2-ЛЕКЦИЯ. ПРИЛОЖЕНИЯ В МОБИЛЬНЫХ УСТРОЙСТВАХ

2.1.Основные виды Android-приложений

Приступая к разработке мобильных приложений хорошо бы иметь представление о том, какие виды приложений существуют. Дело в том, что если удастся определить к какому типу относится приложение, то становится понятнее на какие моменты в процессе его разработки необходимо обращать основное внимание. Можно выделить следующие виды приложений:

- **Приложения переднего плана** выполняют свои функции только, когда видимы на экране, в противном же случае их выполнение приостанавливается. Такими приложениями являются, например, игры, текстовые редакторы, видеопроигрыватели. При разработке таких приложений необходимо очень внимательно изучить жизненный цикл активности, чтобы переключения в фоновый режим и обратно проходили гладко (бесшовно), т. е. при возвращении приложения на передний план было незаметно, что оно вообще куда-то пропадало. Для достижения этой гладкости необходимо следить за тем, чтобы при входе в фоновый режим приложение сохраняло свое состояние, а при выходе на передний план восстанавливало его. Еще один важный момент, на который обязательно надо обратить внимание при разработке приложений переднего плана, удобный и интуитивно понятный интерфейс[1].
- **Фоновые приложения** после настройки не предполагают взаимодействия с пользователем, большую часть времени находятся и работают в скрытом состоянии. Примерами таких приложений могут служить, службы экранирования звонков, SMS-автоответчики. В большинстве своем фоновые приложения нацелены на отслеживание событий, порождаемых аппаратным обеспечением, системой или другими приложениями, работают незаметно. Можно создавать совершенно невидимые сервисы, но тогда они будут неуправляемыми. Минимум действий, которые необходимо позволить пользователю: санкционирование запуска сервиса, настройка, приостановка и прерывание его работы при необходимости.
- **Смешанные приложения** большую часть времени работают в фоновом режиме, однако допускают взаимодействие с пользователем и после настройки. Обычно взаимодействие с пользователем сводится к уведомлению о каких-либо событиях. Примерами таких приложений могут служить мультимедиа-проигрыватели, программы для обмена текстовыми сообщениями (чаты), почтовые клиенты. Возможность реагировать на пользовательский ввод и при этом не терять работоспособности в фоновом режиме является характерной особенностью смешанных приложений. Такие приложения обычно содержат как видимые активности, так и скрытые (фоновые) сервисы, и при взаимодействии с пользователем должны учитывать свое текущее состояние. Возможно, потребуется обновлять графический интерфейс, если приложение находится на переднем плане, или же посылать пользователю уведомления из фонового режима, чтобы держать его в курсе происходящего. И эти особенности необходимо учитывать при разработке подобных приложений.
- **Виджеты** - небольшие приложения, отображаемые в виде графического объекта на рабочем столе. Примерами могут служить, приложения для отображения динамической информации, такой как заряд батареи, прогноз погоды, дата и время. Разумеется, сложные приложения могут содержать элементы каждого из рассмотренных видов. Планируя разработку приложения, необходимо

определить способ его использования, только после этого приступать к проектированию и непосредственно разработке.

2.2.Архитектура приложения, основные компоненты

Архитектура Android приложений основана на идее многократного использования компонентов, которые являются основными строительными блоками. Каждый компонент является отдельной сущностью и помогает определить общее поведение приложения.

Система Android выстроена таким образом, что любое приложение может запускать необходимый компонент другого приложения. Например, если приложение предполагает использование камеры для создания фотографий, совершенно необязательно создавать в этом приложении активность для работы с камерой. Наверняка на устройстве уже есть приложение для получения фотографий с камеры, достаточно запустить соответствующую активность, сделать фотографию и вернуть ее в приложение, так что пользователь будет считать, что камера часть приложения, с которым он работает.

Когда система запускает компонент, она запускает процесс приложения, которому принадлежит компонент, если он еще не запущен, и создает экземпляры классов, необходимых компоненту. Поэтому в отличие от большинства других систем, в системе Android приложения не имеют единой точки входа (нет метода main(), например). В силу запуска каждого приложения в отдельном процессе и ограничений на доступ к файлам, приложение не может напрямую активировать компонент другого приложения. Таким образом для активации компонента другого приложения необходимо послать системе сообщение о намерении запустить определенный компонент, система активирует его.

Можно выделить четыре различных типа компонентов, каждый тип служит для достижения определенной цели и имеет свой особый жизненный цикл, который определяет способы создания и разрушения соответствующего компонента. Рассмотрим основные компоненты Android-приложений.

Активности (Activities). Активность - это видимая часть приложения (экран, окно, форма), отвечает за отображение графического интерфейса пользователя. При этом приложение может иметь несколько активностей, например, в приложении, предназначенном для работы с электронной почтой, одна активность может использоваться для отображения списка новых писем, другая активность - для написания, и еще одна - для чтения писем. Несмотря на то, что для пользователя приложение представляется единым целым, все активности приложения не зависят друг от друга. В связи с этим любая из этих активностей может быть запущена из другого приложения, имеющего доступ к активностям данного приложения. Например, приложение камеры может запустить активность, создающую новые письма, чтобы отправить только что сделанную фотографию адресату, указанному пользователем.

Сервисы (Services). Сервис - компонент, который работает в фоновом режиме, выполняет длительные по времени операции или работу для удаленных процессов. Сервис не предоставляет пользовательского интерфейса. Например, сервис может проигрывать музыку в фоновом режиме, пока пользователь использует другое приложение, может загружать данные из сети, не блокируя взаимодействие пользователя с активностью. Сервис может быть запущен другим компонентом и после этого работать самостоятельно, а может остаться связанным с этим компонентом и взаимодействовать с ним.

Контент-провайдеры (Content providers). Контент-провайдер управляет распределенным множеством данных приложения. Данные могут храниться в файловой системе, в базе данных SQLite, в сети, в любом другом доступном для приложения месте. Контент-провайдер позволяет другим приложениям при наличии у них соответствующих прав делать запросы или даже менять данные. Например, в системе Android есть контент-провайдер, который управляет информацией о контактах пользователя. В связи с этим, любое приложение с соответствующими правами может сделать запрос на чтение и запись

информации какого-либо контакта. Контент-провайдер может быть также полезен для чтения и записи приватных данных приложения, не предназначенных для доступа извне.

Приемники широковещательных сообщений (Broadcast Receivers). Приемник - компонент, который реагирует на широковещательные извещения. Большинство таких извещений порождаются системой, например, извещение о том, что экран отключился или низкий заряд батареи. Приложения также могут инициировать широковещание, например, разослать другим приложениям сообщение о том, что некоторые данные загружены и доступны для использования. Хотя приемники не отображают пользовательского интерфейса, они могут создавать уведомление на панели состояний, чтобы предупредить пользователя о появлении сообщения. Такой приемник служит проводником к другим компонентам и предназначен для выполнения небольшого объема работ, например, он может запустить соответствующий событию сервис.

Все рассмотренные компоненты являются наследниками классов, определенных в Android SDK.

