

## Модуль 2. Объектно-ориентированное программирование

### Тема 2.3. Архитектура приложений под Android. Активности

2 часа

#### Оглавление

2.4. Архитектура приложений под Android. Активности.....	2
2.4.1. Платформа Android.....	2
Компоненты Android .....	2
2.4.2. Создаем Android проект.....	4
Как создать проект в Eclipse ADT .....	4
Как создать проект в Android Studio .....	7
Как загрузить существующий проект под Android.....	8
2.4.3. Запуск приложения .....	10
Как подключить Android устройство .....	10
Запуск приложения на Android устройстве .....	11
2.4.4. Структура проекта .....	11
Упражнение 2.4.1 .....	12
2.4.5. Активности (Activity) .....	13
Создание Активности .....	13
Жизненный цикл Активности .....	13
Стеки Активностей .....	14
Состояния Активностей.....	14
Отслеживание изменений состояния Активности.....	15
Упражнение 2.4.2. ....	17
2.4.6. *Разбор приложения TestBed.....	17
Упражнение 2.4.3 .....	19
Благодарности .....	20

## 2.4. Архитектура приложений под Android. Активности

### 2.4.1. Платформа Android

Android – свободно распространяемая, активно развивающаяся операционная система (ОС) для мобильных устройств. Эта ОС основана на ядре Linux 2.6, включая прикладное программное обеспечение. Первая устойчивая версия Android 1.0 была выпущена 23 сентября 2008 года. На сегодняшний день Android – самая распространенная система для мобильных устройств. Изначально ОС разрабатывалась компанией Android Inc., которую затем купила Google. Googleв альянсе Open Handset Alliance (ОНА) занимается поддержкой и дальнейшим развитием платформы Android .

Инструментарий программной разработки Android SDK находится в свободном доступе и включает в себя интерфейсы прикладного программирования (API) на языке Java. В ИТ ШКОЛЕ SAMSUNG используют IDE Eclipse с плагином ADT (Android Developer Tools) или Android Studio, в состав этих сред разработки включены все инструменты, необходимые для программирования под Android. С декабря 2014 года компания Google официально объявила о прекращении поддержки плагина ADT и переходе на Android Studio.

Скачать Android Studio можно по адресу <http://developer.android.com/sdk/index.html>. IDE Eclipse с плагином ADT размещен в учебном курсе.

#### Компоненты Android

Архитектура системы включает в себя четыре уровня (рис.1):

- приложения,
- система приложений,
- библиотеки,
- ядро Linux.

**Приложения.** Платформа Android не различает по правам предустановленные и сторонние приложения, что позволяет менять программную конфигурацию устройства, в том числе на самостоятельно разработанные. Основным языком разработки под Android является Java, однако, можно так же использовать другие языки программирования, например, C++. Все функциональные возможности системы открыты, так же имеется возможность создания новых компонент.

**Система приложений.** Включает различные службы, курирующие работу составляющих системы.

Activity Manager — диспетчер активности, который отвечает за функционирование приложения и его жизненный цикл.

Resource Manager — диспетчер ресурсов необходим для доступа к используемым внутренним ресурсам (строковым, графическим и т.п.).

Package Manager — диспетчер пакетов, отвечающий за установку и функционирование пакетов прикладных программ.

Window Manager — диспетчер окон, распределяющий активность окон приложений и порядок их отображения.

Telephony Manager — менеджер телефонии следит за типом доступа и параметрами сети

Location Manager — менеджер местоположения — навигационные службы, передающие приложениям информацию о местоположении устройства.

Notification Manager — диспетчер уведомлений позволяет приложению публиковать сообщения в строке состояния.

Content Providers — менеджер внешних ресурсов, открывающий доступ к другим приложениям.

View System — система представлений, используемая для создания внешнего оформления приложения. Имеет расширяемую функциональность.

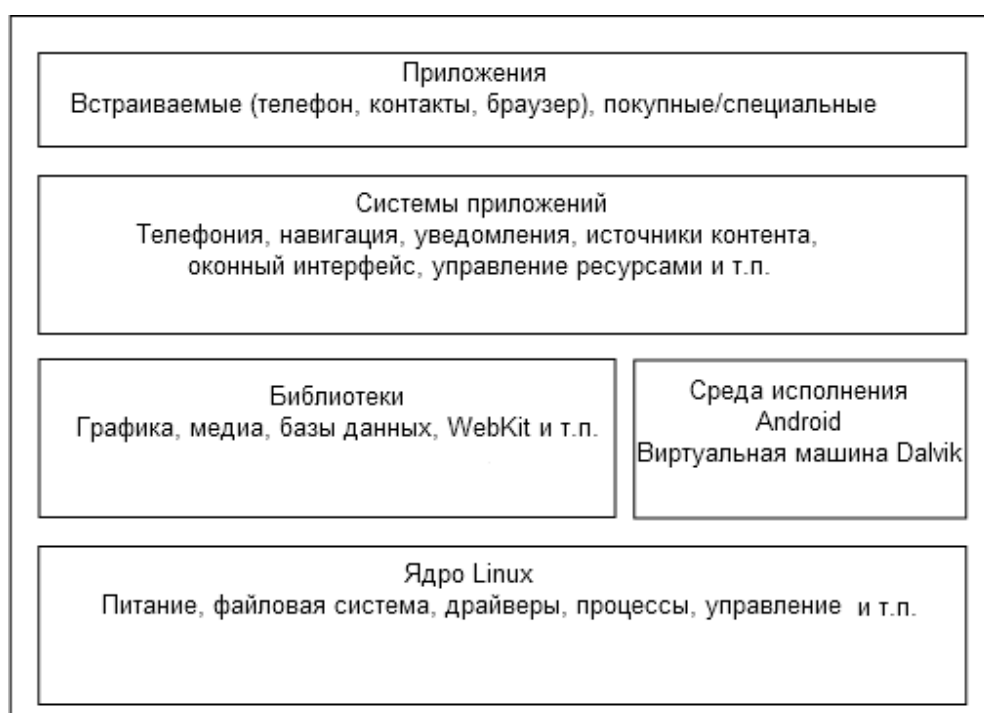


Рисунок 1. Структура ОС Android

**Библиотеки.** Кроме стандартных SLD (2D графика), OpenGL (3D графика), Media Framework (мультимедиа), WebKit (встроенный браузер), FreeType (поддержка шрифтов), SQLite (работа с базой данных), SSL (зашифрованные соединения), разработчики Android создали собственную версию стандартной библиотеки C/C++ - библиотеку Bionic (не поддерживаются исключения C++ и несовместима с GNU libs и POSIX).

На этом же уровне расположены Менеджер поверхностей, позволяющий создавать различные эффекты изображений за счет хранения рисунков в битовых массивах, не отправляя их непосредственно в буфер экрана и Dalvik Virtual Machine - виртуальная машина Java, выполняющая программы. Начиная с версии Android 4.4 KitKat в системе появилась альтернатива DVM - ART (Android Runtime) - среда исполнителя приложений без предварительной программной настройки устройства;

**Ядро Linux.** На этом уровне контролируется аппаратное обеспечение устройства, в том числе работают драйверы межпроцессорного взаимодействия (IPC) и управления питанием. Хотя система и построена на ядре Linux, однако, она имеет некоторые специфические расширения ядра, свойственные Android, а значит, не является Linux-системой;

