id="@+id/menu3"
            android:title="Popup menu item 3">
            <menu>
                <item
                    android:id="@+id/submenu"
                    android:title="Подменю"/>
            </menu>
        </item>
    </group>
    <group android:id="@+id/menugroup2" >
        <item
            android:id="@+id/menu4"
            android:checkable="true"
            android:checked="true"
            android:icon="@mipmap/ic_launcher"
            android:title="Popup menu item 4"/>
        <item
            android:id="@+id/menu5"
            android:title="Popup menu item 5"
            android:enabled="false"/>
        <item
            android:id="@+id/menu6"
            android:title="Popup menu item 6"/>
    </group>

</menu>
```

Рисунок 8.1 – Код программы

15. Переключитесь в режим **Design** и посмотрите, как будет выглядеть меню.

16. Вернитесь в режим **Code** и переведите заголовки в строковые переменные.

17. Обработайте функции всплывающего меню для различных компонентов. Для этого разместите на Activity три компонента: изображение **ImageView**, кнопку **Button** и текст **TextView**. Для каждого компоненты задайте **id**. Пример на рис .8.2.

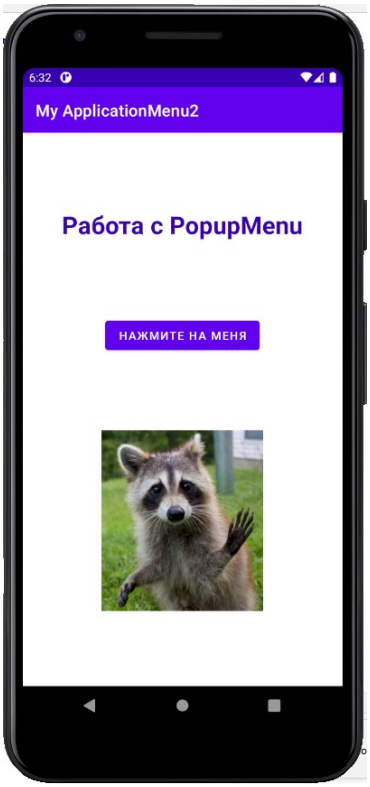


Рисунок 8.2 – Пример запущенного на эмуляторе приложения

18. Переключитесь в режим **Code** и переведите все названия в строковые ресурсы. Для этого встаньте на параметр курсором и нажмите сочетание **Alt+Enter.**

19. Устраните все ошибки в коде. Проверьте работу проекта.

20. Обработайте всплывающее меню для изображения (рис. 8.3):

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val imageView: ImageView = findViewById(R.id.imageView)
        val textView: TextView = findViewById(R.id.textView)
        val button: Button = findViewById(R.id.button)

        val popupMenu = androidx.appcompat.widget.PopupMenu( context: this, imageView)
        popupMenu.inflate(R.menu.popupmenu)
        popupMenu.setOnMenuItemClickListener { it: MenuItem!
            when (it.itemId) {
                R.id.menu1 -> {
                    textView.text = "Нажат пункт PopupMenu 1"
                    true ^setOnMenuItemClickListener
                }
                R.id.menu2 -> {
                    textView.text = "Нажат пункт PopupMenu 2"
                    true ^setOnMenuItemClickListener
                }
                R.id.menu3 -> {
                    textView.text = "Нажат пункт PopupMenu 3"
                    true ^setOnMenuItemClickListener
                }
                else -> false ^setOnMenuItemClickListener

            }
        }

        imageView.setOnClickListener { it: View!
            popupMenu.show()
        }
    }
}
```

Рисунок 8.3 – Код программы

21. Исправьте все ошибки, запустите приложение.

22. Рассмотрим второй способ работы со всплывающим меню. Для этого обработаем меню для кнопки.

23. Создайте xml-файл второго меню. Выберите в режиме **Design** необходимые иконки. Переведите текст в строковые переменные (рис. 8.4).