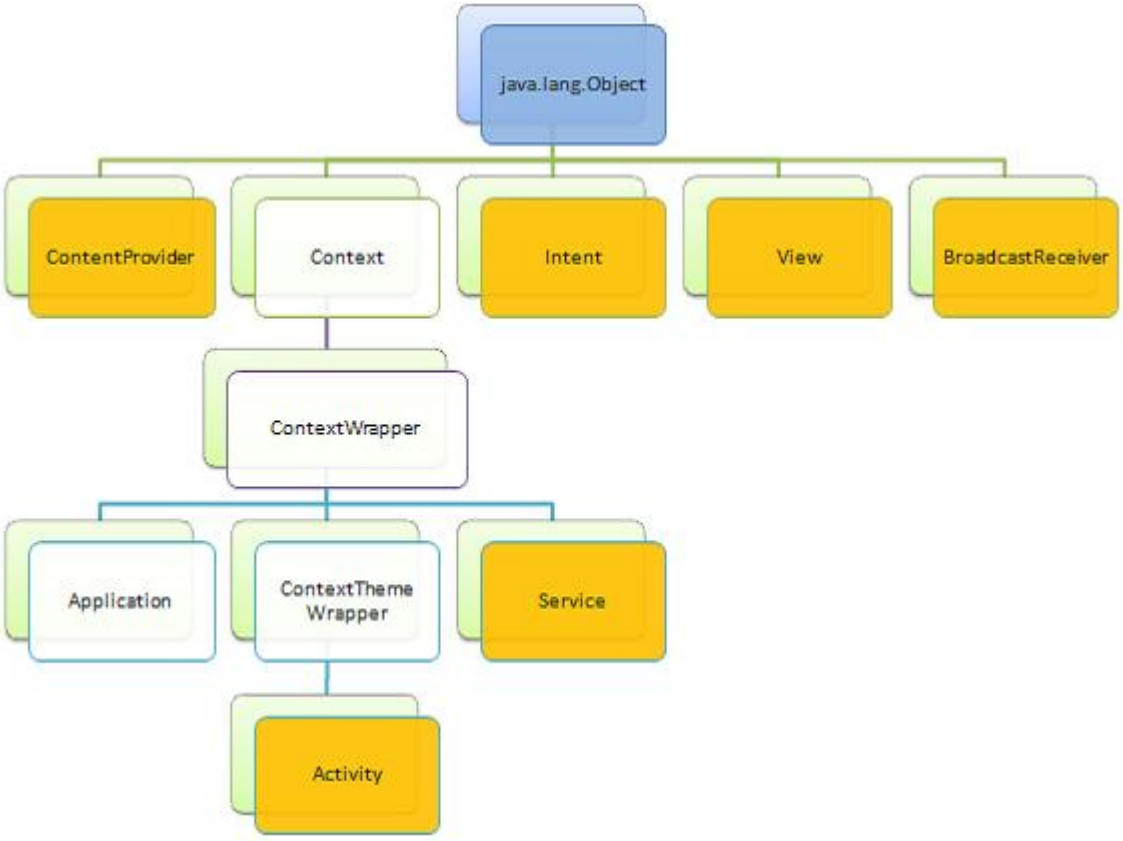


Рис. 2.3. Иерархия классов Android SDK

На рис. 2.3 показана иерархия основных классов Android SDK, с которыми обычно имеет дело разработчик. На самом деле классов намного больше, желтым цветом выделены классы, с которыми разработчик работает непосредственно, наследует от них свои классы. Остальные классы не менее важны, но они реже используются напрямую. Для начала рассмотрим классы Intent и View.

Класс View является основным строительным блоком для компонентов пользовательского интерфейса (UI), он определяет прямоугольную область экрана и отвечает за прорисовку и обработку событий. Является базовым классом для виджетов (GUI widgets), которые используются для создания интерактивных компонентов пользовательского интерфейса: кнопок, текстовых полей и т. д. А также является базовым классом для класса ViewGroup, который является невидимым контейнером для других контейнеров и виджетов, определяет свойства расположения компонентов

пользовательского интерфейса. Интерфейс Android-приложения представляет собой иерархию UI компонентов (см. рис. 2.4), можно описать эту иерархию программно, но более простым и эффективным способом задать расположение элементов интерфейса является XML файл, который предоставляет удобную для восприятия структуру компоновки (layout file). Во время исполнения XML файл автоматически превращается в дерево соответствующих объектов.

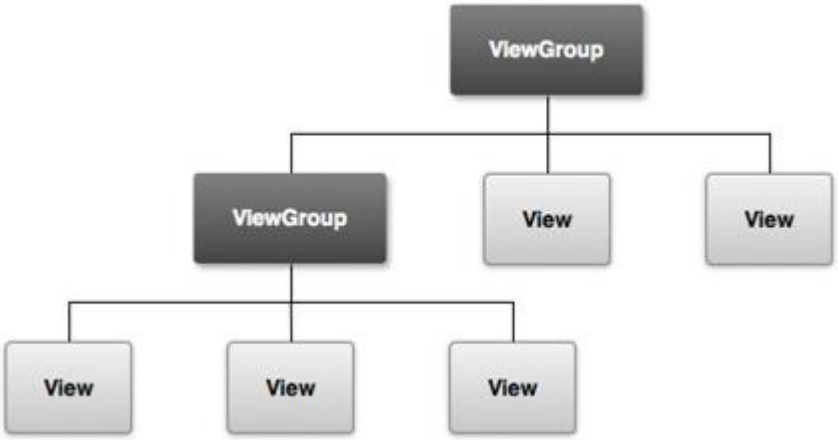


Рис. 2.4. Иерархия компонентов, определяющая компоновку интерфейса пользователя

Объекты-экземпляры класса Intent используются для передачи сообщений между основными компонентами приложений. Известно, что три из четырех основных компонентов: активности, сервисы и приемники широковещательных сообщений, могут быть активированы с помощью сообщений, которые называются намерениями. Такие сообщения являются инструментом позднего связывания компонентов одного или нескольких приложений. Экземпляр класса Intent представляет собой структуру данных, содержащую описание операции, которая должна быть выполнена, и обычно используется для запуска активности или сервиса. В случае с приемниками широковещательных сообщений объект Intent содержит описание события, которое произошло или было объявлено.

Для каждого типа компонентов существуют свои механизмы передачи намерений.

- Чтобы запустить активность или вызвать у работающей активности новое действие, необходимо передать объект-намерение в метод Context.startActivity() или Activity.startActivityForResult().
- Чтобы запустить сервис или доставить новые инструкции работающему сервису, необходимо передать объект-намерение в метод Context.startService(). Также объект-намерение может быть передан в метод Context.bindService(), чтобы связать между собой вызывающий компонент и сервис.
- Чтобы доставить объект-намерение всем заинтересованным приемникам широковещательных сообщений, необходимо передать его в любой из широковещательных методов: Context.sendOrderedBroadcast(), Context.sendStickyBroadcast(), Context.sendBroadcast().

В каждом случае система Android в ответ на намерение находит соответствующий компонент: активность, сервис или множество широковещательных приемников и запускает его если необходимо. В этой системе сообщений не случается накладок: сообщение-намерение, отправленное определенному компоненту, будет получено именно этим компонентом и никем другим.

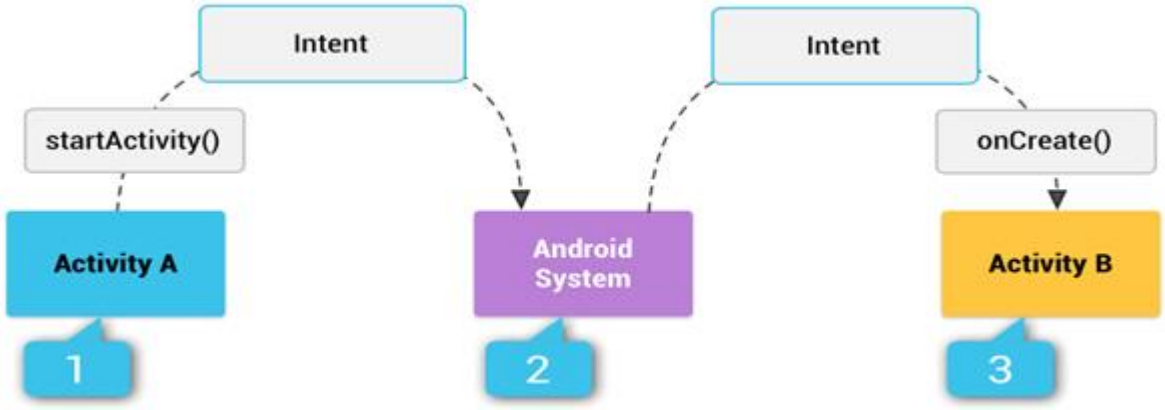


Рис. 2.5. Передача намерений (Intent)

На рис. 2.5 можно увидеть как происходит передача намерений (Intent), в данном случае одна активность запускает другую. Активность А создает намерение (Intent) с описанием действия и передает его в метод startActivity(). Система Android проверяет все приложения на совпадение с намерением, когда совпадение найдено, система запускает соответствующую активность, для чего вызывает метод onCreate() и передает в него объект-намерение Intent.

