

Лабораторная работа №1. Установка и настройка среды программирования Android Studio и создание простого приложения

Цель лабораторной работы: Установка и настройка среды программирования Android Studio и создание первого мобильного приложения

Задачи лабораторной работы:

- Установить и настроить среду программирования **Android Studio Hedgehog | 2023.1.1 (<https://developer.android.com/studio>)**.
 - В Ubuntu установить из snap-пакета в консоли или с помощью приложения Ubuntu Software.
 - Создать первое приложение.
 - Научиться запускать приложение на эмуляторе мобильного устройства.
 - Научиться запускать приложение на мобильном устройстве.
 - Изучить виды разметки, используемых для вёрстки интерфейса приложений.

Содержание

Введение.....	2
Установка и настройка Android Studio.....	2
Системные требования для Android Studio.....	2
Установка Android Studio.....	3
Установка JDK для Android Studio.....	8
Примеры установки.....	9
Изучить структуру приложения.....	9
Критерии оценивания.....	17
Содержание отчета.....	17
Настройка .gitignore для проекта в Android Studio.....	18
Задание 1. Создать простое мобильное приложение.....	19
Задание 2. Выполнить приложение на эмуляторе устройства.....	26
Задание 3. Выполнить приложение на устройстве.....	30
3.1 Настроить устройство.....	30
3.2 Настроить компьютер.....	31
3.3. Настроить среду Android Studio.....	36
Задание 4. Создание приложения с использованием стандартных макетов.....	37
Задание 5. Создание простого приложения с использованием Jetpack Compose.....	38
Контрольные вопросы.....	38
Литература.....	39

Введение

Лабораторная работа посвящена описанию работы в среде Android Studio. В работе рассказывается о программном обеспечении, которое необходимо скачать с официальных сайтов, установить и настроить, создании и запуске простейшего приложения на эмуляторе и мобильном устройстве. В связи с тем, что отладка на устройствах сопряжена с рядом трудностей, приводится подробная инструкция по настройке устройств и операционной системы Windows (версия не ниже 10).

Установка и настройка Android Studio

Большинство приложений для OS Android написано на Java. Одной из самых популярных сред разработки является Android Studio и Eclipse (для неё также необходим JDK) с установленным плагином ADT и Android SDK. В данном курсе работа с Eclipse не рассматривается. Основное внимание в курсе уделяется проектированию мобильных приложений в Android Studio.

Системные требования для Android Studio

Системные требования для ОС Windows:

- Microsoft® Windows® 8/10/11 (32- или 64-бит);
- 3 GB RAM минимум, 8 GB RAM рекомендуется; дополнительно 1 GB для Android Emulator;
- 2 GB свободного места на диске минимально, 4 GB рекомендуется (500 MB для IDE + 1.5 GB для Android SDK и образа системы для эмулятора);
- разрешение экрана: 1280 x 800;
- Для ускорения эмулятора требуется: процессор Intel® с поддержкой аппаратной виртуализации Intel® VT-x, Intel® EM64T (Intel® 64) и аппаратная поддержка защиты от вредоносных программ, например Execute Disable (XD) Bit для процессоров Intel.

Системные требования для ОС macOS

- macOS® 10.15 (Catallina) или выше;
- 3 GB RAM минимально, 8 GB RAM рекомендуется;
- + 1 GB для Android Emulator;
- 2 GB свободного места на диске минимально, 4 GB рекомендуется (500 MB для IDE + 1.5 GB для Android SDK и образов ОС для эмулятора);
- 1280 x 800 минимальное разрешение экрана.

Системные требования для ОС Linux

- GNOME или KDE;
- Ubuntu® 22.04 LTS (Jammy Jellyfish) или выше;
- GNU C Library (glibc) 2.35 или выше;
- 3 GB RAM минимально, 8 GB RAM рекомендуется;
- + 1 GB для Android Emulator;
- 2 GB свободного места на диске минимально, 4 GB рекомендуется (500 MB для IDE + 1.5 GB для Android SDK и образов ОС эмулятора);
- 1280 x 800 минимальное разрешение экрана;

- Для ускорения эмулятора: процессор Intel® с поддержкой аппаратной виртуализации Intel® VT-x, Intel® EM64T (Intel® 64), и аппаратной защитой от вирусов, например Execute Disable (XD) Bit для процессоров Intel, или процессор AMD с поддержкой виртуализации AMD Virtualization™ (AMD-V™).

Установка Android Studio

Скачать среду можно с сайта для разработчиков Android (<http://developer.android.com/>).

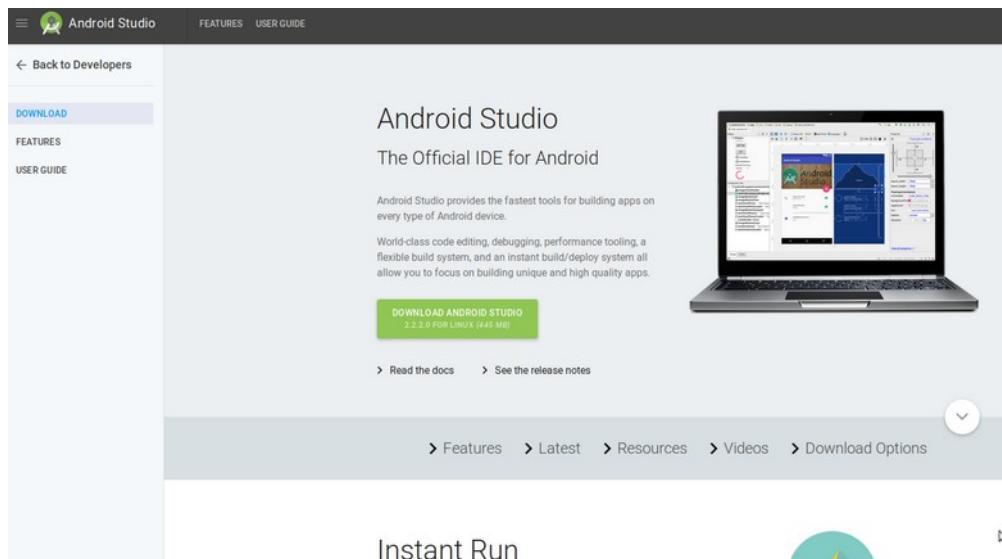


Рис. 1.1. Сайт разработчика

Для того, чтобы скачать среду необходимо принять условия лицензионного соглашения и выбрать вашу версию Windows (32-bit или 64-bit).

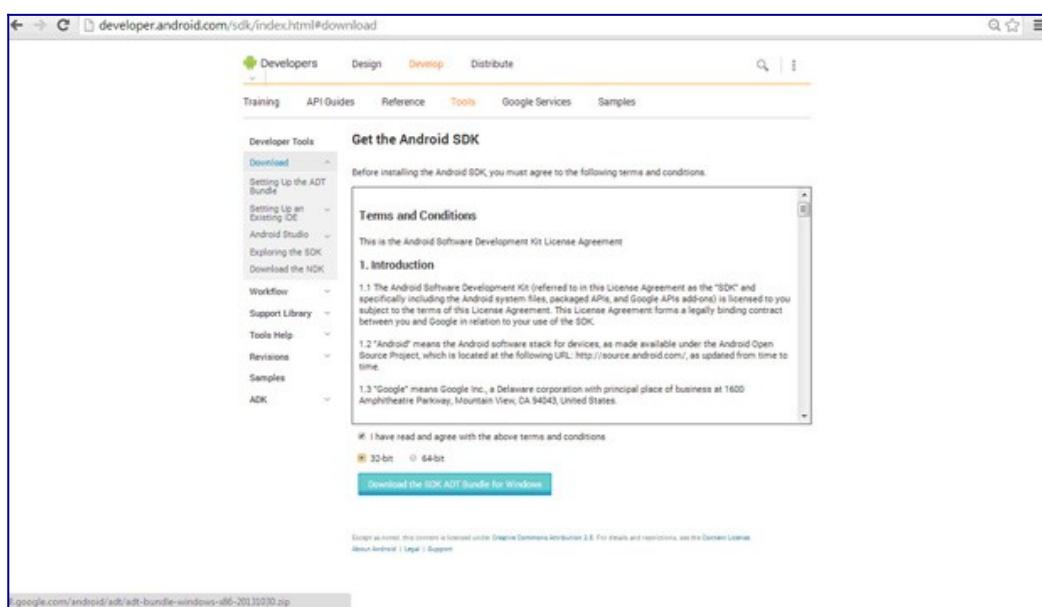


Рис. 1.2. Скачивание среды

После скачивания среды можем перейти установке Android Studio. Ничего необычного в ней не будет — обычный диалог инсталлятора. В процессе нужно будет ответить лишь на один важный вопрос, и то это опционально.

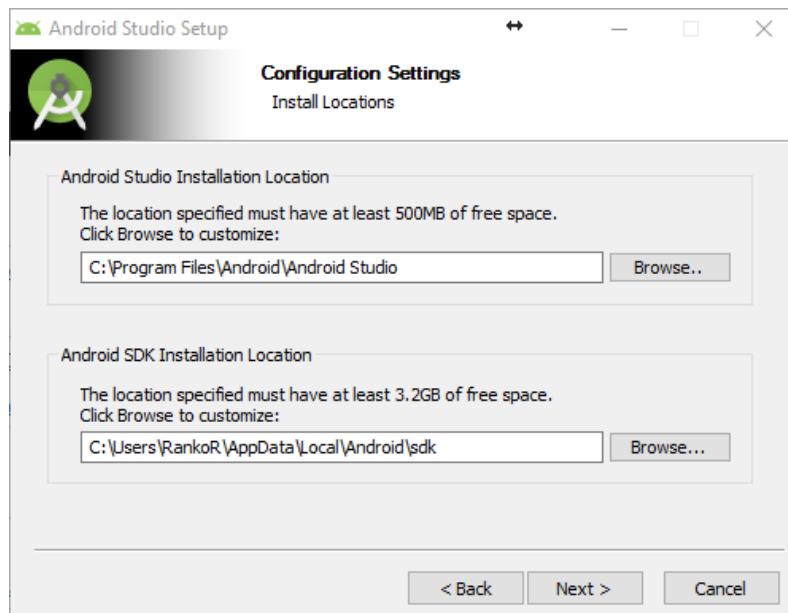


Рис. 1.6. Установка Android Studio

Здесь, как следует из скриншота, установщик спрашивает, куда ставить студию, и куда ставить SDK. Если с самой студией все понятно, то SDK нужно быть внимательным.

Пожалуйста, обратите внимание, что **имя пользователя должно быть указано латиницей и желательно в нижнем регистре**, чтобы не возникало проблем с компиляцией. Так же Android SDK рекомендуется установить не в папке профиля пользователя, а вынести отдельно так, чтобы папка была доступна в случае обновлений, смене версий или для использования разными пользователями системы.

Как опять же следует из скриншота, для установки SDK нужно **минимум 3.2 GB** места на диске. Это минимум, на самом деле, места нужно больше, поскольку через какое-то время вам необходимо будет загружать обновленный SDK. так что, если вы не уверены, что места в будущем хватит — лучше измените местоположение на более вместительный диск.

После этого понадобится стандартно несколько раз нажать на кнопочку «далее», и на этом установка Android Studio завершена.

Далее вам необходимо выбрать (или создать новое) рабочее пространство, т.е. место, где будут находиться ваши проекты. Если поставить галочку, то это рабочее пространство будет выбираться по умолчанию, а противном случае это окно будет появляться при каждом запуске Android Studio.

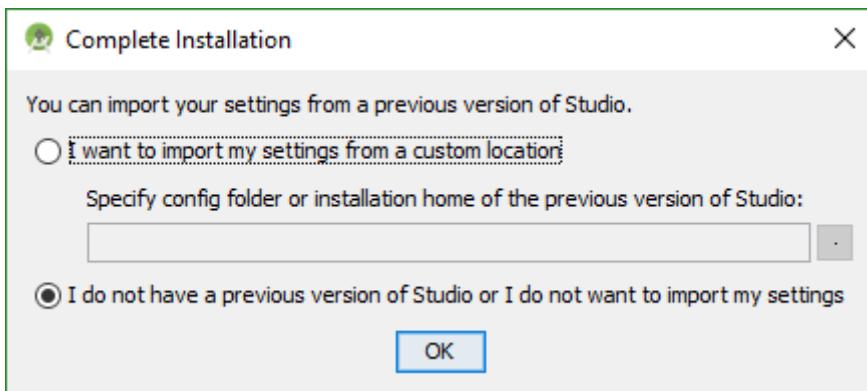


Рис. 1.7. Выбор рабочего пространства

Если Вы скачали сборку Android Studio, в которую не входит Android SDK, то в процессе установки Android SDK будет загружена инсталлятором.

Для разработки и тестирования приложений нам понадобятся SDK-платформы Android. В открывшемся стартовом окне Android Studio жмите пункт **Configure** и далее **SDK Manager**.