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/red"
        android:icon="@android:drawable/presence_video_busy"
        android:iconTint="#F44336"
        android:iconTintMode="src_in"
        android:title="Red" />
    <item
        android:id="@+id/yellow"
        android:icon="@android:drawable/presence_video_away"
        android:iconTint="#FFEB3B"
        android:iconTintMode="src_in"
        android:title="Yellow" />
    <item
        android:id="@+id/green"
        android:icon="@android:drawable/presence_video_online"
        android:iconTint="#4CAF50"
        android:iconTintMode="src_in"
        android:title="Green" />
</menu>
```

Рисунок 8.4 – Код программы

24. Дообработайте код для второго всплывающего меню с иконками, переведите текст в строковые переменные (рис. 8.5):

```
        val popupMenu2 = PopupMenu( context: this, button)
        popupMenu2.inflate(R.menu.popup_menu)
        popupMenu2.setOnMenuItemClickListener { it: MenuItem!
            when (it.itemId) {
                R.id.red -> {
                    textView.background = ColorDrawable(Color.RED)
                    textView.text = "Вы выбрали красный цвет"
                }
                R.id.yellow -> {
                    textView.background = ColorDrawable(Color.YELLOW)
                    textView.text  = "Вы выбрали желтый цвет"
                }
                R.id.green -> {
                    textView.background = ColorDrawable(Color.GREEN)
                    textView.text  = "Вы выбрали зеленый цвет"
                }
            }
            false ^setOnMenuItemClickListener
            }
        if (Build.VERSION.SDK_INT >=Build.VERSION_CODES.Q) {
            popupMenu2.setForceShowIcon(true)
        }

        button.setOnClickListener { it: View!
            popupMenu2.show()
        }
    }

}
```

Рисунок 8.5 – Код программы

25. Проверьте работу проекта.

26. Самостоятельно создайте третье всплывающее меню для изменения размера textView. Обработайте изменение размера текста в коде и закрепите всплывающее меню за компонентом textView.

27. Сохраните проект.

Задание для самостоятельной работы: выполните п. 26 лабораторной работы.

Лабораторная работа №9.
Анимация

Цель: изучить особенности разработки приложений с использованием анимации.

Ход работы

1. Запустите Android Studio.
2. Создайте новый проект для демонстрации работы анимации в Android.
3. Выберите тип Activity – **Empty**.
4. Нажмите кнопку «Next».
5. Укажите необходимые параметры для проекта.
6. Нажмите кнопку «Finish».
7. Дождитесь окончания процессе сборки проекта, процесс может занять несколько минут.
8. Проверьте необходимость обновления скриптов сборки приложения Gradle. Для этого выберите в списке элементов проекта узел Gradle Scripts, откройте двойным щелчком мыши файл build.gradle, который относится к модулю. Если библиотеку необходимо обновить, строка в модуле будет подсвечена. При наведении на подсвеченную строку курсом мыши, Android Studio предложит обновить библиотеку.
9. Запустите проект и убедитесь, что обновления и сборка проекта закончились успешно. Для этого нажмите на кнопку «Run» (зеленый треугольник на панели инструментов). На забудьте выбрать эмулятор перед запуском.
10. Дождитесь на эмуляторе загрузки операционной системы Android. Убедитесь, что эмулятор работает.

11. В Android Studio есть возможность анимировать объекты. Создадим объекты в xml и обработаем движение на экране. Для этого создайте файл **sun.xml** в папке **res / drawable** (правой кнопкой мыши по папке **drawable** – **New** – **File**) со следующим кодом (рис. 9.1):

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

Рисунок 9.1 – Код программы

12. Создайте файл **sky.xml** в папке **res / drawable** (правой кнопкой мыши по папке **drawable** – **New** – **File**) со следующим кодом (рис. 9.2):

```
<?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="#ff0000ff" />

</shape>
```

Рисунок 9.2 – Код программы

13. Создайте файл **grass.xml** в папке **res / drawable** (правой кнопкой мыши по папке **drawable** – **New** – **File**) со следующим кодом (рис. 9.3):

```
<?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="#ff003300"
        android:startColor="#ff009900" />

</shape>
```

Рисунок 9.3 – Код программы

14. Добавьте в файл строковых ресурсов следующие переменные (рис. 9.4):

