



*Белорусский государственный университет
информатики и радиоэлектроники*

Эргономика мобильных приложений

Преподаватель: Меженная Марина Михайловна

К.Т.Н., доцент,

доцент кафедры инженерной психологии и эргономики
а 609-2

mezhennaya@bsuir.by



Кафедра инженерной психологии и эргономики

Лекция 2:

Архитектура и основные компоненты системы Android. Создание и запуск Android проекта.

План лекции:

1. Архитектура Android.
2. Основные компоненты Android.
3. Создание Android проекта.
4. Структура Android проекта.
5. Запуск Android проекта.

Архитектура Android

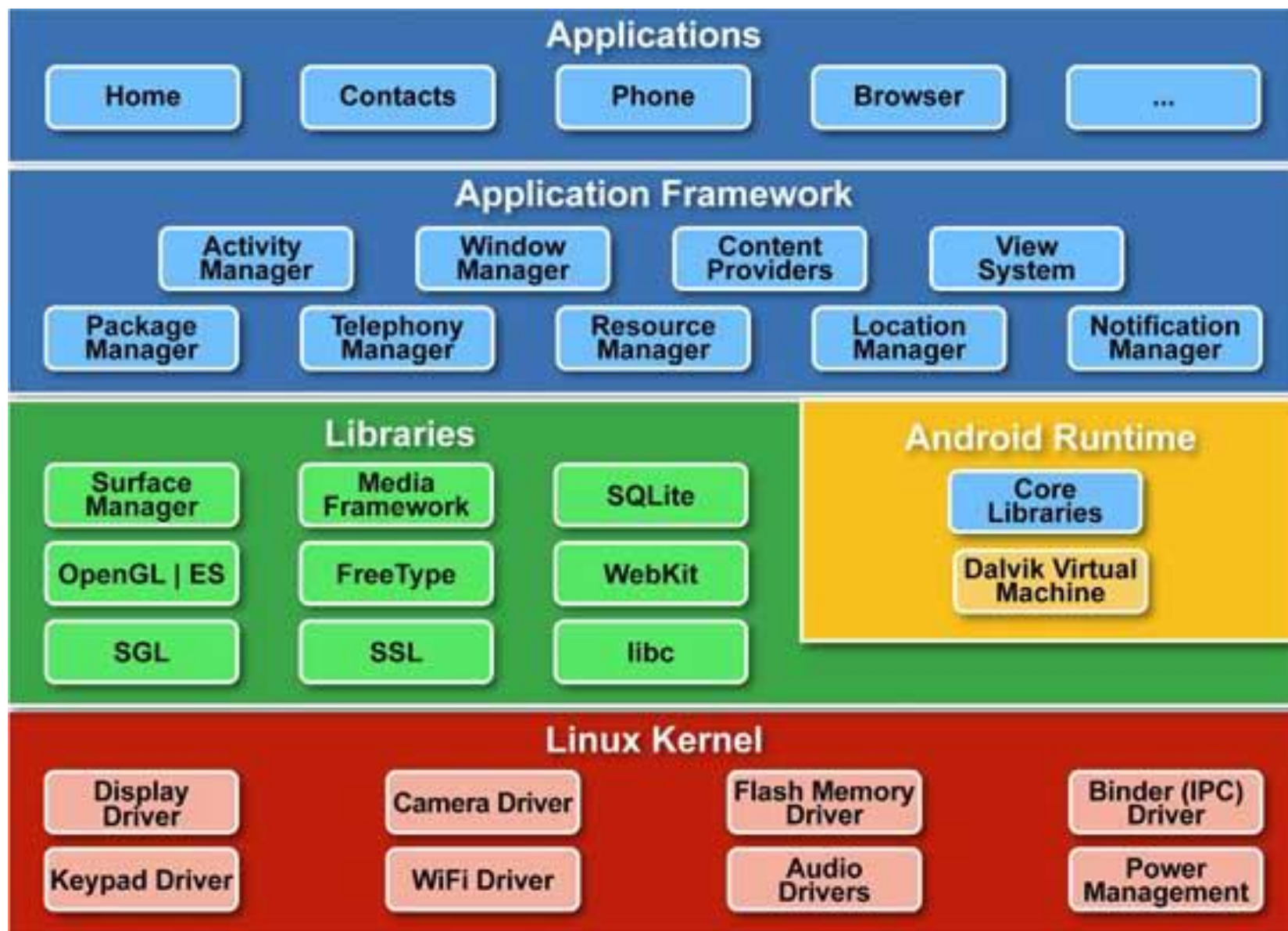
Система Android предназначена для мобильных устройств. Это означает, что наивысший приоритет отдается **сохранению энергии батареи** и эффективному **управлению ограниченными ресурсами памяти**.

Система Android включает операционную систему, программное обеспечение промежуточного слоя (middleware), а также основные пользовательские приложения (e-mail, календарь, карты, браузер, контакты и др.).

Архитектуру Android принято делить на уровни:

- уровень ядра;
 - уровень библиотек и среды выполнения;
 - уровень каркаса приложений;
 - уровень приложений.
-

Архитектура Android



Архитектура Android: уровень ядра

Ядро Linux обеспечивает функционирование системы и отвечает за безопасность, управление памятью, энергосистемой и процессами, а также предоставляет сетевой стек и модель драйверов.

Драйвер IPC обеспечивает механизм Inter-process Communication, который является основным механизмом взаимодействия между процессами.



Архитектура Android: уровень библиотек

Набор библиотек (Libraries) предназначен для обеспечения важнейшего базового функционала приложений. Библиотеки реализованы на C/C++ .



Архитектура Android: уровень библиотек

Surface Manager – композитный менеджер окон: отвечает за подготовку графики в буфере, а затем ее полного (композитного) отображения на экране. Это позволяет системе создавать интересные бесшовные эффекты, прозрачность окон и плавные переходы.

Media Framework – библиотеки записи и воспроизведения аудио и видео контента, а также вывода статических изображений.

SQLite – легковесная и производительная реляционная СУБД, используемая в Android в качестве основного движка для работы с базами данных, используемыми приложениями для хранения информации.

Архитектура Android: уровень библиотек

OpenGL ES (OpenGL for Embedded Systems) – открытый движок для работы с 3D-графикой.

FreeType – высококачественный движок для шрифтов и отображения текста.

WebKit – готовый движок для веб-браузера; обрабатывает HTML, JavaScript.

SGL (Skia Graphics Engine) – открытый движок для работы с 2D-графикой.

SSL – библиотеки для поддержки одноименного криптографического протокола, обеспечивающий безопасную передачу данных по сети.

Libc – стандартная библиотека языка C, настроенная для работы на устройствах на базе Linux. Носит название Bionic.

Архитектура Android: уровень библиотек и среды выполнения

Для того, чтобы одно и то же приложение могло работать на разном аппаратном обеспечении, компания Google использовала контейнер-ориентированную архитектуру. В такой архитектуре двоичный код выполняется программным контейнером (виртуальной машиной) и изолируется от деталей конкретного аппаратного обеспечения.



Архитектура Android: уровень библиотек и среды выполнения

В Android 4.4 появилась возможность сменить виртуальную машину Dalvik на ART (Android Runtime).

В отличие от Dalvik, который использует JIT-компиляцию (Just-in-Time: во время выполнения приложения), ART компилирует приложение единожды во время его установки (Ahead-of-Time). За счет этого достигается повышение скорости работы программ и одновременно увеличение времени работы от батареи.

Однако при этом увеличивается время установки приложения, занимаемое место во внутренней памяти устройства.

Для обеспечения обратной совместимости ART использует тот же байт-код, что и Dalvik.

Архитектура Android: уровень каркаса приложений

На этом уровне находятся основные службы Android:

- менеджер деятельности (Activity Manager)
- менеджер пакетов (Package Manager)
- менеджер окон (Window Manager)
- менеджер ресурсов (Resource Manager)
- телефонный менеджер (Telephony Manager)
- менеджер местоположения (Location Manager)
- менеджер уведомлений (Notification Manager)



Архитектура Android: уровень каркаса приложений

Набор представлений (**Views**) используется для создания визуальных компонентов приложений (списков, текстовых полей, таблиц, кнопок или даже встроенного web-браузера).