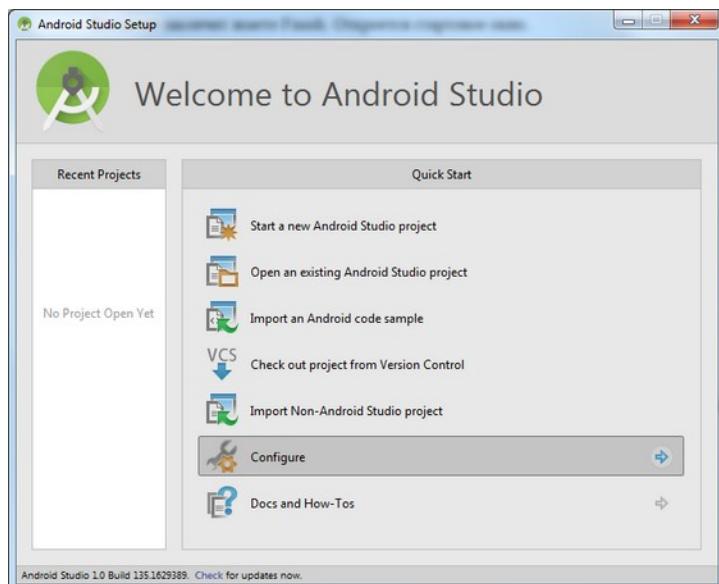
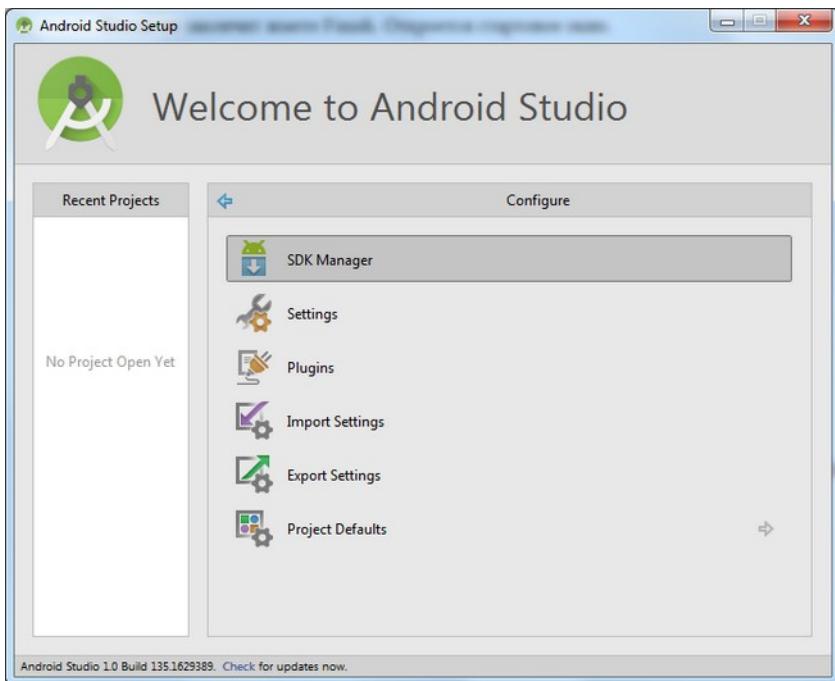
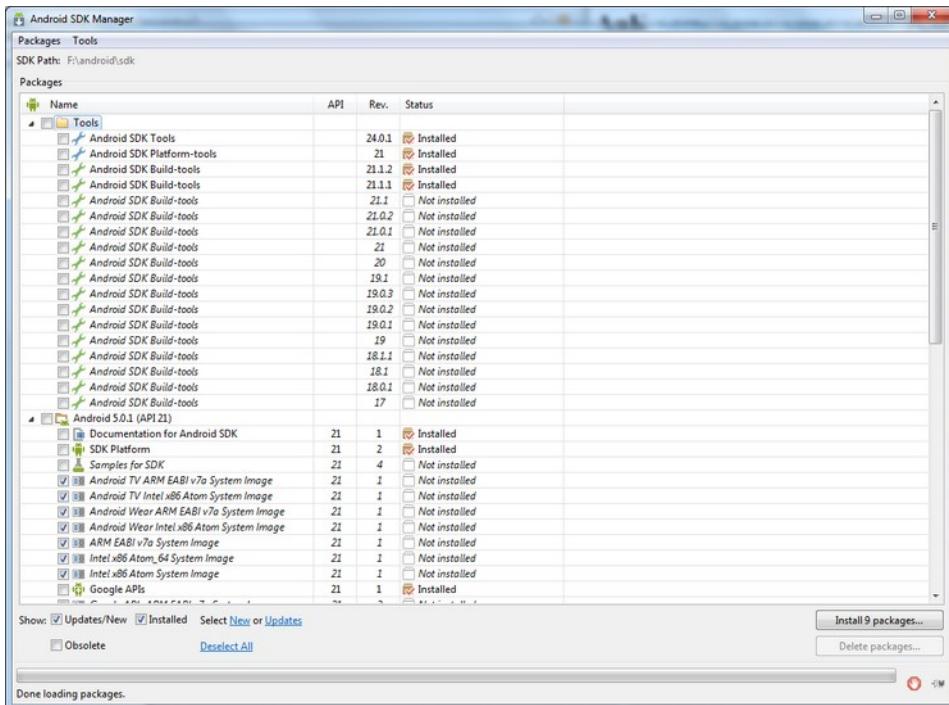


Рис. 1.8. Настройка Android SDK Manager



Открылся SDK Manager



Здесь нам показывают SDK-компоненты, которые мы можем скачать, обновить или удалить.

Сначала идет папка **Tools** — в ней находятся утилиты, необходимые для разработки под Android. Далее идет список версий Android. Далее указана папка **Extras**, в которой обычно находятся дополнительные библиотеки.

Справа от каждого компонента в списке виден его статус: **Installed** — установлен, **Not installed** — не установлен, **Update available** — доступно обновление.

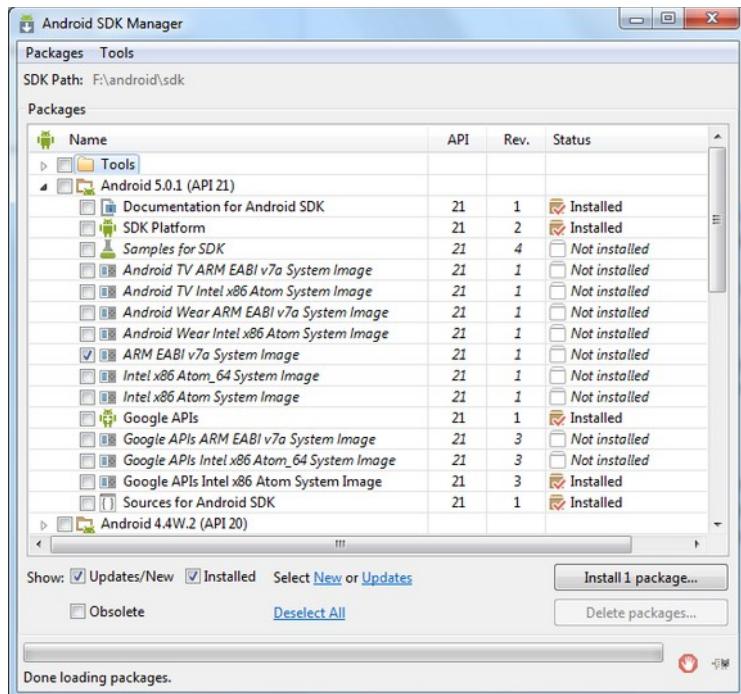
Нас сейчас интересуют папки с именами версий Android. На текущий момент последняя доступная версия — Android 14.0 (API 34). Установите версию Android (API) и версию платформы, используемую на Вашем телефоне. Далее по тексту рекомендации даются для Android 9.0.

Как минимум для разработки нам необходимы два компонента в этой папке:

1) **SDK Platform** — используется для разработки. Говоря простыми словами, здесь содержатся все программные компоненты системы Android, которые мы будем использовать при создании приложений — т.е. окна, кнопки и т.п.

2) **ARM EABI v7a System Image** — образ Android системы. Используется для создания эмулятора Android, который нужен будет для тестирования приложений прямо на компе, без подключения реальных устройств.

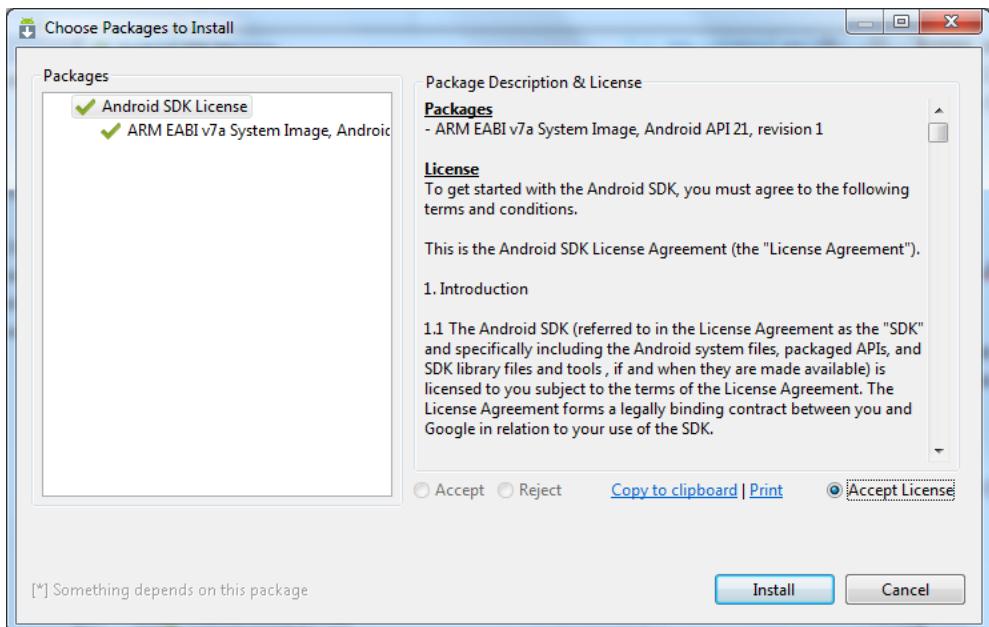
Т.е. мы сможем создать приложение, использующее компоненты и возможности Android версии 9.0 и запустить это приложение на эмуляторе версии 9.0. Этого вполне достаточно для начала. Выбираем эти пункты в папке Android 9.0 (API 28).



Если вам вдруг понадобятся другие версии Android, заходите в SDK Manager и устанавливайте эту пару компонентов для нужной вам версии.

Снизу справа жмем кнопку **Install <число> packages**, чтобы он установил все компоненты, которые мы выделили. Учтите, что платформы могут весить несколько сотен мегабайт.

Появится окно со списком установки (у вас может быть другое содержимое).



Отмечаем в нем **Accept license** и нажимаем **Install**. После завершения установки откроется окно создания проекта.

По документу <https://developer.android.com/studio/intro/studio-config> проверить настройки окружения для Android Studio и привести в соответствие, если настройки не оптимизированы.

Установка JDK для Android Studio

Для Android Studio рекомендуется использовать OpenJDK (<https://developer.android.com/studio/intro/studio-config#jdk>). Копия последней версии OpenJDK поставляется в комплекте с Android Studio 2.2 и выше, и Google рекомендует использовать эту версию JDK для ваших проектов Android. Чтобы использовать пакетный JDK, выполните следующее:

Откройте проект в Android Studio и выберите **Файл > Настройки... > Сборка, выполнение, развертывание > Инструменты сборки > Gradle** (Android Studio > Настройки... > Сборка, выполнение, развертывание > Инструменты сборки > Gradle на Mac) или **File > Settings... > Build, Execution, Deployment > Build Tools > Gradle** (Android Studio > Preferences... > Build, Execution, Deployment > Build Tools > Gradle a Mac).

В Gradle JDK выберите параметр Embedded JDK.

Нажмите «OK».

По умолчанию версия языка Java, используемая для компиляции вашего проекта, основана на compileSdkVersion вашего проекта (поскольку разные версии Android поддерживают разные версии Java). При необходимости вы можете переопределить эту версию Java по умолчанию, добавив следующий блок compileOptions в файл build.gradle:

```
android {
```

```
compileOptions {  
    sourceCompatibility JavaVersion.VERSION_1_6  
    targetCompatibility JavaVersion.VERSION_1_6  
}  
}  
Дополнительные сведения о настройке JDK см. по ссылке
```

<https://developer.android.com/studio/intro/studio-config#jdk>.

Примеры установки

1. <https://developer.android.com/studio/install.html>
2. <https://www.c-sharpcorner.com/article/how-to-download-and-install-android-studio-in-windows-10/>
3. <http://www steptoinstall com/how-to-install-android-studio-on-windows-step-by-step.html>
4. <https://www.onlinetutorialspoint.com/android/how-to-install-android-studio-on-windows-10.html>
5. <https://javarush.ru/quests/lectures/questgoogleandroid.level03.lecture07>
6. <https://askubuntu.com/questions/634082/how-to-install-android-studio-on-ubuntu>

Изучить структуру приложения

Создайте новый проект в среде Android Studio. Процесс создания нового проекта и описание основных настроек подробно рассмотрен в лабораторной работе к первой лекции.

В процессе создания проекта, мы назвали его ProjectN, среда разработки подготавливает необходимые папки и файлы. Полный иерархический список обязательных элементов проекта можно увидеть на вкладке Package Explorer (аналогичную информацию предоставляет вкладка Project Explorer), иерархия полученных папок и файлов для нашего проекта изображена на рис. 1.9.

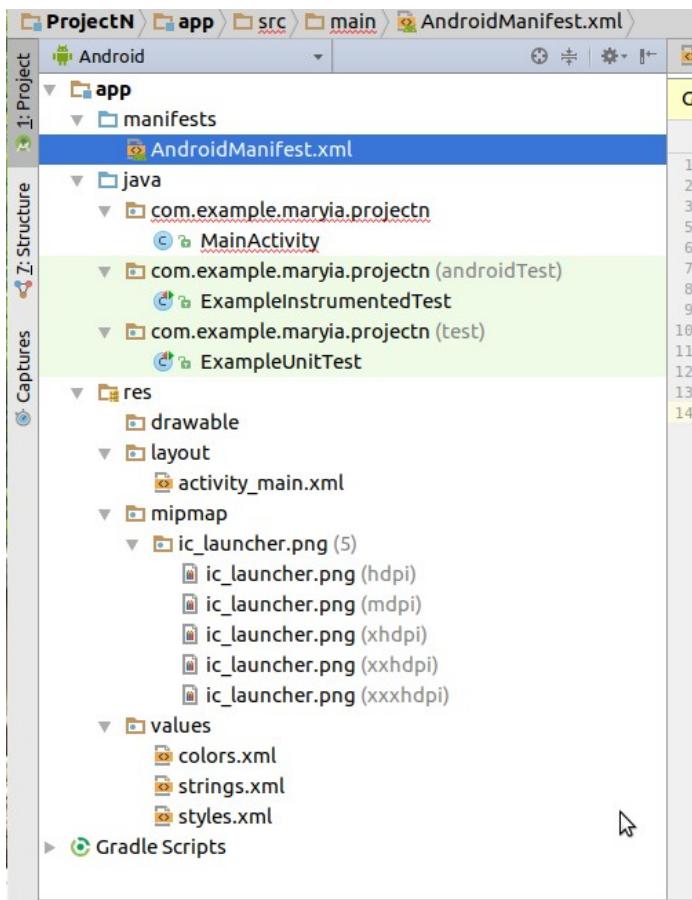


Рис. 1.9. Структура проекта ProjectN

По умолчанию Android Studio отображает проект в представлении Android (Android view). Данное представление не отображает существующую файловую иерархию на диске, но организовано по модулям и типам файлов, чтобы упростить навигацию между ключевыми исходными файлами проекта. Скрывая для этого определённые виды файлов и каталогов, которые не часто используются. Ряд структурных элементов проекта, которые соответствуют файловой структуре на диске включают следующие:

- Вывод всех конфигурационных файлов для сборки в верхнеуровневой группе **Gradle Script**.
 - Вывод всех файлов манифестов для каждого модуля на уровне группы модуля. (для тех случаев, когда создаются отдельные файлы манифестов для сборки и платформ).
 - Вывод всех альтернативных файлов ресурсов в отдельной группе вместо разделения на каталоги в зависимости от типа ресурса. Например, все иконки разных размеров находятся в одной папке.

Каждый модуль приложения Android включает следующие группы файлов:

- **manifests** — содержит файл `AndroidManifest.xml`.
 - **java** — содержит исходные файлы Java, разделенные по пакетам, включа код тестов JUnit.

- **res** — содержит все ресурсы, такие как xml разметка, строки UI, изображения, размещаемые в соответствующих подкаталогах:
 - **layout** - в данной папке содержатся xml-файлы, которые описывают внешний вид форм и их элементов, пока там находится только activity_main.xml;
 - **values** - содержит XML файлы, которые определяют простые значения, таких ресурсов как, строки, числа, цвета, темы, стили, которые можно использовать в данном проекте.

Чтобы просмотреть файловую структуру проекта, включая все скрытые файлы в представлении Android (Android View), необходимо переключиться в представление **Project** в выпадающем меню окна **Project** (см. рис. 1.10.).