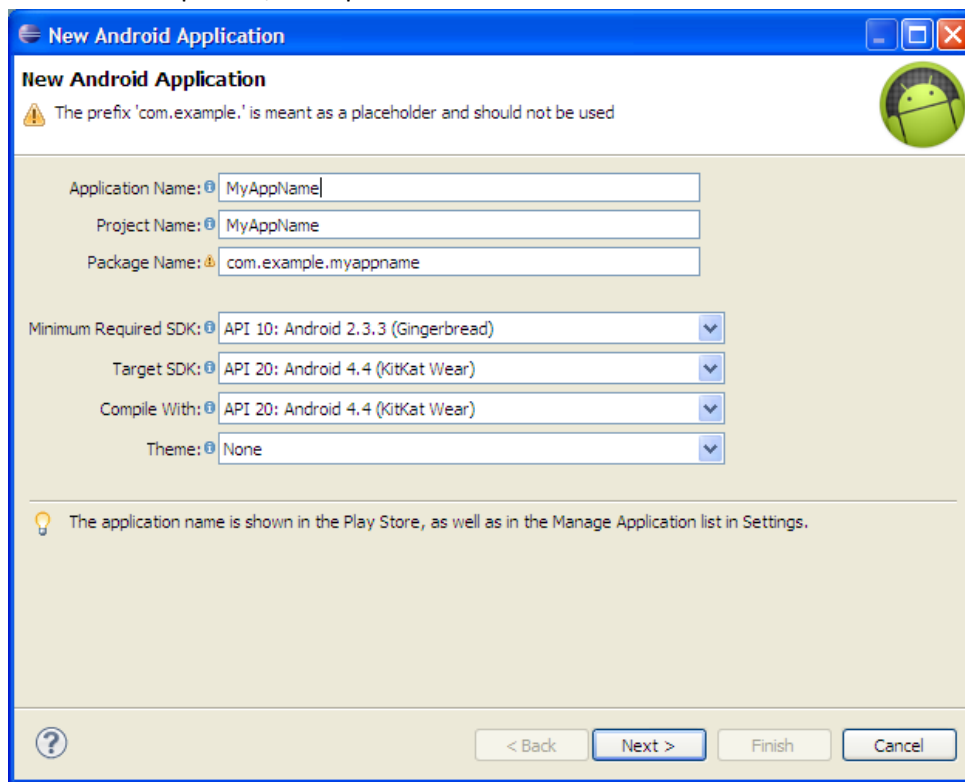


*Важно, чтобы разные программисты имели возможность использовать одинаковые имена классов в своих проектах. Имя домена сайта разработчика уникально по определению, поэтому размещение файлов в пакете с таким именем гарантирует отсутствие конфликтов. При этом для удобства имена пишутся в обратном порядке. При этом естественным образом соблюдается иерархичность. Например, следующий проект IT школы Samsung при импорте автоматически расположится в папке ru/samsung/itschool, рядом с TestBed.*

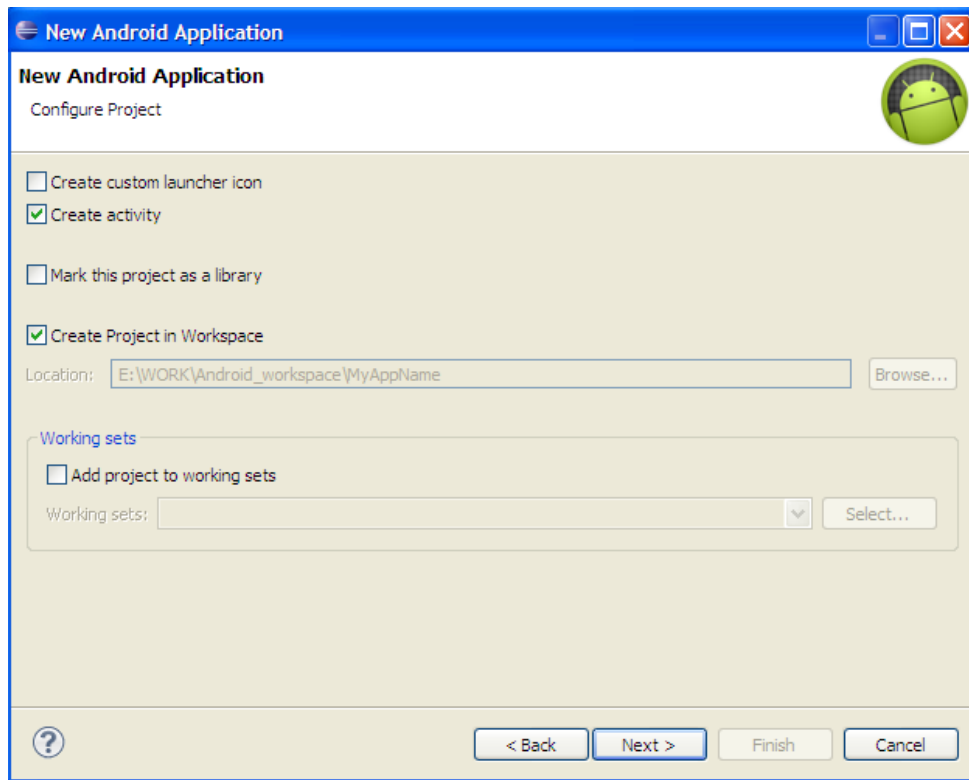
## 2.4.2. Создаем Android проект

### Как создать проект в Eclipse ADT

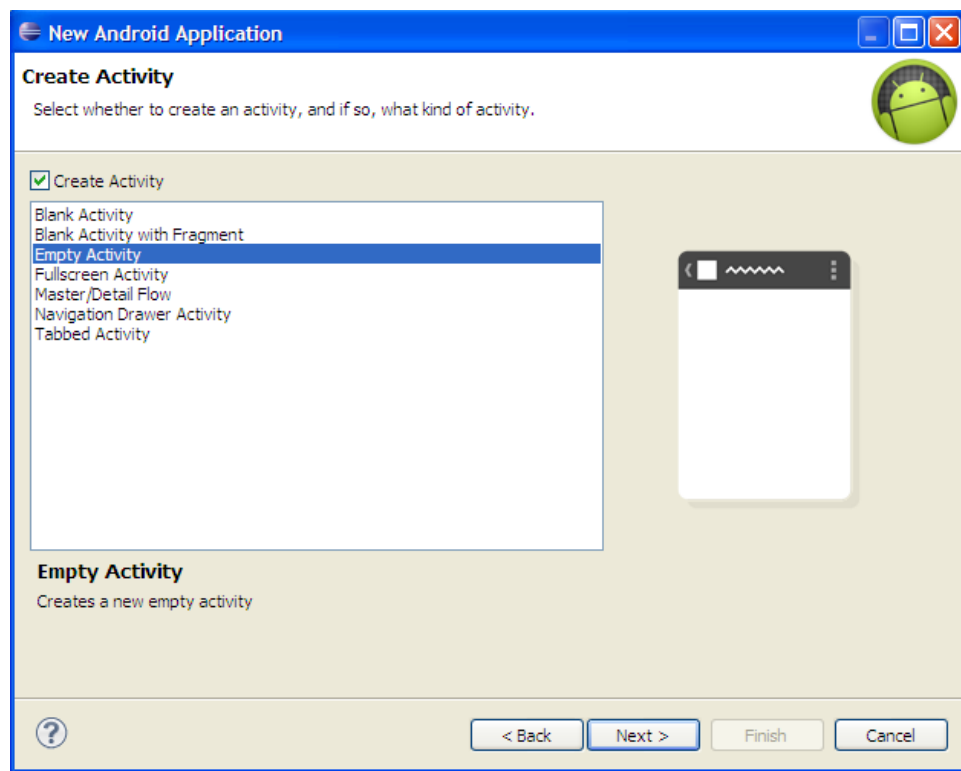
- 1) Открываем мастер создания проекта: **File⇒New ⇒Android Application Project**
- 2) Указываем имя проекта, номера API и отказываемся от темы.