Контент-провайдеры (**Content Providers**) управляют данными, которые одни приложения открывают для других, чтобы те могли их использовать для своей работы.

Менеджер ресурсов (**Resource Manager**) обеспечивает доступ к ресурсам без функциональности (не несущими кода), например, к строковым данным, графике, файлам.

Менеджер оповещений (**Notification Manager**) отображает уведомления для пользователя в строке состояния.

Менеджер действий (**Activity Manager**) управляет жизненными циклами приложений.

Менеджер местоположения (**Location Manager**) обновляет данные о текущем географическом положении устройства.

Архитектура Android: уровень приложений

Уровень приложений включает стандартные приложения Android, а также загруженные дополнительные приложения из магазина Android.

Платформа Android не делает разницы между стандартными приложениями и сторонним программным обеспечением, таким образом, ключевые приложения, входящие в стандартный набор, можно заменить при желании альтернативными приложениями.

Программы для Android пишутся на языке Java.



Основные компоненты Android-приложения

Всего в Android приложениях существует четыре типа компонентов:

1. Деятельность (**Activity**),
2. Служба (**Service**),
3. Приемник широковещательных намерений (**Broadcast Receiver**),
4. Контент-провайдер (**Content Provider**).

Взаимодействие компонентов осуществляется с помощью объектов **Intent**.

Основные компоненты Android-приложения: Activity

Activity представляет собой визуальный пользовательский интерфейс для приложения - окно.

Activity может также использовать дополнительные окна, например всплывающее диалоговое окно

Все деятельности реализуются как подкласс базового класса Activity.

Основные компоненты Android-приложения: Service

Компонент Service не имеет визуального интерфейса пользователя и выполняется в фоновом режиме в течение неопределенного периода времени, пока не завершит свою работу.

В общем случае это процесс, который функционирует, когда приложение не в фокусе. Примером такого процесса может стать прослушивание музыки в то время, когда пользователь делает что-то другое или получение данных по сети без блокирования текущей активности.

Приложения могут подключаться к компоненту Service или запускать его, если он не запущен, а также останавливать уже запущенные компоненты.

Основные компоненты Android-приложения:

Broadcast Receiver

Broadcast Receiver используется для отслеживания внешних событий и реакции на них. Приложение может иметь несколько компонентов Broadcast Receiver, чтобы ответить на любые объявления, которые оно считает важными. При этом каждый компонент BroadcastReceiver будет определять код, который выполнится после возникновения конкретного внешнего события. С помощью класса NotificationManager можно сообщить пользователю информацию, требующую его внимания.

Примером оповещений может быть сигнал о том, что информация загружена на устройство и доступна к использованию.

Основные компоненты Android-приложения:

Content Provider

Content Provider делает определенный набор данных, используемых приложением, доступным для других приложений. Данные могут быть сохранены в файловой системе, в базе данных SQLite, в сети, или в любом другом месте, к которому приложение может иметь доступ.

Контент-провайдеры для безопасного доступа к данным используют механизм разрешений. Посредством content provider другое приложение может запрашивать данные и, если выставлены соответствующие разрешения, изменять их.

Например, система Android содержит content provider, который управляет пользовательской информацией о контактах.

Intent

Intent – специальные классы в коде приложения, которые определяют и описывают запросы приложения на выполнение каких-либо операций. Намерения добавляют слой, позволяющий оперировать компонентами с целью их повторного использования и замещения.

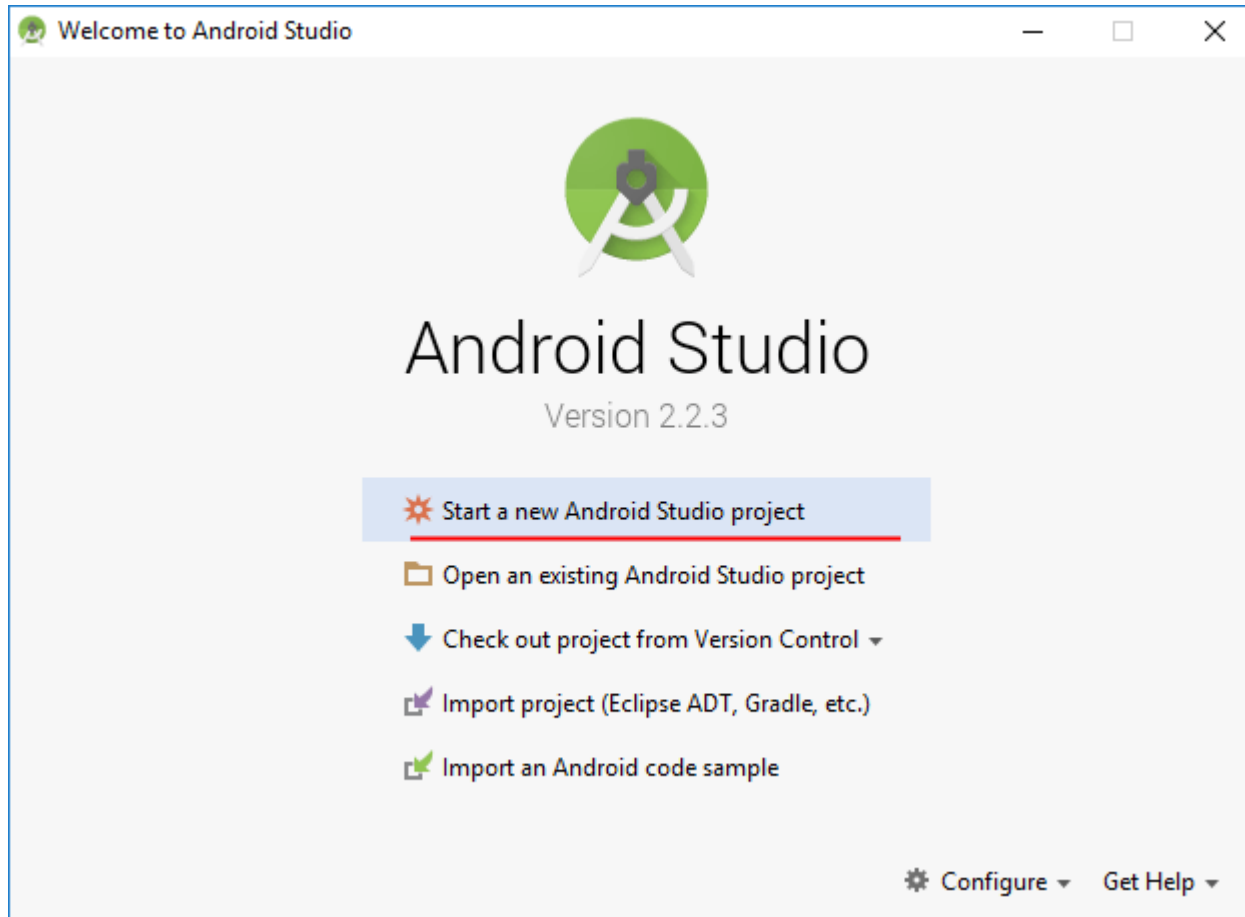
В некоторых случаях это может быть очень мощным средством интеграции приложений. Главная особенность платформы Android состоит в том, что одно приложение может использовать элементы других приложений при условии, что эти приложения разрешают использовать свои компоненты. При этом ваше приложение не включает код другого приложения или ссылки на него, а просто запускает нужный элемент другого приложения.