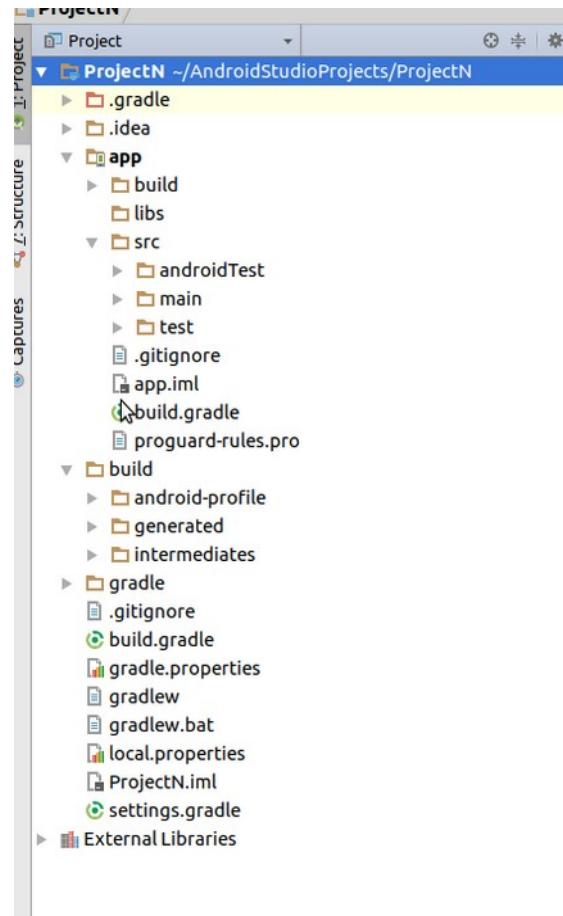


Рис. 1.10. Представление Project

В представлении Project (Project view) можем увидеть больше файлов и каталогов, наиболее важные из них:

module-name/

build/ — содержит сгенерированные файлы сборки.

libs/ — содержит библиотеки

src/ — содержит код и файлы ресурсов модуля, распределенные по подкаталогам.

androidTest/ — содержит код тестов, которые выполняются на устройстве Android.

main/ — содержит исходный код и ресурсы. Именно в этой папке размещаются все классы, создаваемые в процессе разработки приложения. Сейчас в этой папке в пакете com.example.projectn размещается единственный класс MainActivity.java. Этот класс определяет главную и единственную активность в этом приложении.

Комментарий 1: Имя пакету присваивается в процессе создания приложения в поле **Package Name**, использовать com.example не рекомендуется, т. к. пакет с таким именем нельзя загрузить в Google Play. Часто рекомендуют использовать в качестве имени пакета название сайта программиста, записанное в обратном порядке, можно просто использовать свои имя и фамилию. Последнее слово в имени пакета формируется автоматически и совпадает с именем проекта.

Комментарий 2: Имя файлу присваивается в процессе создания приложения на этапе настройки активности. Имя определяется в поле **Activity Name**.

AndroidManifest.xml — описывает структуру приложения и каждого из его компонентов.

java/ — исходный код на java

jni/ — код, используемый Java Native Interface (JNI). Подробнее см. документацию по Android NDK.

gen/ — содержит файлы java, сгенерированные Android Studio, такие как R.java файл и интерфейсы, созданные из файлов AIDL.

res/ — содержит ресурсы приложение, такие как изображения, файлы разметки, файлы локализации и др.

assets/ — содержит файлы, которые при компиляции включаются в apk файл как есть. Просматривать данную директорию можно по URI и читать файлы как поток байтов, используя AssetManager. Например, в данной папке рекомендуется размещать текстуры и данные для игр.

test/ — содержит код для локальных тестов, исполняемых на хостовой JVM.

build.gradle (module) — определяет конфигурации сборок определенного модуля.

build.gradle (project) — определяет конфигурацию, применяемую ко всем модулям. Этот файл основной для проекта, так как его можно использовать для ревизии всего кода проекта

Рассмотрим файл **AndroidManifest.xml** - файл в формате xml, который описывает основные свойства проекта, разрешение на использование ресурсов устройства и др. Сразу после создания приложения файл AndroidManifest.xml выглядит так, как показано на рис. 1.11. Это файл манифеста проекта.

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ProjectN"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Рис. 1.11. Файл AndroidManifest.xml только созданного проекта

Рассмотрим подробно файл манифеста. В сгенерированном файле apk может содержаться только один файл AndroidManifest.xml, но проект Android Studio может содержать несколько файлов манифестов: файл манифеста проекта, манифест сборки и манифести импортированных библиотек. Поэтому при создании приложения Gradle build объединяет все файлы манифеста в один файл манифеста, который упаковывается в файл apk приложения.

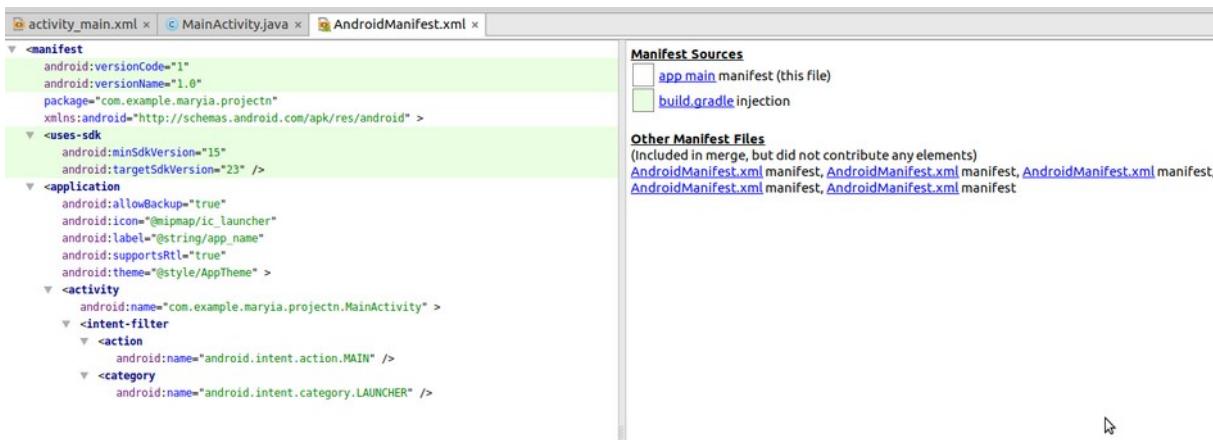


Рис. 1.12. Объединённый файл манифеста созданного проекта

В Android Studio используется инструмент слияния манифестов, который объединяет все элементы XML из каждого файла, следуя правилам слияния. В результате слияния получается файл манифеста, содержащий все требуемые элементы (см. рис. 1.12).

Первый обязательный элемент <manifest> является корневым элементом файла, должен содержать обязательный элемент <application> и все остальные элементы по необходимости. Рассмотрим основные атрибуты этого элемента:

xmlns:android	- определяет пространство имен Android, всегда должен иметь значение: "http://schemas.android.com/apk/res/android". Обязательный атрибут.
package	- полное имя пакета, в котором располагается приложение. Обязательный атрибут. Имя должно быть уникальным, может содержать заглавные и строчные латинские буквы, числа и символ подчёркивания. Однако начинаться должно только с буквы. Для избежания конфликтов с другими разработчиками рекомендуется использовать имя вашего сайта (если он есть) записанное в обратном порядке. В нашем случае пакет имеет имя "com.example.projectn" и наше приложение не удастся разместить в Google Play (но мы на это и не претендуем). Внимание: если Вы опубликовали свое приложение, Вы не можете менять имя пакета , т.к. имя пакета служит уникальным идентификатором для приложения и в случае его смены приложение будет рассматриваться, как совсем

	другое и пользователи предыдущей версии не смогут его обновить.
android:versionCode	- внутренний номер версии приложения не виден пользователю. Этот номер используется только для определения является ли одна версия более современной по сравнению с другой, больший номер показывает более позднюю версию.
android:versionNumber	- номер версии, является строкой и используется только для того, чтобы показать пользователю номер версии приложения.
android:shareUserID, android:sharedUserLabel, android:installLocation	- эти атрибуты в нашем файле манифеста не представлены, про их назначение можно почитать по ссылке: http://developer.android.com/guide/topics/manifest/manifest-element.html .

Рассмотрим элемент `<uses-sdk>`, который показывает совместимость приложения с версиями Android. Основные атрибуты:

android:minSdkVersion	- указывает значение минимального уровня API, необходимого для работы приложения. Система Android не позволит установить приложение, если уровень API ниже, чем уровень, указанный в этом атрибуте. Внимание: если этот атрибут не указан, система установит значение по умолчанию равным "1", которое означает, что приложение совместимо со всеми версиями Android. И в случае, если приложение не совместимо со всеми версиями, установка пройдет на любую версию Android, а во время работы приложение сломается, когда попытается обратиться к недоступным элементам API. Поэтому необходимо всегда указывать значение этого атрибута.
android:targetSdkVersion	- указывает уровень API целевой платформы Android приложения, если этот атрибут пропущен, по умолчанию принимается значение android:minSdkVersion.
android:maxSdkVersion	- указывает максимальное значение уровня API, под который разрабатывалось приложение. Если значение этого атрибута ниже, чем уровень API соответствующий версии Android, на которую устанавливается приложение, то система не позволит установить такое приложение.

Если внимательно посмотреть на файл манифеста рис. 1.11 и 1.12, можно заметить, что данный атрибут в нем отсутствует. На самом деле разработчикам не рекомендуются задавать значение этого атрибута. Во-первых, это значение будет препятствовать использованию приложений на новых версиях Android при их появлении, несмотря на то, что все новые версии полностью обратно-совместимы. Во-вторых, стоит иметь ввиду, что задание этого атрибута может привести к тому, что

приложение будет удалено с устройства после обновления Android до более высокого уровня API.

Подробнее с элементом <uses-sdk> и его атрибутами можно ознакомиться по ссылке: <http://developer.android.com/guide/topics/manifest/uses-sdk-element.html>.

Рассмотрим элемент <application>, который является обязательным элементом манифеста, полностью определяет состав приложения. Представляет собой контейнер для элементов <activity>, <service>, <receiver>, <provider> (и не только), каждый из которых определяет соответствующий компонент приложения. Содержит набор атрибутов, действие которых распространяется на все компоненты приложения. Рассмотрим атрибуты элемента <application>, представленные в манифесте на рис. 1.11, 1.12:

android:allowBackup	- определяет разрешение для приложения участвовать в резервном копировании и восстановлении. Если значение этого атрибута false, то для приложения никогда не может быть создана резервная копия, даже если проводится резервное копирование всей системы целиком. По умолчанию значение этого атрибута равно true.
android:icon	- определяет иконку для приложения целиком, а также иконку по умолчанию для компонентов приложения, которая может быть переопределена атрибутом android:icon каждого компонента. Задается как ссылка на графический ресурс, содержащий изображение, в нашем случае значение этого атрибута равно "@drawable/ic_launcher".
android:label	- определяет видимый для пользователя заголовок приложения целиком, а также заголовок по умолчанию для компонентов приложения, который может быть переопределен атрибутом android:label каждого компонента. Задается как ссылка на строковый ресурс, в нашем случае значение атрибута равно "@string/app_name".
android:theme	- определяет тему по умолчанию для всех активностей приложения, может быть переопределен атрибутом android:theme каждой активности. Задается как ссылка на стилевой ресурс, в нашем случае значение атрибута равно "@style/AppTheme".

На самом деле у элемента <application> гораздо больше атрибутов, чем нам удалось рассмотреть, найти полный список атрибутов с описаниями можно по ссылке: <http://developer.android.com/guide/topics/manifest/application-element.html>.

В приложении всего одна активность, других компонентов нет, в связи с этим элемент <application> в манифесте содержит ровно один элемент <activity> и больше никаких других элементов не содержит. Рассмотрим элемент <activity>, который определяет активность. Для каждой активности обязательно необходим свой элемент <activity> в манифесте. Рассмотрим атрибуты элемента <activity>, представленные в манифесте на рис. 1.11:

android:name	- определяет имя класса, который задает активность. Значение атрибута должно полностью определять имя класса с указанием пакета, в котором располагается класс. В нашем случае атрибут имеет значение: "com.example.projectn.MainActivity". Можно использовать сокращенную запись ".MainActivity", в этом случае добавляется имя пакета, определенное соответствующим атрибутом элемента <manifest>.
android:configChanges	- перечисляет изменения конфигурации, которыми может управлять активность. Если конфигурация меняется во время работы, то по умолчанию активность останавливается и перезапускается. Если же изменение конфигурации указано в этом атрибуте, то при появлении этого изменения активность не перезапускается, вместо этого она продолжает работать и вызывает метод onConfigurationChanged(). В нашем случае атрибут имеет значение "orientation keyboardHidden screenSize", т. е. при смене ориентации экрана, смене размера экрана и изменении доступности клавиатуры не произойдет перезапуск активности.
android:label	- определяет видимый пользователю заголовок активности, если он отличается от общего заголовка приложения. Задается как ссылка на строковый ресурс, в нашем случае значение атрибута равно "@string/app_name"(т.е. можно было и не задавать).
android:theme	- определяет тему активности, если она отличается от общей темы приложения, заданной соответствующим атрибутом элемента <application>. Задается как ссылка на стилевой ресурс, в нашем случае значение атрибута равно "@style/FullscreenTheme".

На самом деле у элемента <activity> гораздо больше атрибутов, чем нам удалось рассмотреть, найти полный список атрибутов с описаниями можно по ссылке: <http://developer.android.com/guide/topics/manifest/application-element.html>.

В манифесте для нашего приложения элемент <activity> содержит ровно один элемент: <intent-filter>, определяющий типы намерений, которые может принимать активность. Этот элемент содержит два элемента: <action> и <category>.

Первый элемент определяет действия, которые проходят в фильтр намерений, при этом <intent-filter> должен содержать хотя бы один элемент <action>, в противном случае ни один объект-намерение не сможет пройти через фильтр и активность не возможно будет запустить. Элемент <action> имеет единственный атрибут android:name="android.intent.action.MAIN".

Второй элемент определяет имя категории в фильтре намерений. Имеет единственный атрибут android:name="android.intent.category.LAUNCHER".

На этом разбор манифеста приложения закончим, подробно с описанием всех элементов этого файла можно познакомиться по ссылке: <http://developer.android.com/guide/topics/manifest/manifest-intro.html>.

Чаще всего, при создании приложения приходится иметь дело с папками **src**, **res/layout** и **res/values**, т.к. там находятся основные файлы проекта.

Критерии оценивания

Оценка 4

Выполнены задания 1-3. Представлен код проекта из задания 1 в репозитории. Лабораторная работа сдана с задержкой в 1 неделю.

Оценка 5-6

Выполнены задания 1-3 и проект на java из задания 4. Представлен отчет и исходный код проектов в репозитории, ответы на контрольные вопросы. Репозиторий оформлен согласно требованиям. Лабораторная работа сдана с задержкой в 1 неделю.

Оценка 7-8

Выполнены задания 1-4. Представлен отчет, исходный код проектов в репозитории, ответы на контрольные вопросы. Репозиторий оформлен согласно требованиям. Лабораторная работа сдана с задержкой в 1 неделю.