Обратим внимание на организацию исполнения приложений в ОС Android. Как уже было отмечено приложения под Android разрабатываются на языке программирования Java, компилируется в файл с расширением .apk, после этот файл используется для установки приложения на устройства, работающие под управлением Android. После установки каждое Android приложение "живет" в своей собственной безопасной "песочнице", рассмотрим, как это выглядит:

- операционная система Android является многопользовательской ОС, в которой каждое приложение рассматривается как отдельный пользователь;
- по умолчанию, система назначает каждому приложению уникальный пользовательский ID, который используется только системой и неизвестен приложению;
- система устанавливает права доступа ко всем файлам приложения следующим образом: доступ к элементам приложения имеет только пользователь с соответствующим ID;
- каждому приложению соответствует отдельный Linux процесс, который запускается, как только это необходимо хотя бы одному компоненту приложения, процесс прекращает работу, когда ни один компонент приложения не использует его или же системе требуется освободить память для других (возможно, более важных) приложений;
- каждому процессу соответствует отдельный экземпляр виртуальной машины Dalvik, в связи с этим код приложения исполняется изолировано от других приложений.

Перечисленные идеи функционирования приложения в ОС Android реализуют принцип минимальных привилегий, т. е. каждому приложению, по умолчанию, разрешен доступ только к компонентам, необходимым для его работы и никаким больше. Таким образом обеспечивается очень безопасная среда функционирования приложений.

Однако, в случае необходимости приложения могут получить доступ к данным других приложений и системным сервисам (услугам). В случае, когда двум приложениям необходимо иметь доступ к файлам друг друга, им присваивается один и тот же пользовательский ID. Для экономии системных ресурсов такие приложения запускаются в одном Linux процессе и делят между собой один и тот же экземпляр виртуальной машины, в этом случае приложения также должны быть подписаны одним сертификатом.

В случае же, когда приложению требуется доступ к системным данным, например, контактам, SMS сообщениям, картам памяти, камере, Bluetooth и т. д., пользователю необходимо дать приложению такие полномочия во время установки его на устройство.

2.3. Установка и настройка инструментальных средств

Большинство приложений для OS Android написано на Java. Одной из самых популярных сред разработки является Eclipse (для неё также необходим JDK) с установленным плагином ADT и Android SDK. Раньше приходилось ставить все компоненты отдельно. Сейчас появилась версия среды Eclipse с уже настроенными дополнениями - ADT Bundle. Здесь есть минимум инструментов, необходимый для разработки приложений. С этой версией мы и будем работать. Однако в ней есть далеко не всё, поэтому, если при разработке какого-либо проекта вам потребуются инструменты, не входящие в ADT Bundle, вы можете скачать их с сайта разработчиков и дополнить свою среду. Скачать среду можно с сайта для разработчиков Android (http://developer.android.com/sdk/index.html).

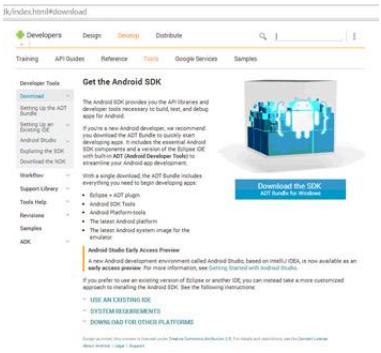


Рис. 1. Сайт разработчика

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

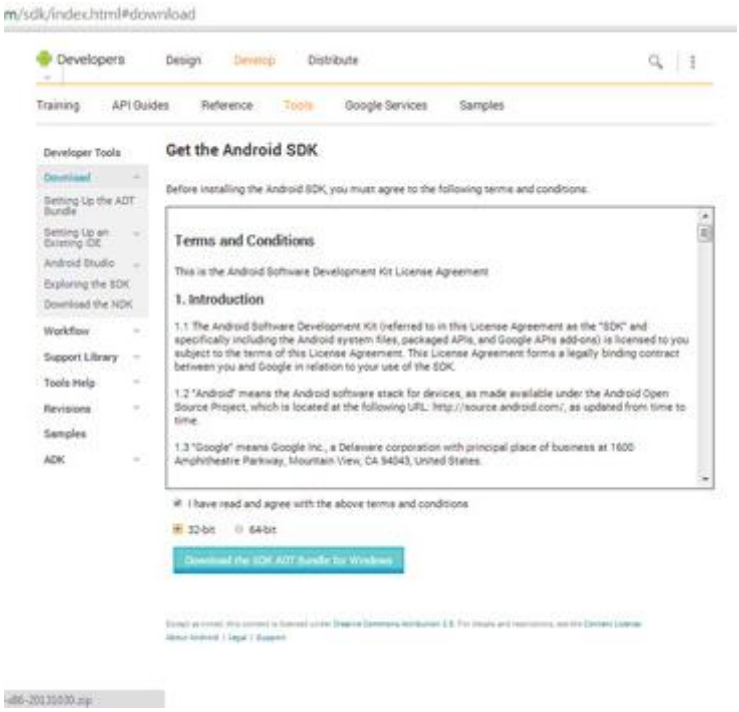


Рис. 2. Скачивание среды

После скачивания распакуйте архив в ту папку, где собираетесь работать (среда не требует специальной установки). После распаковки зайдите в папку и запустите Eclipse. Здесь возможна небольшая проблема: если у вас не установлен JDK, среда не запустится и потребует указать путь к папке с JDK или установить его. Скачать JDK можно с сайта Oracle.

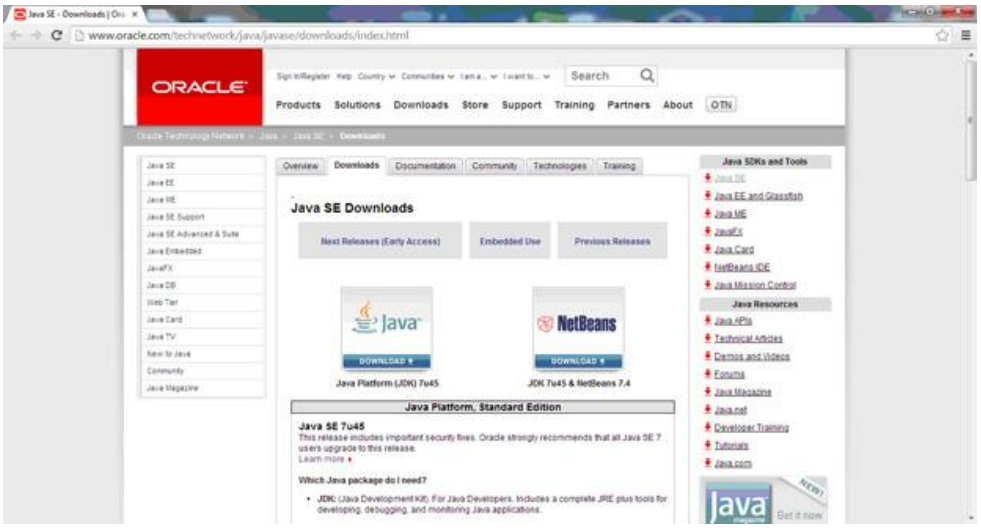


Рис. 3. Сайт компании Oracle

Чтобы скачать JDK нужно сначала принять условия лицензионного соглашения, а затем выбрать нужную версию.

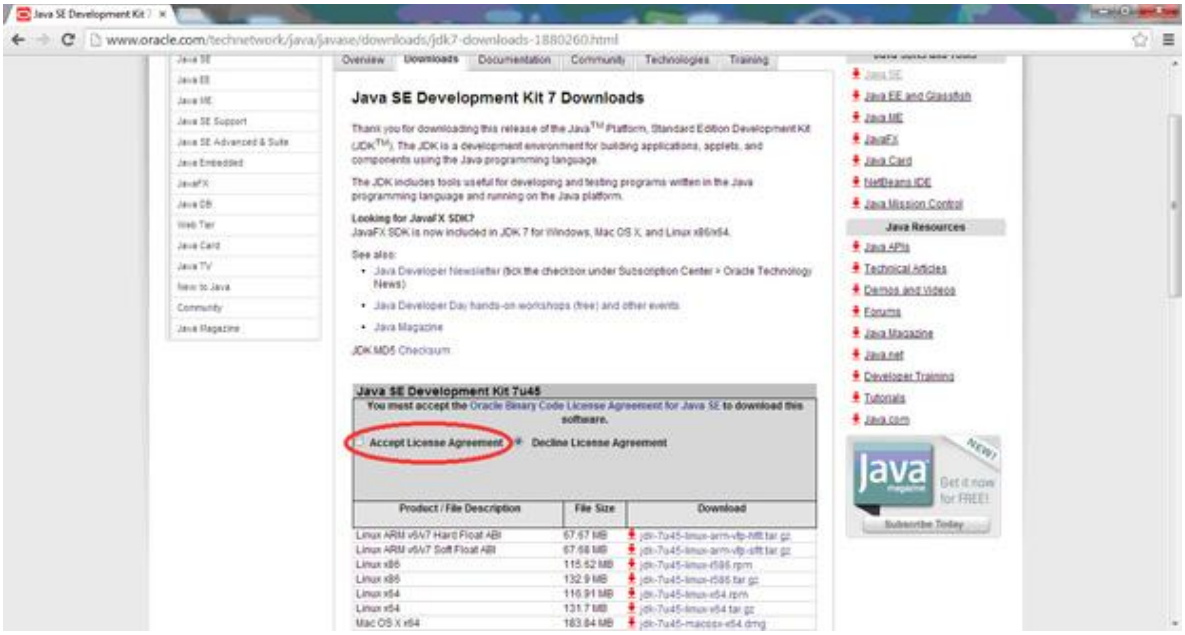


Рис. 4. Скачивание JDK

После скачивания запустите setup-файл и установите JDK.