Создание Android проекта

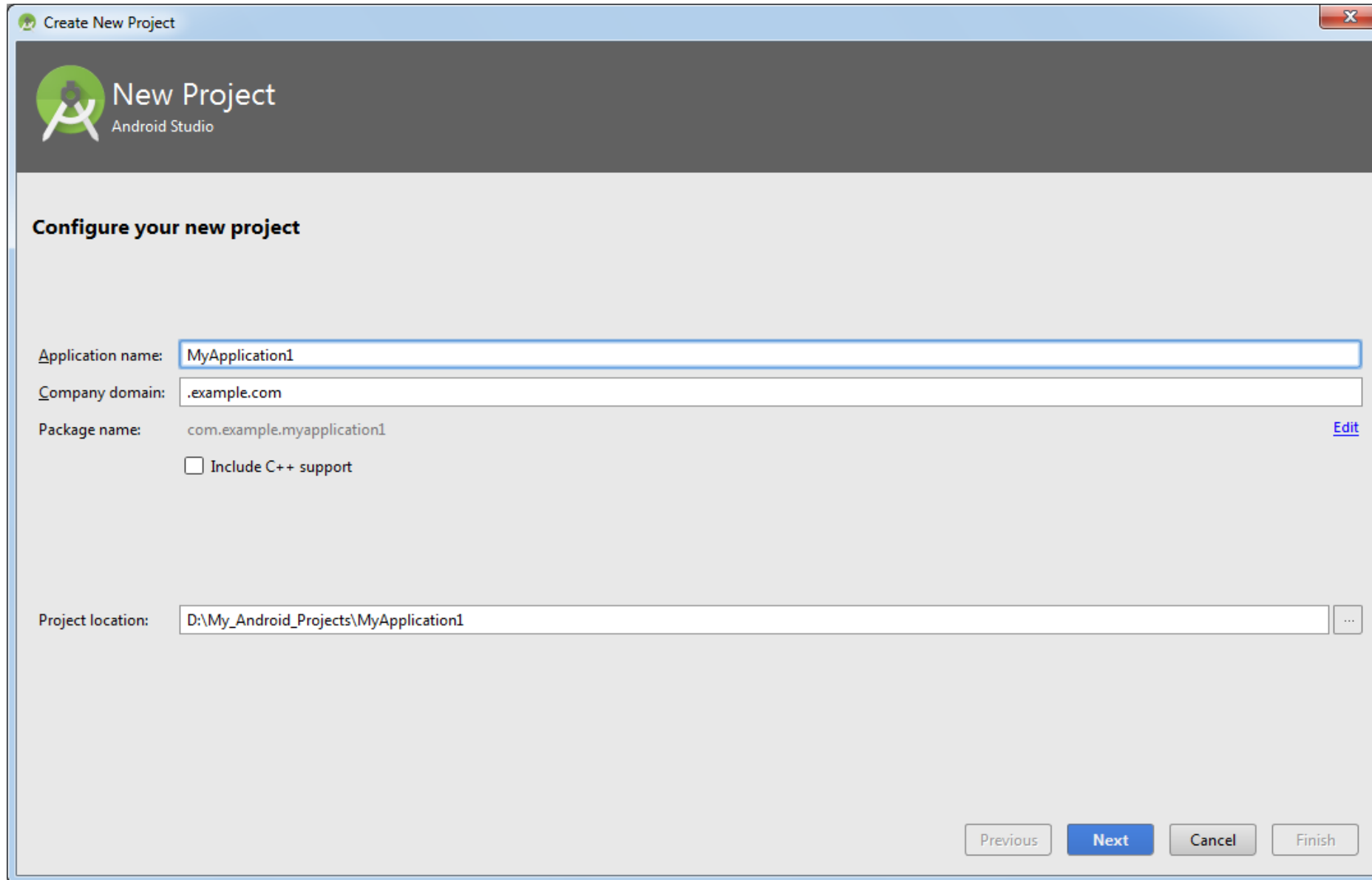
При создании проекта, в нем создается модуль.
Модуль по сути и является приложением.
А проект - контейнер для модуля.




Создание Android проекта: Start a new Android Studio project



Создание Android проекта: Ввод Application name, Company Domain



Create New Project

 New Project
Android Studio

Configure your new project

Application name:

Company domain:

Package name: [Edit](#)


☐ Include C++ support

Project location: ...

Previous Next Cancel Finish

Создание Android проекта: Minimum SDK

Create New Project

 Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

☒ Phone and Tablet

Minimum SDK

API 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available.
By targeting API 15 and later, your app will run on approximately **100,0%** of the devices that are active on the Google Play Store.
[Help me choose](#)

☐ Wear

Minimum SDK

API 21: Android 5.0 (Lollipop)

☐ TV

Minimum SDK

API 21: Android 5.0 (Lollipop)

☐ Android Auto

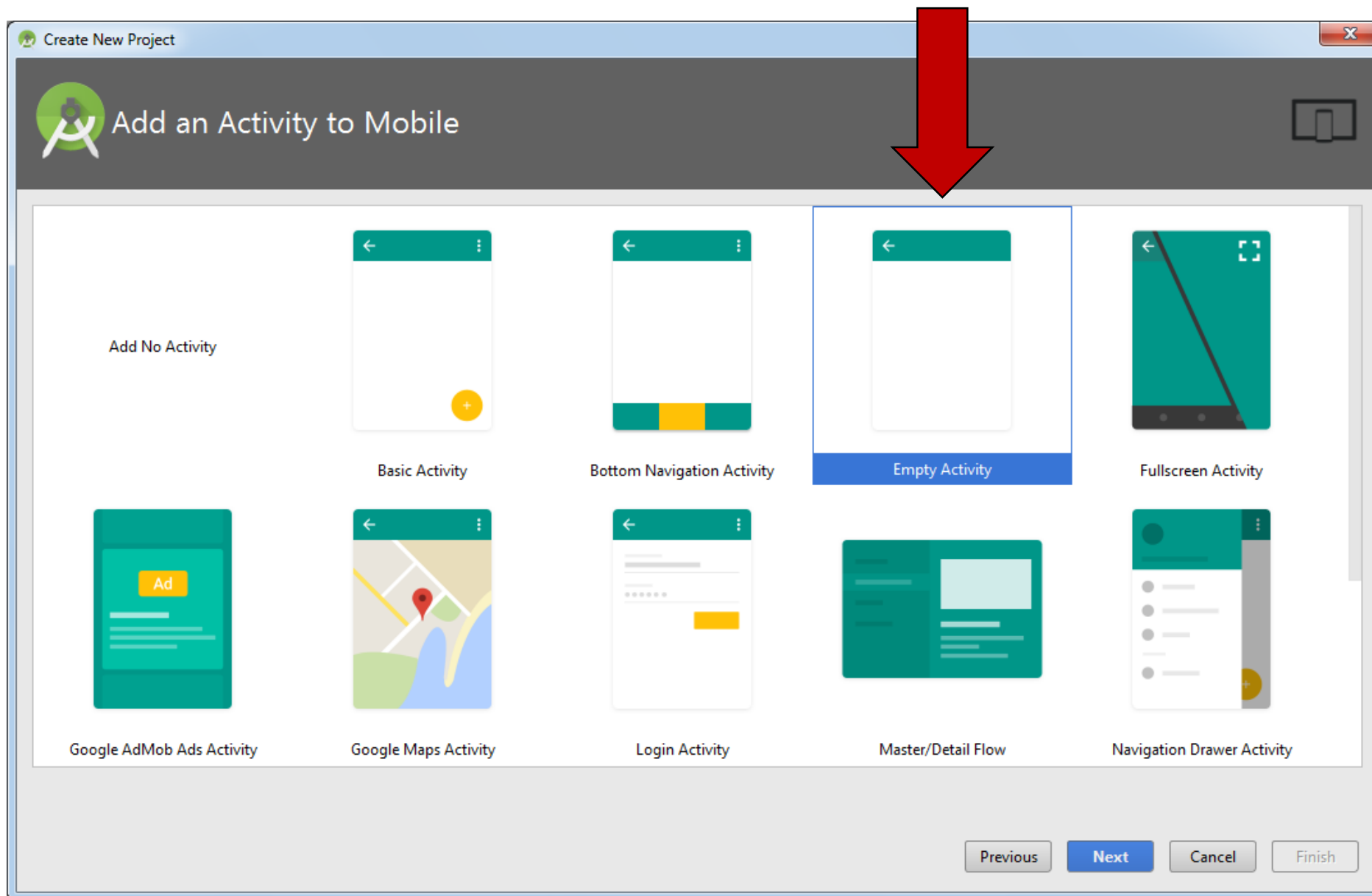
Previous

Next

Cancel


Finish


Создание Android проекта: Empty Activity



Создание Android проекта: имена файлов проекта-«заготовки»