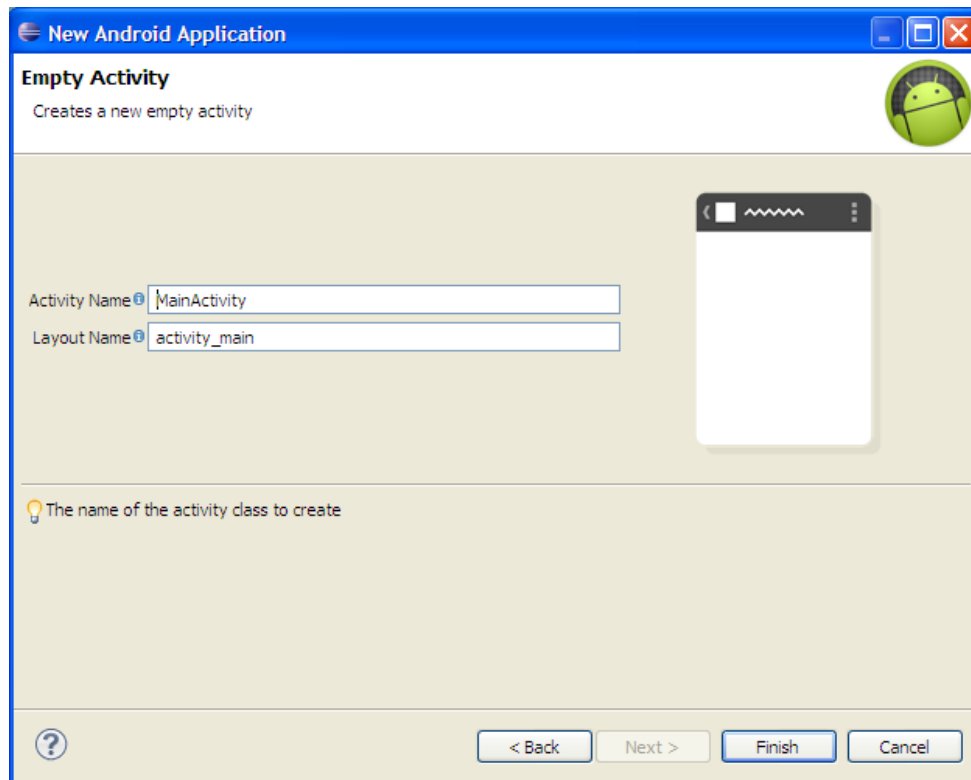
- 3) В следующем окне, приводим к такому виду и нажимаем Next.



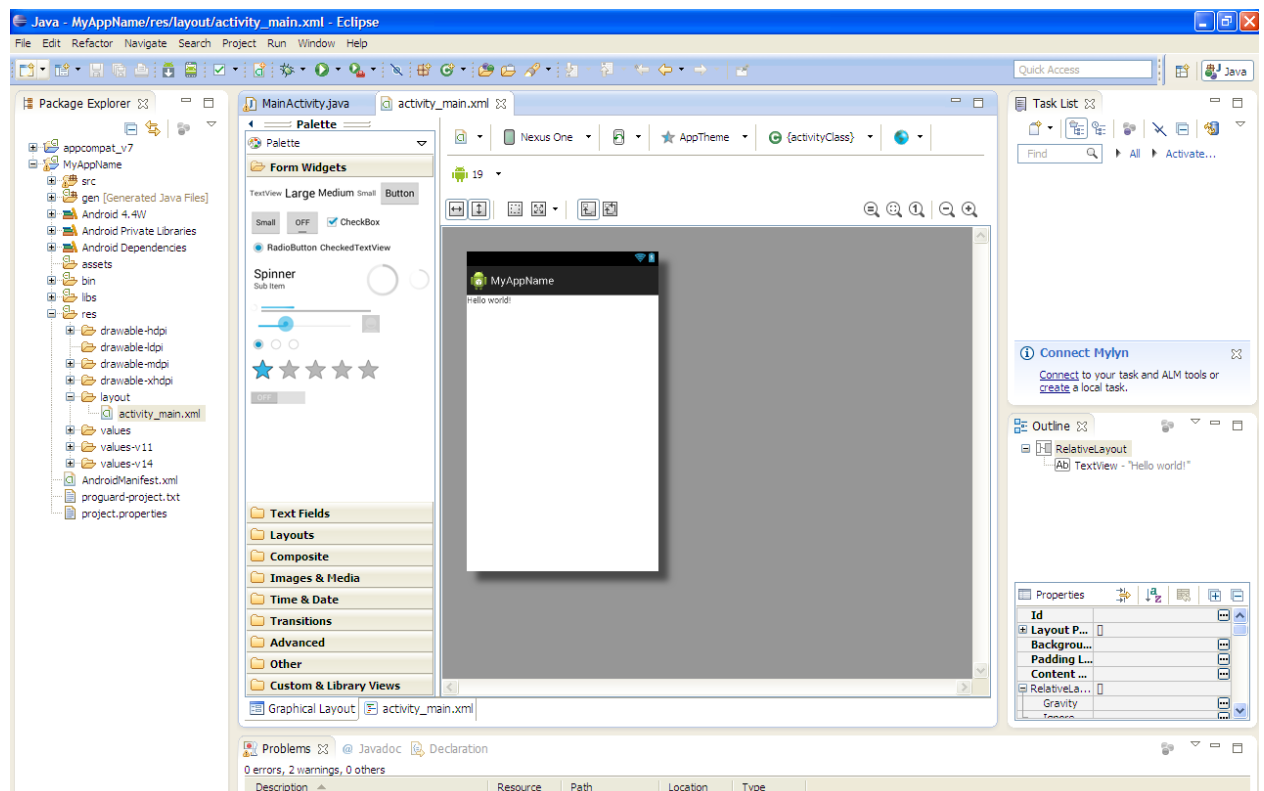
4) Выбираем пустую активность (Empty Activity)



5) Выбираем Finish

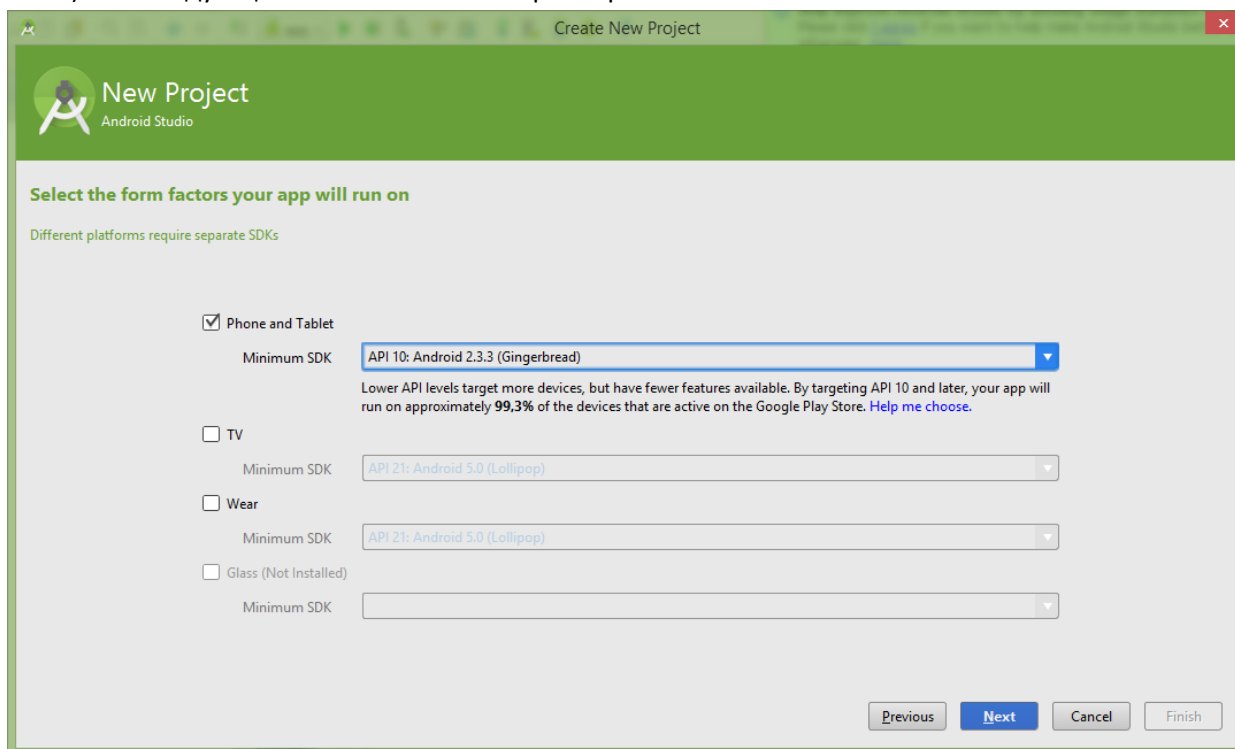


6) В созданном проекте открываем макет (layout) и выбираем версию API рендерера №19