Рис. 5. Установка JDK

После установки JDK среда должна запуститься. 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 Eclipse.

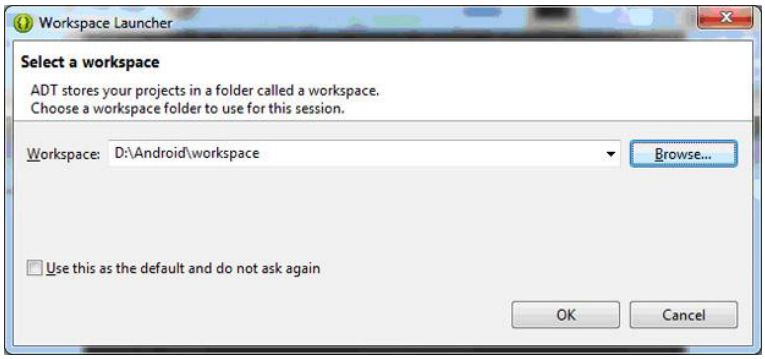


Рис. 6. Выбор рабочего пространства

Затем появляется окно, в котором разработчики предлагают отправлять статистику для дальнейшего улучшения SDK. Вы можете согласиться или отказаться.

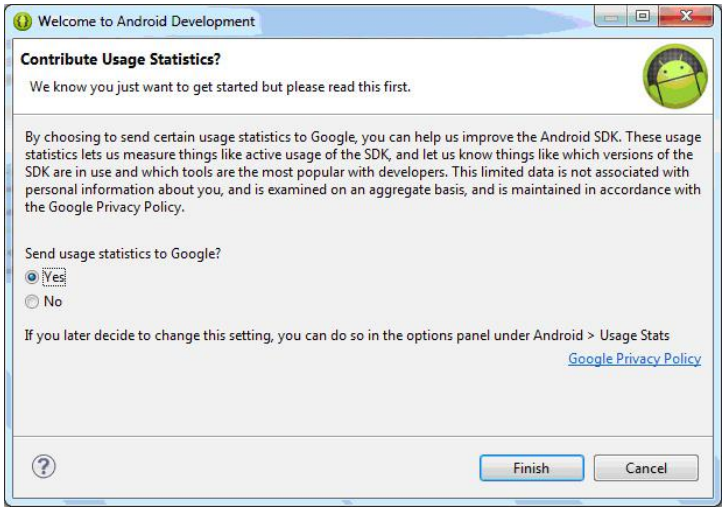


Рис. 7. Отправка статистики

Обратите внимание на значок Android SDK Manager, находящийся на панели инструментов (его также можно найти в меню Window). С его помощью вы сможете добавлять в свою среду новые инструменты.

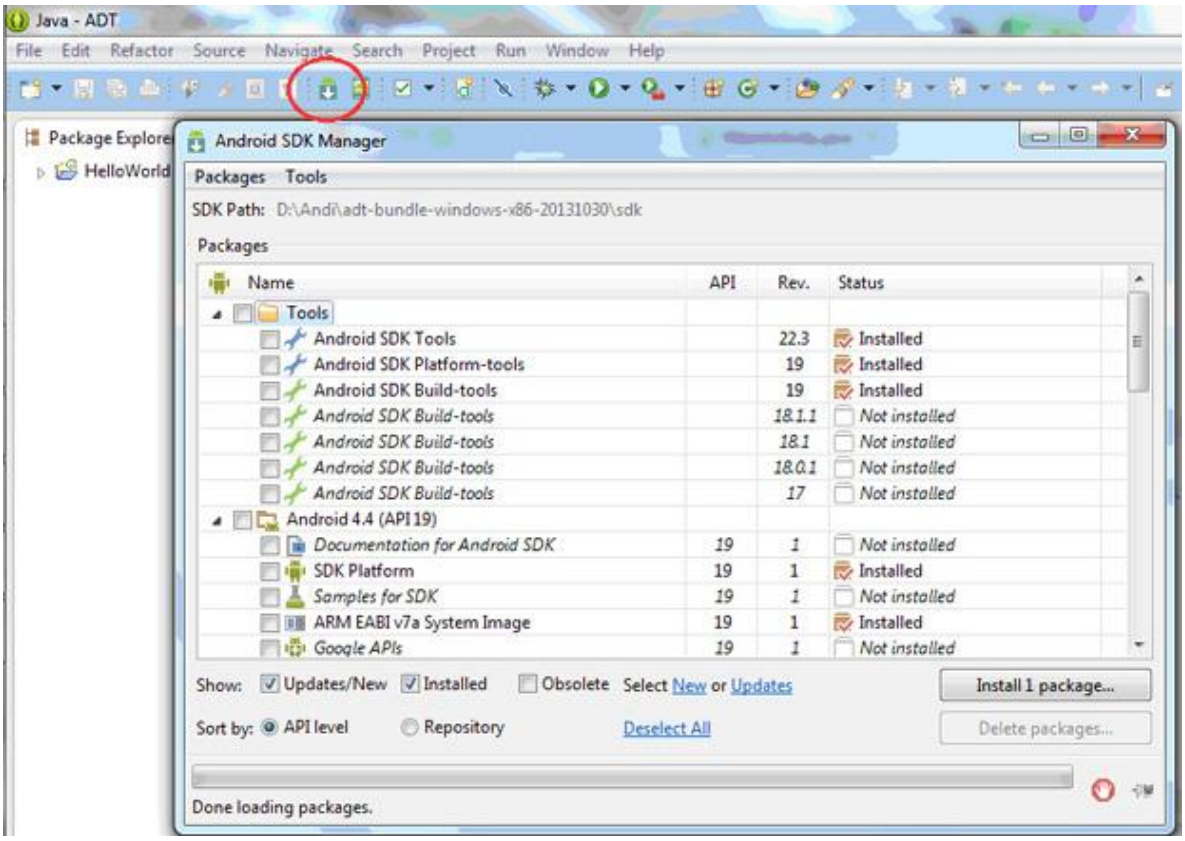


Рис. 8. Android SDK Manager

2.4. Этапы создания приложений

Чтобы создать приложение, зайдите в меню **File->New->Android Application Project**.