Create New Project

 Customize the Activity



Creates a new empty activity

Activity Name: MainActivity

☒ Generate Layout File

Layout Name: activity_main

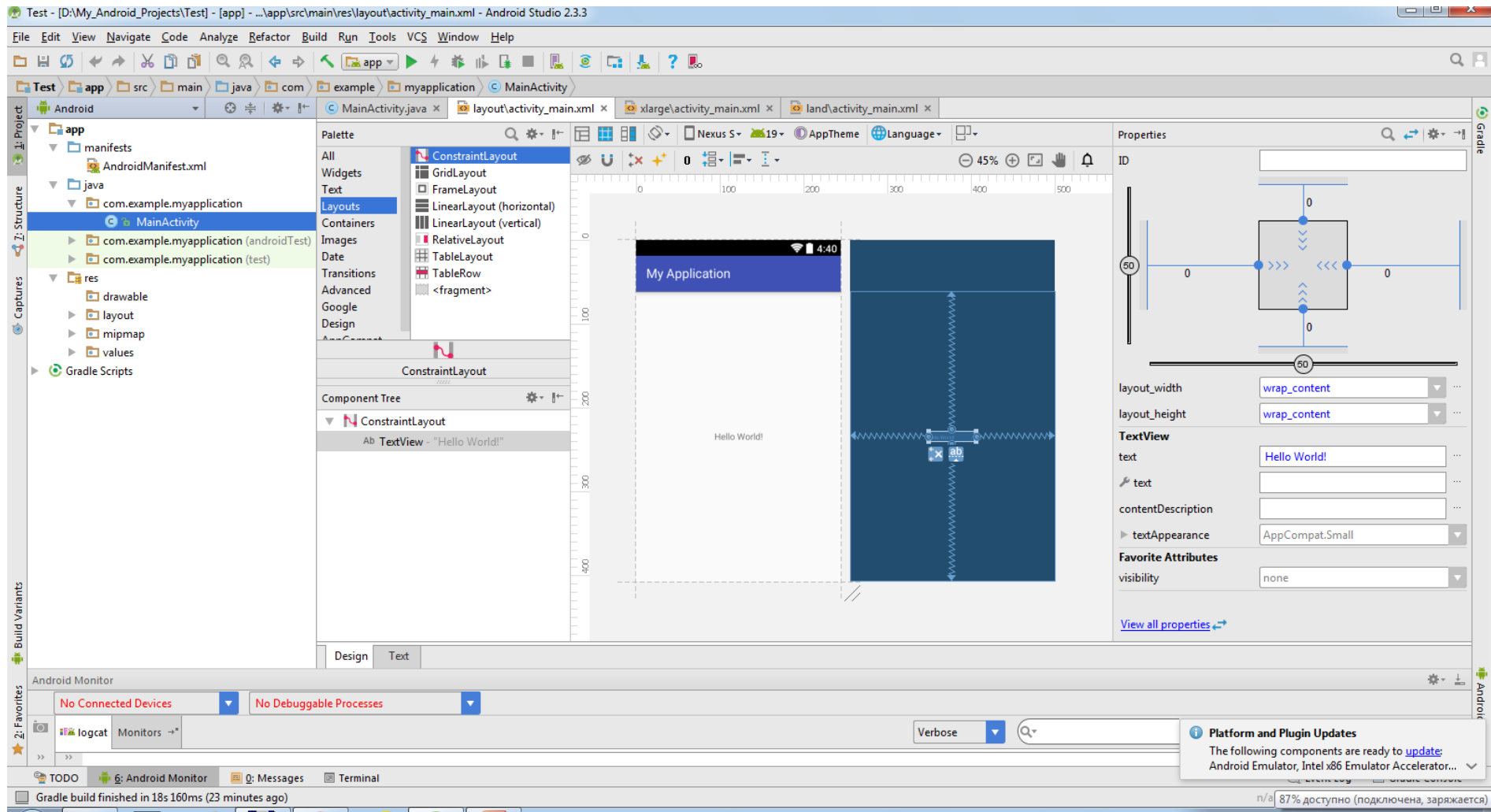
☒ Backwards Compatibility (AppCompat)

Empty Activity

The name of the activity class to create

PreviousNextCancelFinish

Проект создан:



Структура проекта:

Файл `AndroidManifest.xml` – манифест или конфиг-файл приложения.

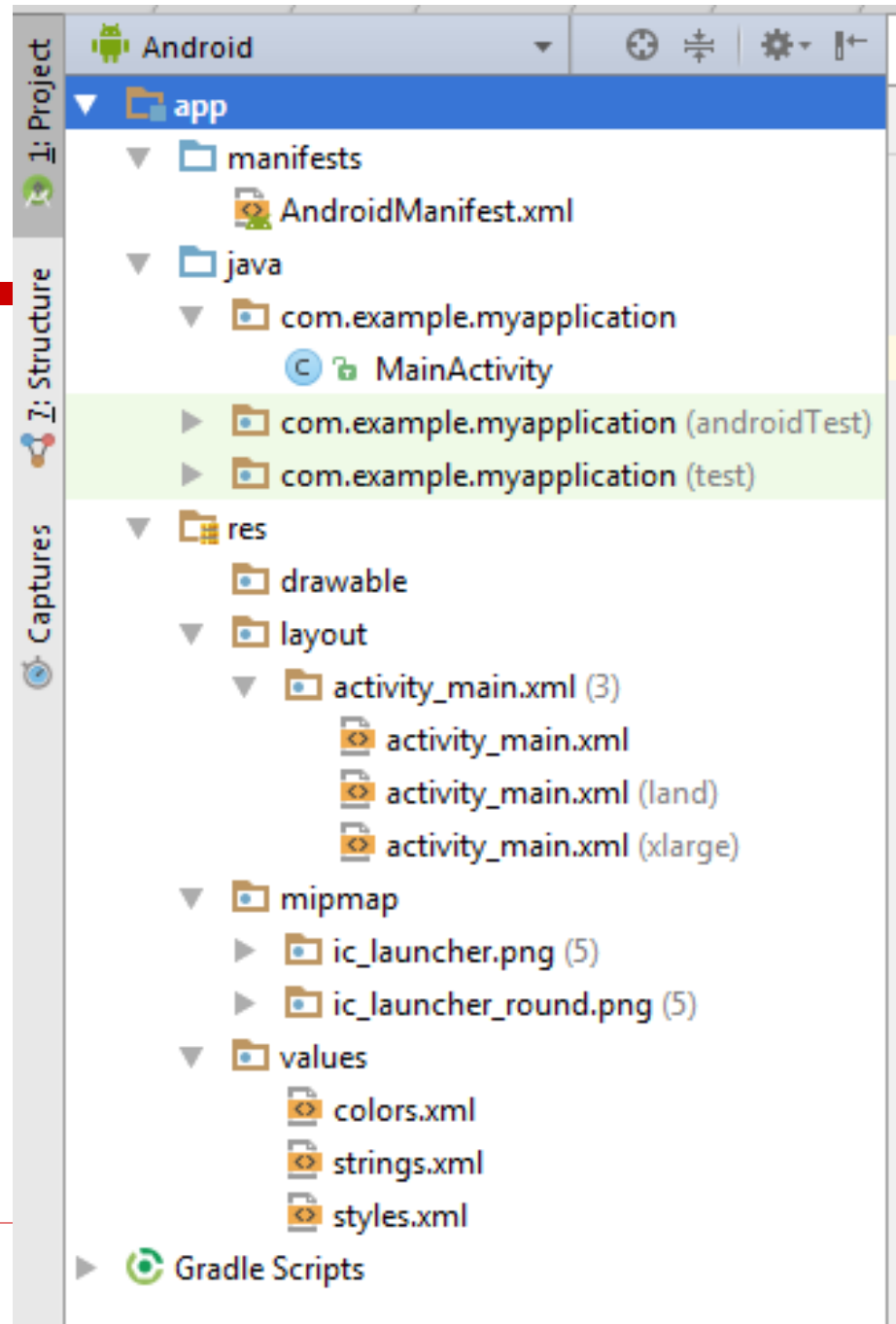
Папке `java` содержит весь код приложения.