Оценка 9

Выполнены задания 1-5. Представлен отчет, ответы на контрольные вопросы и исходный код проектов в репозитории. Репозиторий оформлен согласно требованиям. Лабораторная работа сдана в срок.

Содержание отчета

1. Цель работы.
2. Скриншоты приложений.
3. Ответы на контрольные вопросы.

Отчет должен быть опубликован в git-репозитории на github. Все результаты лабораторной работы должны быть опубликованы в git-репозитории, ссылка на который доступна в курсе «Программирование мобильных и встраиваемых систем»:

В файле Readme проекта на github должна быть ссылка на отчёт. Отчет опубликовать во внешнем хранилище или в репозитории в каталоге /docs. Если в лабораторной работе необходимо написать программу/ы, то отчёт должен результаты тестов по каждой программе и ответы на контрольные вопросы.

Пример оформления файла Readme может быть таким:

Overview

Report on LabRabota1.

Projects

// Описать разработанные проекты.

How to build

// Рекомендации по сборке проекта/ов, включая требования к версии Android ОС.

Author

Your name and group number.

Additional Notes

//дополнительные комментарии.

Каждая лабораторная работа содержит тексты задач и контрольные вопросы, ответы на которые проверяются преподавателем при приёме работы у студента.

Выполнение студентом лабораторной работы и сдача её результатов преподавателю происходит следующим образом:

1. Студент выполняет разработку программ.
2. В ходе разработки студент обязан следовать указаниям к данной задаче (в случае их наличия). Исходные тексты программ следует разрабатывать в соответствии с требованиями к оформлению, приведёнными в приложении.
3. Студент выполняет самостоятельную проверку исходного текста каждой разработанной программы и правильности её работы, а также свои знания по теме лабораторной работы.

Настройка .gitignore для проекта в Android Studio

! Каждый проект в репозитории должен содержать файл .gitignore.

Пример .gitignore представлен ниже. Еще один пример .gitignore – [github Android.gitignore example](#).

```
# built application files
*.apk
*.ap_
*.aab

# Mac files
.DS_Store

# files for the dex VM
*.dex

# Java class files
*.class
```

```
# generated files
bin/
gen/

# Ignore gradle files
.gradle/
build/

# Local configuration file (sdk path, etc)
local.properties

# Proguard folder generated by Eclipse
proguard/
proguard-project.txt

# Eclipse files
.project
.classpath
.settings/

# Android Studio/IDEA
*.iml
.idea

# External native build folder generated in Android Studio 2.2 and later
.externalNativeBuild

# Google Services (e.g. APIs or Firebase)
google-services.json

# Android Studio captures folder
captures/

# Others
.cxx
```

Задание 1. Создать простое мобильное приложение

Изучить пример, представленный в задании ниже, и материалы из статей:

<https://cig-rdlab.gitbook.io/android/lectures/lecture-1/3.-struktura-android-proekta>

<https://cig-rdlab.gitbook.io/android/lectures/lecture-1/simple-app>

и реализовать простое мобильное приложение на языке Java или Kotlin на выбор. Интерфейс приложения состоит из кнопки **Открыть** и скрытого textView. При нажатии на кнопку выводится Ваше имя и фамилия, номер группы и название проекта. У приложения должна быть своя иконка, отличная от стандартной.

Добавить в репозиторий .gitignore (см. Настройка .gitignore для проекта в Android Studio). Опубликовать приложение в репозиторий в

ветке **project1** в каталоге, например **lab1-project1_ваша-фамилия**, заменив **ваша-фамилия** на Вашу фамилию.

Итак, наконец, мы подошли к самому главному — созданию проекта. Чтобы создать проект, зайдите в меню **File->New->Android Application Project**.

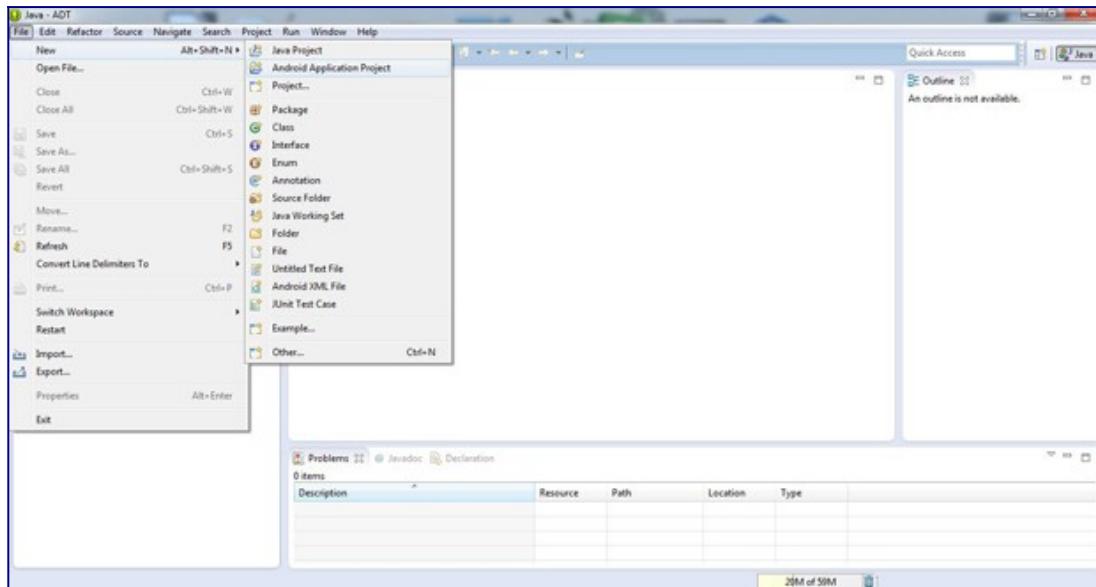


Рис. 1.13. Создание проекта

В появившемся окне обязательно нужно прописать имя приложения, имя проекта, а также имя пакета (package). Лучше не оставлять его именем example, т.к. пакет с таким именем нельзя разместить в Google Play.

Minimum Required SDK — минимальная версия Android, которую будет поддерживать приложение. Чаще всего по умолчанию указывается версия 5.0, чтобы поддерживать как можно больше устройств. Если определенная функция вашего приложения работает только на более новых версиях Android, и это не является критическим для основного набора функций приложения, вы можете включить ее в качестве опции на версиях, которые поддерживают его.

Target SDK - версия Android, под которую будет написано ваше приложение; определяет максимальную версию Android, на которой вы тестировали приложение. Это нужно для режимов совместимости.

Compile With определяет, возможности какой версии Android будет использовать приложение.

Оставьте пока установки, заданные по умолчанию в качестве значений для этого проекта.

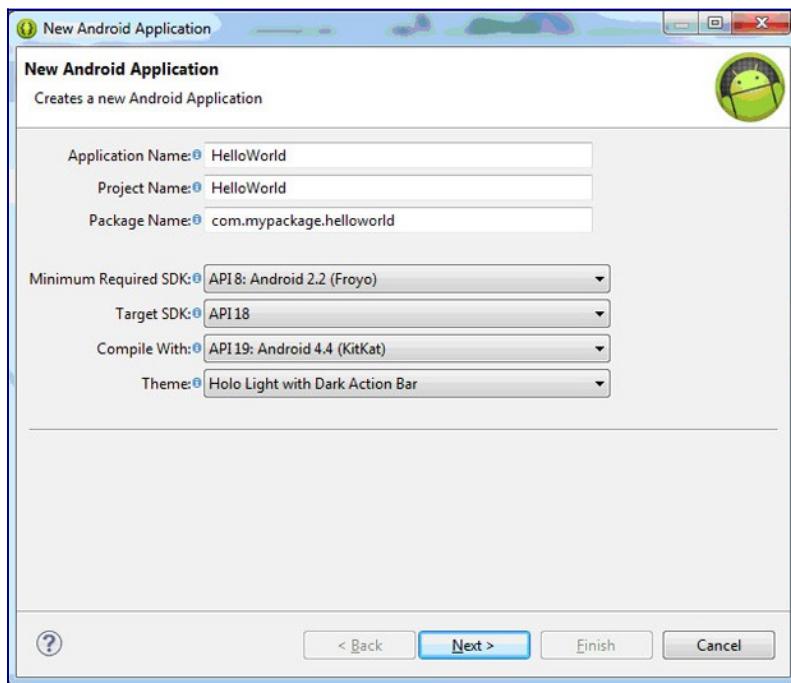


Рис. 1.14. Наименование проекта

Следующее окно можно пропустить без изменений.

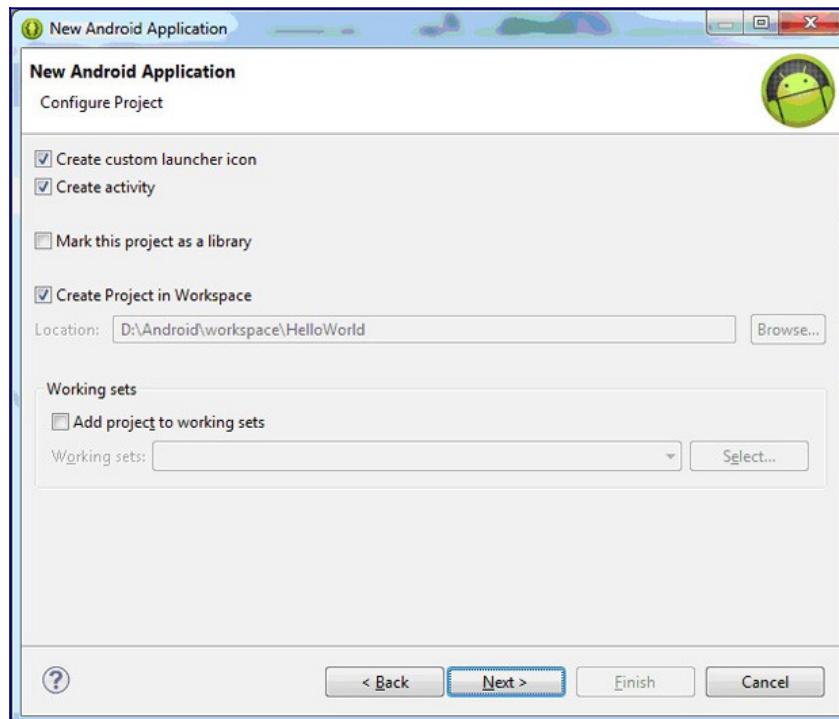


Рис. 1.15. Конфигурация проекта

Create custom launcher icon - создать значок приложения.

Create activity - создать Activity (активность, деятельность).

Mark this project as library - создать проект, как библиотеку. Сейчас в этом нет необходимости, наше приложение в других проектах использоваться не будет.

Create Project in Workspace - создать проект в папке Workspace. В этой папке будут храниться все наши проекты.

Следующий этап - создание иконки. Можете оставить стандартную или создать свою собственную. В нашем примере изменена цветовая гамма, форма, а также выбрана фигурка из клипарта.

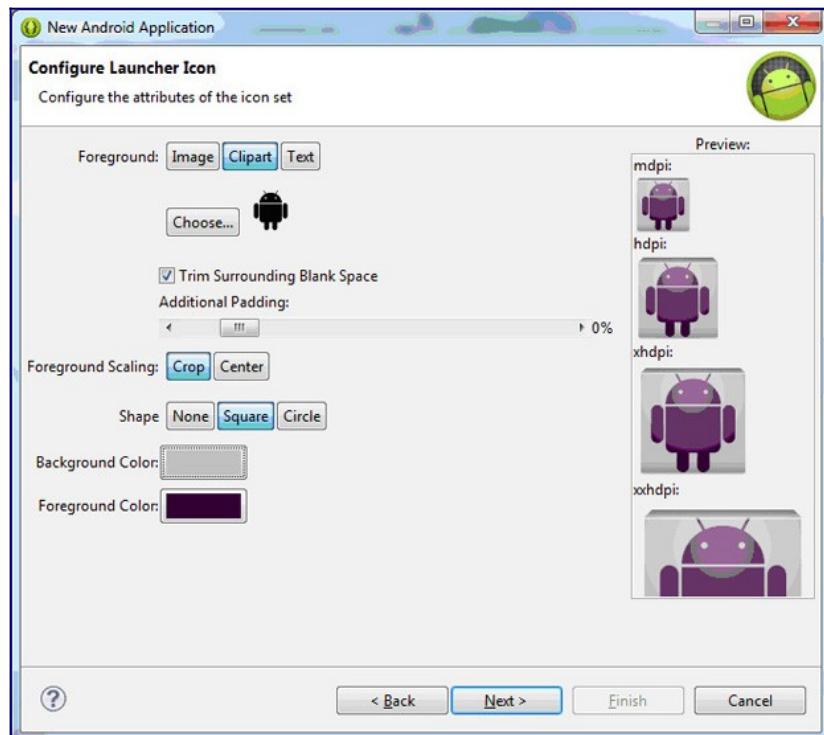


Рис. 1.16. Создание иконки приложения

Большинство приложений на Android имеют свой экран (форму, окно), которое называется операцией или активностью (Activity).

Следующее два окна создают пустую операцию (activity). В первом ничего пока менять не нужно. Во втором вы можете переименовать свою операцию (activity) (в приложении их может быть несколько).

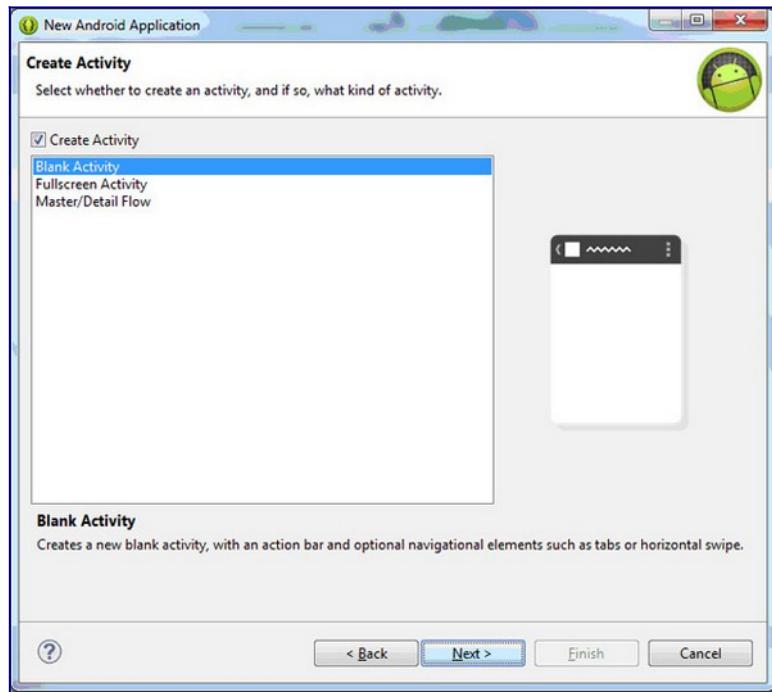


Рис. 1.17. Создание активности

Blank Activity - шаблон, предназначенный для мобильных телефонов.

Fullscreen Activity - шаблон, позволяющий растянуть приложение на весь экран (без навигационной панели и статус-бара).

Master/Detail Flow - шаблон, предназначенный для планшетных компьютеров.