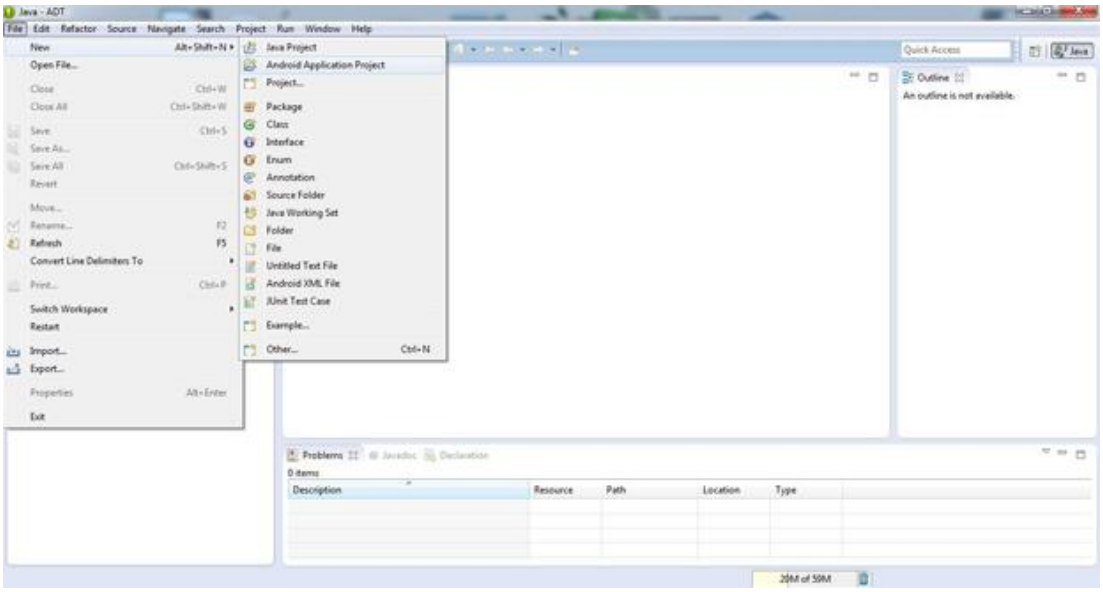


Рис. 9. Создание приложения

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. Конечно, учебные приложения туда не загружают, однако, следует иметь это в виду на будущее.

Minimum Required SDK - минимальная версия Android, которую будет поддерживать приложение. Чаще всего по умолчанию указывается версия 2.2, 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.

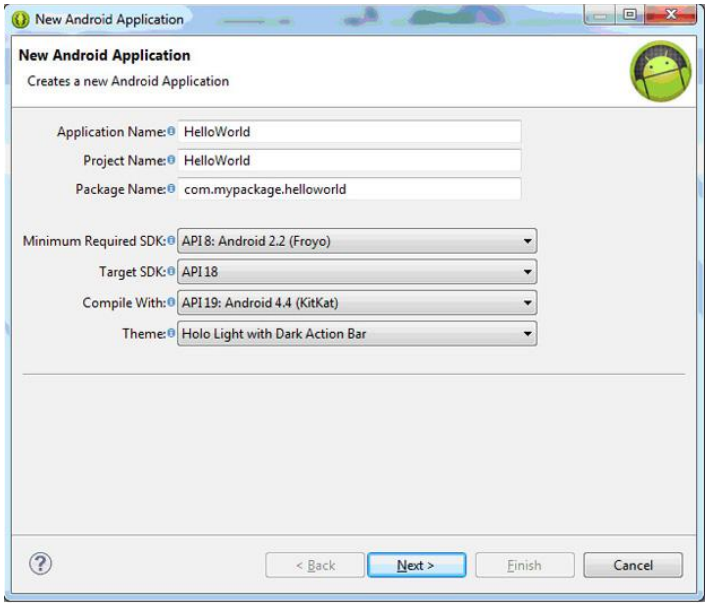


Рис. 10. Наименование приложения

Следующее окно можно пропустить без изменений.

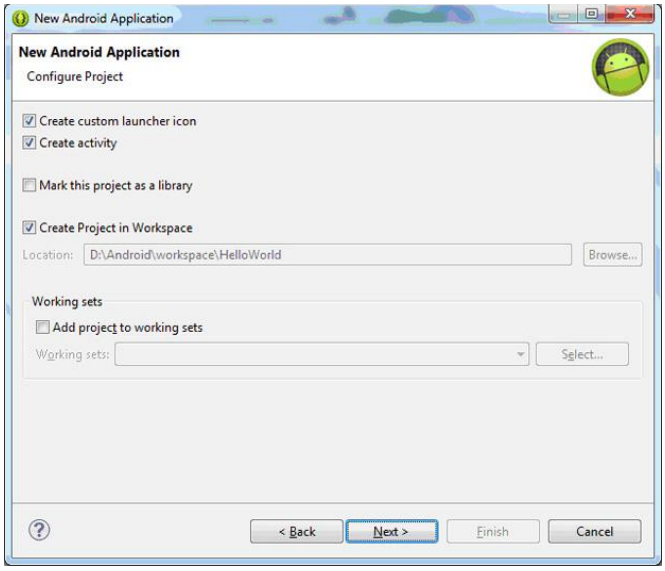


Рис. 11. Конфигурация приложения

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.

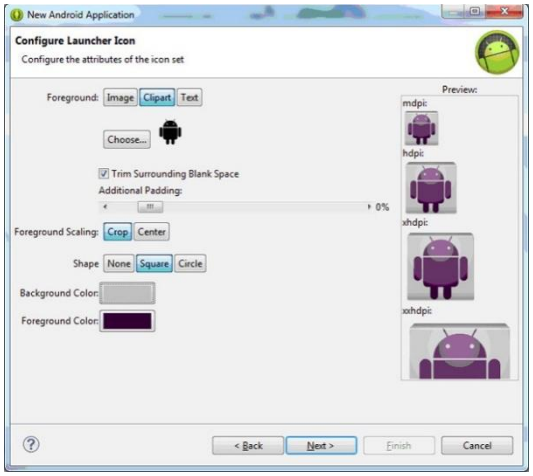


Рис. 12. Создание иконки приложения

Большинство приложений на Android имеют свой экран (форму, окно), которое называется активностью или деятельностью (Activity). Следующее два окна создают пустую активность. В первом ничего пока менять не нужно. Во втором вы можете переименовать свою активность (в приложении их может быть несколько).

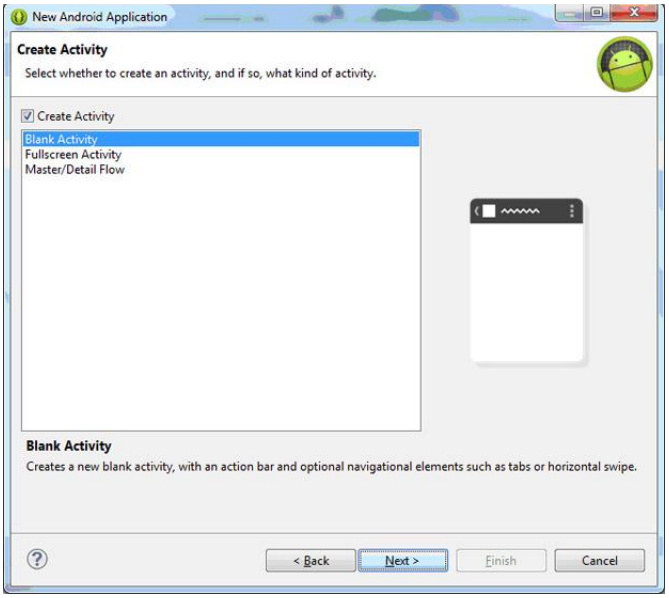


Рис. 13. Создание активности

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.

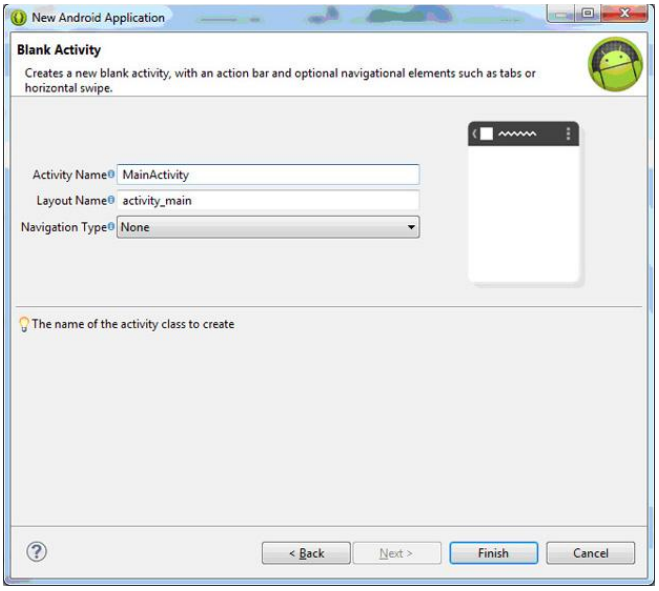


Рис. 14. Переименование активности

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.