```
<resources>
    <string name="app_name">My ApplicationAnimation</string>
    <string name="sun">Солнце</string>
    <string name="grass">Трава</string>
    <string name="sky">Небо</string>
    <string name="clock">Часы</string>
    <string name="hour">Стрелка</string>
</resources>
```

Рисунок 9.4 – Код программы

15. Расположите в **activity_main.xml** три компонента **ImageView**, которые будут отображать солнце, небо и траву соответственно (рис. 9.5).

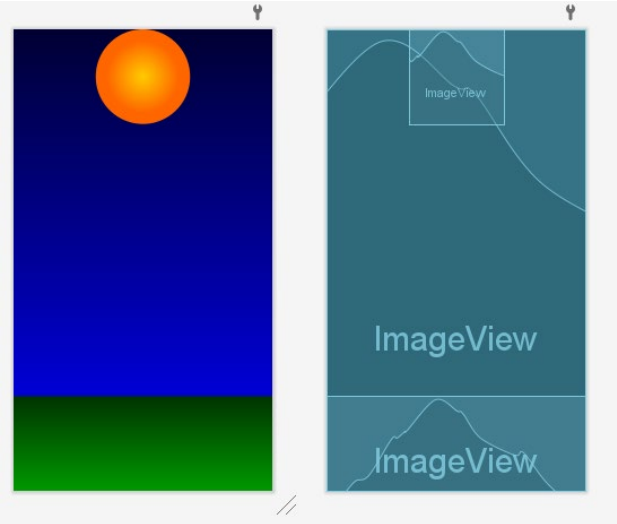


Рисунок 9.5 – Дизайн приложения

16. Определите их местоположение на **Activity**, укажите **id**. Свойство **android:src** позволяет разместить ранее созданные файлы xml с изображениями. Например, на рис. 9.6:

```
<ImageView
    android:id="@+id/sky"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:contentDescription="@string/sky"
    android:src="@drawable/sky"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<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"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<ImageView
    android:id="@+id/grass"
    android:layout_width="0dp"
    android:layout_height="150dp"
    android:layout_alignParentBottom="true"
    android:contentDescription="@string/grass"
    android:src="@drawable/grass"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
```

Рисунок 9.6 – Код программы

17. Запустите эмулятор, проверьте размещение объектов (рис. 9.7).

Рисунок 9.7 – Запущенное на эмуляторе приложение

18. Обработаем анимацию. Для этого создайте в папке **res** папку **anim** (с помощью контекстного меню, пункт **New** – **Directory**.

19. Создайте в папе **anim** файл **sun_rise.xml**, которые отвечает за анимацию (рис. 9.8):

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="5000"
    android:fillAfter="true"
    android:interpolator="@android:anim/accelerate_decelerate_interpolator"
    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>
```

Рисунок 9.8 – Код программы

20. Пояснения: блок **set** позволяет установить настройки анимации. Свойство **android:duration** отвечает за длительность анимации (здесь 5 секунд). Параметр **fillAfter** управляет состоянием анимации – она не должна переключаться в начало. Параметр **android:interpolator** использует системную константу для небольшого ускорения от начала к середине анимации и торможения от середины к концу анимации. Внутри **set** расположены следующие блоки: **scale** – изменение размеров, **translate** – позиции и **alpha** – прозрачности. Например, в **scale** фигура солнца увеличиваться от своего изначального размера в полтора раза, равномерно от своей середины. Блок **translate** организует перемещение солнца 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**).

21. Создадим код (рис. 9.9):

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val imageView: ImageView = findViewById(R.id.sun)
        val sunRiseAnimation: Animation = AnimationUtils.loadAnimation( context: this, R.anim.sun_rise)
        imageView.startAnimation(sunRiseAnimation)