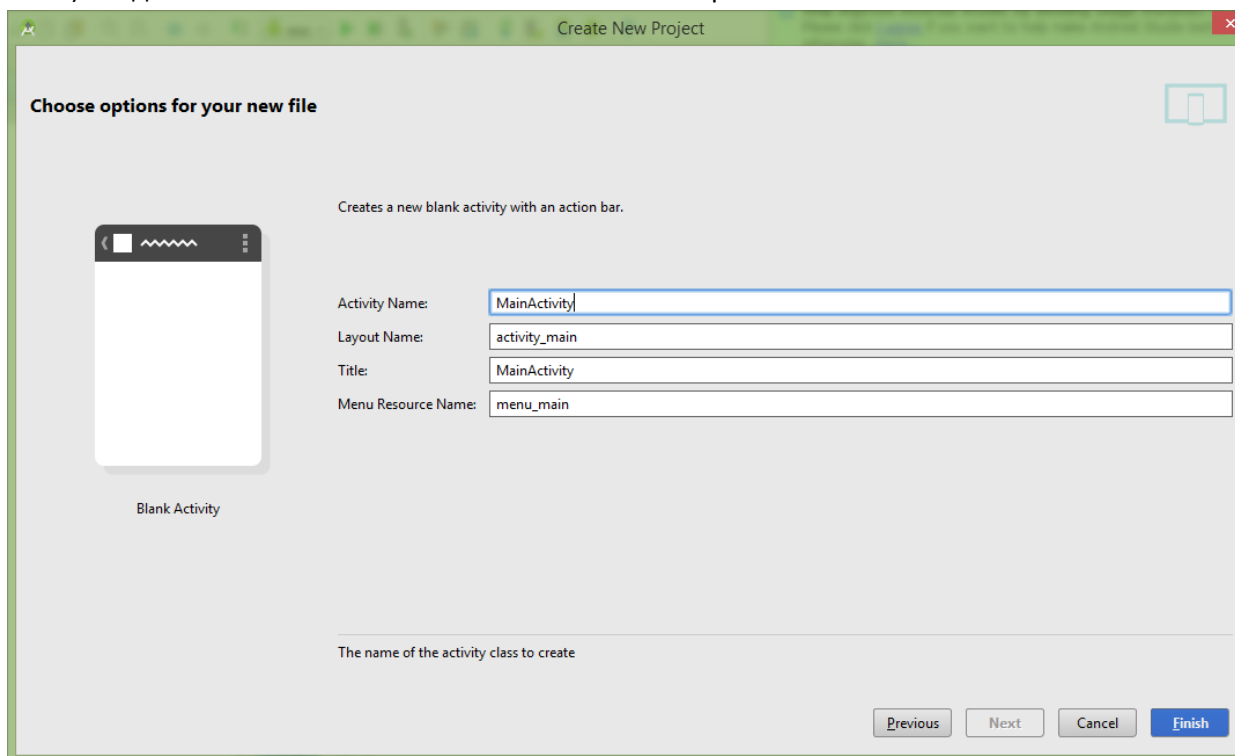


## Как создать проект в Android Studio

- 1) Открываем мастер создания проекта: **File⇒New ⇒Project**
- 2) Задаем имя , домен и папку для сохранения проекта
- 3) На следующем шаге оставляем параметры:

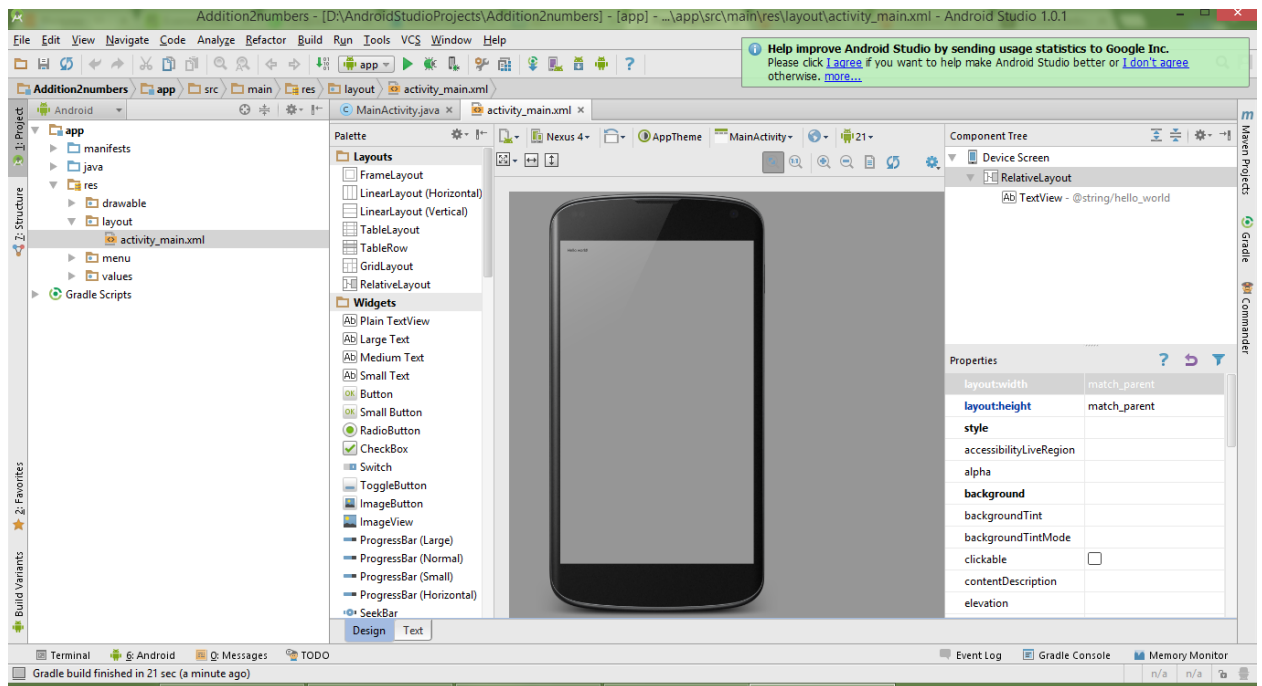


- 4) На следующем шаге выбираем проект Blank Activity.
- 5) Задаем наименования основных компонентов приложения:

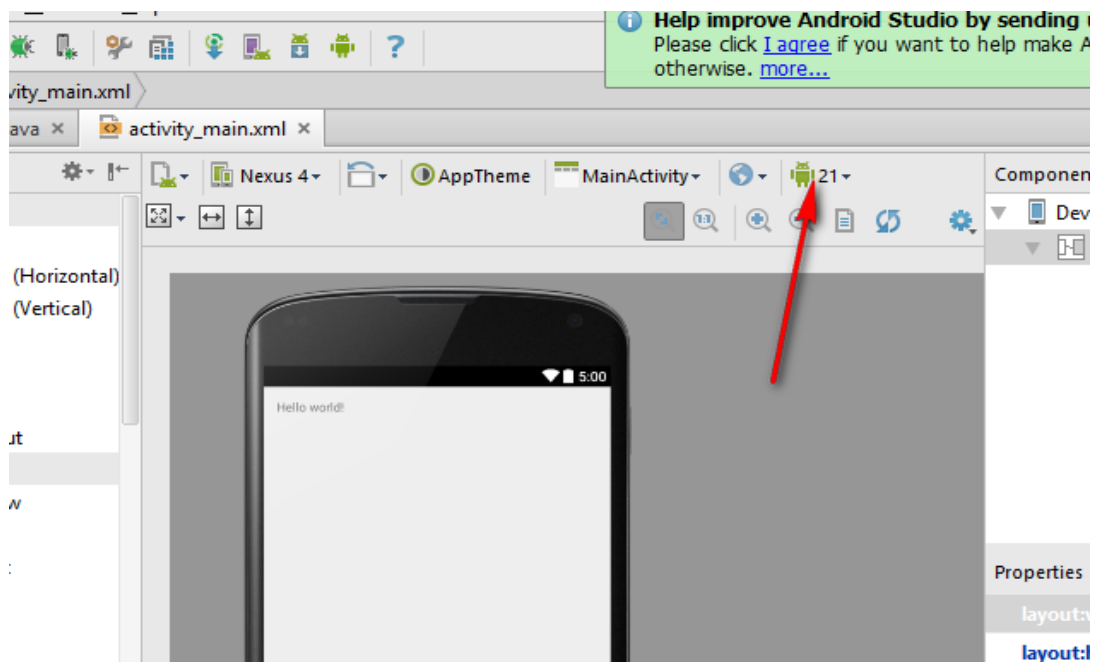


- 6) Завершаем создание пустого кнопкой Finish

В результате открывается окно проекта:



Если будут проблемы с отображением окна (как на рисунке), то переключитесь на другой API для отображения:



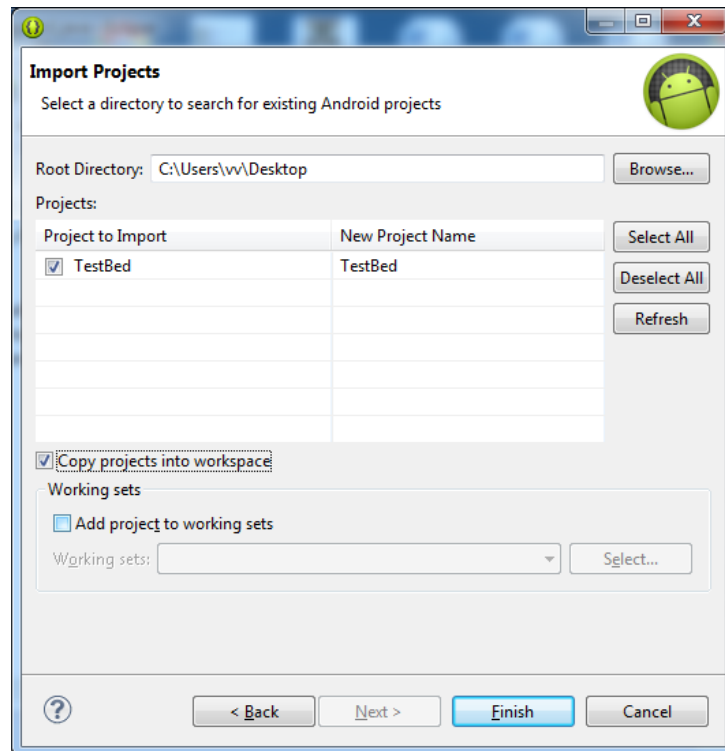
### Как загрузить существующий проект под Android

Для загрузки существующего проекта скопируйте проект из указанной преподавателем папки на сервере или системы обучения и импортируйте его.

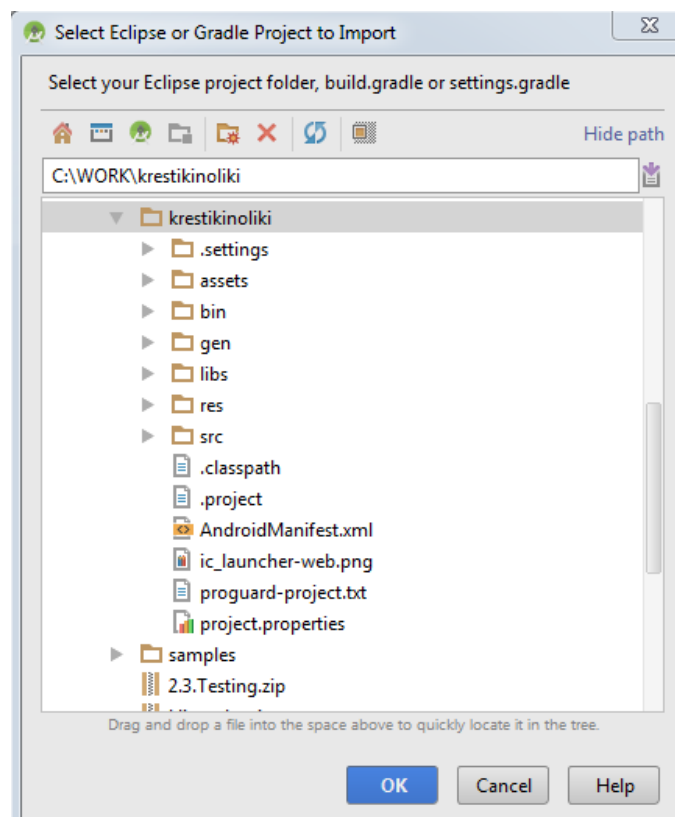
В Eclipse ADT необходимо в меню выбрать **File ⇒ Import**, в появившемся списке **Android ⇒ Existing Android Code into Workspace**

В появившемся окне желательно ставить флажок **Add project to working sets**, чтобы все проекты были в одном месте.





В Android Studio для импорта необходимо в меню выбрать **File ⇒ New ⇒ Import Project...**




При импорте проекта из Eclipse, дополнительно на следующем шаге надо указать директорию, куда перестроится проект под Android Studio. Можно использовать ту же, откуда производится импорт.

Проекты из обоих IDE в большинстве случаев совместимы и импортируются друг в друга.

### 2.4.3. Запуск приложения

Для запуска приложения можно использовать физическое Android устройство либо его эмулятор. При этом надо отметить, что эмулятор очень требователен к производительности компьютера, на котором работает IDE, и не всегда может заменить реальное устройство. Поэтому мы рекомендуем использовать первый способ.




	<p><b>Очень Важное Замечание!</b></p> <p>При работе с Android-проектами IDE заметно притормаживает. Нужно к этому очень терпеливо относиться. Android Studio, например, особенно долго грузиться в первый раз, потом это будет происходить быстрее. А Eclipse сразу после загрузки может показать много ошибок в проекте, но через полминуты все станет нормально. Если при этом немедленно пытаться исправить ситуацию, щелкая мышкой по управляющим элементам среды, и фактически, запуская новые команды, скорее всего Eclipse зависнет совсем и его придется перезагружать. Будьте терпеливы!</p>
---	---

#### Как подключить Android устройство

Прежде всего, нужно подготовить мобильное устройство к загрузке программ.

Для этого проверьте, что

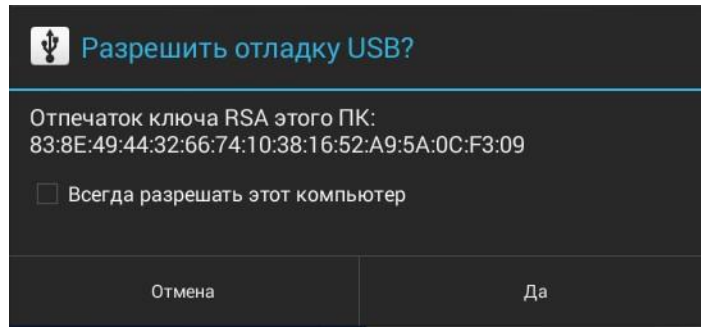
- на компьютере разработчика установлены драйвера для этого устройства;
- на самом устройстве включена отладка по USB (см. “Параметры разработчика” в Настройках<sup>1</sup>);
- устройство подключено к компьютеру проводом USB.