Папка `res` используется для файлов-ресурсов различного типа.

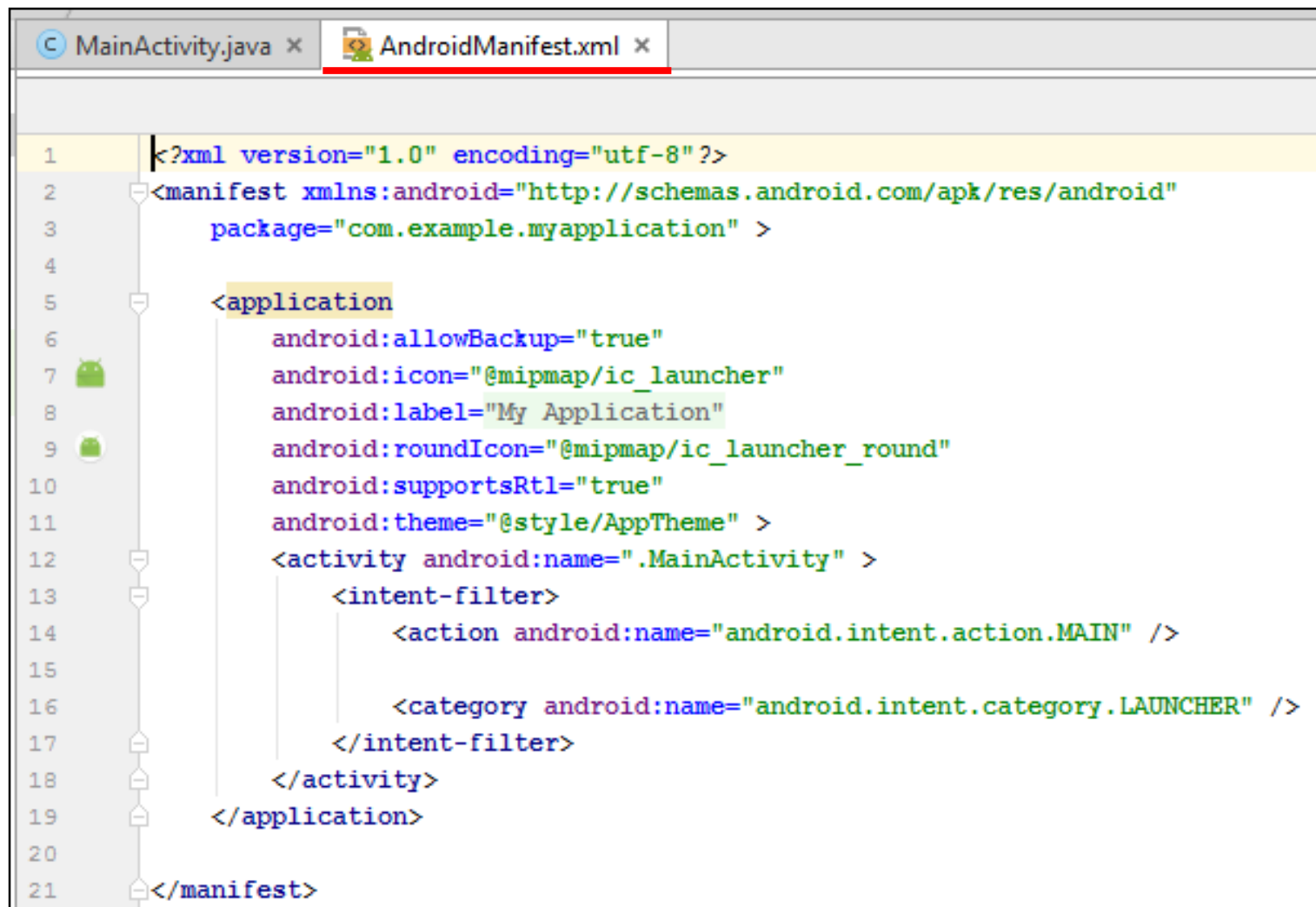
`res/drawable/` - для изображений (PNG, JPEG и т. д.);

`res/layout/` - для XML-файлов разметки (компоновка графических элементов окон приложения);

`res/values/` - для строковых ресурсов, стилей и т. д.



Файл AndroidManifest.xml

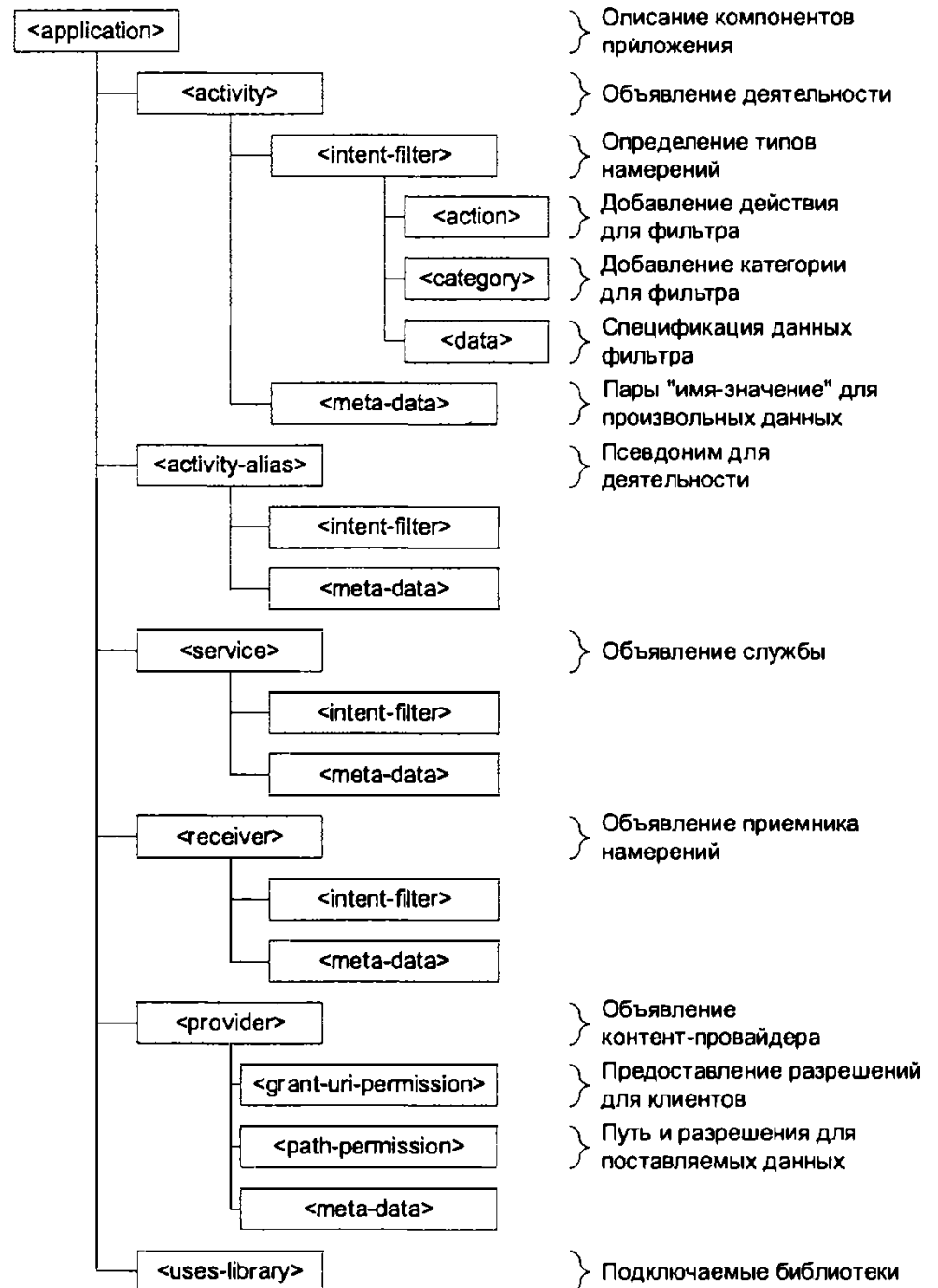


```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.example.myapplication" >
4
5     <application
6         android:allowBackup="true"
7         android:icon="@mipmap/ic_launcher"
8         android:label="My Application"
9         android:roundIcon="@mipmap/ic_launcher_round"
10        android:supportRtl="true"
11        android:theme="@style/AppTheme" >
12        <activity android:name=".MainActivity" >
13            <intent-filter>
14                <action android:name="android.intent.action.MAIN" />
15
16                <category android:name="android.intent.category.LAUNCHER" />
17            </intent-filter>
18        </activity>
19    </application>
20
21 </manifest>
```