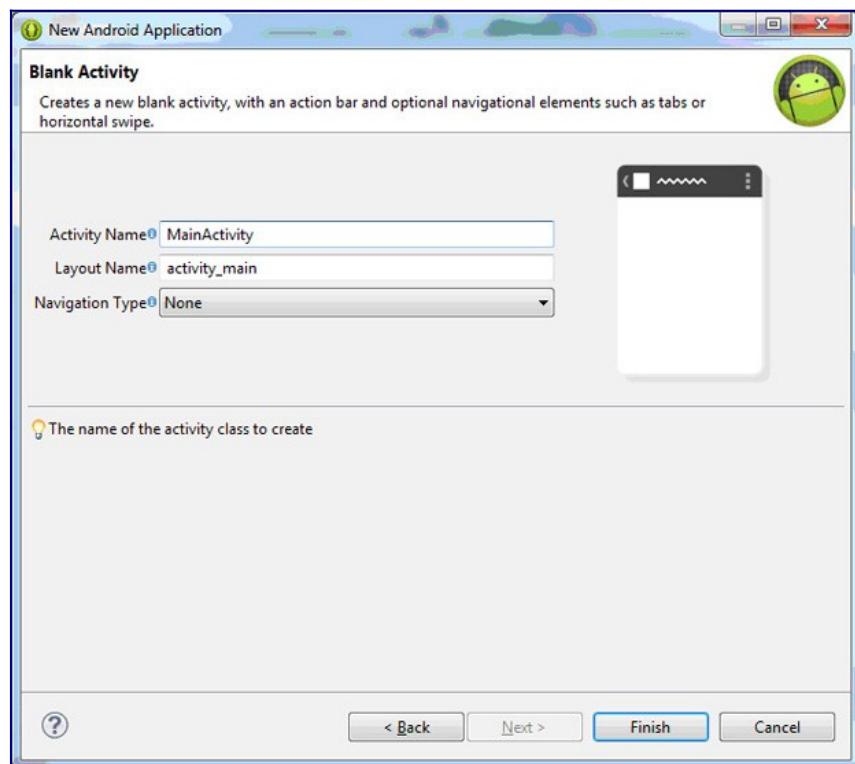


Рис. 1.18. Переименование активности

Итак, вы создали свой первый проект. Конечно, это всего лишь встроенное в среду приложение для проверки корректной установки инструментария, однако множество приложений создаются именно из него.

Посмотрим на его структуру. Она показана в области слева.

В первую очередь нас интересует файл активности. Он находится в папке **src** в вашем пакете. Он имеет расширение **.java**.

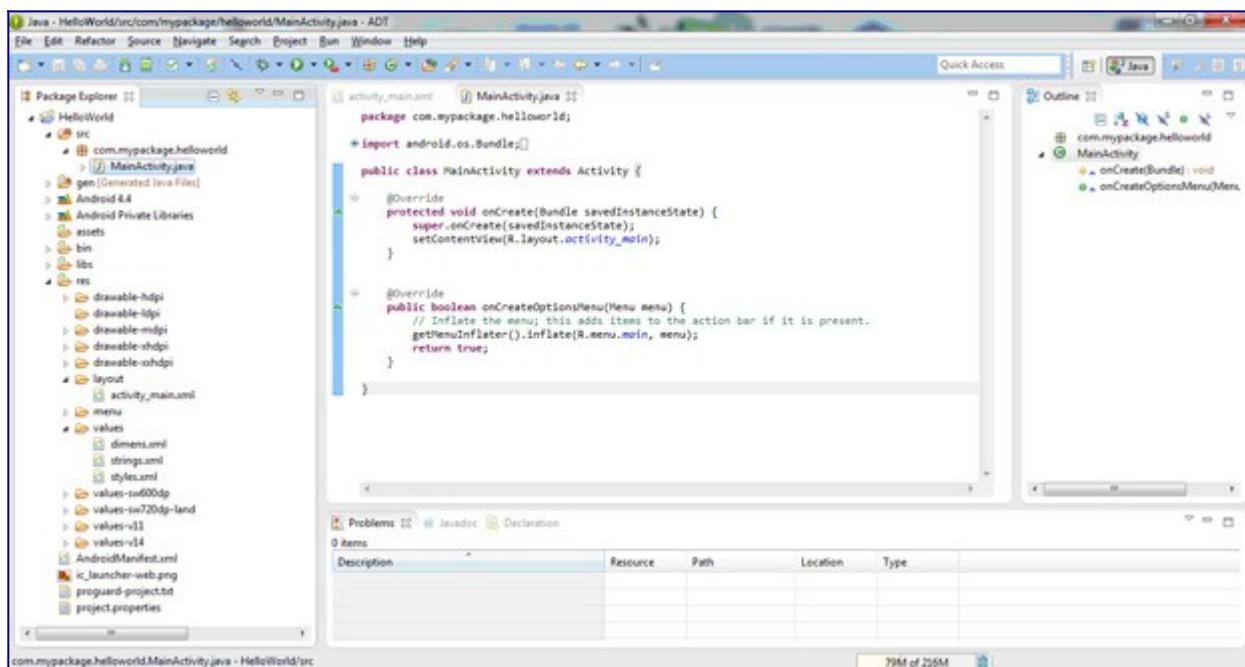


Рис. 1.19. Активность

В папке **res** в подпапке **layout** находится xml-файл, который является оболочкой нашей активности. Именно этот файл будет виден на экране устройства.

С xml-файлами можно работать как в режиме графического редактора, так и непосредственно редактировать код.

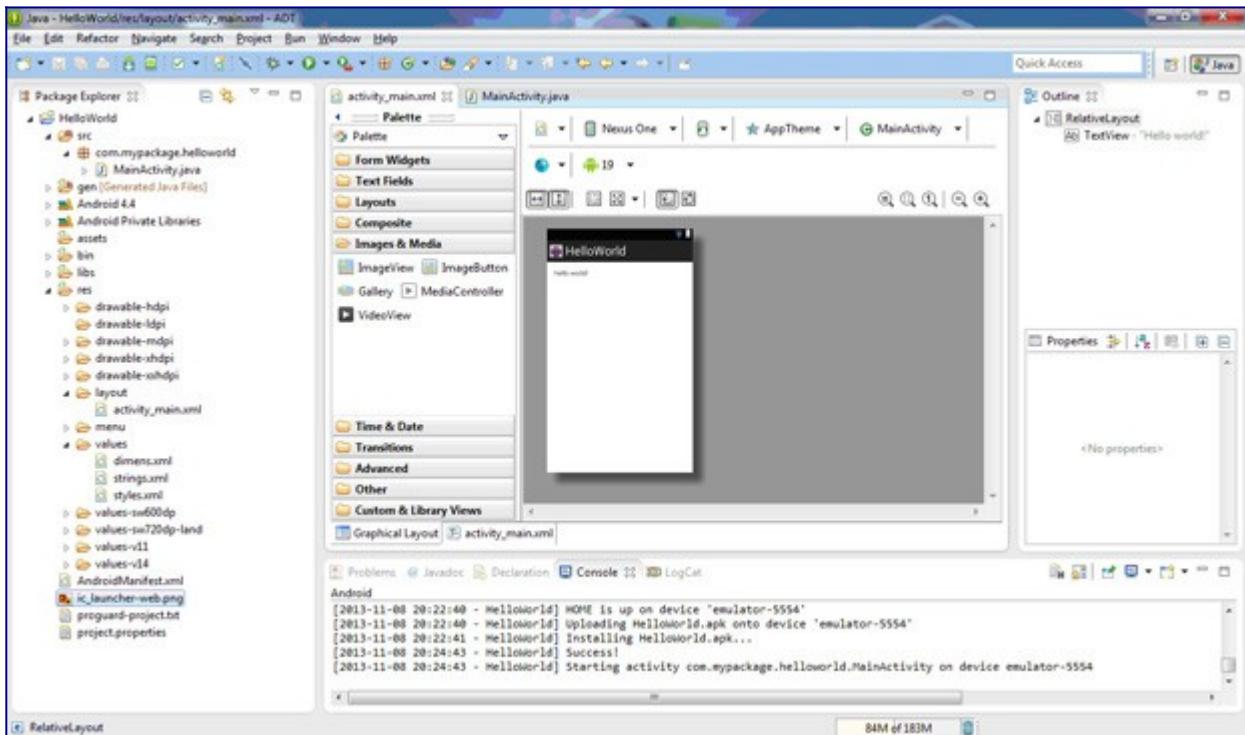


Рис. 1.20. Xml-файл. Графический редактор

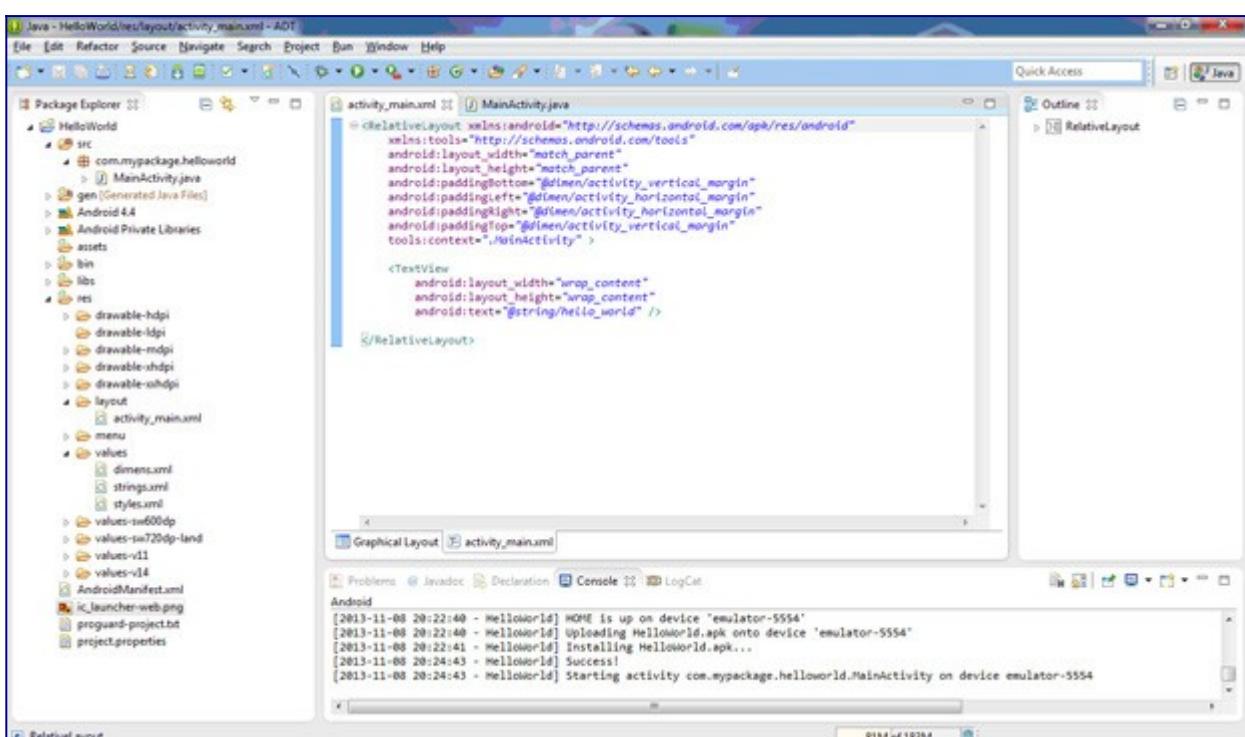


Рис. 1.21. Xml-файл

Задание 2. Выполнить приложение на эмуляторе устройства

В первую очередь нужно создать эмулятор устройства. Это можно сделать, нажав на кнопку на панели инструментов, изображающую смартфон. Если кнопки нет на панели, ее можно найти в меню **Window**.

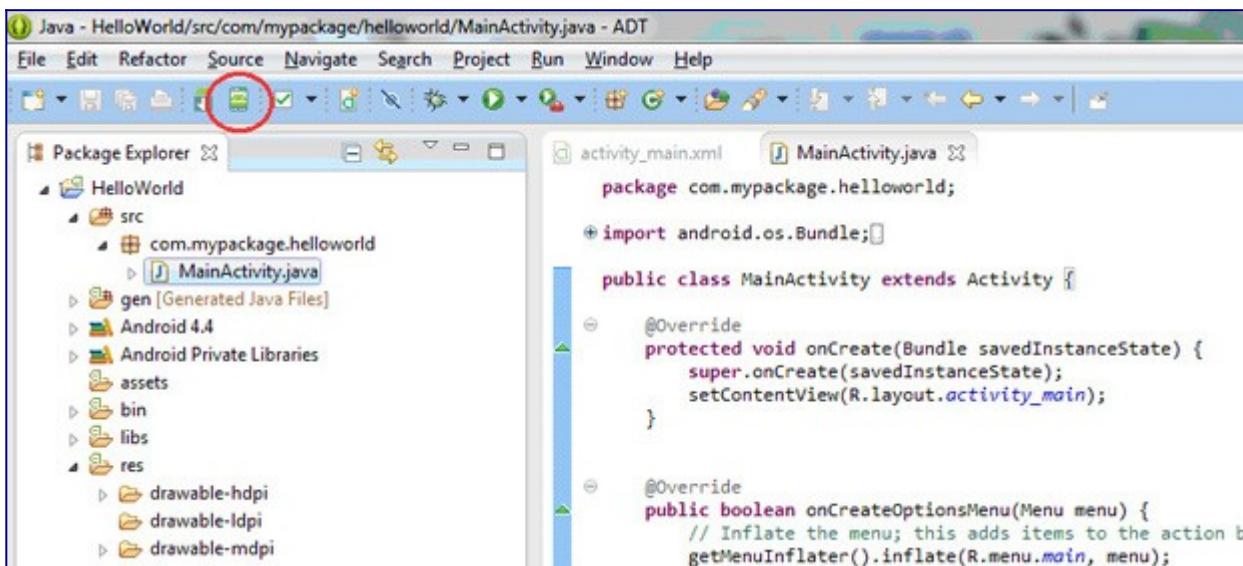


Рис. 1.22. Запуск проекта

Откроется **Android Virtual Device Manager**. Пока в нем нет ни одного виртуального устройства.

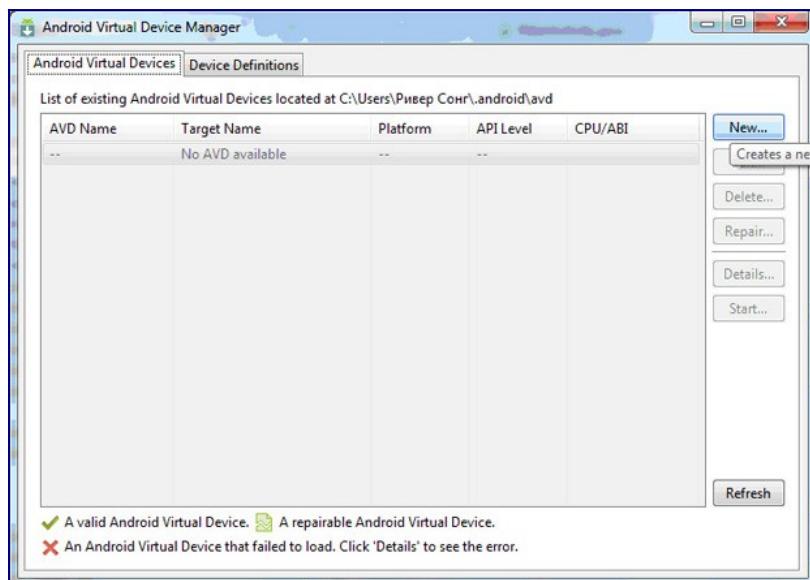


Рис. 1.23. Android Virtual Device Manager

Чтобы создать виртуальное устройство, нажмите кнопку **New**. Появится окно создания. Вам нужно назвать устройство и выбрать обязательные характеристики: **Device** - модель вашего устройства, и **Target** - версия Android. Также можно изменять дополнительные параметры: размер sd-карты, встроенной памяти и т.п.