    }
}
```

Рисунок 9.9 – Код программы

22. Исправьте все ошибки, запустите приложение.

23. Добавим анимацию часов в правом нижнем углу. За время восхода стрелку на часах совершит два оборота. Для этого создадим изображение часов и стрелок.

24. Создайте файл **clock.xml** в папке **res** – **drawable** (рис. 9.10):

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

    <item>
        <shape
            android:dither="true"
            android:shape="oval" >
            <gradient
                android:endColor="#ffffffff"
                android:gradientRadius="100"
                android:startColor="#66ffffff"
                android:type="radial"
                android:useLevel="false" />

            <size
                android:height="100dp"
                android:width="100dp" />

            <stroke
                android:width="2dp"
                android:color="#99000000" />
        </shape>
    </item>
    <item
        android:bottom="44dp"
        android:left="48dp"
        android:right="48dp"
        android:top="5dp">
        <shape android:shape="rectangle" >
            <solid android:color="#99000000" />
        </shape>
    </item>

</layer-list>
```

Рисунок 9.10 – Код программы

25. Создайте файл **clock_turn.xml** в папке **res** – **anim** для анимации часов. В анимации указано значение 720 градусов, чтобы часы сделали полный оборот два раза. Хотя вращается вся фигура, пользователю будет казаться, что вращается только стрелка (рис. 9.11).

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

    <rotate
        android:fromDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toDegrees="720" />

</set>
```

Рисунок 9.11 – Код программы

26. Разместите на **Activity** новый **ImageView** для часов и разместите его в правый нижний угол (рис. 9.12).

```xml
<ImageView
    android:id="@+id/clock"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginBottom="16dp"
    android:contentDescription="@string/clock"
    android:padding="10dp"
    android:src="@drawable/clock"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent" />
```

Рисунок 9.12 – Код программы

27. Обработайте код (рис. 9.13):

```
val imageView1: ImageView = findViewById(R.id.clock)
val clockTurnAnimation: Animation = AnimationUtils.loadAnimation( context: this, R.anim.clock_turn)
imageView1.startAnimation(clockTurnAnimation)
```

Рисунок 9.13 – Код программы

28. Добавим часовую стрелку. Для этого создайте файл **hour_hand.xml** в папке **res** – **drawable.** Основные отличия от предыдущего файла – прозрачный круг и более короткая стрелка. При наложении на часы с минутной стрелкой, мы увидим часовую стрелку, а прозрачный круг мешать не будет (рис. 9.14):

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

    <item>
        <shape
            android:dither="true"
            android:shape="oval" >
            <solid android:color="#00000000" />

            <size
                android:height="100dp"
                android:width="100dp" />
        </shape>
    </item>
    <item
        android:bottom="44dp"
        android:left="48dp"
        android:right="48dp"
        android:top="15dp">
        <shape android:shape="rectangle" >
            <solid android:color="#99000000" />
        </shape>
    </item>

</layer-list>
```

Рисунок 9.14 – Код программы

29. Разместите на **Activity** новый **ImageView** для стрелки часов и разместите его в правый нижний угол строго поверх часов (рис. 9.15).

```
<ImageView
    android:id="@+id/hour_hand"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true"
    android:contentDescription="@string/clock"
    android:padding="10dp"
    android:src="@drawable/hour_hand"
    app:layout_constraintEnd_toEndOf="@+id/sky"
    app:layout_constraintTop_toTopOf="@+id/clock" />
```

Рисунок 9.15 – Код программы

30. Создайте файл анимации часовой стрелки **hour_turn.xml** в папке **res** – **anim.** Начальная позиция установлена в значении 180 градусов, что соответствует 6 часам. При анимации стрелка повернётся на 60 градусов и будет соответствовать 8 часам. За это время минутная стрелка сделает два полных оборота, что соответствует двум часам (8-6) (рис. 9.16):

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

    <rotate
        android:fromDegrees="180"
        android:pivotX="50%"
        android:pivotY="50%"
        android:toDegrees="240" />

</set>
```

Рисунок 9.16 – Код программы

31. Обработайте код (рис. 9.17):