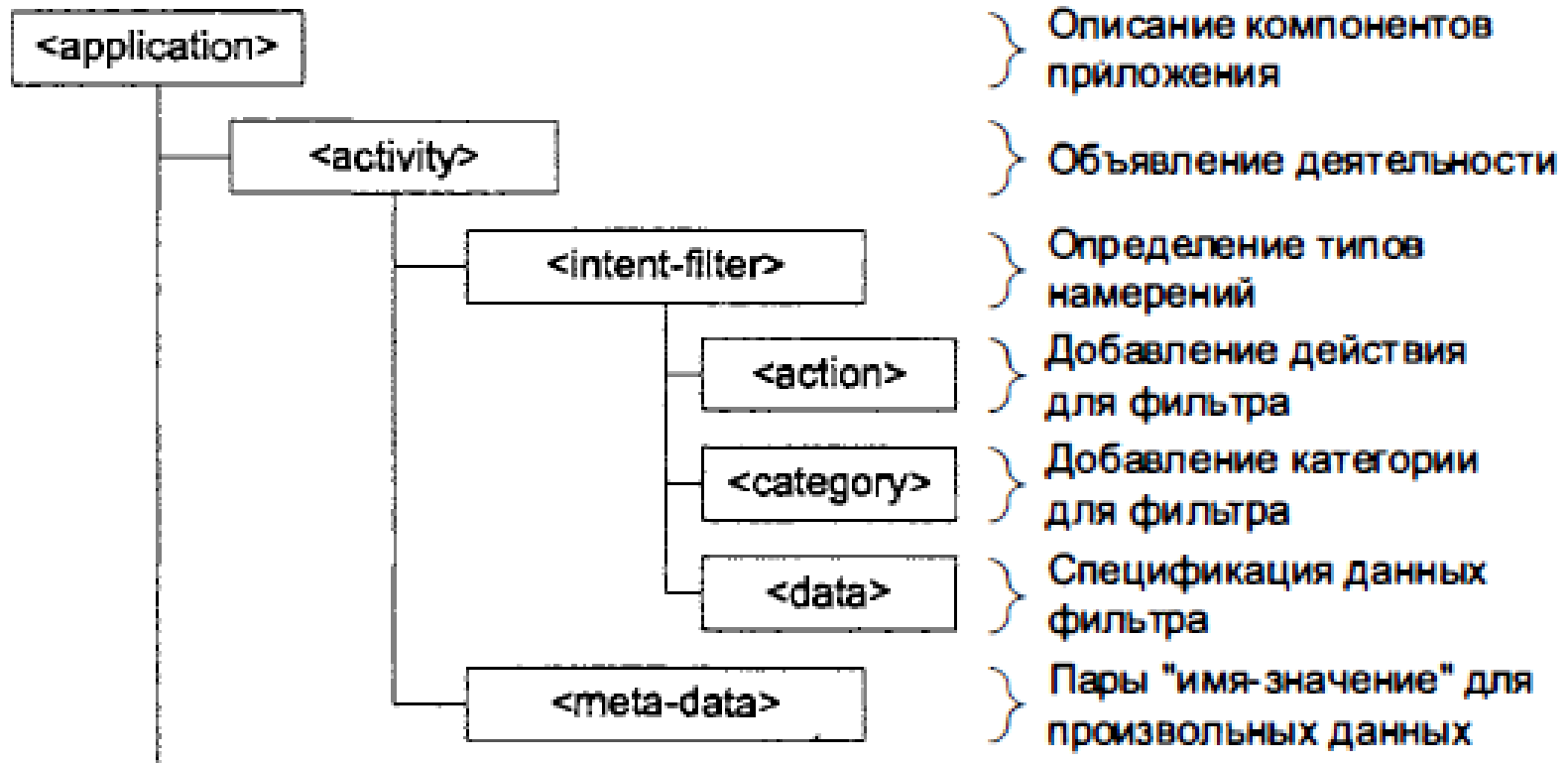
Файл AndroidManifest.xml



Структура элемента <application>



Структура элемента <application>



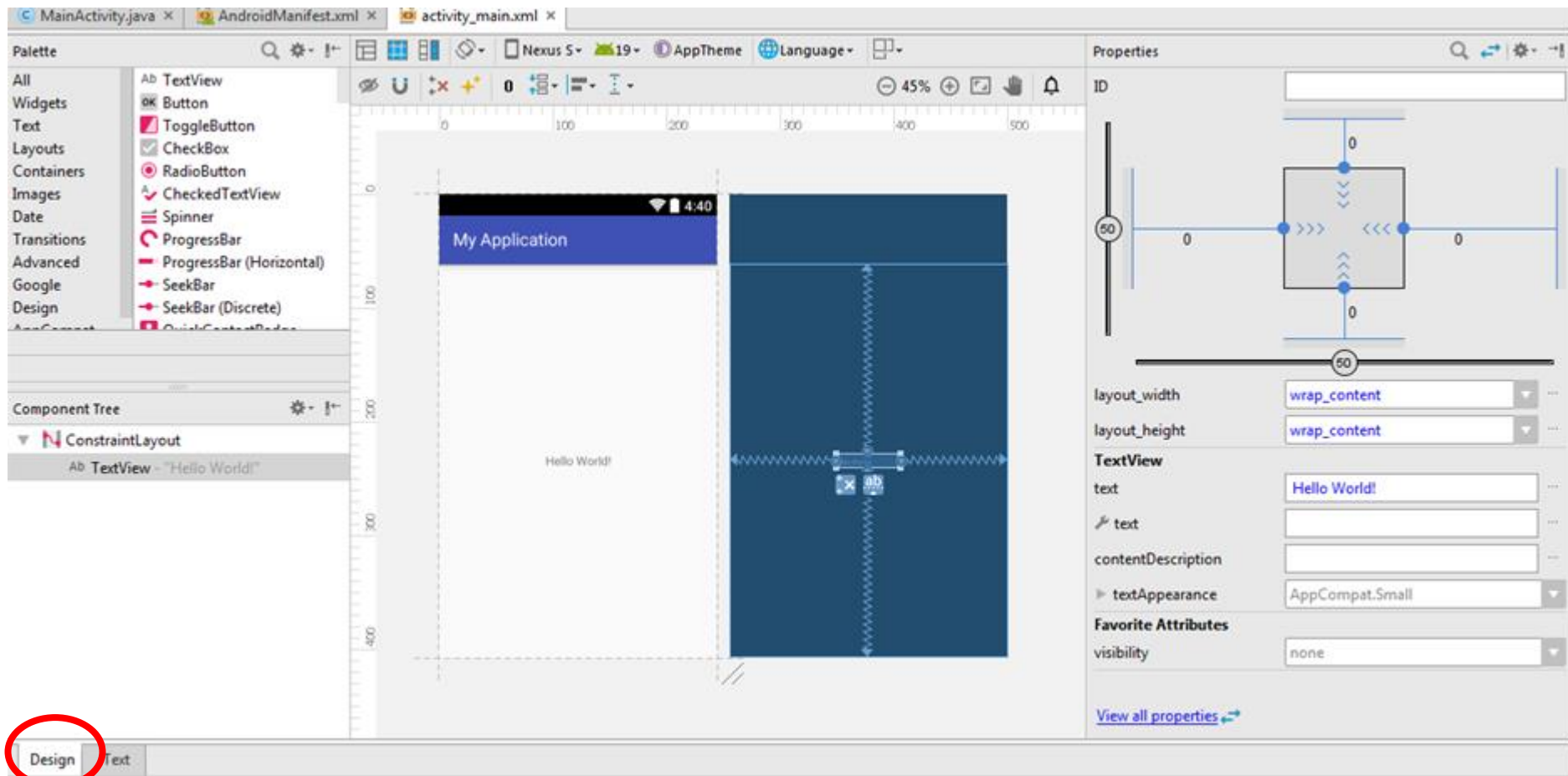
Содержимое созданного проекта

В результате описанного пошагового создания нового проекта мы получаем одно Activity, которое обозначено в файле манифеста как `launcher`, т.е. отображаемое при запуске нашего модуля.

Activity имеет:

1. графическое представление (файл **activite_main.xml** в папке `res/layout`),
 2. java-класс с указанием используемого графического представления (файл **MainActivity** в папке `java`). Логика работы Activity, основанная на обработке событий графического интерфейса, прописывается в java-классе.
-

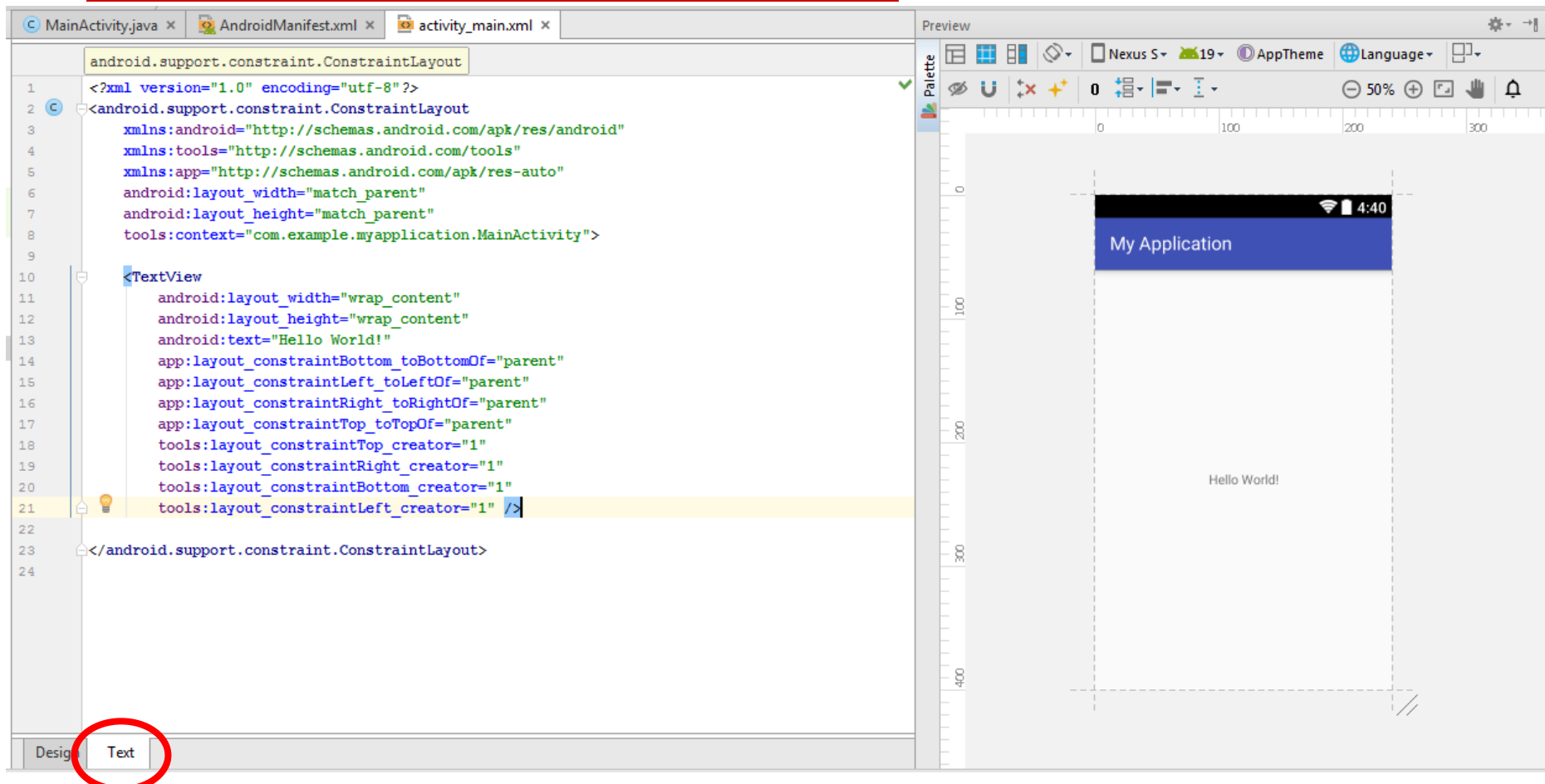
Графическое представление Activity: Layout: activite_main.xml, вкладка Design



Графическое представление Activity:

Layout: activite_main.xml,

вкладка Text



MainActivity.java