	<p><b>Эмулятор Android устройства</b></p> <p>В составе Android SDK имеется утилита Android Virtual Device (AVD)— эмулятор мобильного устройства, запускаемый на компьютере. Эмулятор нужен для отладки и тестирования приложения для разных устройств непосредственно в среде разработки. Можно создать несколько устройств с разными параметрами конфигурации и даже с разными версиями системы Android. Создать эмулятор можно в командной строке при помощи утилиты <code>android.bat</code>, находящейся в каталоге <code>tools</code> или с использованием Менеджера виртуальных устройств (AVD Manager), входящего в состав любой среды разработки для Android.</p> <p>В среде Eclipse менеджер виртуальных устройств AVD Manager вызывается командой меню <code>Window   Android Virtual Device Manager</code> либо кнопкой  в панели инструментов. В среде Android Studio AVD Manager вызывается командой меню <code>Tools -&gt; Android -&gt; AVD Manager</code> или кнопкой  в панели инструментов.</p>
---	---

<sup>1</sup> На некоторых устройствах с ОС Android 4.2 и выше в настройках отсутствует этот пункт. Чтобы его открыть, найдите пункт «Об устройстве» (или «О телефоне») в нем нажмите на «Номер сборки» 7 раз подряд, после чего откроется меню “Параметры разработчика”.

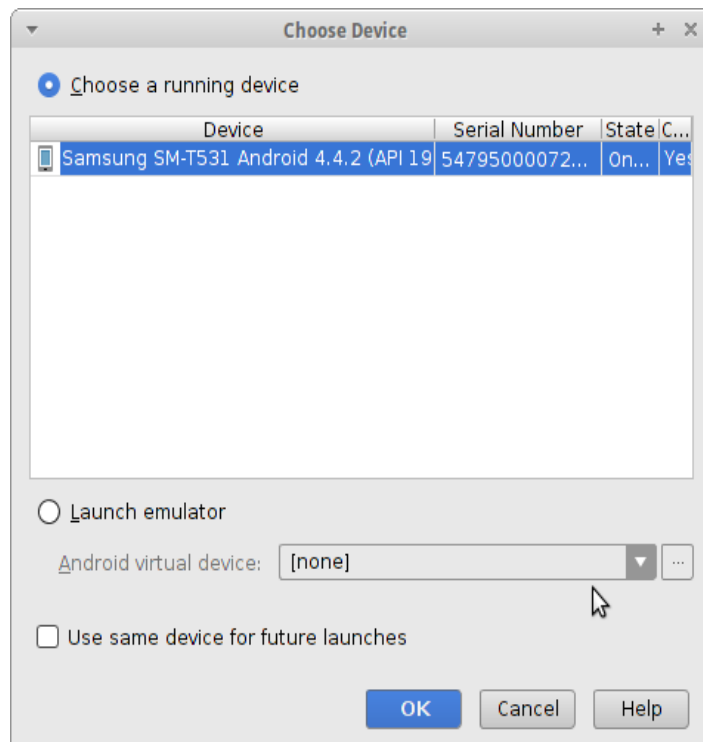
### Запуск приложения на Android устройстве

1. После подготовки Android устройства при подключении через USB на устройстве появится сообщение вида:



2. Запустите приложение в Android Studio нажав SHIFT+F10 или 

3. Далее Android Studio предложит выбрать устройство



4. А на экране устройства запустится приложение с сообщением “Hello world!”

Для Eclipse ADT шаги запуска практически аналогичны.

### 2.4.4. Структура проекта

Итак, что же получилось?

Как мы уже знаем, для отображения структуры проекта в Eclipse предусмотрено окно Package Explorer. Структура проекта меняется в зависимости от версии платформы, для которой ведется разработка (то есть от уровня API), но основные составляющие для всех проектов одинаковы.

Для Eclipse свойственно разделять составляющие проекта на:

- файлы кодов (программные java-файлы),
- файл разметки,
- файлы ресурсов (гипертекстовые xml-файлы).

Как уже говорилось, имена файла разметки и файла кодов можно задать при создании проекта. В нашем проекте это имена MainActivity.java и activity\_main.xml, которые Eclipse сгенерировал по умолчанию.

Файл java-кода MainActivity.java располагается в пакете проекта, содержащемся в папке src (у нас это пакет com.example.my\_aap). Именно в этом файле создаются классы объектов и описываются методы работы с ними.

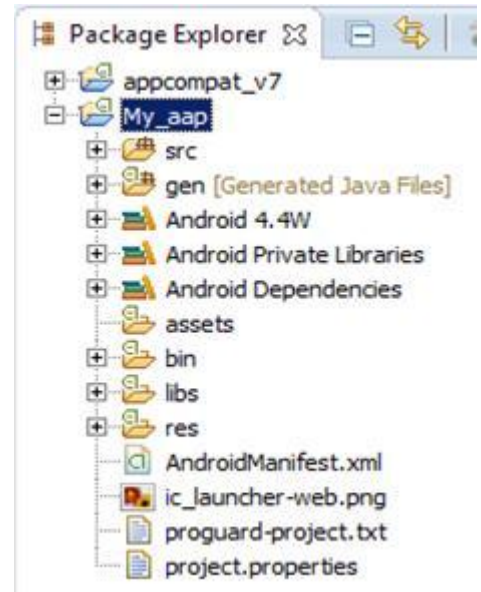
Файлы разметки и ресурсов проекта хранятся в папка res.

Ресурсных файлов в проекте много: это поддержка стилей, размеров и строковых констант для разных языковых настроек (папки values-\*), графики для разных параметров экрана (папки drawable-\*), разметки меню (папка menu) и непосредственно разметки элементов пользовательского интерфейса (папка layout), упакованного содержимого БД..

Отдельно расположенный файл AndroidManifest.xml содержит информацию о компонентах приложения, предназначенную для операционной системы. Файл манифеста позволяет определить, что за компонент и каковы условия, при которых он может быть запущен, по этому, если какой-либо компонент не описан в этом файле, то система не сможет с ним работать. Плагин ADT создает и редактирует файл манифеста автоматически, то есть нам не нужно заботиться о его заполнении.

Код приложения являет собой «активную» часть приложения. Ресурсы отделены от кода, что дает следующие преимущества:

- упрощается командная работа над проектом;
- изменение внешнего вида приложения зачастую вообще не требует модификации кода;
- локализация приложения для разных стран требует, как правило, всего лишь перевода строк из UI (хранящихся в строковых ресурсах) на язык нужной страны;
- упрощается адаптация внешнего вида приложения к разнообразным экранам.



## Упражнение 2.4.1

Редактирование файлов ресурсов

1. В созданном проекте My\_aap измените текстовые константы в файле res/values/string.xml:
  - "Hello, World" на строку "Hello, Name", вместо Name поставьте свое имя;
  - "app-name" на строку "PROJECT ANDROID".
2. Сохраните проект и запустите приложение на планшете.
3. Скопируйте папку res/values и назовите копию res/values-ru. Переведите значения строковых констант в файле string.xml на русский язык и сохраните проект. Переключите на планшете язык на русский и запустите приложение.

## 2.4.5. Активности (Activity)

Каждая **Активность** – это экран (по аналогии с web-формой), который приложение может показывать пользователям. Чем сложнее создаваемое приложение, тем больше экранов (Активностей) потребуется. При создании приложения потребуется, как минимум, начальный (главный) экран, который обеспечивает основу пользовательского интерфейса приложения. При необходимости этот интерфейс дополняется второстепенными Активностями, предназначенными для ввода информации, ее вывода и предоставления дополнительных возможностей. Запуск (или возврат из) новой Активности приводит к «перемещению» между экранами UI.

Большинство Активностей проектируются таким образом, чтобы использовать все экранное пространство, но можно также создавать полупрозрачные или плавающие диалоговые окна.