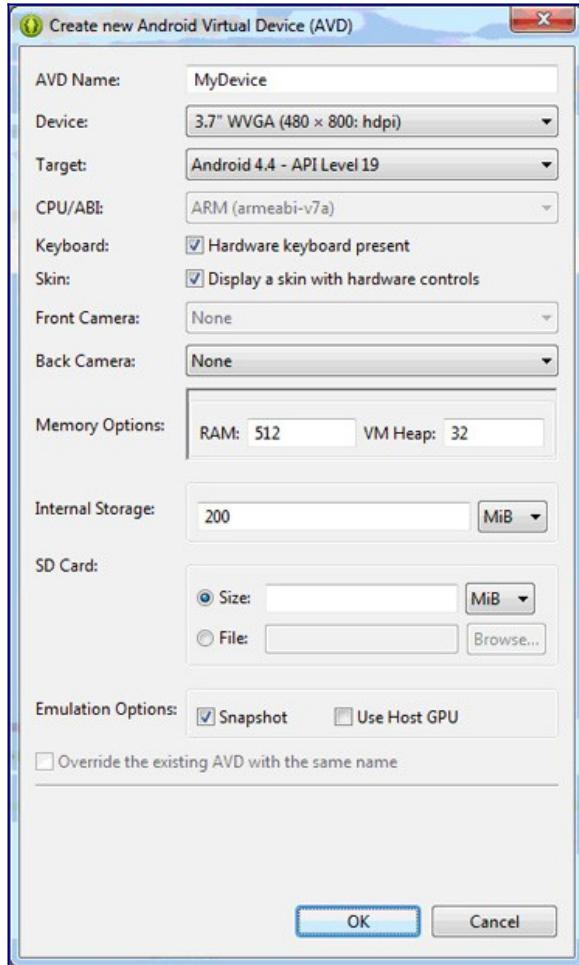


Рис. 1.24. Создание AVD

Теперь можно запускать приложение. Для этого нужно нажать на кнопку **Run** (белый треугольник в зеленом кружочке) на панели инструментов. Проблемы с запуском можно отследить в консоли.

Если приложение не запускается, попробуйте нажать на черный треугольник справа от кнопки **Run**, выбрать **Run Configurations**, затем во вкладке **Target** выбрать созданное устройство и запустить проект снова.

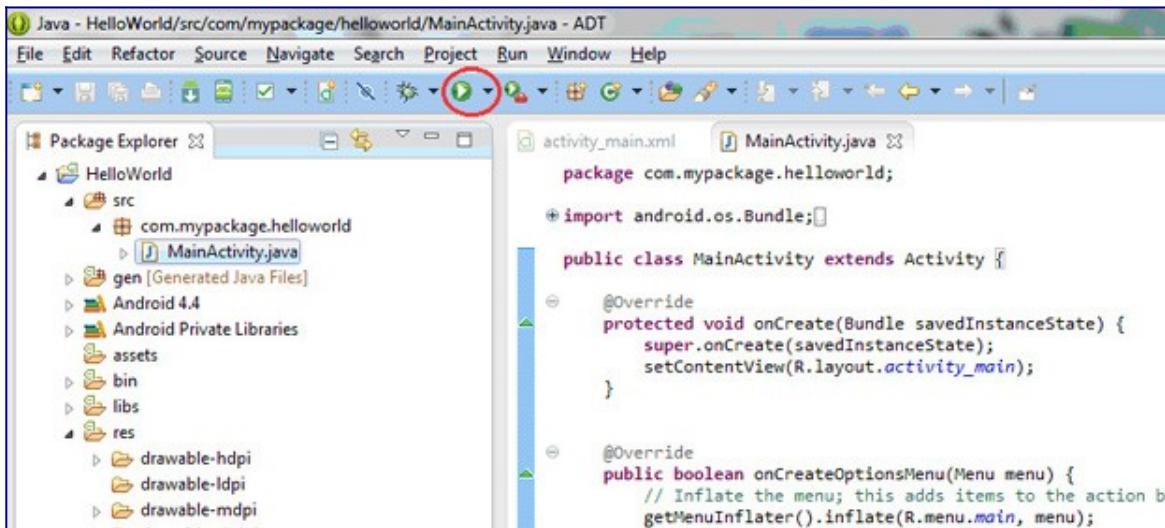


Рис. 1.25. Запуск приложения

Если все сделано правильно, должен загрузиться эмулятор. Время запуска зависит от размера оперативной памяти на вашем компьютере. В дальнейшем эмулятор можно не закрывать, приложения будут запускаться в работающем.

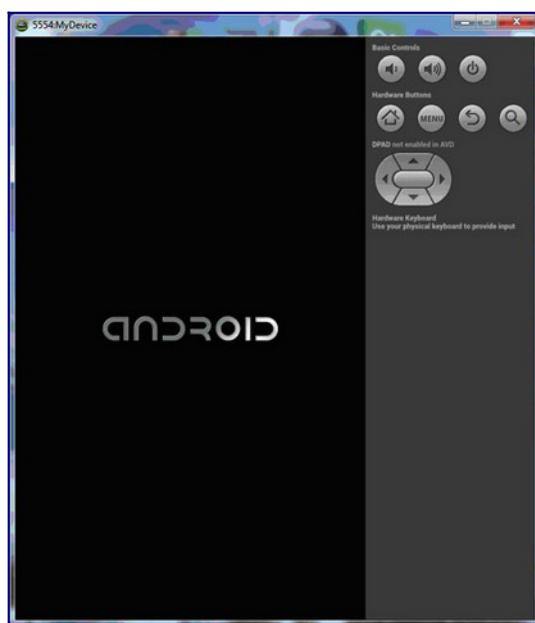


Рис. 1.26. Запуск эмулятора

Рис. 1.27. Запущенный эмулятор

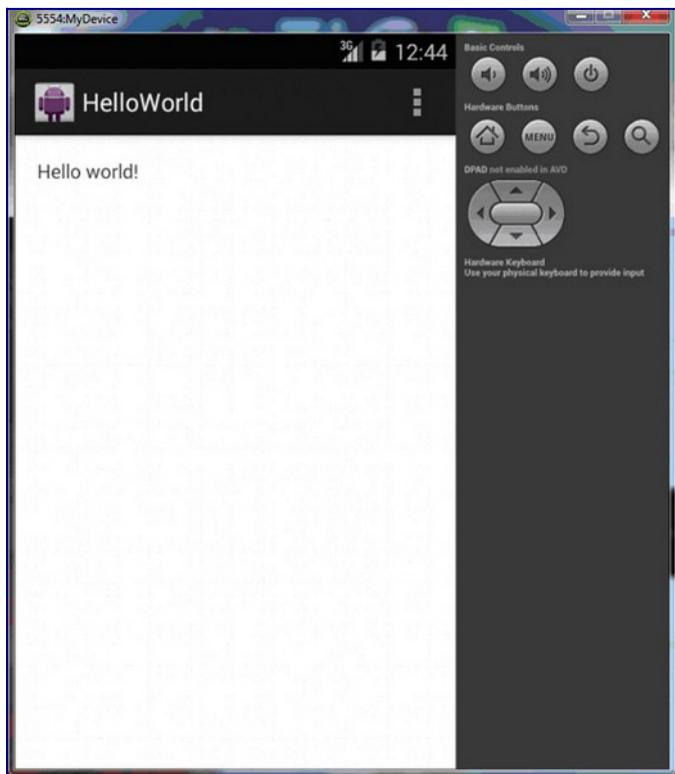


Рис. 1.28. Приложение Hello, world!

Если ваше приложение сразу не загрузилось, его можно найти в меню приложений устройства.

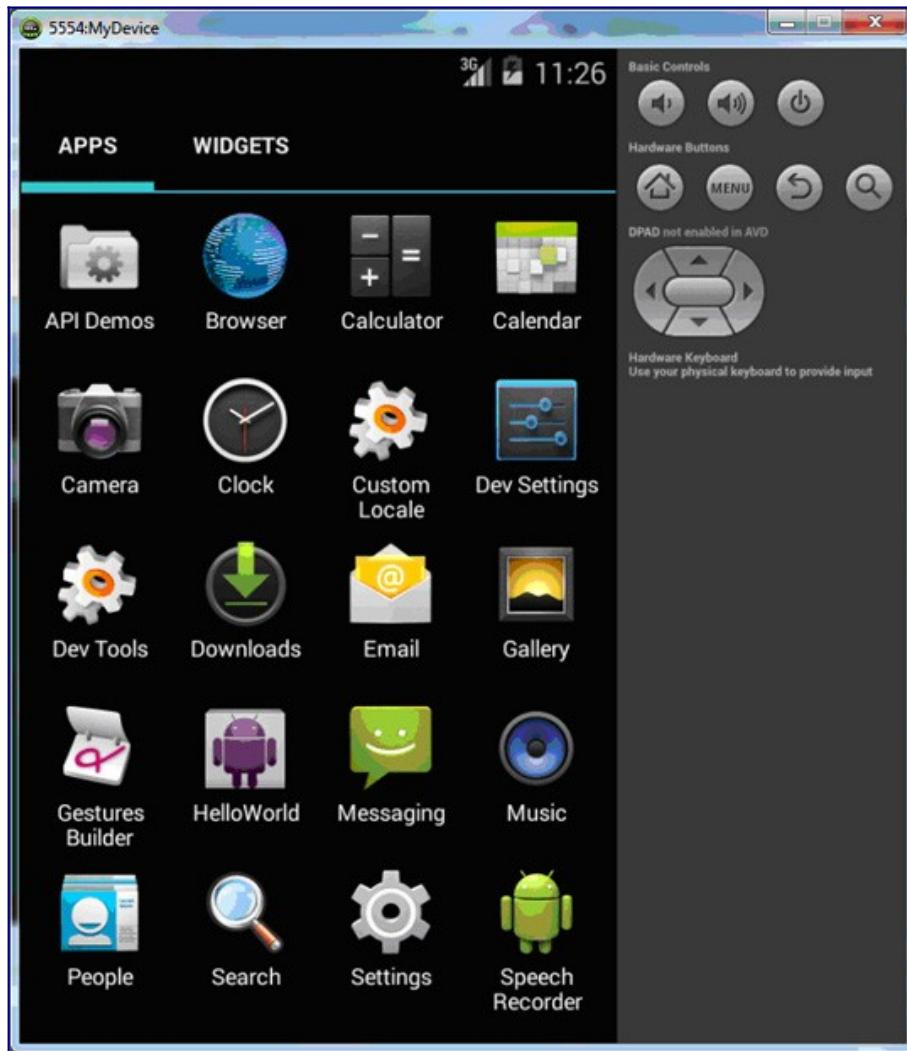


Рис. 1.29. Меню приложений

Задание 3. Выполнить приложение на устройстве

Прежде чем запустить проект на реальном устройстве, необходимо:

- Настроить устройство
- Настроить компьютер
- Настроить среду

В данной версии представлено руководство для предыдущих версий устройств и не предусматривает настройку для новых устройств. Актуальные рекомендации по настройке устройства и эмулятора описаны в статье по созданию приложения <https://developer.android.com/training/basics/firstapp/running-app> или в руководстве <https://developer.android.com/studio/run/device>. **Ознакомьтесь с ними, если рекомендации ниже НЕ подходят для Вашего устройства.**

3.1 Настройте устройство

Настройка устройства состоит в следующем:

1. Включить режим отладки по USB.

2. Для запуска файлов с расширением *.apk, полученных не из магазина приложений Google Play, необходимо разрешить установку приложений из альтернативных источников.

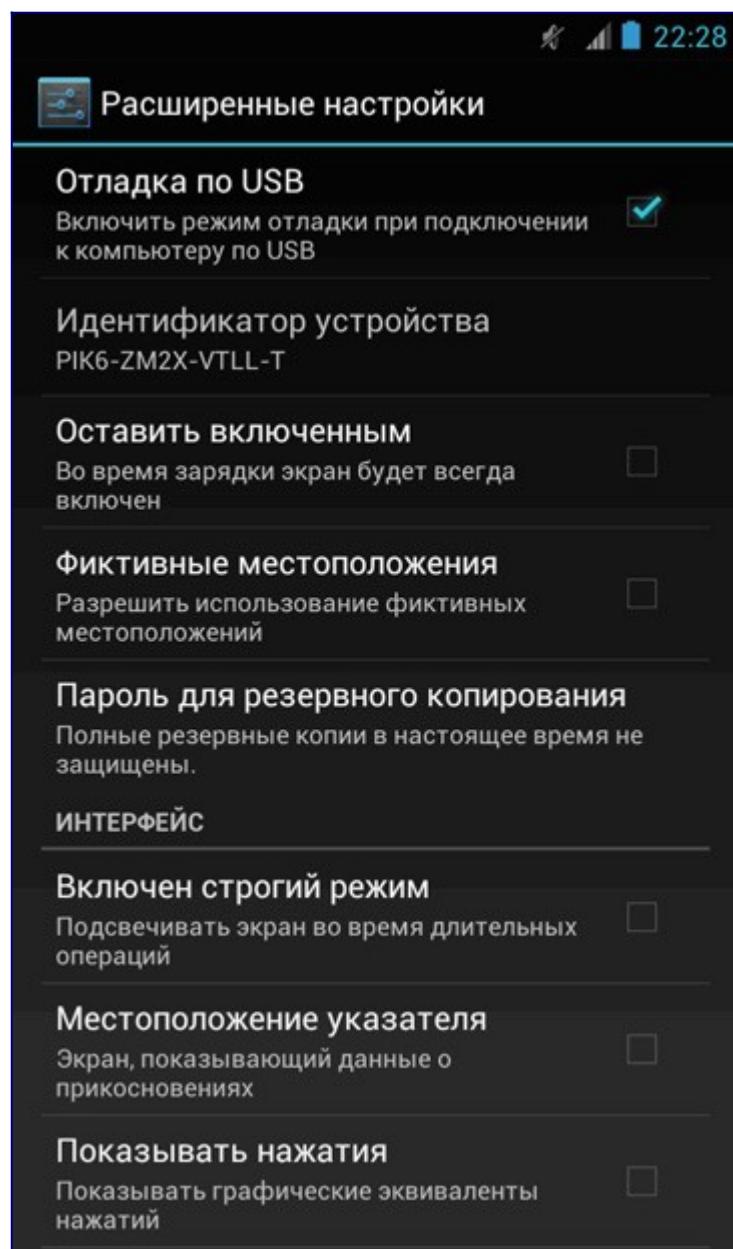


Рис. 1.30. Настройка устройства

3.2 Настроить компьютер

В ряде случаев установка драйвера не требуется. Если же телефон недоступен, то потребуется выполнить следующие действия:

1. Зайти в Диспетчер устройств (**Пуск->Панель управления->Система и безопасность->Система->Диспетчер устройств**) и найти ваше устройство

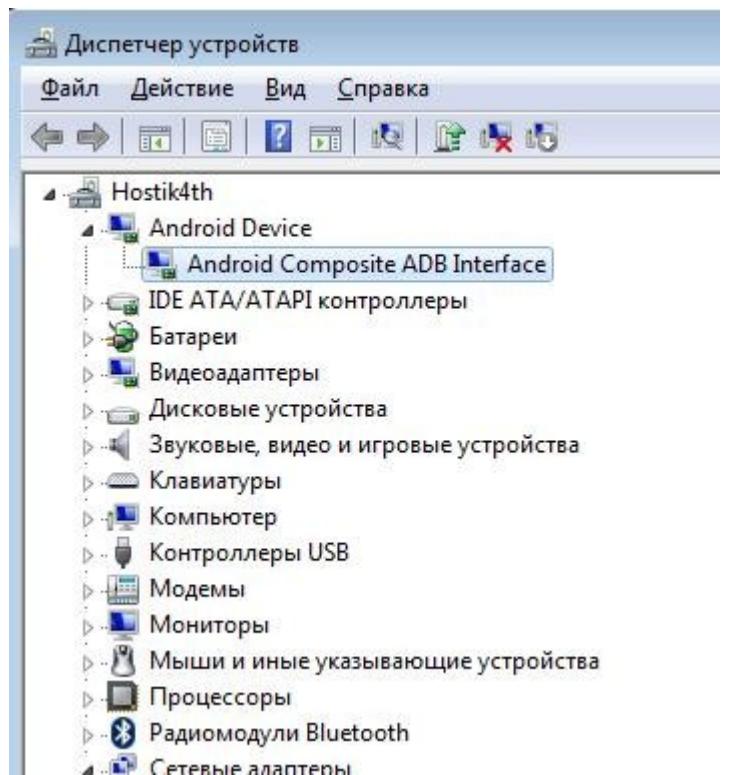
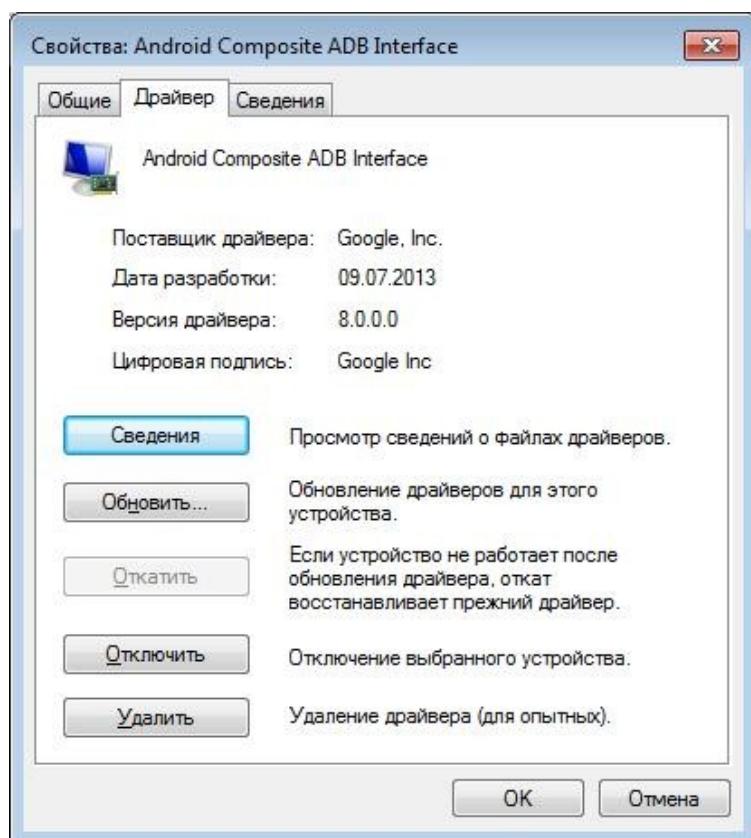


Рис. 1.31. Диспетчер устройств

2. Щелкнуть по нему правой кнопкой мыши и вызвать меню **Свойства**.



3. Во вкладке **Драйвер** нажать на кнопку **Обновить** драйвер.

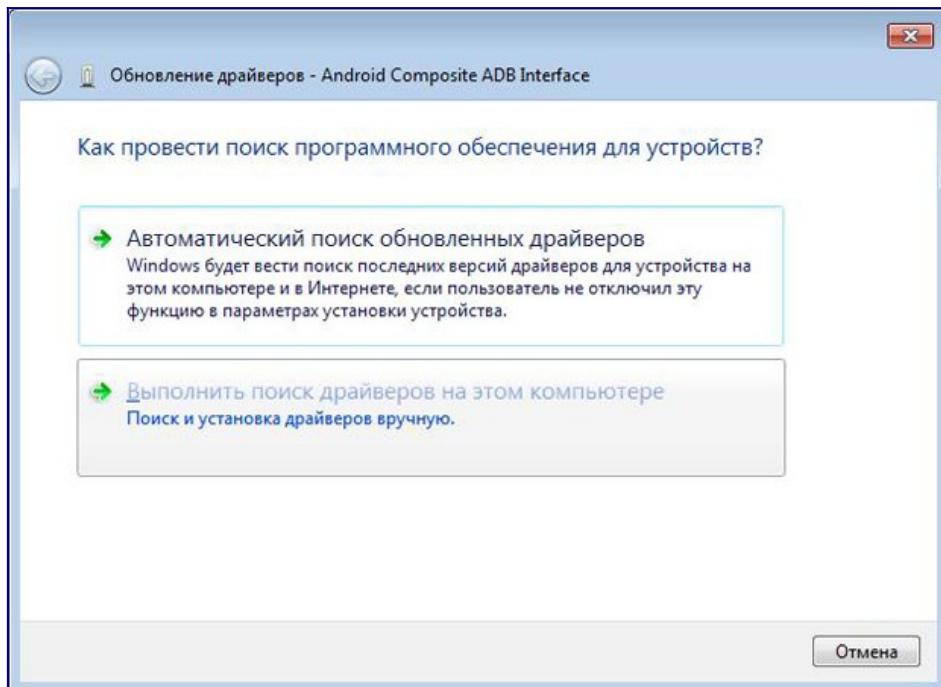


Рис. 1.30. Обновление драйвера

4. В появившемся окне выбрать **Выполнить поиск драйвера на этом компьютере**.

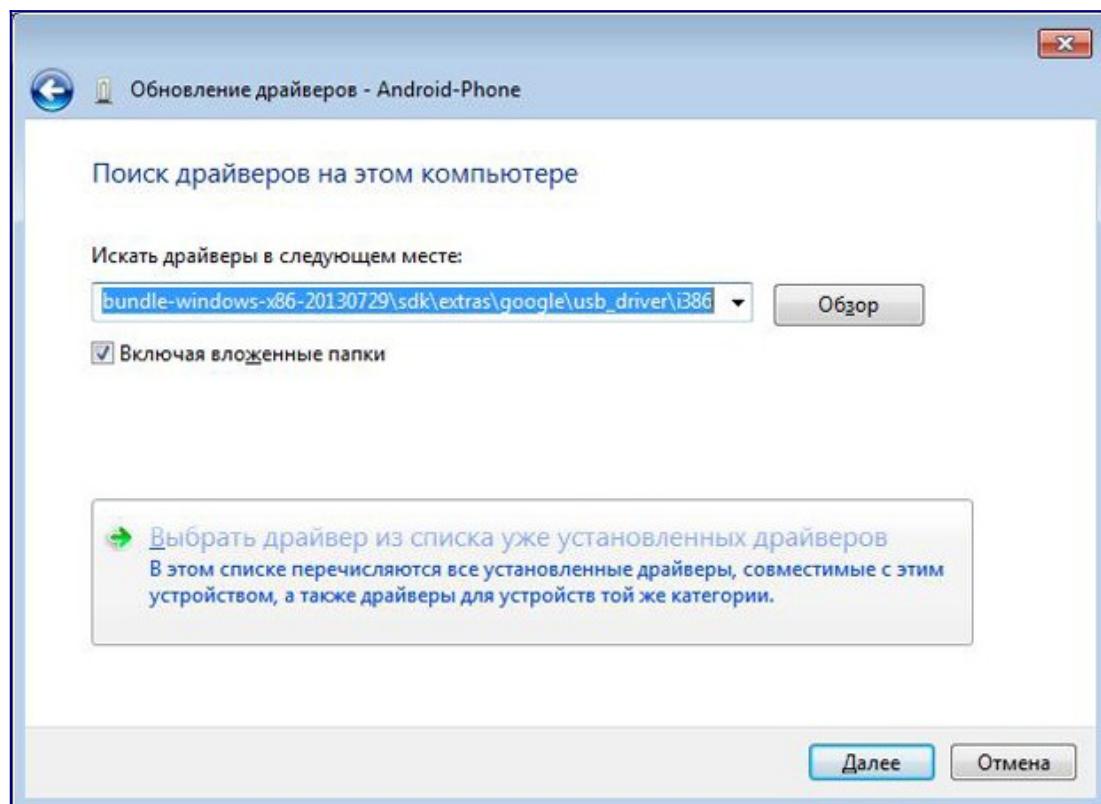


Рис. 1.32. Поиск драйверов на компьютере

5. В следующем окне нужно прописать путь, откуда будет установлен драйвер.

Или можно отыскать драйвер следующим образом:

- В следующем окне нажмите **Установить с диска**

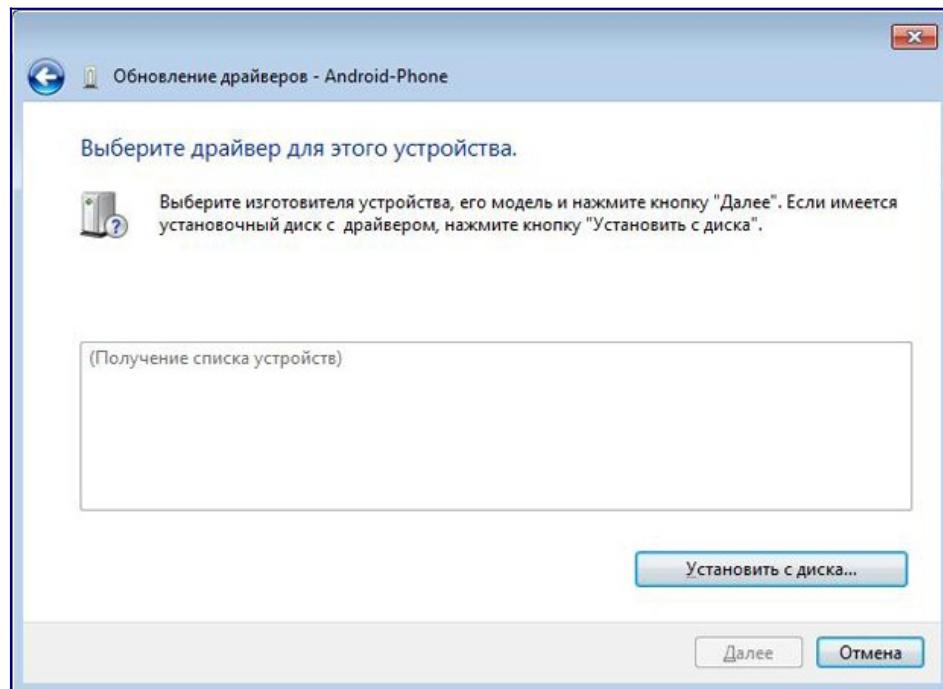


Рис. 1.33. Выбор драйвера устройства

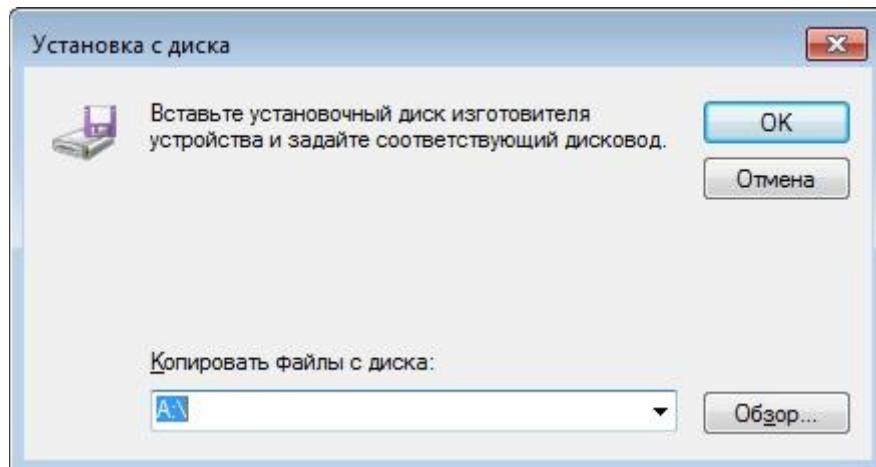


Рис. 1.34. Поиск драйвера

- Далее выберите файл android_winusb.inf

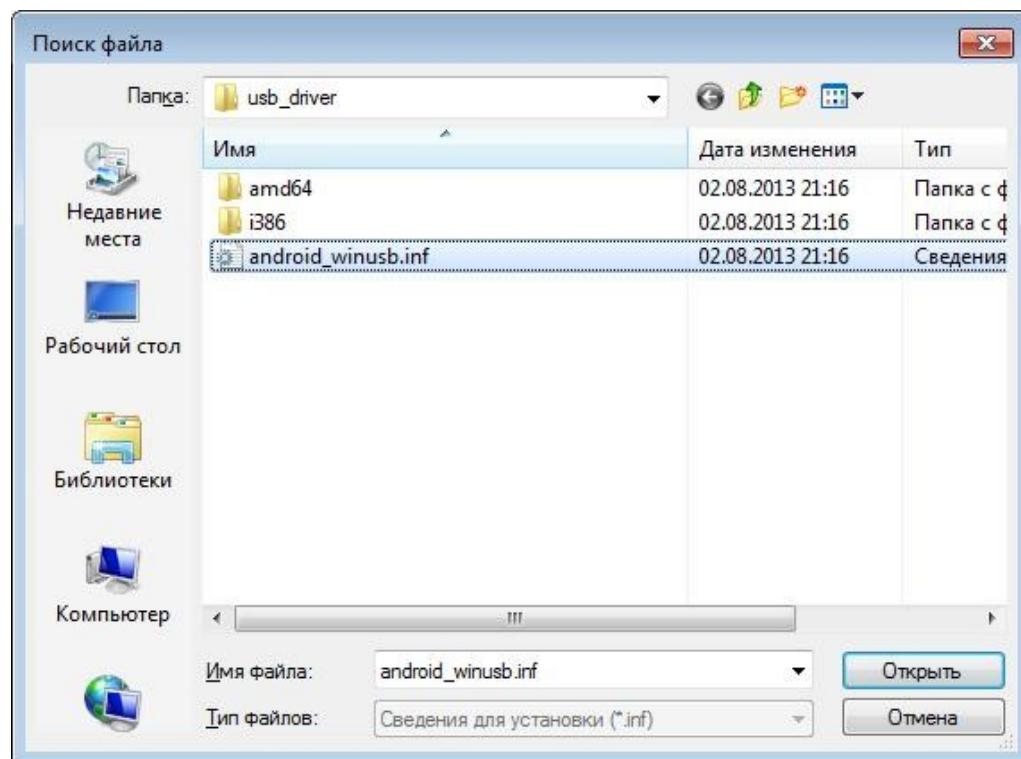


Рис. 1.35. Сведения для установки

6. Нажмите **Далее** в этом и следующем окнах.
7. Из предложенного списка выберите **Android ADB Interface** и нажмите **Далее и Да**.