The screenshot shows an IDE window with three tabs: MainActivity.java, AndroidManifest.xml, and activity_main.xml. The MainActivity.java tab is active, displaying the following code:

```
1 package com.example.myapplication;
2
3 import ...
4
5
6 public class MainActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12     }
13 }
14
```

The code is formatted with syntax highlighting. The package name is in blue, the class name is in black, and the method name is in black. The code is enclosed in a yellow highlight box. The IDE interface includes a left margin with line numbers and a right margin with a lightbulb icon.

Запуск приложения

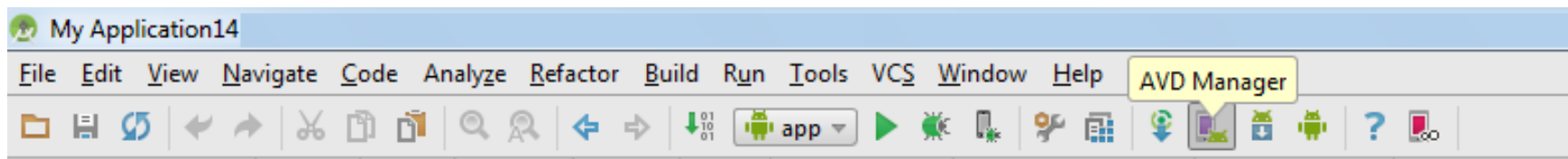
Необходимо определиться с устройством запуска.

Здесь возможны три варианта:

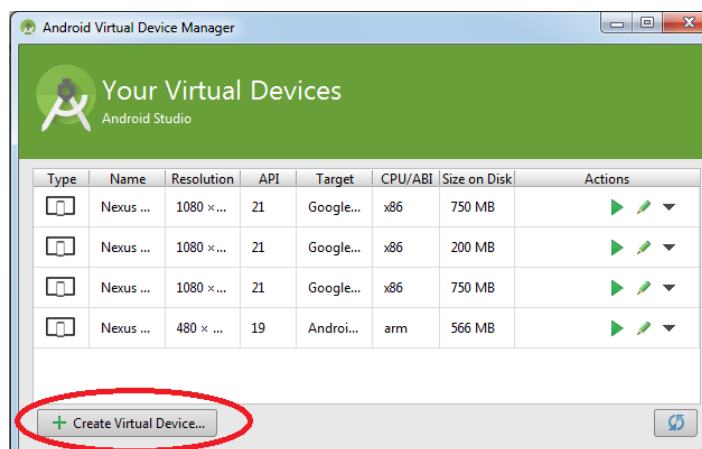
- эмулятор устройств Android Studio,
- виртуализация устройства посредством Genymotion,
- **подключение реального устройства.**

Запуск приложения: эмулятор Android Studio

Если выбрали первый вариант, то предварительно нужно создать устройство в AVD Manager.