### Создание Активности

Для создания новой Активности наследуется класс **Activity** или его подкласс (ListActivity, FragmentActivity и т. п. ). Внутри реализации класса необходимо определить пользовательский интерфейс и реализовать требуемый функционал. В настоящий момент при создании Активности без Фрагментов Android SDK автоматически генерирует следующий код:

```
import android.app.Activity;
import android.os.Bundle;

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

Базовый класс Activity представляет собой пустой экран, который не особенно полезен, поэтому первое, что вам нужно сделать, это создать пользовательский интерфейс с помощью Представлений (View) и разметки (Layout).

**Представления** (View) – это элементы UI, которые отображают информацию и обеспечивают взаимодействие с пользователем. Android предоставляет несколько классов разметки (Layout), называемых также View Groups, которые могут содержать внутри себя несколько Представлений, для создания пользовательского интерфейса приложения.

Чтобы назначить пользовательский интерфейс для Активности, внутри обработчика событий onCreate используется метод setContentView().

### Жизненный цикл Активности

Приложения Android не могут контролировать свой жизненный цикл, ОС сама управляет всеми процессами и, как следствие, Активностями внутри них. При этом, состояние Активности помогает ОС определить приоритет родительского для этой Активности Приложения (Application). А приоритет Приложения влияет на то, с какой вероятности его работа (и работа дочерних Активностей) будет прервана системой.

## Стеки Активностей

Состояние каждой Активности определяется ее позицией в стеке (LIFO) Активностей, запущенных в данный момент. При запуске новой Активности представляемый ею экран помещается на вершину стека. Если пользователь нажимает кнопку «назад» или эта Активности закрывается каким-то другим образом, на вершину стека перемещается (и становится активной) нижележащая Активность.

На приоритет приложения влияет его самая приоритетная Активность. Когда диспетчер памяти ОС решает, какую программу закрыть для освобождения ресурсов, он учитывает информацию о положении Активности в стеке для определения приоритета приложения.

## Состояния Активностей

Активности могут находиться в одном из четырех возможных состояний:

**Активное (Active).** Активность находится на переднем плане (на вершине стека) и имеет возможность взаимодействовать с пользователем. Android будет пытаться сохранить ее работоспособность любой ценой, при необходимости прерывая работу других Активностей, находящихся на более низких позициях в стеке для предоставления необходимых ресурсов. При выходе на передний план другой Активности работа данной Активности будет приостановлена или остановлена.

**Приостановленное (Paused).** Активность может быть видна на экране, но не может взаимодействовать с пользователем: в этот момент она приостановлена. Это случается, когда на переднем плане находятся полупрозрачные или плавающие (например, диалоговые) окна. Работа приостановленной Активности может быть прекращена, если ОС необходимо выделить ресурсы Активности переднего плана. Если Активность полностью исчезает с экрана, она останавливается.

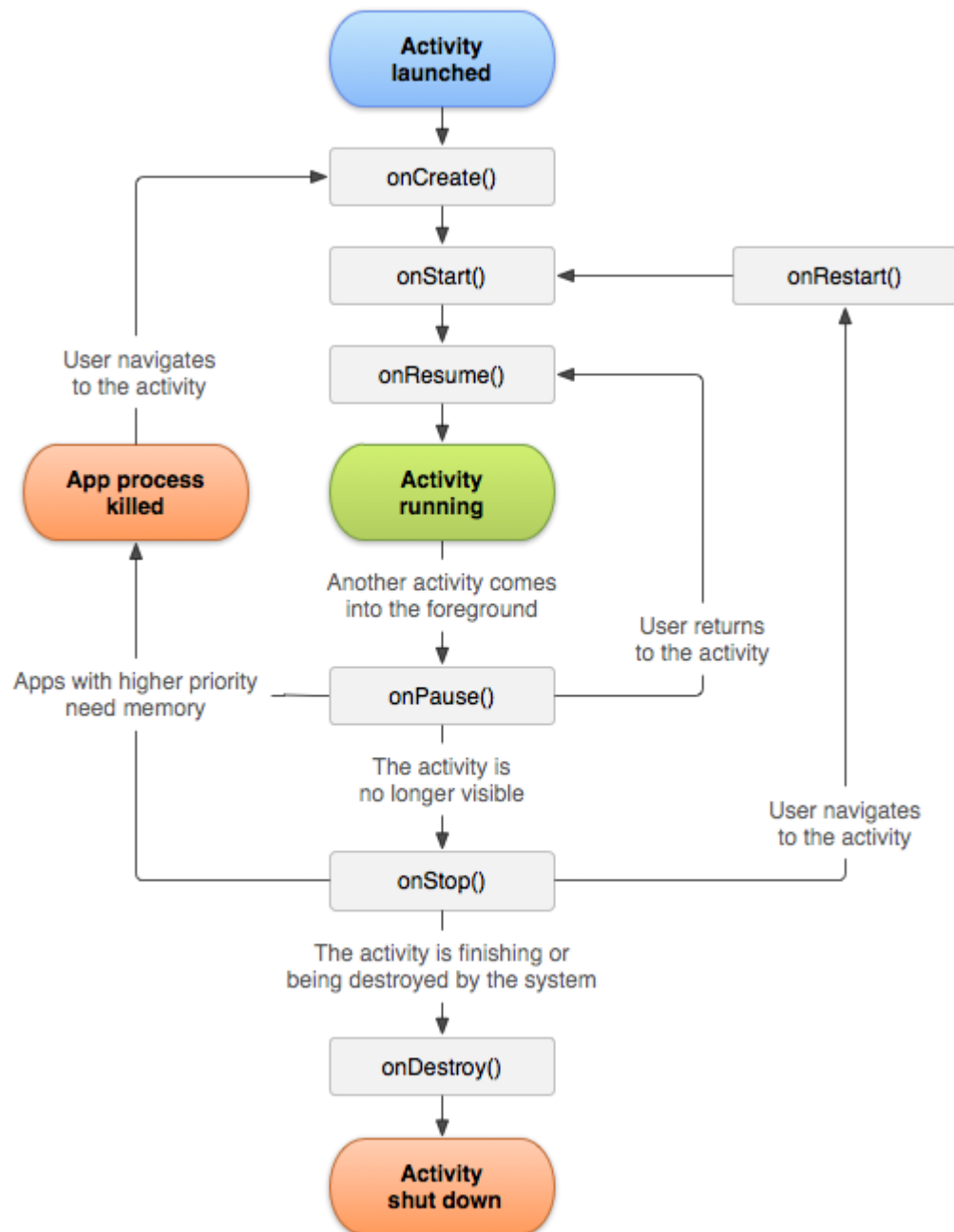
**Остановленное (Stopped).** Активность невидима, она находится в памяти, сохраняя информацию о своем состоянии. Такая Активность становится кандидатом на преждевременное закрытие, если системе потребуется память для чего-то другого. При остановке Активности разработчику важно сохранить данные и текущее состояние пользовательского интерфейса (состояние полей ввода, позицию курсора и т. д.). Если Активность завершает свою работу или закрывается, она становится неактивной.

**Неактивное (Inactive).** Когда работа Активности завершена, и перед тем, как она будет запущена, данная Активности находится в неактивном состоянии. Такие Активности удаляются из стека и должны быть (пере)запущены, чтобы их можно было использовать.

Изменение состояния приложения – недетерминированный процесс и управляется исключительно менеджером памяти Android. При необходимости Android вначале закрывает приложения, содержащие неактивные Активности, затем остановленные и, в крайнем случае, приостановленные.

Для обеспечения полноценного интерфейса приложения, изменения его состояния должны быть незаметными для пользователя. Меняя свое состояние с приостановленного на остановленное или с неактивного на активное, Активность не должна внешне меняться. При остановке или приостановке работы Активности разработчик приложения должен обеспечить сохранение состояния Активности, чтобы его можно было восстановить при выходе Активности на передний

план. Для этого в классе Activity имеются обработчики событий, переопределение которых позволяет разработчику отслеживать изменение состояний Активности.



### Отслеживание изменений состояния Активности

Обработчики событий класса Activity позволяют отслеживать изменения состояний соответствующего объекта **Activity** во время всего жизненного цикла. Ниже показан пример с заглушками для таких методов – обработчиков событий:

```

public class ExampleActivity extends Activity {
    // Вызывается при создании Активности
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Инициализирует Активность.
    }
}

```