```
val imageView2: ImageView = findViewById(R.id.hour_hand)
val hourTurnAnimation: Animation = AnimationUtils.loadAnimation( context: this, R.anim.hour_turn)
imageView2.startAnimation(hourTurnAnimation)
```

Рисунок 9.17 – Код программы

32. Проверьте работу проекта.
33. Сохраните проект.

Задание для самостоятельной работы: создайте проект с анимацией. Требования: минимум три объекта, различные траектории движения. Информацию по тегам для прорисовки в xml можно найти в сети Интернет.

Контрольные вопросы

1. Опишите достоинства разработки для Android.
2. Опишите недостатки разработки для Android.
3. Назовите этапы разработки мобильного приложения.
4. Какова архитектура операционной системы Android.
5. Опишите процесс загрузки операционной системы Android.
6. Перечислите возможности Android Studio при разработке мобильных приложений.
7. Назовите шаги установки Android Studio.
8. Перечислите параметры создания проекта в Android Studio.
9. Какие шаги необходимо выполнить для создания проекта в Android Studio.
10. Какова структура проекта в Android Studio.
11. Роль файла AndroidManifest.xml.
12. Структура файла AndroidManifest.xml.
13. Назначение компонентов файла AndroidManifest.xml.
14. Язык программирования Kotlin.
15. Достоинства и недостатки разработки приложения на языке программирования Kotlin.
16. Способы подключения библиотек в Android Studio.
17. Что такое Activity.
18. Назовите способы создания Activity.
19. Перечислите особенности жизненного цикла Activity.
20. Что такое макет приложения.
21. Способы создания макета приложения.
22. Что такое Layout? Основные виды Layout.
23. Основные атрибуты Layout.

24. Реализация переходов между Activity.
25. Что такое ресурсы в Android-приложении?
26. Типы ресурсов
27. Способы использования ресурсов, примеры.
28. Работа с ресурсами из кода приложения.
29. Работа с ресурсами в XML-файле.
30. Как обработать событие (например, нажатие кнопки) в Android Studio.

Список рекомендованных источников

1. Введение в разработку приложений для ОС Android : учебное пособие / Ю. В. Березовская, О. А. Юфрякова, В. Г. Вологдина [и др.]. – 3-е изд. – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. – 427 с. – ISBN 978-5-4497-0890-8. – Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. – URL: https://www.iprbookshop.ru/102000.html (дата обращения: 15.07.2021). – Режим доступа: для авторизир. пользователей.

2. Информационная система «Единое окно доступа к образовательным ресурсам. Раздел. Информатика и информационные технологии» https://habr.com/. Например, открытые лекции Технотрека «Разработка под Android» https://habr.com/ru/company/mailru/ blog/345252/.

3. Официальный сайт Android Studio для разработчиков https://developer.android.com/studio?hl=ru.

4. Официальный сайт языка программирования Kotlin https://kotlinlang.org/.

5. Пирская, Л. В. Разработка мобильных приложений в среде Android Studio : учебное пособие / Л. В. Пирская. – Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2019. – 123 с. – ISBN 978-5-9275-3346-6. – Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. – URL: https://www.iprbookshop.ru/100196.html (дата обращения: 15.07.2021). – Режим доступа: для авторизир. пользователей.

6. Разработка на Android – открытый курс лекций в рамках образовательного проекта «Технотрек Mail.ru Group» при МФТИ https://www.youtube.com/playlist?list=PLrCZzMib1e9ptI7bPXFG8X5xEiCBt5qYE.

7. Руководство по языку программирования Kotlin на русском языке https://kotlinlang.ru/.

8. Семакова, А. Введение в разработку приложений для смартфонов на ОС Android : учебное пособие для СПО / А. Семакова. – Саратов : Профобразование, 2021. – 102 с. – ISBN 978-5-4488-0994-1. – Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. – URL: https://www.iprbookshop.ru/102187.html (дата обращения: 15.07.2021). – Режим доступа: для авторизир. пользователей.

Учебное издание

Носова Людмила Сергеевна

РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Ответственный редактор
Е. Ю. Никитина

Корректор
В.Е. Жабаков

Компьютерная верстка
В. М. Жанко

Подписано в печать 30.10.2021. Формат 60x84 1/16. Усл. печ. л. 6,57. Тираж 500 экз. Заказ 503.

Южно-Уральский научный центр Российской академии образования. 454080, Челябинск, проспект Ленина, 69, к. 454.
Учебная типография Федерального государственного бюджетного образовательного учреждения высшего образования «Южно-Уральский государственный гуманитарно-педагогический университет. 454080, Челябинск, проспект Ленина, 69.