Вы сможете создать только те устройства, для которых скачали базовые компоненты в SDK Manager!

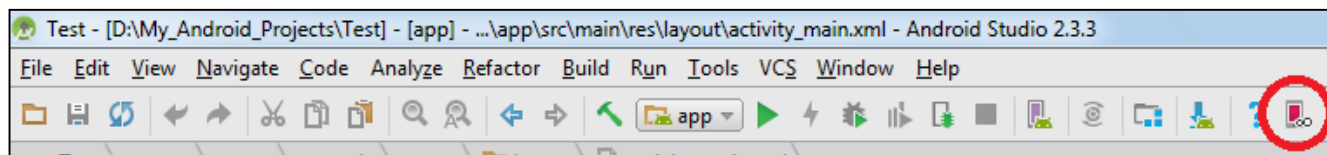


Далее запускаем модуль и запускаем эмулятор.

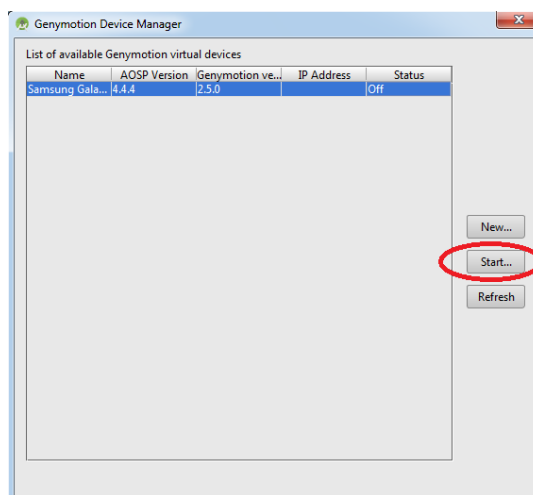


Запуск приложения: Genymotion

Если Вы выбрали вариант Genymotion, нажимаем на соответствующую иконку:



и запускаем созданное ранее устройство:



Далее запускаем модуль.



Запуск приложения: реальное устройство

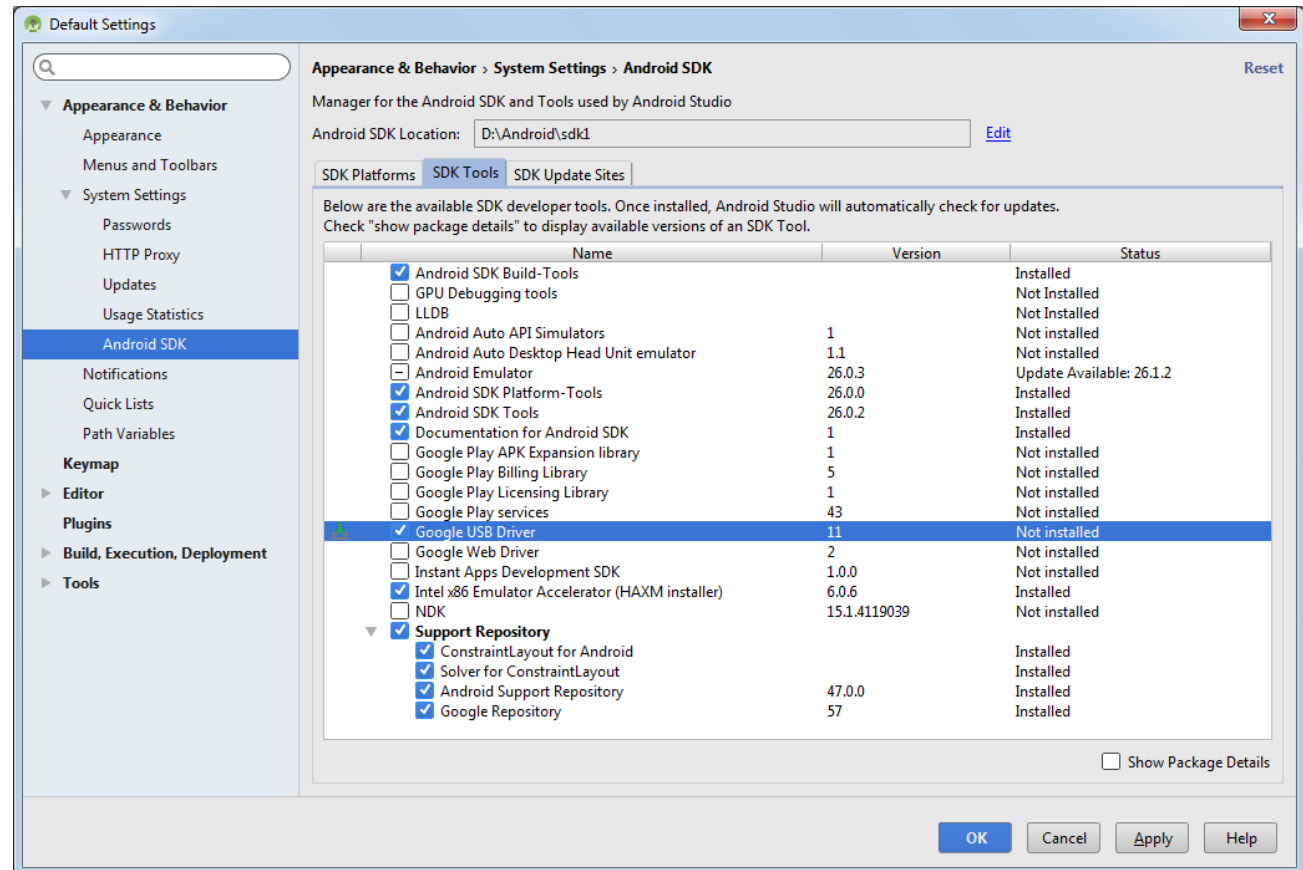
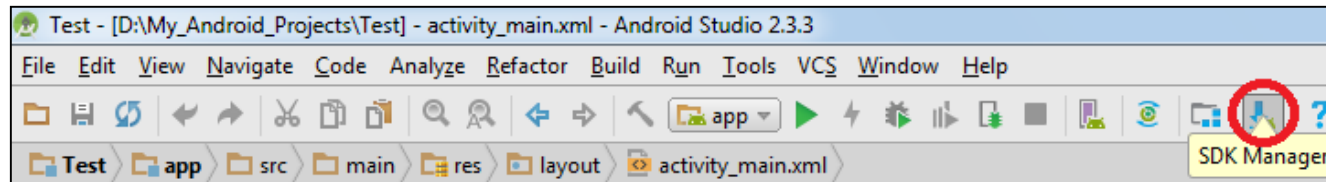
Для тестирования на реальном устройстве подключаем его к ПК и запускаем модуль.

При условии соблюдения всех настроек по работе с реальными устройствами Вы увидите на смартфоне свое приложение.

Необходимые настройки:

- В папке SDK Manager установлен компонент Google USB Driver.
- На Вашем устройстве входим в Настройки → Опции → Об устройстве. Нажимаем на Номер сборки несколько раз до появления сообщения «Режим разработчика включен». После этого у Вас появится пункт меню Параметры разработчика (Настройки → Опции). Устанавливаем здесь опцию Отладка USB.
- При необходимости установите на ПК драйвер для моб. устройств.

Запуск приложения: реальное устройство



Где посмотреть и при необходимости изменить минимальную версию SDK платформы