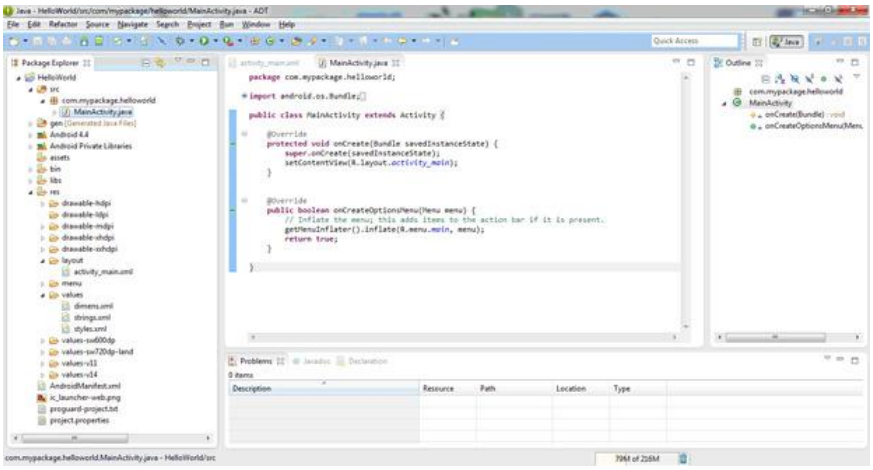


Рис. 15. Активность

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.

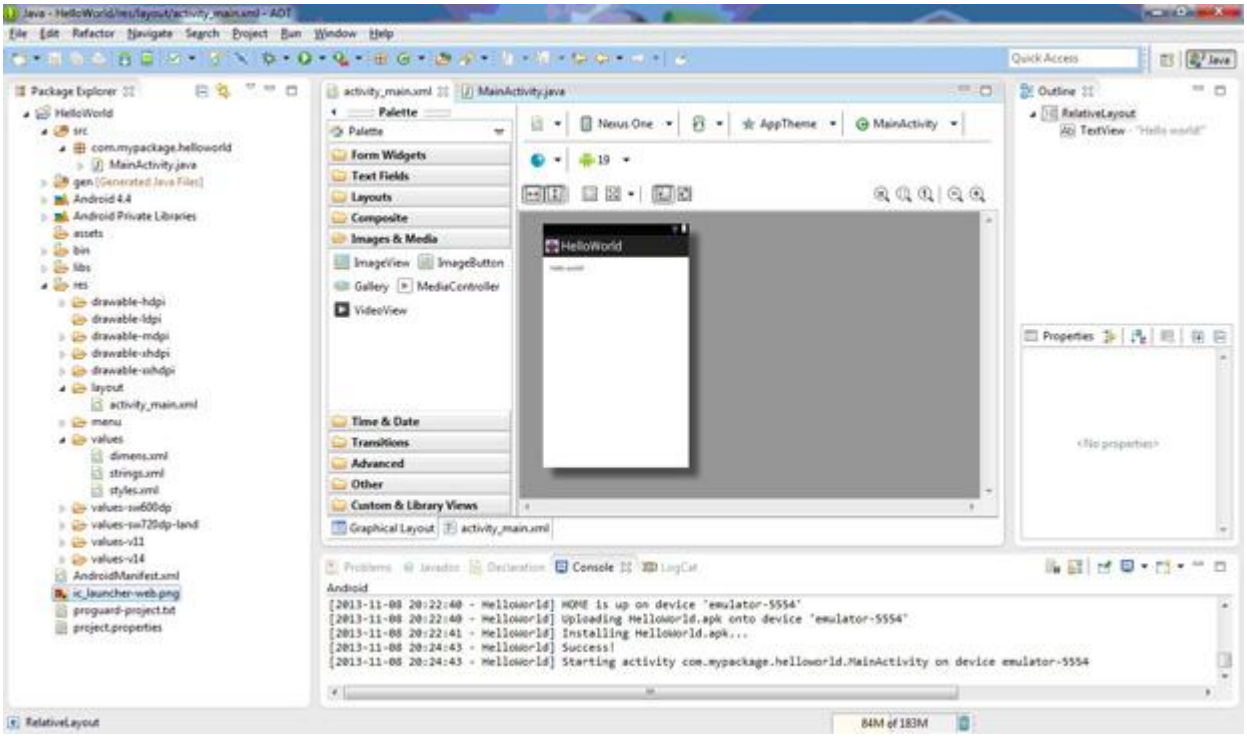


Рис. 16. Xml-файл. Графический редактор

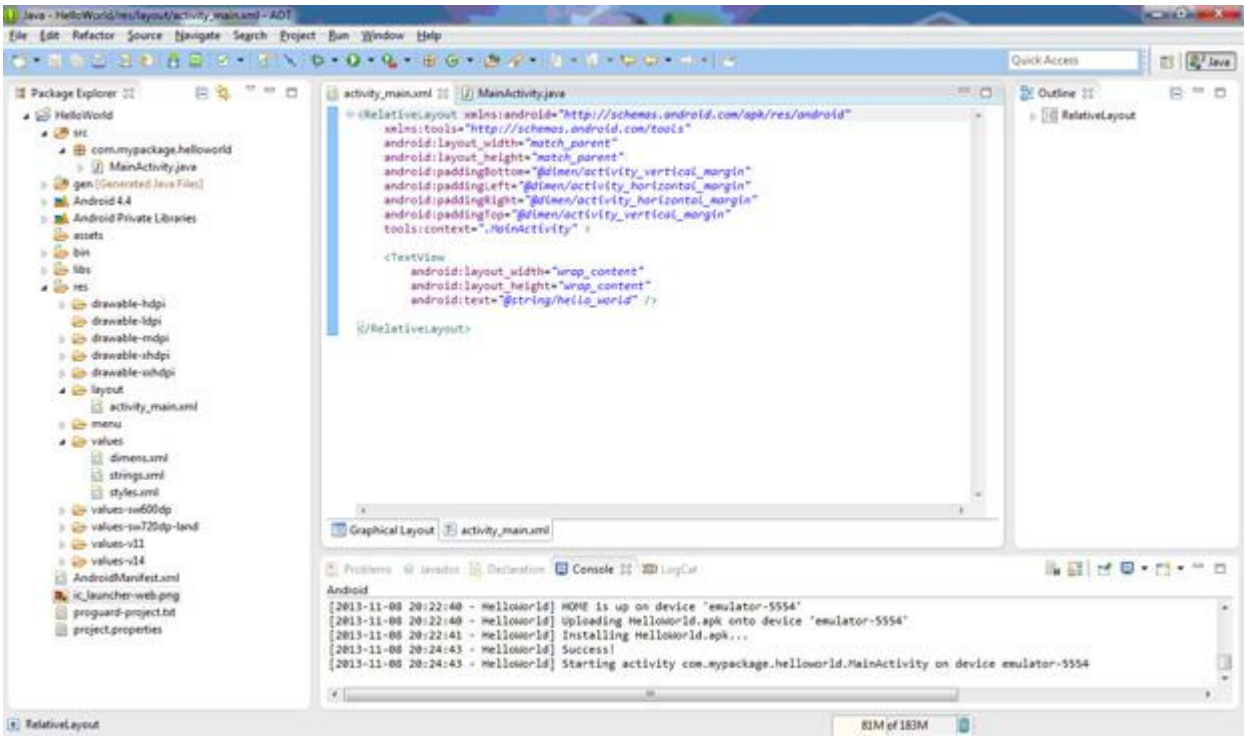


Рис. 17. Xml-файл

Для запуска приложения на эмуляторе 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.**