```
// Вызывается после завершения метода onCreate
// Используется для восстановления состояния UI
@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    // Восстановить состояние UI из объекта savedInstanceState.
    // Данный объект также был передан методу onCreate.
}

// Вызывается когда Активность стала видимой
@Override
protected void onStart() {
    super.onStart();
    // Прodelать необходимые действия для Активности, видимой на экране
}

// Должен вызываться в начале видимого состояния.
// На самом деле Android вызывает данный обработчик только
// для Активностей, восстановленных из неактивного состояния
@Override
protected void onResume() {
    super.onResume();
    // Восстановить приостановленные обновления UI,
    // потоки и процессы, «замороженные, когда
    // Активность была в неактивном состоянии
}

// Вызывается перед выходом из активного состояния,
// позволяя сохранить состояние в объекте savedInstanceState
@Override
protected void onSaveInstanceState(Bundle savedInstanceState) {
    super.onSaveInstanceState(savedInstanceState);
    // Объект savedInstanceState будет в последующем
    // передан методам onCreate и onRestoreInstanceState
}

// Вызывается перед выходом из активного состояния
@Override
protected void onPause() {
    super.onPause();
    // «Заморозить» обновления UI, потоки или «трудоемкие» процессы,
    // ненужные, когда Активность не на переднем плане
}

// Вызывается перед выходом из видимого состояния
@Override
protected void onStop() {
    super.onStop();
    // «Заморозить» обновления UI, потоки или «трудоемкие» процессы,
    // ненужные, когда Активность не на переднем плане.
    // Сохранить все данные и изменения в UI.
}

// Вызывается перед уничтожением активности
@Override
```



```
protected void onDestroy() {  
    super.onDestroy();  
    // Освободить все ресурсы, включая работающие потоки, соединения с БД.  
}  
}
```

## Упражнение 2.4.2.

Отслеживание состояния активности

1. В проекте My\_aar из предыдущего упражнения откройте файл кода MainActivity.java и измените метод onCreate().

```
import android.os.Bundle;  
import android.support.v7.app.ActionBarActivity;  
  
public class MainActivity extends ActionBarActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Toast.makeText(this, "i creating", Toast.LENGTH_SHORT).show();  
    }  
}
```

Здесь команда `super.onCreate()` показывает, что данный метод наследуется от класса-родителя `Activity`, а объект `LENGTH_SHORT` из класса `android.widget.Toast` (класса всплывающих окон) — это текстовая строка "i creating", которая показывается короткий период времени.

2. Переопределите так же методы `onStart()`, `onPause()`, `onStop()`. Сохраните и запустите проект на планшете. Понаблюдайте за всплывающими окнами в процессе работы приложения. Изучите другие объекты класса `Toast` и посмотрите их работу в вашем приложении.

```
@Override  
protected void onStart() {  
    super.onStart();  
    Toast.makeText(this, "i starting", Toast.LENGTH_SHORT).show();  
}
```

## 2.4.6. \*Разбор приложения TestBed

Рассмотрим приложение TestBed:

- Интерфейс приложения состоит всего из трёх объектов:
  - **EditText** - поле для ввода текста.
  - **TextView** - поле для отображения текста (результат выполнения программы, отладочная информация)
  - **Button** - кнопка для закрытия приложения.

- Классы **AndroidInputStream** и **AndroidOutputStream** реализуют ввод/вывод в приложении и используются в главной активности приложения - **MainActivity**.

Разберём код **MainActivity**:

1. Весь пользовательский код в приложении TestBed содержится в методе **main** класса **MyProgram**.
2. Класс активности **MainActivity** содержит вспомогательный класс **RunUserProgram**. **RunUserProgram** перенаправляет ввод/вывод пользовательской программы в экземпляры классов **AndroidInputStream** / **AndroidOutputStream** и запускает функцию **main** из класса **MyProgram** код. Рассмотрим фрагмент класса **RunUserProgram** :

```
private class RunUserProgram extends AsyncTask<Void, Void, String> {

    @Override
    protected String doInBackground(Void... params) {

        try {
            // Связываются элементы UI и потоки ввода/вывода пользовательской программы
            MyProgram.in = new Scanner(in);
            MyProgram.out = new PrintStream(out);

            MyProgram.out.println("Program started.");
            MyProgram.main(new String[0]); // Запускается пользовательская программа
            MyProgram.out.println("The End.");
        } catch (Throwable error) {
            // Обработка ошибок
        }
        return getString(R.string.program_finished);
    }
}
```

3. Класс **MainActivity** связывает экземпляры классов **AndroidInputStream** и **AndroidOutputStream** с элементами пользовательского интерфейса (объектами **EditText** и **TextView**), добавляет обработчик события нажатия на **Enter** и параллельно создаёт экземпляр класса **RunUserProgram**. Рассмотрим фрагмент класса **MainActivity** :

```
public class MainActivity extends Activity {

    // Эти объекты используются для связи элементов UI и потоков ввода/вывода в
    // пользовательской программы. Созданные объекты используются в обработке нажатия на
    // Enter и в классе RunUserProgram
    public AndroidOutputStream out=new AndroidOutputStream(new TextFlusher(){...});
    public AndroidInputStream in = new AndroidInputStream();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        consoleWrite = (TextView) findViewById(R.id.consoleWrite);
        valuePrompt = (EditText) findViewById(R.id.valuePrompt);
        closeButton = (Button) findViewById(R.id.closeButton);

        hOut = new PrintoutHandler();
        hIn = new ScanInHandler();
    }
}
```

```

        (new RunUserProgram()).execute(); // Создаётся новый экземпляр класса
        RunUserProgram и вызывается метод doInBackground()

        // Создаётся обработчик нажатия на Enter: необходимо значение из EditText
        перенести во входной поток пользовательской программы и очистить поле EditText
        valuePrompt.setOnKeyListener(new View.OnKeyListener() {
            public boolean onKey(View v, int keyCode, KeyEvent event) {
                if (event.getAction() == KeyEvent.ACTION_DOWN
                    && (keyCode == KeyEvent.KEYCODE_ENTER)) {
                    String value = valuePrompt.getText().toString();
                    in.process(value);
                    valuePrompt.setText("");
                    valuePrompt.setVisibility(View.INVISIBLE);

                    return true;
                }
                return false;
            }
        });
    }
}

```

### Упражнение 2.4.3

Рассмотрим пример приложения, решающего линейное уравнение:

```

public class MainActivity extends ActionBarActivity {
    /** Вызывается при нажатии пользователем на кнопку Решить */
    public void solveEquation(View view) {
        double a = Double.parseDouble( ((EditText)
        findViewById(R.id.coefficient_a)).getText().toString());
        double b = Double.parseDouble( ((EditText)
        findViewById(R.id.coefficient_b)).getText().toString());
        double c = Double.parseDouble( ((EditText)
        findViewById(R.id.coefficient_c)).getText().toString());
        TextView result = (TextView) findViewById(R.id.result);
        result.setText("" + String.valueOf((c - b) / a));
    }
    /* Все последующее является шаблоном, сгенерированным средой разработки.
    * Не стоит прямо сейчас разбираться во всех подробностях.
    * Этот код при загрузке читает конфигурационные xml файлы и создает необходимые Java
    объекты.
    */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                .add(R.id.container, new PlaceholderFragment())
                .commit();
        }
    }

    @Override

```

```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}

public static class PlaceholderFragment extends Fragment {

    public PlaceholderFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_main, container, false);
        return rootView;
    }
}
```

## Задание 2.4.1.

Модифицировать приложение так, чтобы оно решало квадратное уравнение, а так же корректно обрабатывало ситуацию нулевых коэффициентов.

## Благодарности

Компания Samsung Electronics выражает благодарность за участие в подготовке данного материала преподавателям ИТ ШКОЛЫ SAMSUNG Коноркину Ивану Олеговичу, Дружинской Елене Владимировне, Салпагарову Солтану Исмаиловичу и Сергиенко Антону Борисовичу.