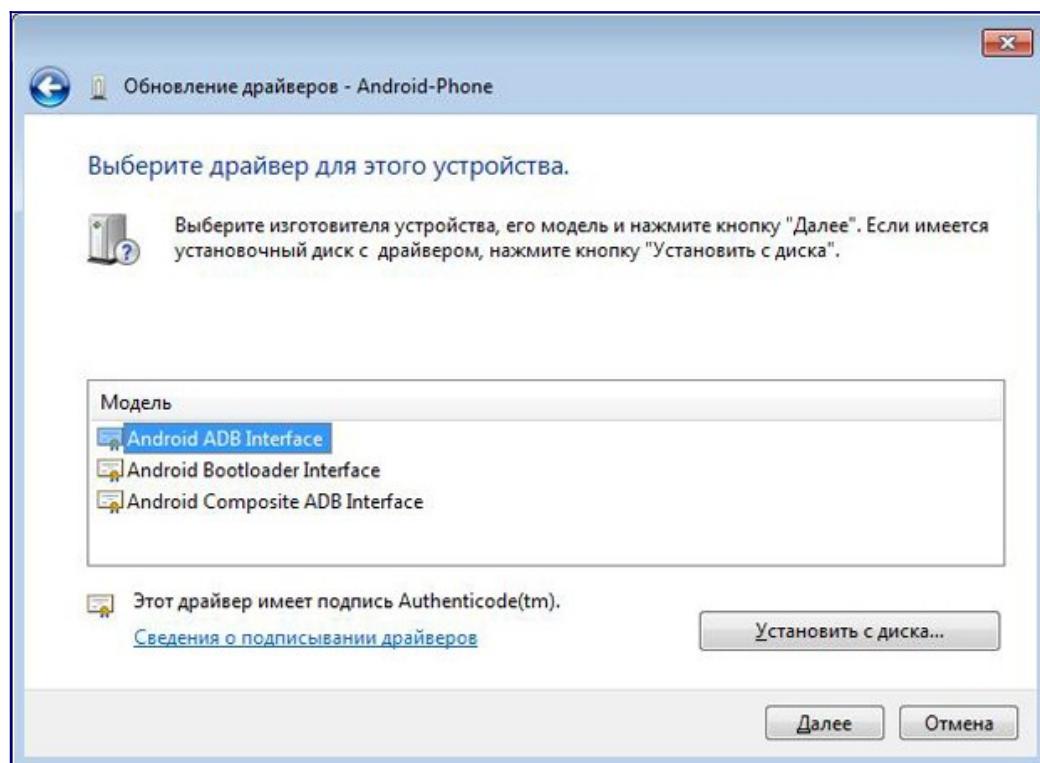


Рис. 1.36. Завершение установки драйвера

3.3. Настроить среду Android Studio

1. В Android Studio зайдите в меню **Run\Debug Configurations** и перейдите на вкладку **Target**. Поставьте флажок напротив **Always prompt to pick device**.

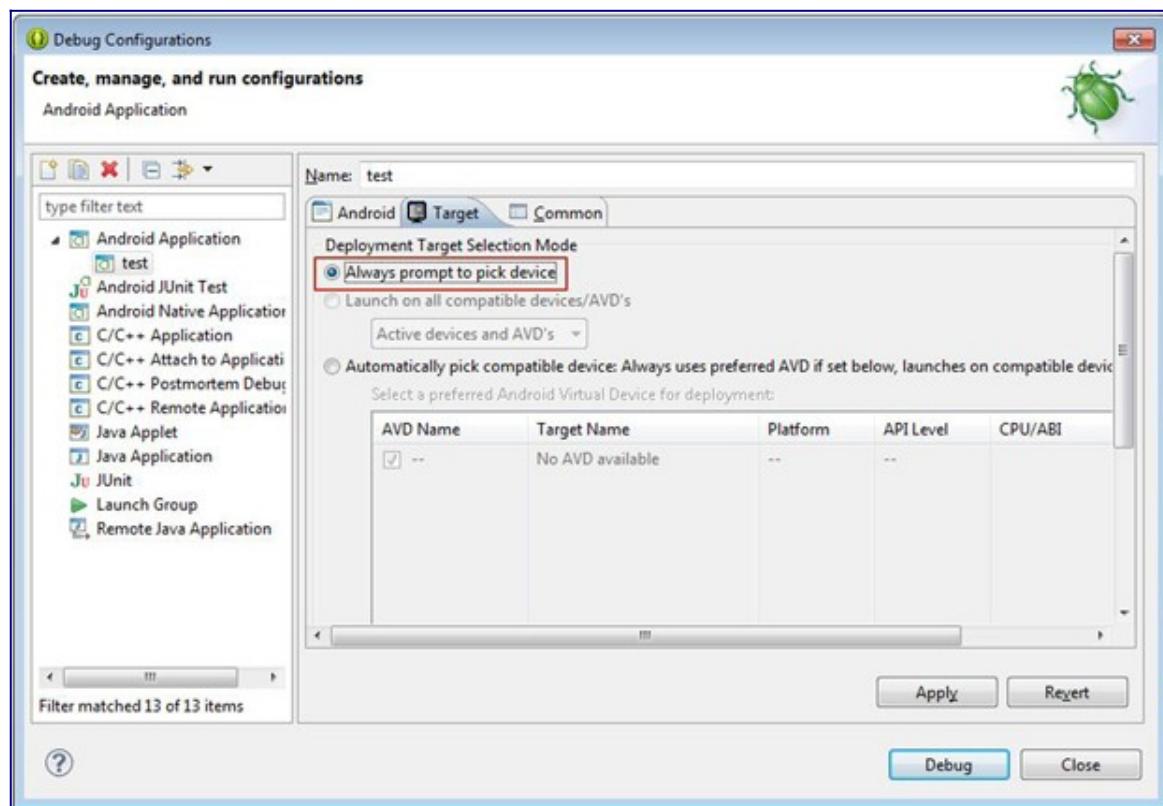


Рис. 1.37. Настройка среды

2. В открывшемся окне выберите подключенное устройство, поставив флажок напротив **Choose a running Android Device**

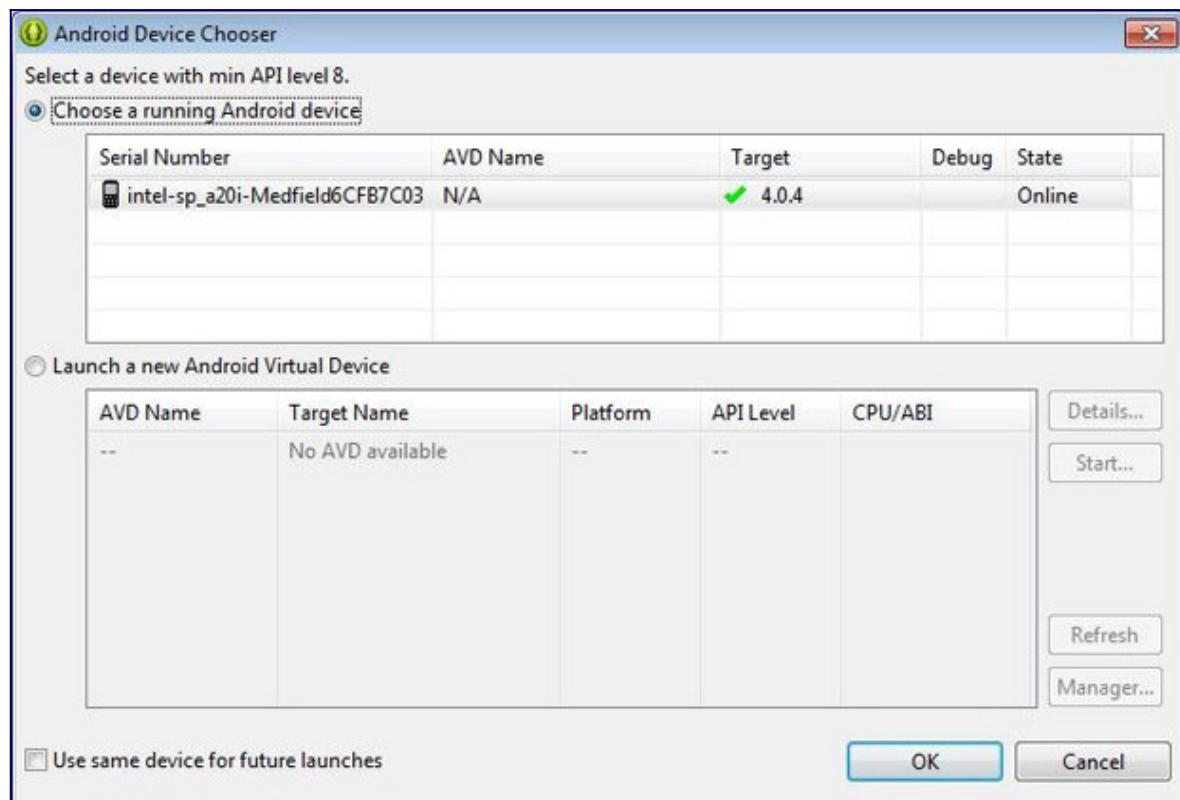


Рис. 1.38 Выбор устройства

3. Теперь можно запустить приложение.

Задание 4. Создание приложения с использованием стандартных макетов

1) Изучить материалы лабораторной работы и дополнительные материалы:

<https://developer.android.com/studio/projects/index.html>

<https://developer.android.com/studio/projects/create-project.html>

<https://developer.android.com/training/index.html>

<https://developer.android.com/guide/topics/ui/declaring-layout?hl=ru>

<https://betacode.net/10423/android-ui-layouts>

2) Изучить создание макета с разметкой Constraint Laouyt:

<https://startandroid.ru/ru/uroki/vse-uroki-spiskom/489-urok-180-constraintlayout-osnovy.html>

Layouts in Android Studio — https://youtu.be/OCceWupZ_Ik

<https://developer.android.com/training/constraint-layout>

3) Изучить создание макета с разметкой Constraint Laouyt и Activity на Kotlin -

<https://tutorial.eyehunts.com/android/constraint-layout-advantage-and-example-in-kotlin/>

4) Изучить создание макета с разметкой Linear Laouyt и Activity на Kotlin -

<https://tutorial.eyehunts.com/android/android-linearlayout-tutorial-example-android-kotlin/>

5) Изучить материалы по подключению репозитория на github:

<http://developer.alexanderklimov.ru/android/studio/git.php>

<https://code.tutsplus.com/ru/tutorials/working-with-git-in-android-studio--cms-30514>

<https://medium.com/code-yoga/how-to-link-android-studio-with-github-312037a13b99>

6) Добавить в репозиторий каждого проекта файл .gitignore (см. Настройка .gitignore для проекта в Android Studio).

7) Проект на Java с макетом Constraint Layout:

1. Создайте проект на Java, содержащий следующие элементы:

- ImageView — Ваше фото
- TextView — аккаунт в соцсети VK или Telegram
- TextView — список интересов/хобби
- Button — кнопка «Подписаться», при нажатии на которую выводится ссылка на страницу Вашей соцсети.

- У приложения должна быть своя иконка, отличная от стандартной.
2. Подключите внешний репозиторий на github, созданный для лабораторной работы 1.
 3. Опубликуйте проект в репозитории в ветке project2 в каталоге *lab1-project2_1_ваша-фамилия*, заменив *ваша-фамилия* на Вашу фамилию.
 4. При защите проекта продемонстрировать запуск приложения в эмуляторе или на устройстве.

8) Проект на Kotlin с макетом Linear Layout:

1. Создайте проект на Kotlin, как и в пункте 6, но используйте макет LinearLayout. У приложения должна быть свой иконка.
2. Опубликуйте проект в репозитории в ветке project2 в каталоге *lab1-project2_2_ваша-фамилия*, заменив *ваша-фамилия* на Вашу фамилию.
3. При защите проекта продемонстрировать запуск приложения в эмуляторе или на устройстве.

Задание 5. Создание простого приложения с использованием Jetpack Compose

- 1) Изучить материалы Unit 1 <https://developer.android.com/courses/android-basics-compose/course> и статьи <https://developer.android.com/codelabs/basic-android-kotlin-compose-first-app#1>
- 2) Реализовать приложение из задания 4 на языке Kotlin с использованием Jetpack Compose.
- 3) Разработать иконку приложения.
- 4) Добавить файл .gitignore в каталог проекта.
- 5) Опубликуйте проект в репозитории в ветке project3 в каталоге *lab1-project3_ваша-фамилия*, заменив *ваша-фамилия* на Вашу фамилию.
- 6) При защите проекта продемонстрировать запуск приложения в эмуляторе или на устройстве.

Контрольные вопросы

1. Какое представление позволяет просмотреть файловую структуру проекта, включая скрытые файлы?
2. Какая структура папок (групп) в представлении Android View?
3. Какая структура папок в представлении Project View?

4. Может ли пакет apk содержать несколько файлов AndroidManifest.xml? Почему?
5. Может ли проект Android Studio несколько файлов манифестов? Каких?
6. Какой элемент в файле manifest.xml является корневым? Какой обязательный элемент он должен содержать?
7. Что определяет атрибут xmlns:android? Является ли он обязательным?
8. Как определить в манифесте иконку приложения?
9. Для чего предназначен атрибут android:label элемента <application> в манифесте? Элемента <activity>?
10. Перечислите достоинства и недостатки макета Constraint Layout пользовательского интерфейса в Adnroid.
11. Охарактеризуйте библиотеку Jetpack Compose, ее достоинства, недостатки и возможности для построения пользовательского интерфейса мобильного приложения.
12. Какие основные шаги необходимо выполнить для подключения репозитория на github?
13. Как создать коммит в Android Studio и просмотреть историю коммитов?
14. Как создать ветку в репозитории в Android Studio?
15. Как получить изменения из внешнего репозитория и опубликовать их во внешний репозиторий?

Литература

1. <http://startandroid.ru/ru/uroki/vse-uroki-spiskom/9-urok-2-ustanovka-i-nastrojka-sredy-razrabotki.html>
2. <http://www.fandroid.info/urok-2-ustanovka-i-nastrojka-android-studio-ustanovka-jdk-nastrojka-android-sdk/>
3. <https://android-school.ru/ustanovka-android-studio-emulator-hello-world/>
4. <http://web-academy.com.ua/stati/14-stati/63-ustanovka-android-studio-instruktsiya>