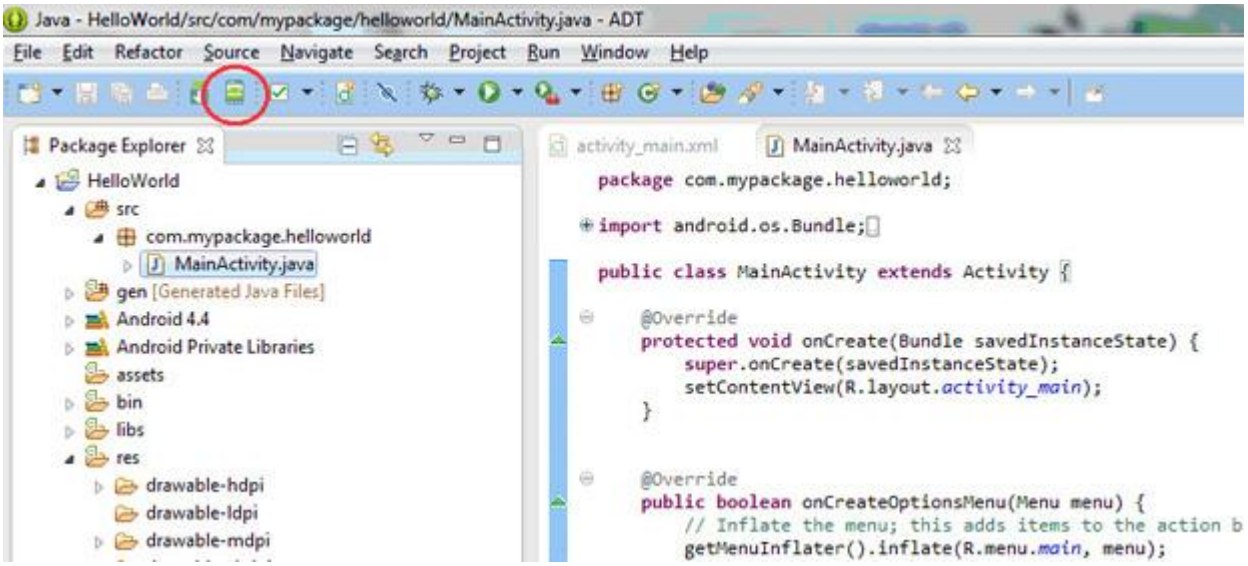


Рис. 18. Запуск приложения

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

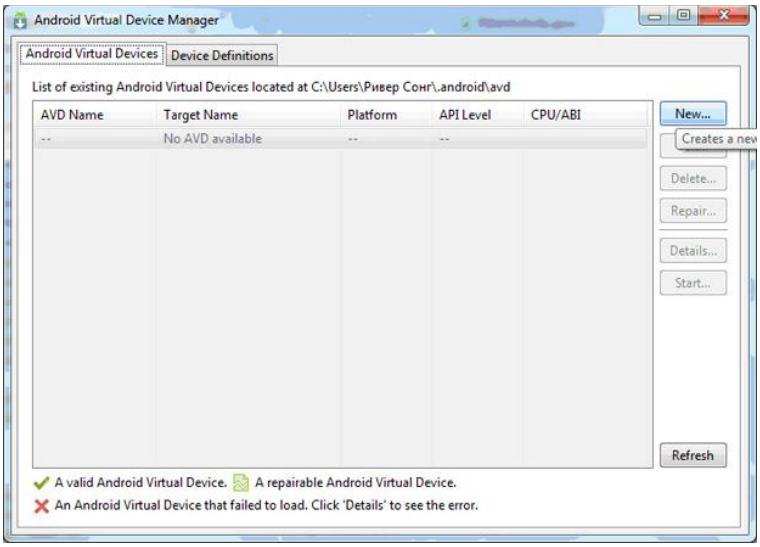


Рис. 19. Android Virtual Device Manager

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.

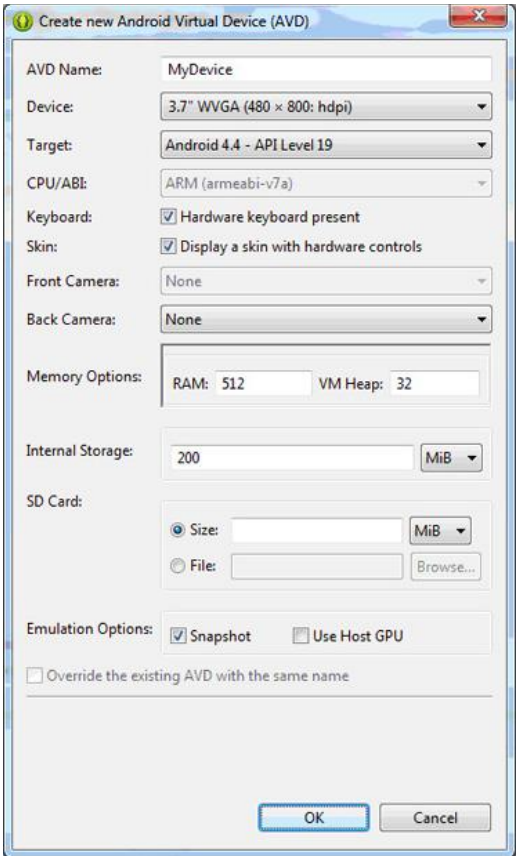


Рис. 20. Создание AVD

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.

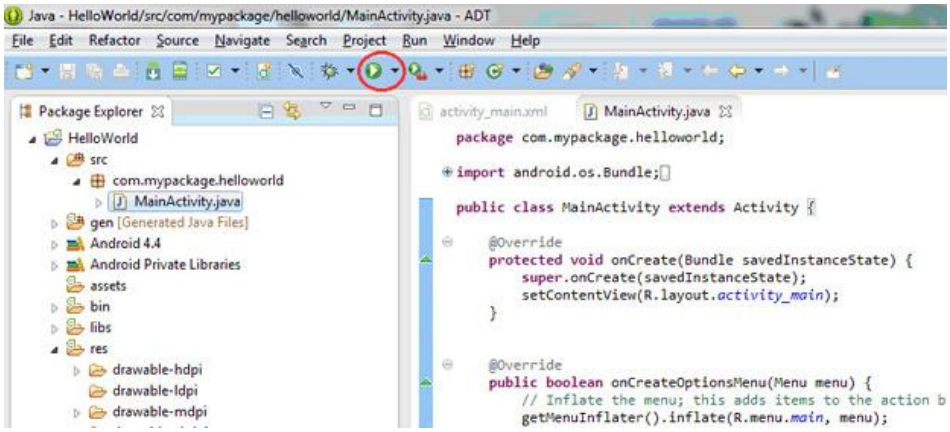


Рис. 21. Запуск приложения

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

Рис. 22. Запуск эмулятора

Рис. 23. Запущенный эмулятор

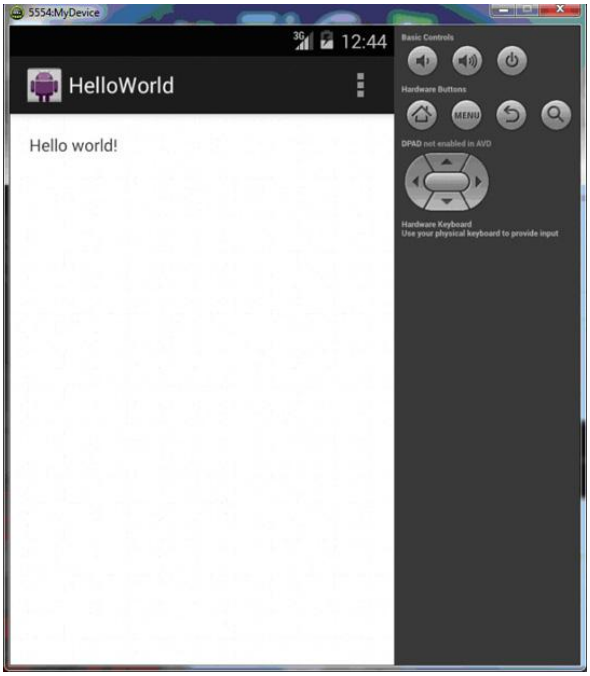


Рис. 24. Приложение Hello, world!

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

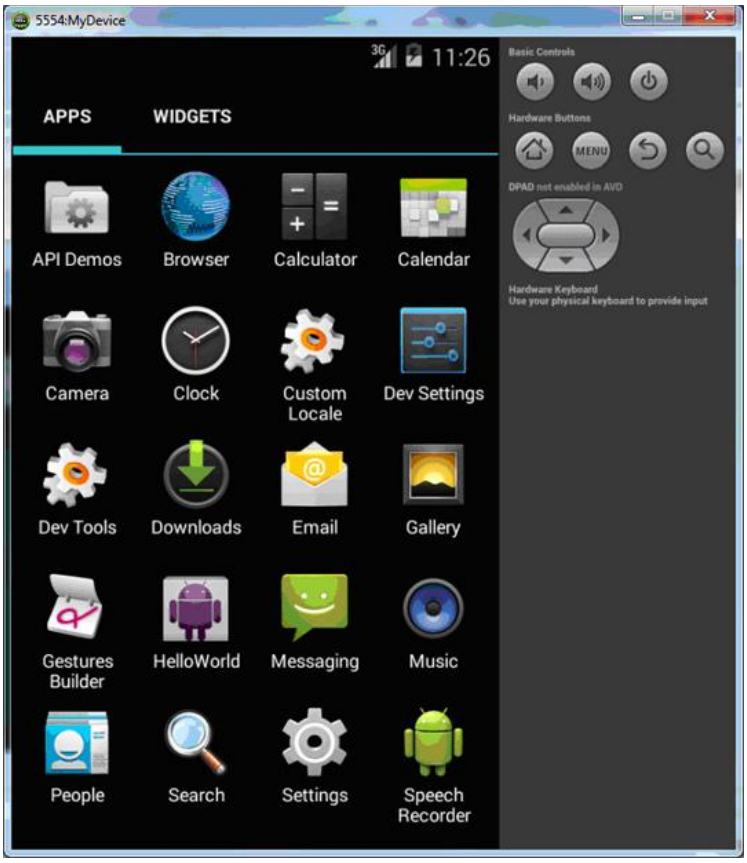


Рис. 25. Меню приложений

2.5. Изменение свойств приложений

Нередко приложению требуется сохранять небольшие кусочки данных для дальнейшего использования, например, данные о пользователе, настройки конфигурации и т.д. Для этого в Android существует концепция Preferences или настройки. Настройки представляют собой группу пар ключ-значение, которые используются приложением. В качестве значений могут выступать данные следующих типов: Boolean, Float, Integer, Long, String, набор строк. Настройки общими для всех activity в приложении, но также могут быть и настройки непосредственно для отдельных activity. Для работы с разделяемыми настройками в классе Activity (точнее в его базовом классе Context) имеется метод **getSharedPreferences()**:

```
import android.content.SharedPreferences;
//.....
SharedPreferences settings = getSharedPreferences("PreferencesName", MODE_PRIVATE);
```

Первый параметр метода указывает на название настроек. В данном случае название - "PreferencesName". Если настроек с подобным названием нет, то они создаются при вызове данного метода. Второй параметр указывает на режим доступа. В данном случае режим описан константой MODE_PRIVATE. Класс android.content.SharedPreferences предоставляет ряд методов для управления настройками:

- contains(String key): возвращает true, если в настройках сохранено значение с ключом key
- getAll(): возвращает все сохраненные в настройках значения
- getBoolean (String key, boolean defValue): возвращает из настроек значение типа Boolean, которое имеет ключ key. Если элемента с таким ключом не окажется, то возвращается значение defValue, передаваемое вторым параметром
- getFloat(String key, float defValue): возвращает значение типа float с ключом key. Если элемента с таким ключом не окажется, то возвращается значение defValue
- getInt(String key, int defValue): возвращает значение типа int с ключом key
- getLong(String key, long defValue): возвращает значение типа long с ключом key
- getString(String key, String defValue): возвращает строковое значение с ключом key
- getStringSet(String key, Set<String> defValues): возвращает массив строк с ключом key
- edit(): возвращает объект SharedPreferences.Editor, который используется для редактирования настроек

Для управления настройками используется объект класса **SharedPreferences.Editor**, возвращаемый метод edit(). Он определяет следующие методы:

- clear(): удаляет все настройки
- remove(String key): удаляет из настроек значение с ключом key
- putBoolean(String key, boolean value): добавляет в настройки значение типа boolean с ключом key
- putFloat(String key, float value): добавляет в настройки значение типа float с ключом key
- putInt(String key, int value): добавляет в настройки значение int с ключом key
- putLong(String key, long value): добавляет в настройки значение типа long с ключом key
- putString(String key, String value): добавляет в настройки строку с ключом key
- putStringSet(String key, Set<String> values): добавляет в настройки строковый массив
- commit(): подтверждает все изменения в настройках
- apply(): также, как и метод commit(), подтверждает все изменения в настройках, однако измененный объект SharedPreferences вначале сохраняется во временной памяти, и лишь затем в результате асинхронной операции записывается на мобильное устройство

Рассмотрим пример сохранения и получения настроек в приложении. Допустим, у нас следующий файл layout с пользовательским интерфейсом:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <EditText
        android:id="@+id/nameBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Введите имя"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

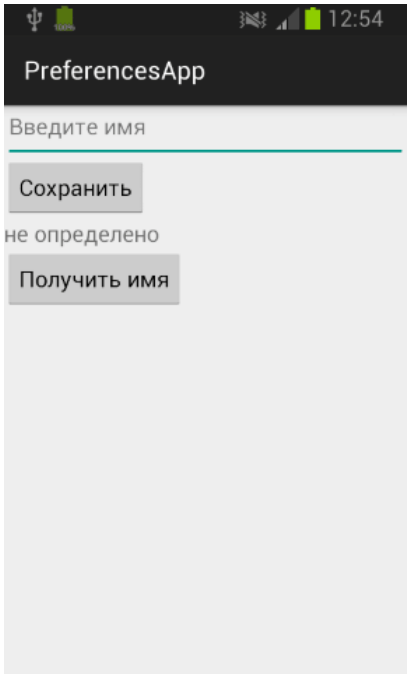
```
        android:text="Сохранить"
        android:onClick="saveName"/>
    <TextView
        android:id="@+id/nameView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18dip"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Получить имя"
        android:onClick="getName"/>
</LinearLayout>
```

На экране будут две кнопки - для сохранения и для вывода ранее сохраненного значения, а также поле для ввода и текстовое поля ля вывода сохраненной настройки. Определим методы обработчики кнопок в классе Activity:

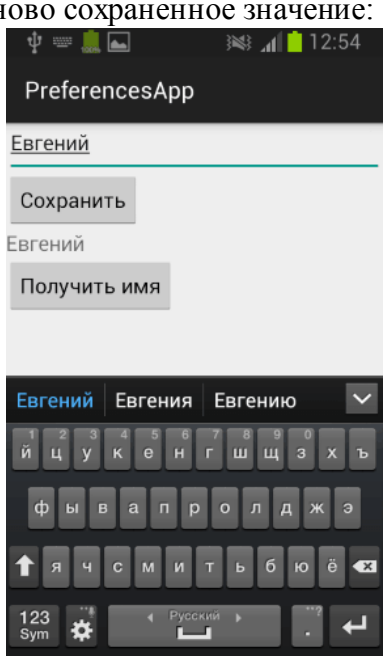
```
package com.example.eugene.preferencesapp;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
import android.widget.EditText;
import android.view.View;
import android.content.SharedPreferences;
public class MainActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void saveName(View view) {
        // получаем введенное имя
        EditText nameBox = (EditText) findViewById(R.id.nameBox);
        String name = nameBox.getText().toString();
        // сохраняем его в настройках
        SharedPreferences settingsActivity = getSharedPreferences("PersonalAccount",
MODE_PRIVATE);
        SharedPreferences.Editor prefEditor = settingsActivity.edit();
        prefEditor.putString("Name", name);
        prefEditor.commit();
    }
    public void getName(View view) {

        // получаем сохраненное имя
        TextView nameView = (TextView) findViewById(R.id.nameView);
        SharedPreferences settingsActivity = getSharedPreferences("PersonalAccount",
MODE_PRIVATE);
        String name = settingsActivity.getString("Name","не определено");
        nameView.setText(name);
    }
```

}

При отсутствии настроек при попытке их получить, приложение выведет значение по умолчанию:



Теперь сохраним и выведем заново сохраненное значение:



Кроме общих настроек каждая activity может использовать приватные, к которым доступ из других activity будет невозможен. Для получения настроек уровня activity используется метод **getPreferences(MODE_PRIVATE)**:

```
import android.content.SharedPreferences;
//.....................
SharedPreferences settingsActivity = getPreferences(MODE_PRIVATE);
```

То есть в отличие от общих настроек здесь не используется название группы настроек в качестве первого параметра, как в методе getSharedPreferences(). Однако вся

остальная работа по добавлению, получению и изменению настроек будет аналогична работает с общими настройками.

Контрольные вопросы:

1. Виды приложений.
2. Как можно изменять свойства приложений?
3. Какую архитектуру имеет приложения Андроид?
4. Опишите фоновые приложения.
5. Как работают смешанные приложения?