Приложение к ППССЗ
по специальности 09.02.07

Методические указания
по выполнению практических работ
ОП.12 Менеджмент в профессиональной деятельности
основной профессиональной образовательной программы
по специальности
09.02.07. Информационные системы и программирование
Уровень подготовки - базовый
Год начала подготовки-2023

2023

СОДЕРЖАНИЕ

1 Пояснительная записка

Данные методические рекомендации составлены в соответствии с содержанием рабочей программы ПМ.12. Разработка децентрализованных приложений специальности 09.02.07 Информационные системы и программирование.

ПМ.12. Разработка децентрализованных приложений изучается в течение IV семестра. Общий объем времени, отведенный на практические занятия по УД, составляет в соответствии с учебным планом и рабочей программой – 64 часа

Практические работы проводятся после изучения соответствующих разделов и тем МДК.01.03 Разработка мобильных приложений. Выполнение обучающимися практических работ позволяет им понять, где и когда изучаемые теоретические положения и практические умения могут быть использованы в будущей практической деятельности.

В результате выполнения практических работ, предусмотренных программой по МДК.01.03 Разработка мобильных приложений, обучающийся должен:
уметь:
- выполнять отладку и тестирование программы на уровне модуля;
- осуществлять разработку кода программного модуля на современных языках программирования;

знать:
- основные этапы разработки программного обеспечения;
- основные принципы технологии структурного и объектно-ориентированного программирования

Вышеперечисленные умения, знания и практический опыт направлены на формирование следующих профессиональных и общих компетенций обучающихся**:**
ПК.1.2. Разрабатывать программные модули в соответствии с техническим заданием
ПК 1.6. Разрабатывать модули программного обеспечения для мо-бильных платформ.
ОК 1. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам
ОК 2. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности
ОК3. Планировать и реализовывать собственное профессиональное и личностное развитие.
ОК 4. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.
ОК 5. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.
ОК6. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей
ОК7. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.
ОК8 Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности
ОК 9. Использовать информационные технологии в профессиональной деятельности
ОК10. Пользоваться профессиональной документацией на государственном и иностранном языках

2 Перечень практических работ УД МДК.01.03 Разработка мобильных приложений

Название практических работ	**Количество часов**

Тема 1.3.1 Основные платформы и языки разработки мобильных приложений	
Практическая работа №1 Установка инструментария и настройка среды для разработки мобильных приложений	8
Практическая работа №2 Установка среды разработки мобильных приложений с применением виртуальной машины	6
Тема 1.3.2 Создание и тестирование модулей для мобильных приложений	
Практическая работа №3 Создание эмуляторов и подключение устройств»	6
Практическая работа №4 Настройка режима терминала»	4
Практическая работа №5 Создание нового проекта»	6
Практическая работа №6 Изучение и комментирование кода»	6
Практическая работа №7 Лабораторная работа «Изменение элементов дизайна»	4
Практическая работа №8Обработка событий: подсказки»	4
Практическая работа №9 Обработка событий: цветовая индикация»	4
Практическая работа №10 Подготовка стандартных модулей»	4
Практическая работа №11 Обработка событий: переключение между экранами	4
Практическая работа №12 Передача данных между модулями	4
Практическая работа №13 Тестирование и оптимизация мобильного приложения	4

3 Инструктивно-методические указания по выполнению практических работ

Практическая работа №1 Установка инструментария и настройка среды для разработки мобильных приложений

Установка и настройка компонентов среды разработки

Приложения для Android, как и большинство приложений для коммуникаторов, разрабатываются на стандартном ПК, где ресурсы не ограничены (по сравнению с мобильным устройством) и загружаются на целевой коммуникатор для отладки, тестирования и последующего использования. Приложения можно отлаживать и тестировать на реальном устройстве под управлением Android или на эмуляторе. Для первоначальной разработки и отладки удобнее использовать эмулятор, а затем выполнять окончательное тестирование на реальных устройствах.

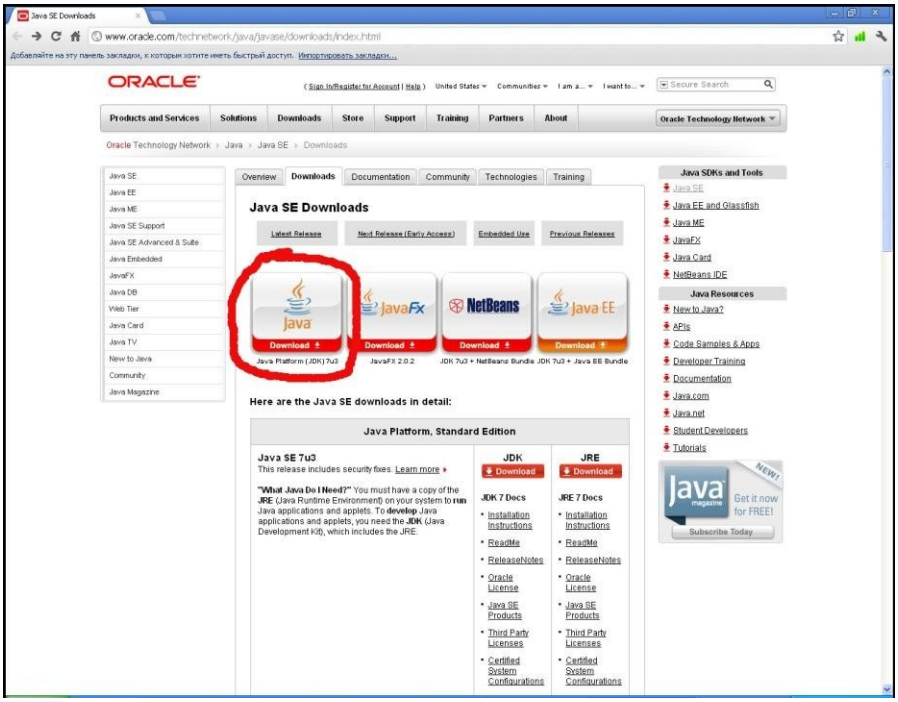
Разработчика предоставляется возможность использовать средства разработки приложений для Android на ПК под управлением любой из распространенных операционных систем: Windows, Linux и Mac OS X. Ниже будет детально описан процесс установки компонентов среды разработки под Windows XP, для прочих операционных систем отличия весьма незначительны.

Установка JDK

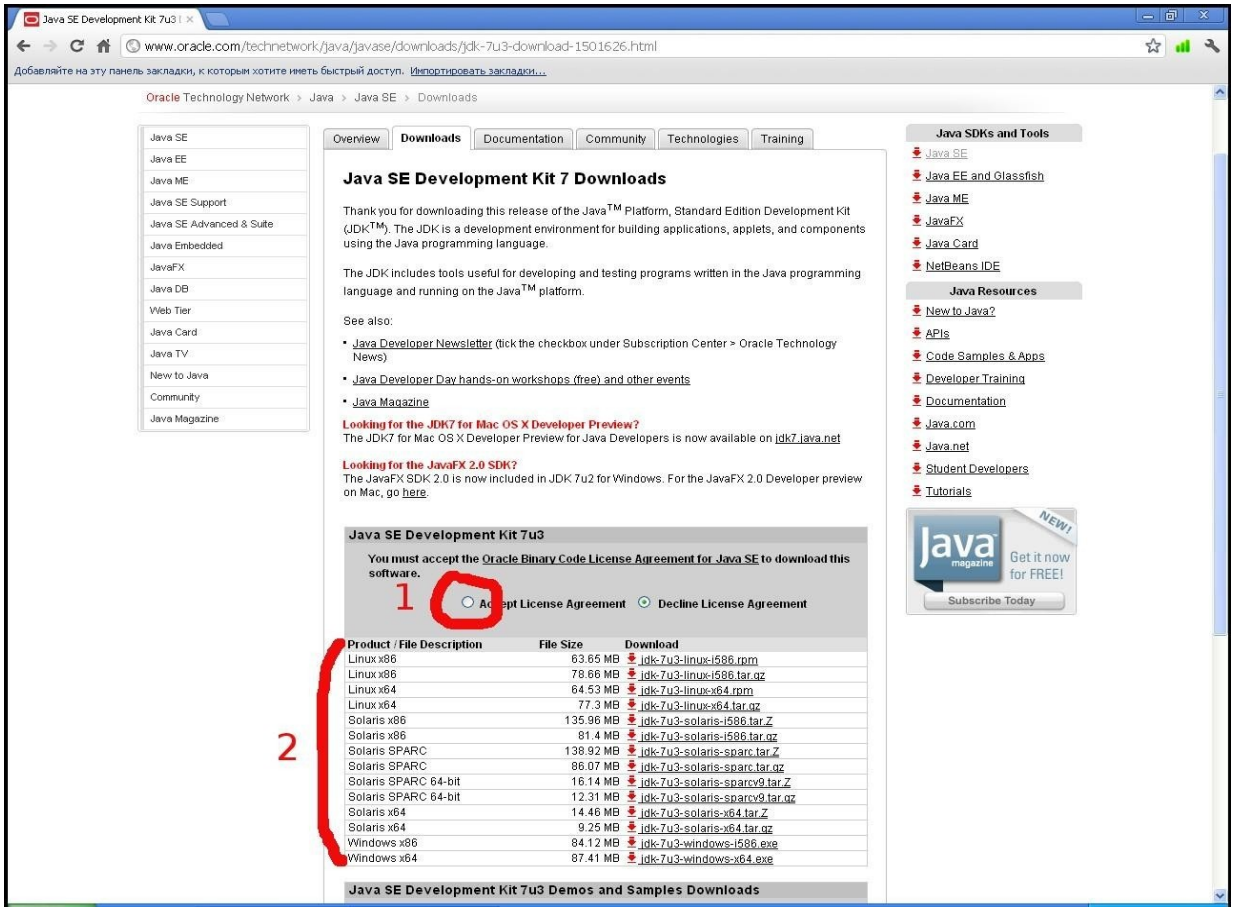
Для Android SDK требуется JDK версии не ниже 5 (на момент написания данного методического руководства была актуальна 7-я версия Sun JDK и 6-я версия Open-JDK). Для Windows традиционно используется Sun JDK, который можно бесплатно загрузить на сайте разработчика:

Страница загрузки Sun JDK

http://www.oracle.com/technetwork/java/javase/downloads/index.html



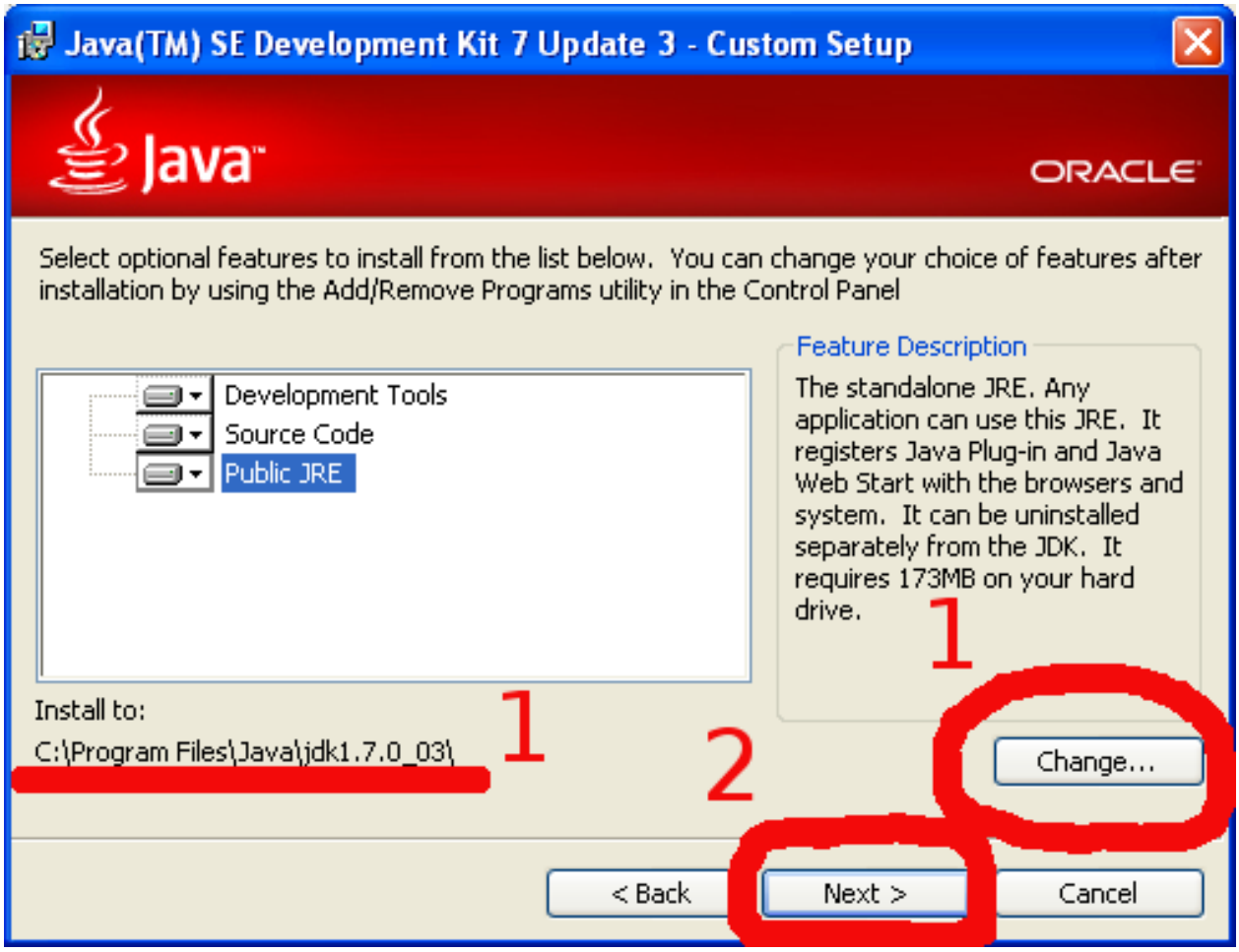
После принятия условий лицензионного соглашения *Oracle Binary Code License Agreement*



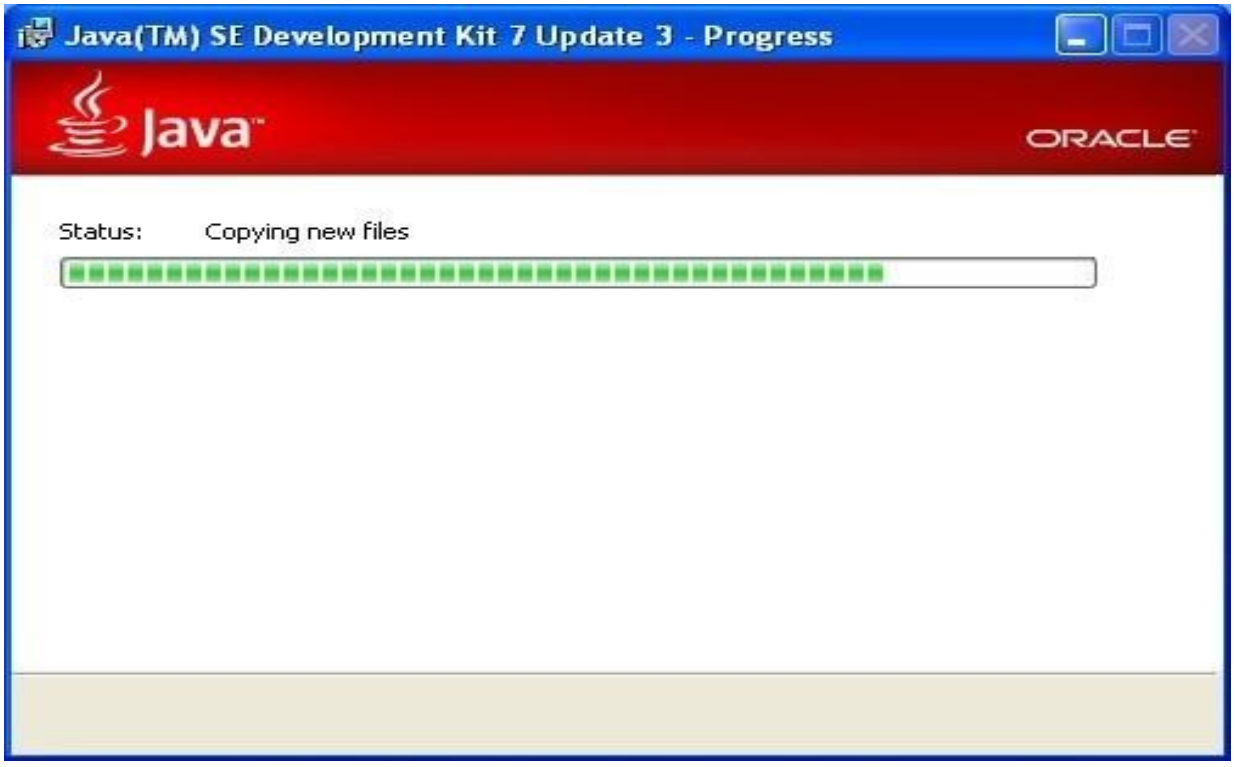
for Java SE **(1)** можно выбрать подходящую для вашей платформы ссылку**(2):**

Установка загруженного JDK достаточно проста:

Если каталог для установки по умолчанию **(1)** не подходит, его можно изменить **(2)**, а затем продолжить установку:



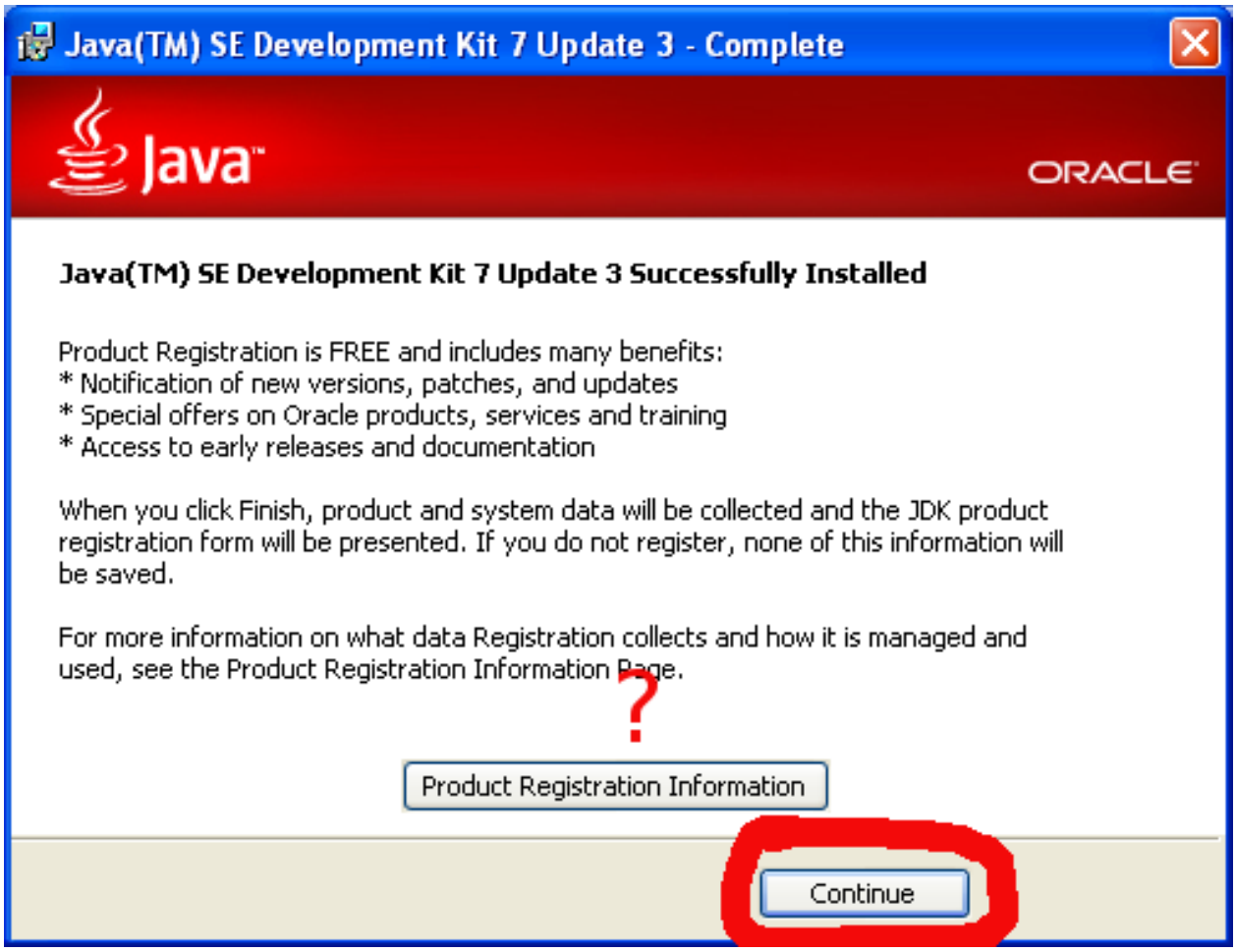
Далее все идет вполне ожидаемо:



При установке Java Runtime так же можно изменить каталог для установки:

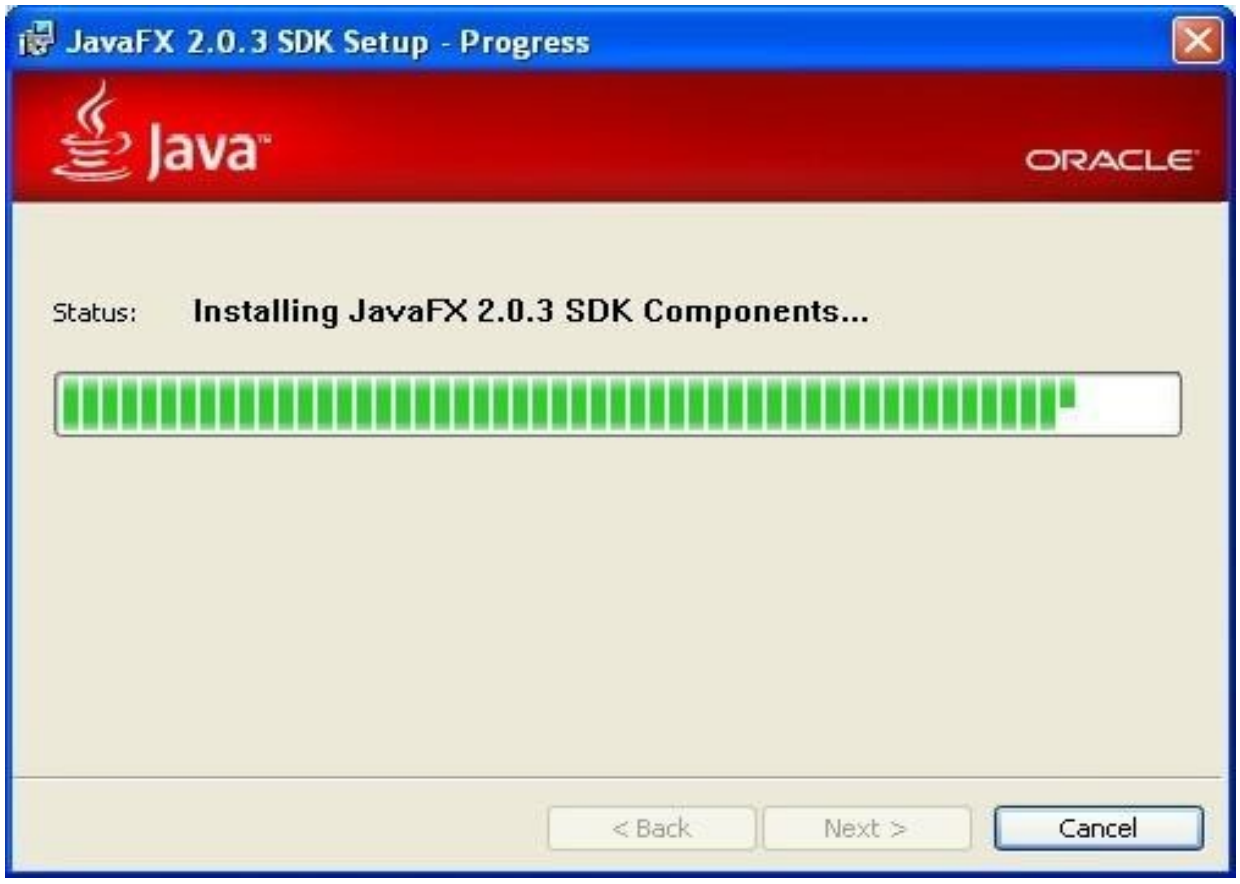
и продолжить установку:

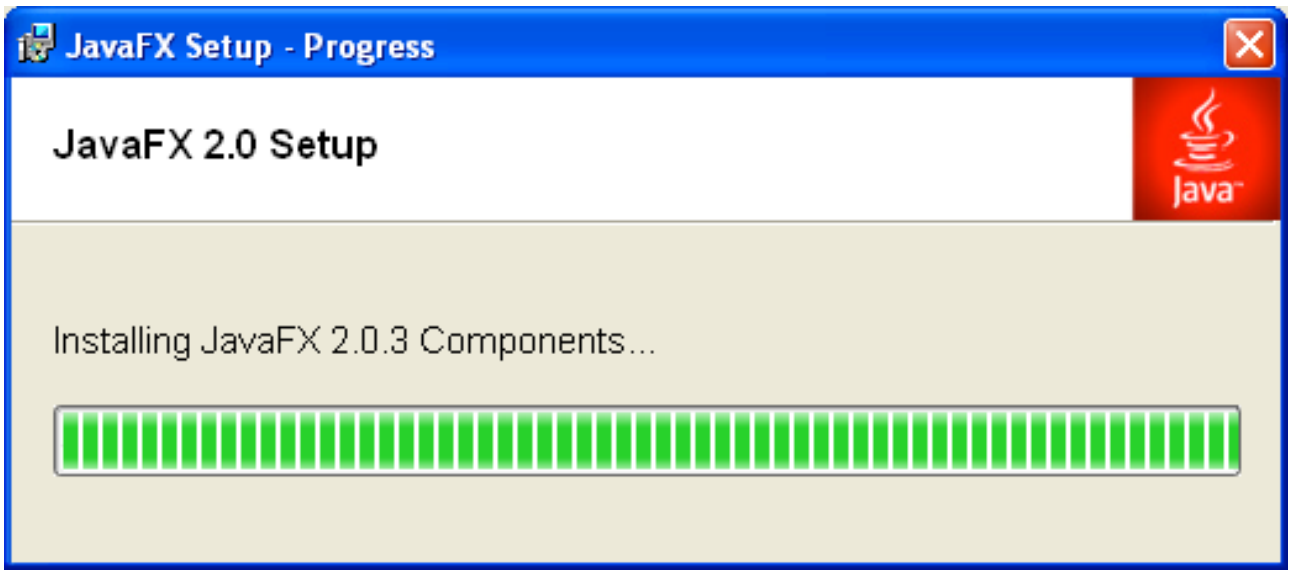
Далее при желании можно зарегистрировать установленный продукт:



И установить JavaFX SDK, если не жалко 50 МБ дискового пространства:

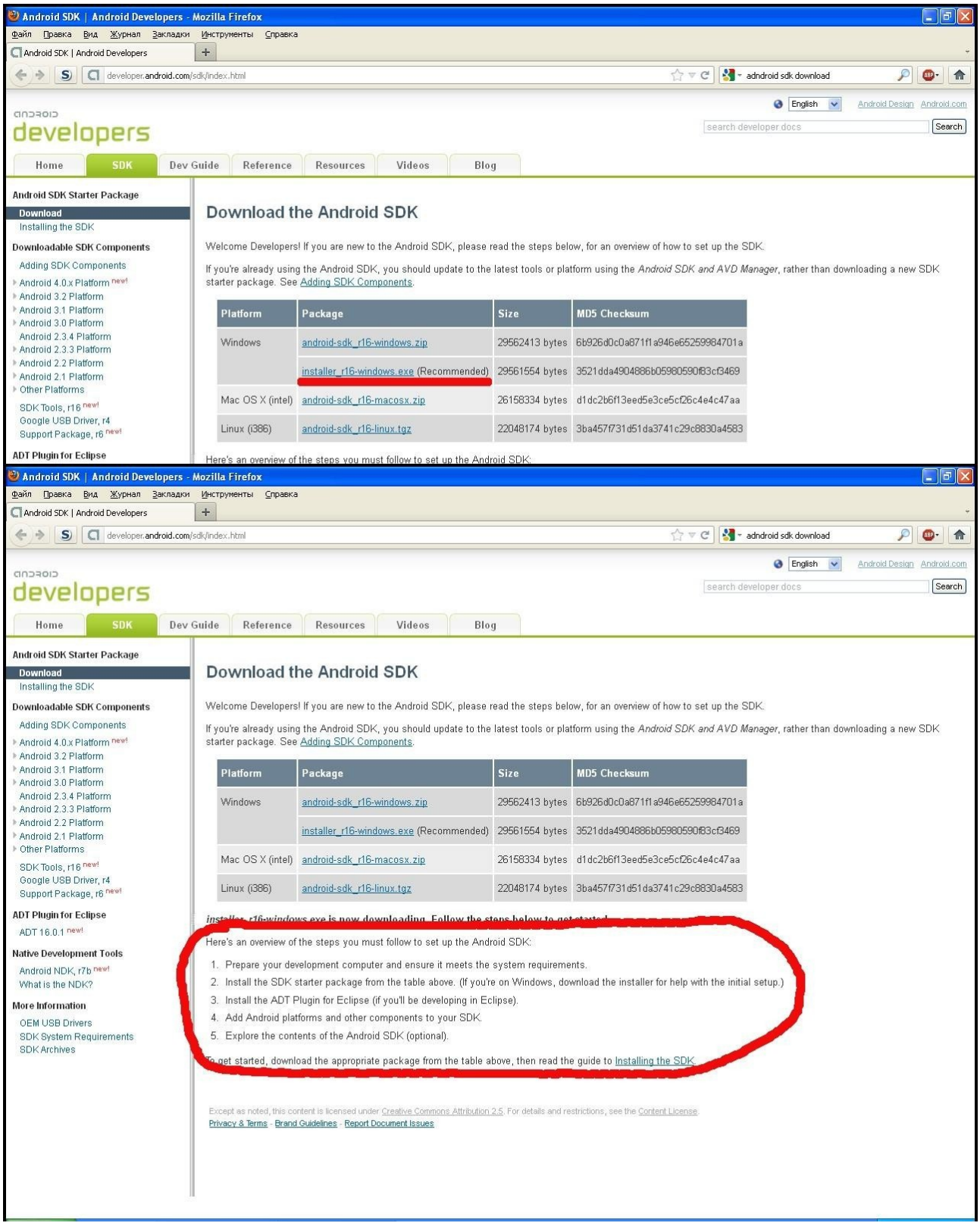

JavaFX SDK Setup
Java
ORACLE
Setup Options
To install the SDK in this folder, click Next.
The SDK will need approximately 50MB of disk space.
Folder:
C:\Program Files\Oracle\JavaFX 2.0 SDK\
Browse...
1
2
< Back
Next >


JavaFX 2.0.3 SDK Setup - Progress
Java
ORACLE
Status:
Installing JavaFX 2.0.3 SDK Components...
< Back
Next >
Cancel



И, наконец, установка JDK закончена:

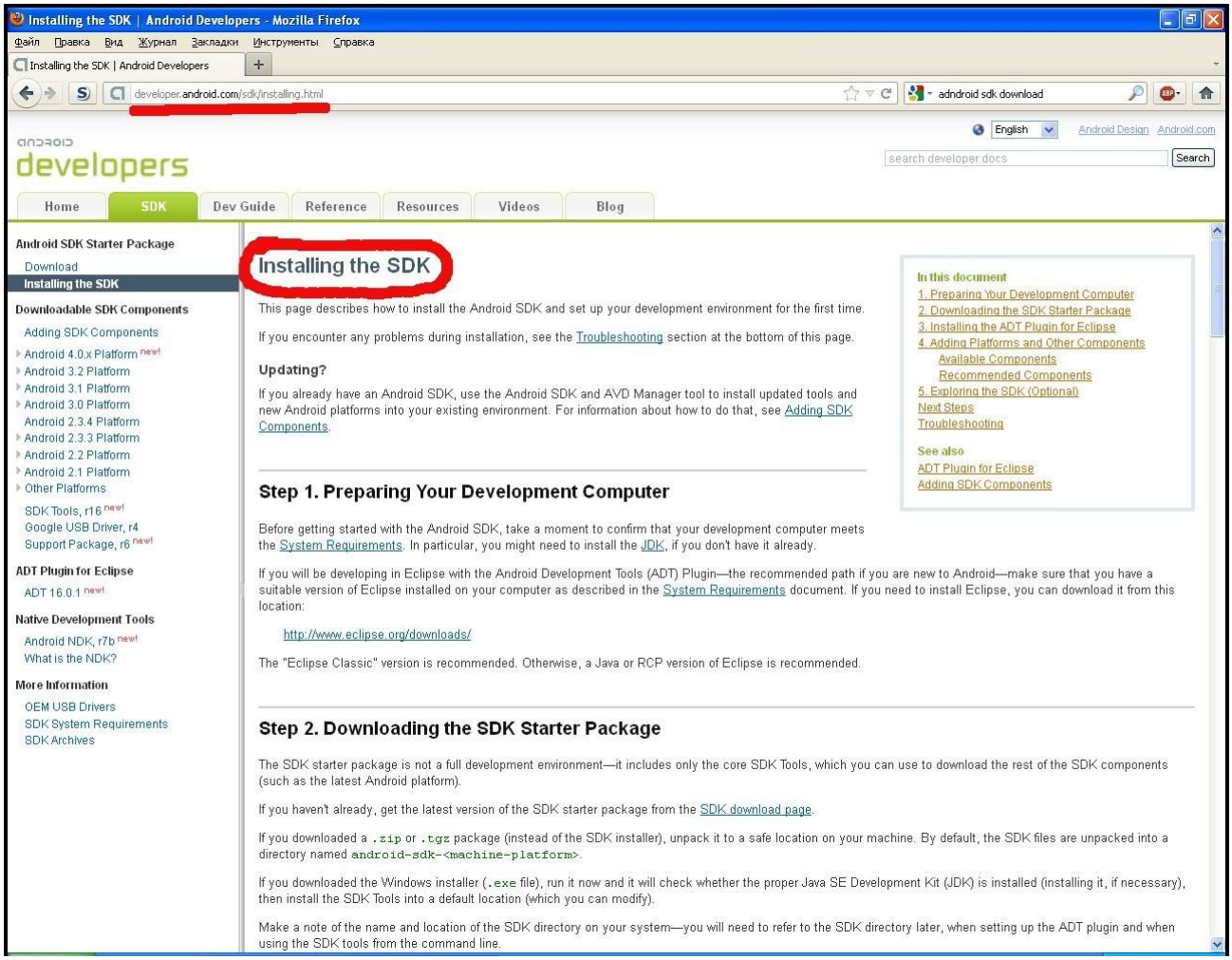
Установка Android SDK



На странице **http://developer.android.com/sdk** выбираем **installer_r16-windows.exe**

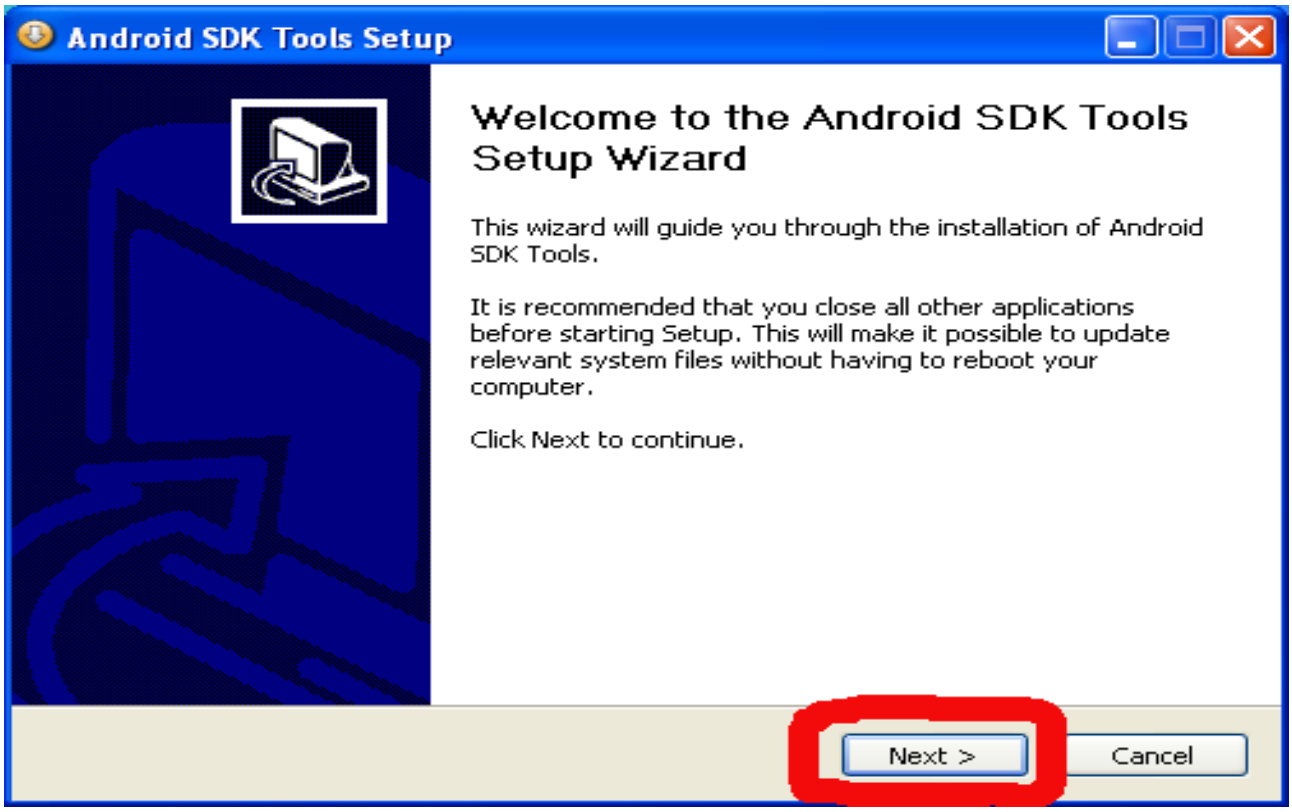
Здесь же имеются дополнительные инструкции по установке, а также ссылка на пошаговую инструкцию по установке SDK на все поддерживаемые платформы:

http://developer.android.com/sdk/installing.html

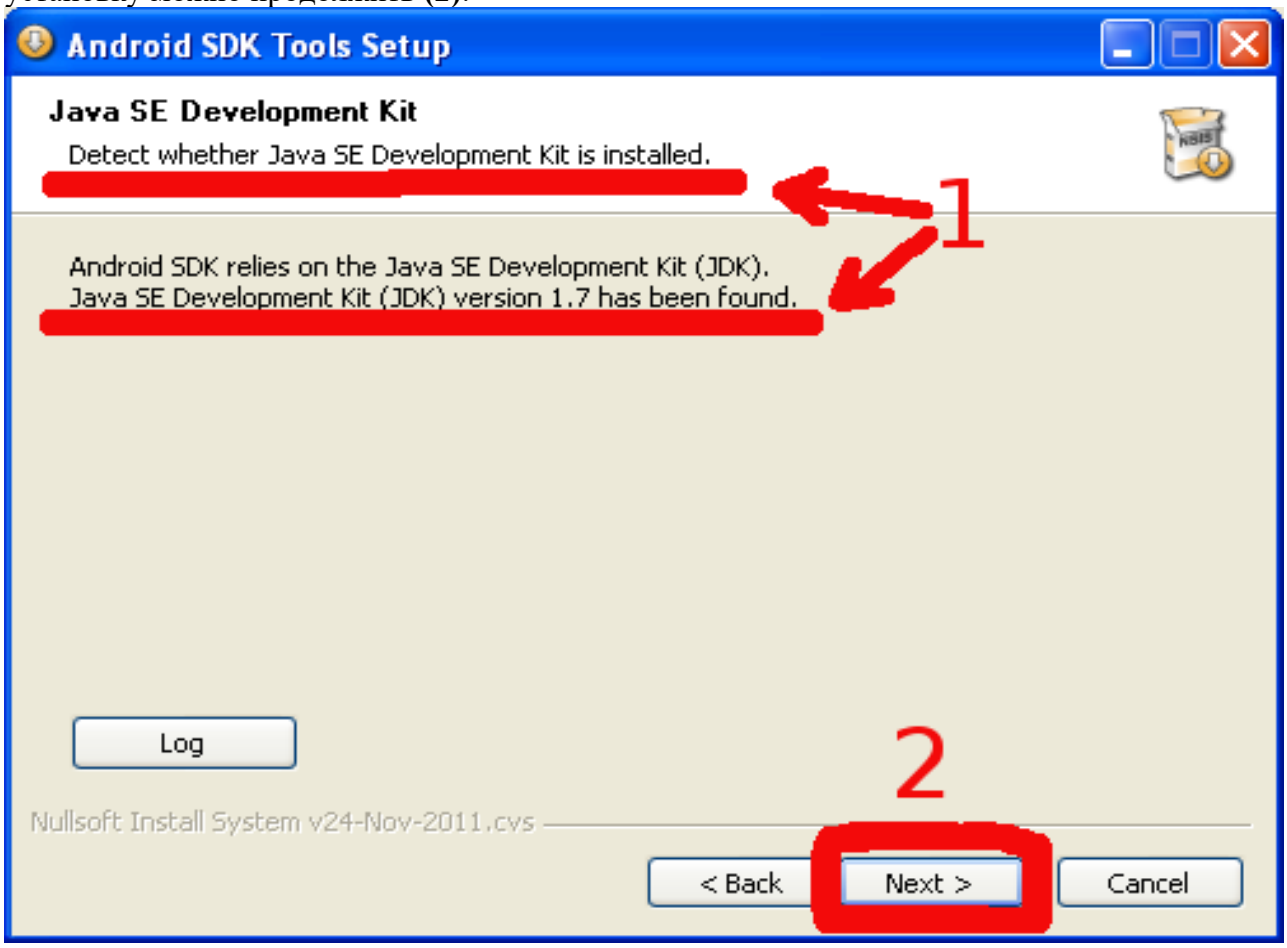


Ресурс http://developer.android.com является основным источником информации о платформе Android «из первых рук». В нем содержится огромное количество самых разнообразных сведений, необходимых разработчику, начиная с описания API и кончая такими вещами, как рекомендации по дизайну приложений и повышению производительности приложений. В данном методическом пособии в дальнейшем будут встречаться ссылки на конкретные страницы, посвященные изучаемым темам.

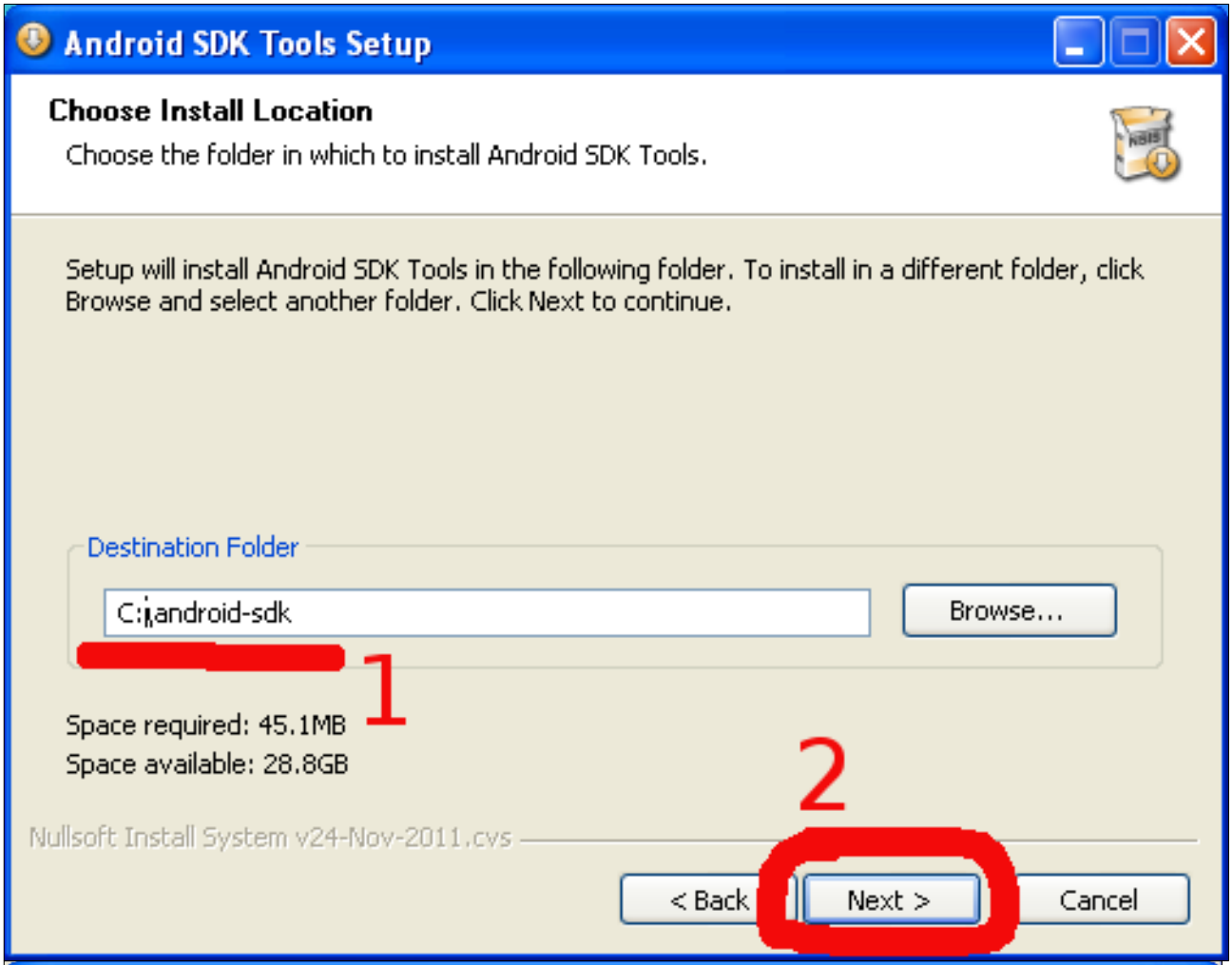
Установка загруженного Android SDK также не отличается особой сложностью:

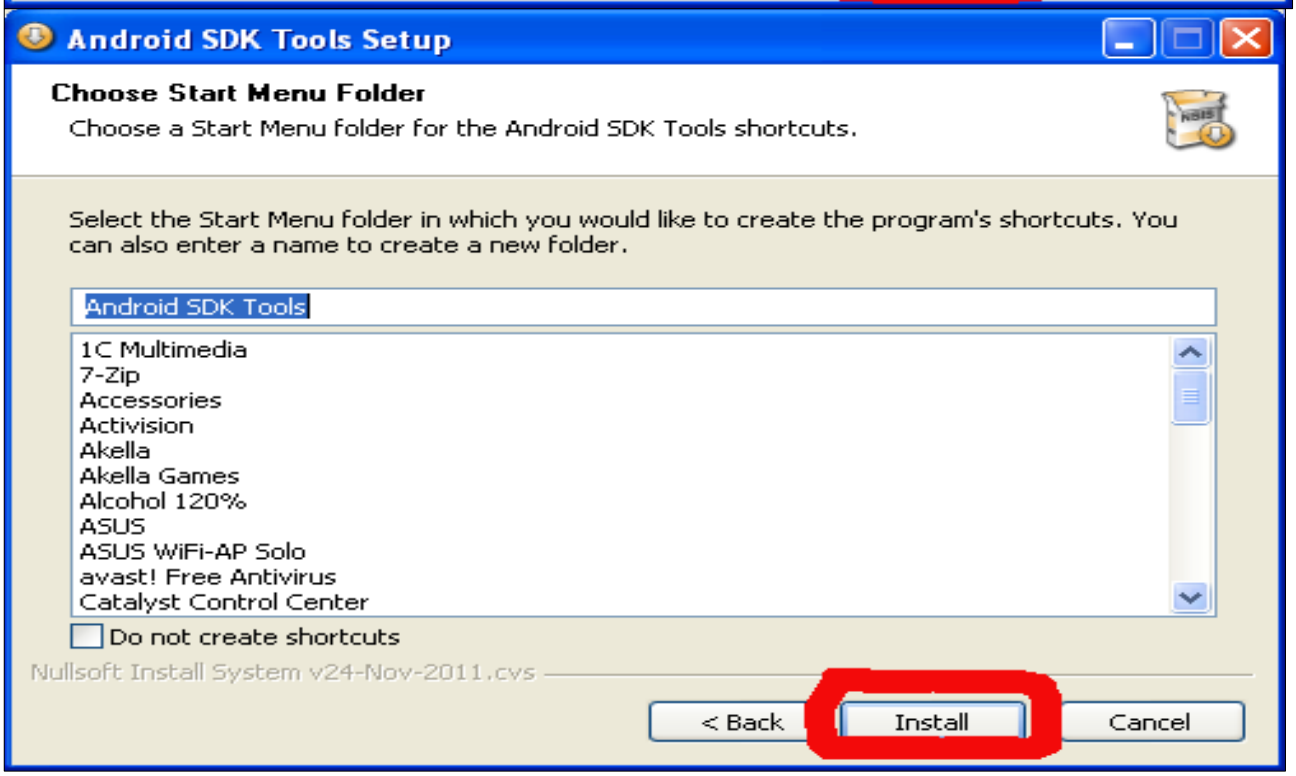


Мастер установки обнаружит (если сможет) установленную версию JDK **(1)**, после чего установку можно продолжить **(2)**:

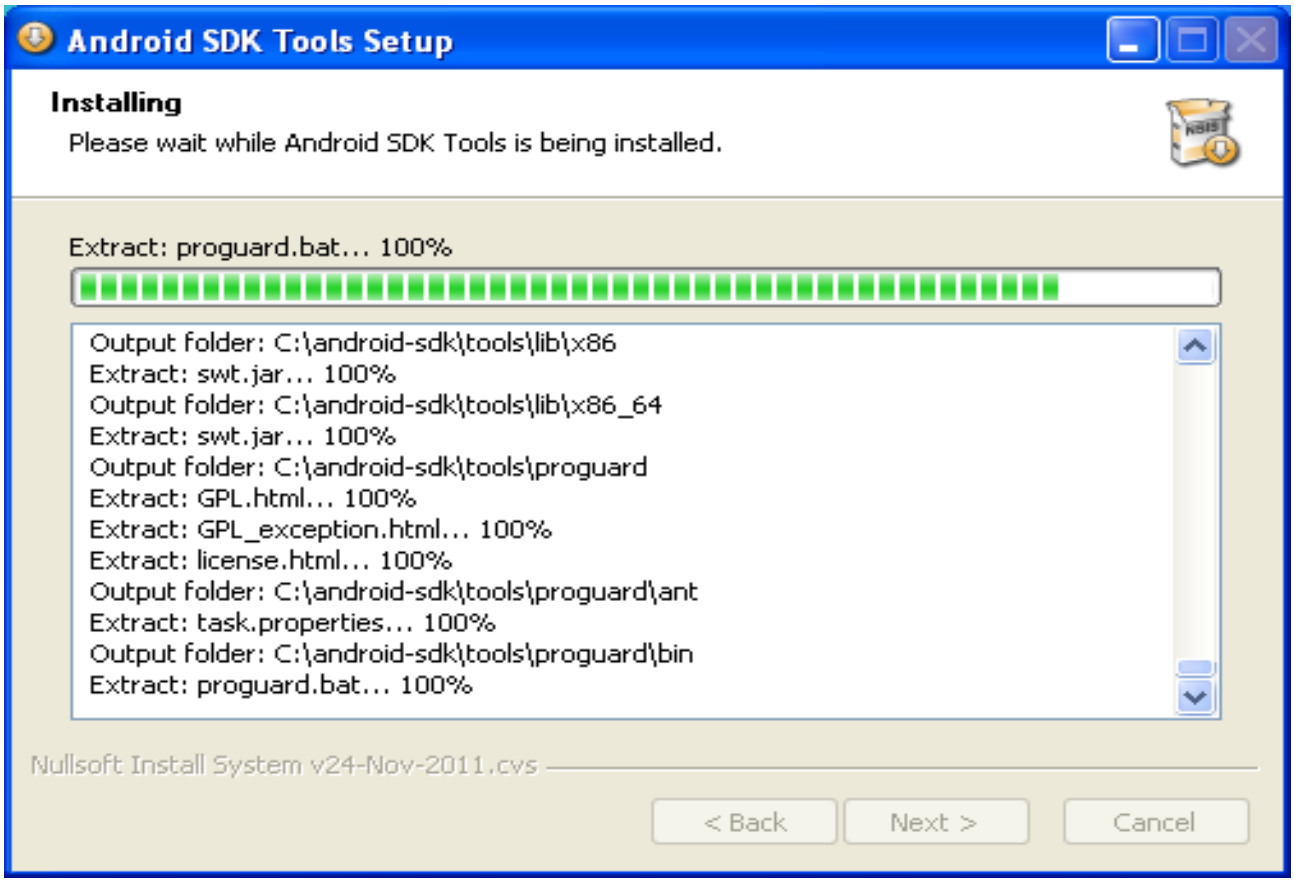


Если планируется запускать некоторые инструменты, входящие в Android SDK, из командной строки, имеет смысл выбрать каталог для установки ближе к корню файловой системы:

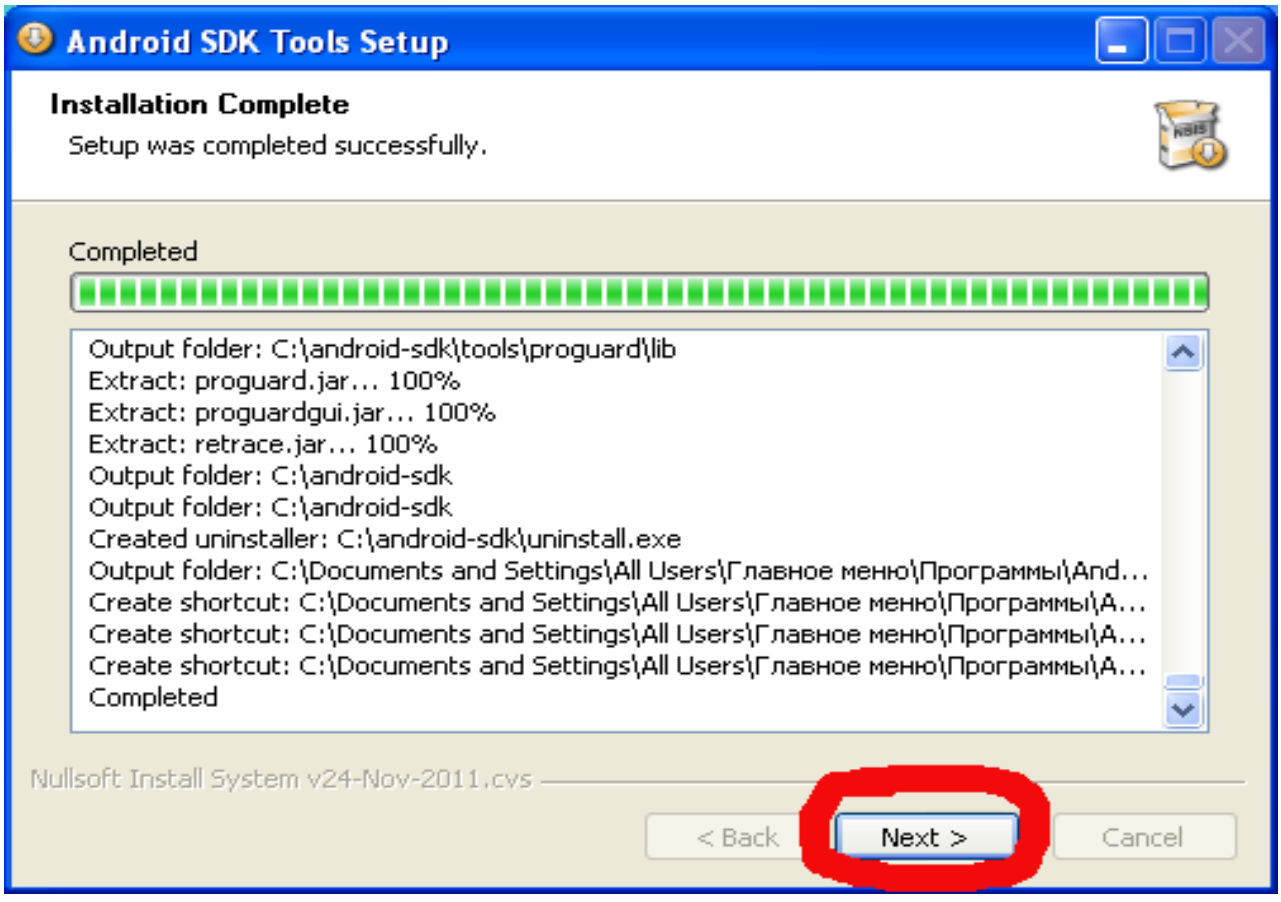




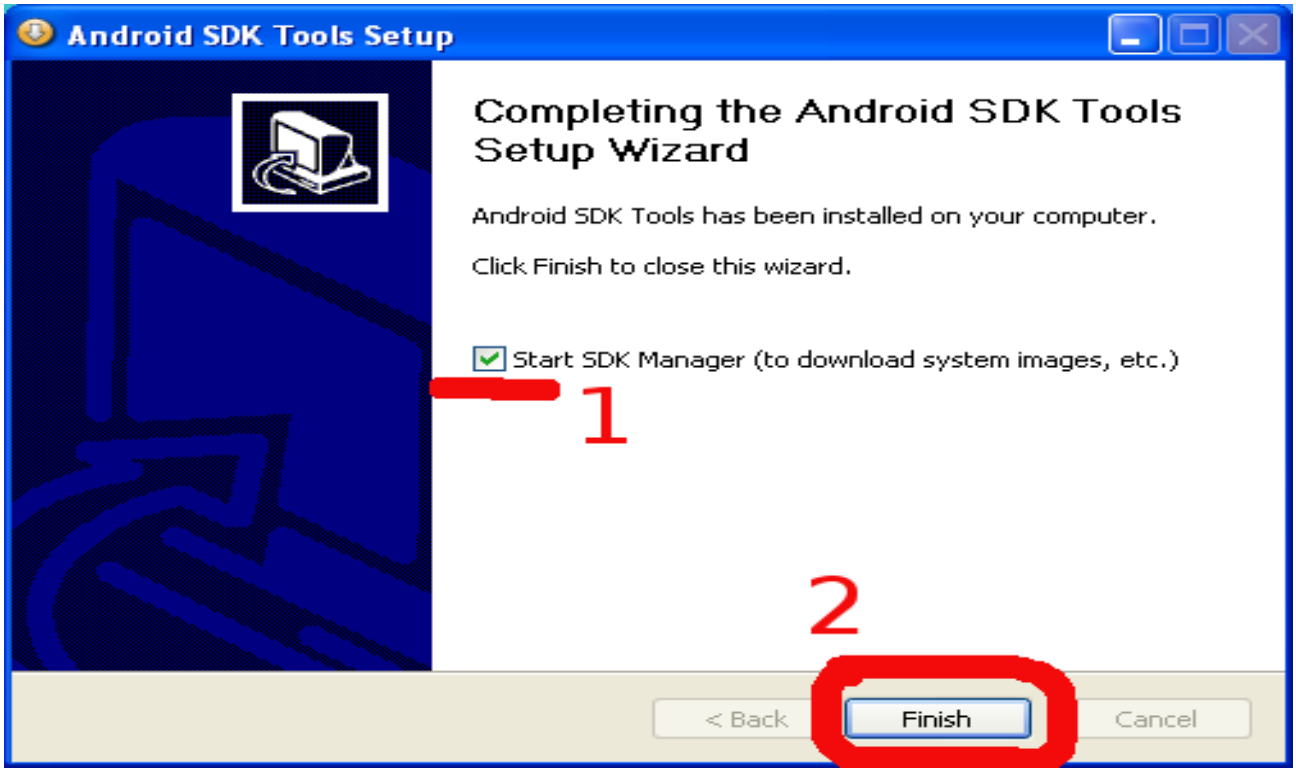
Установка начинается



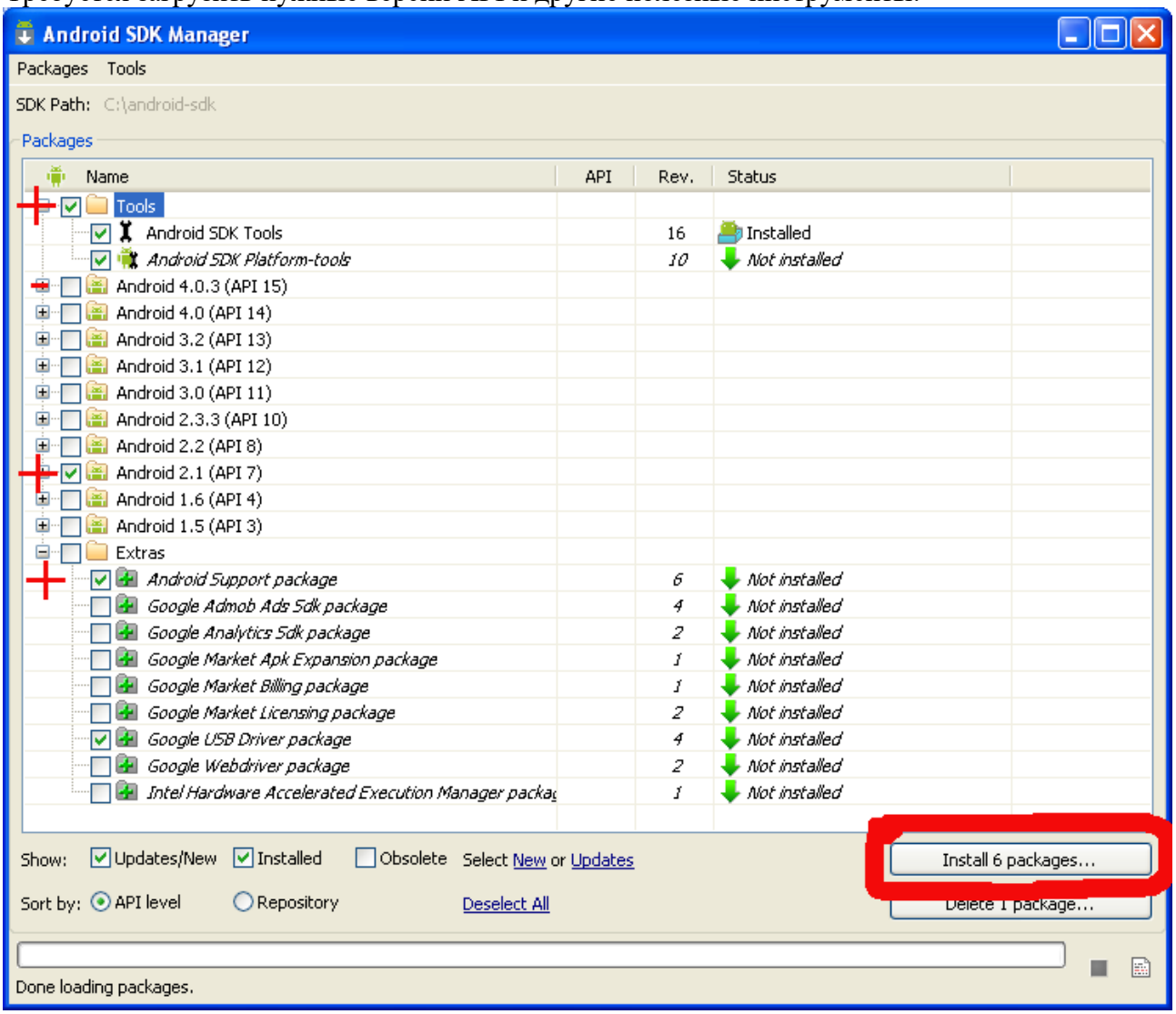
и заканчивается:

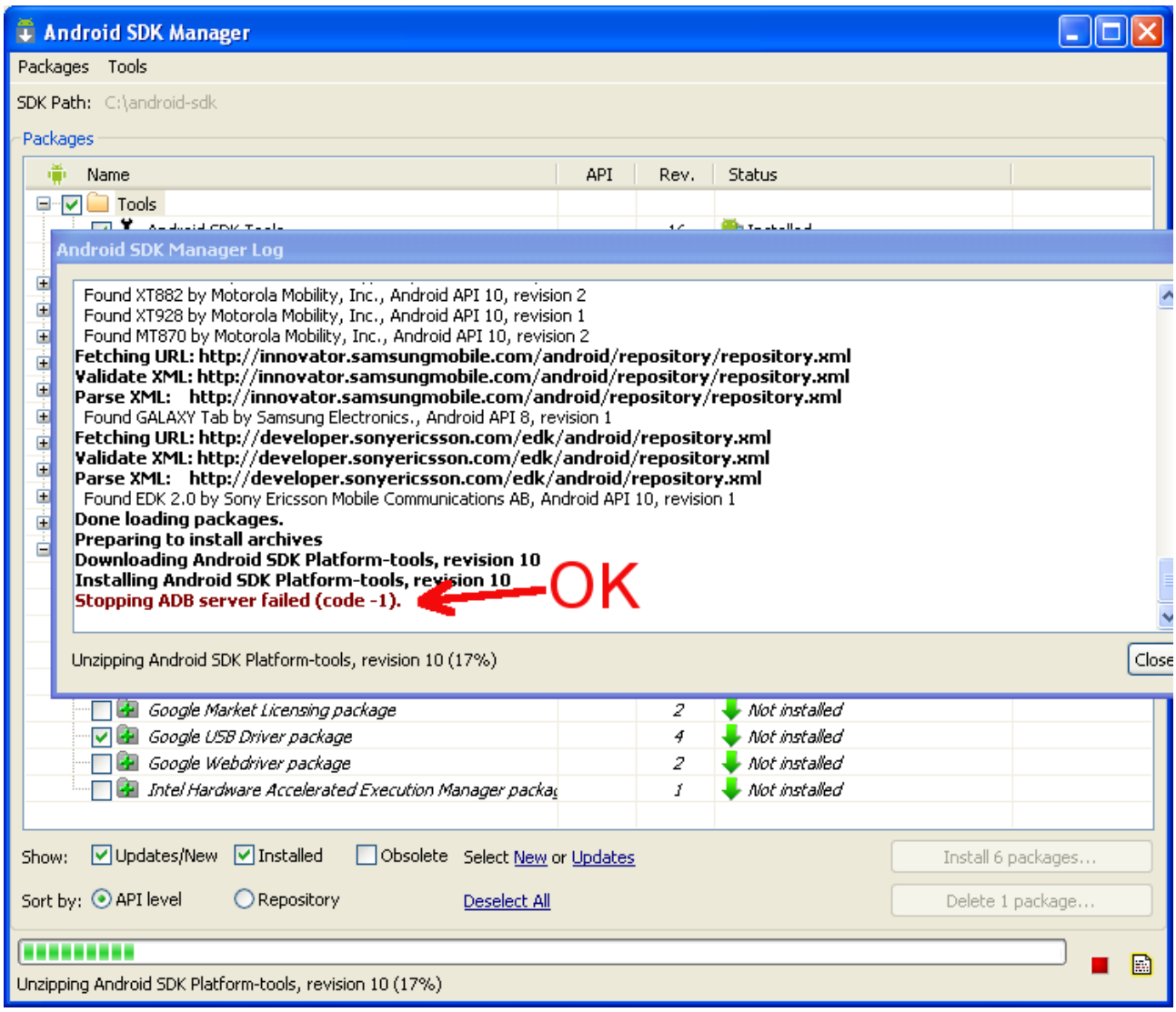


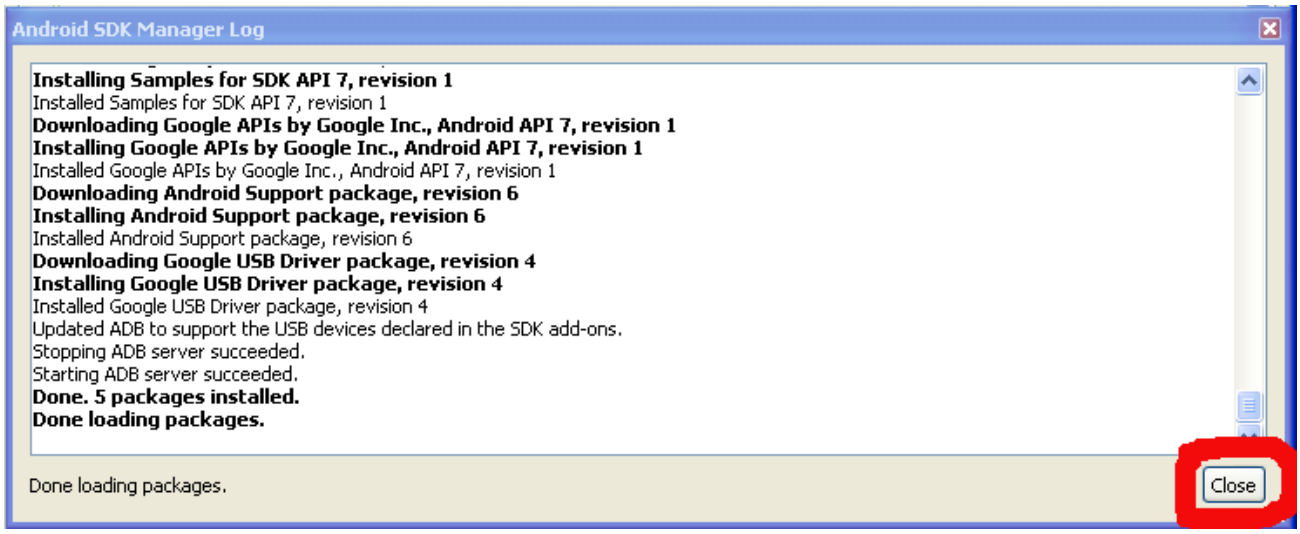
Можно сразу поставить галку**(1)** и запустить SDK Manager для окончательной настройки**(2)**:



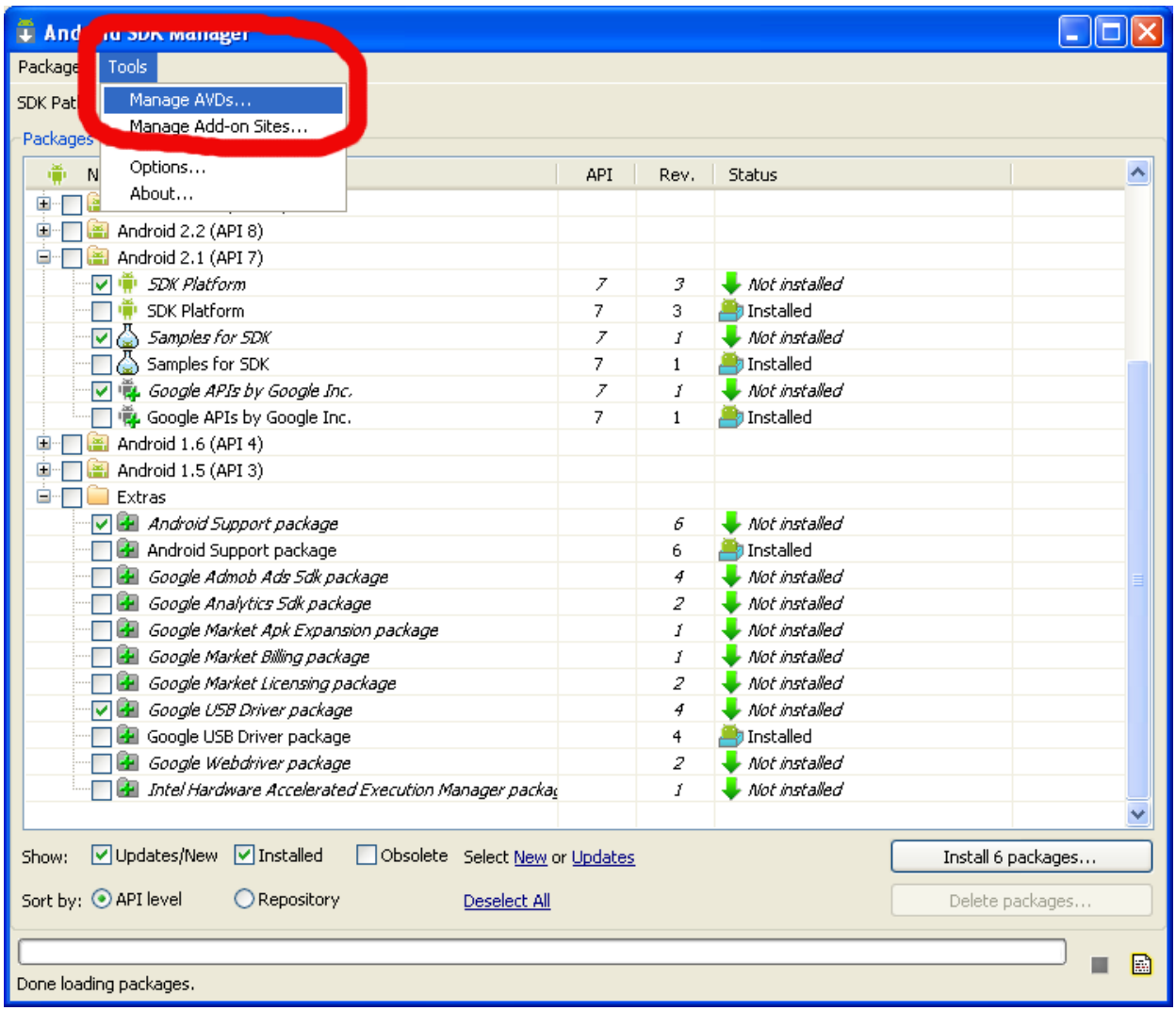
Требуется загрузить нужные версии API и другие полезные инструменты:



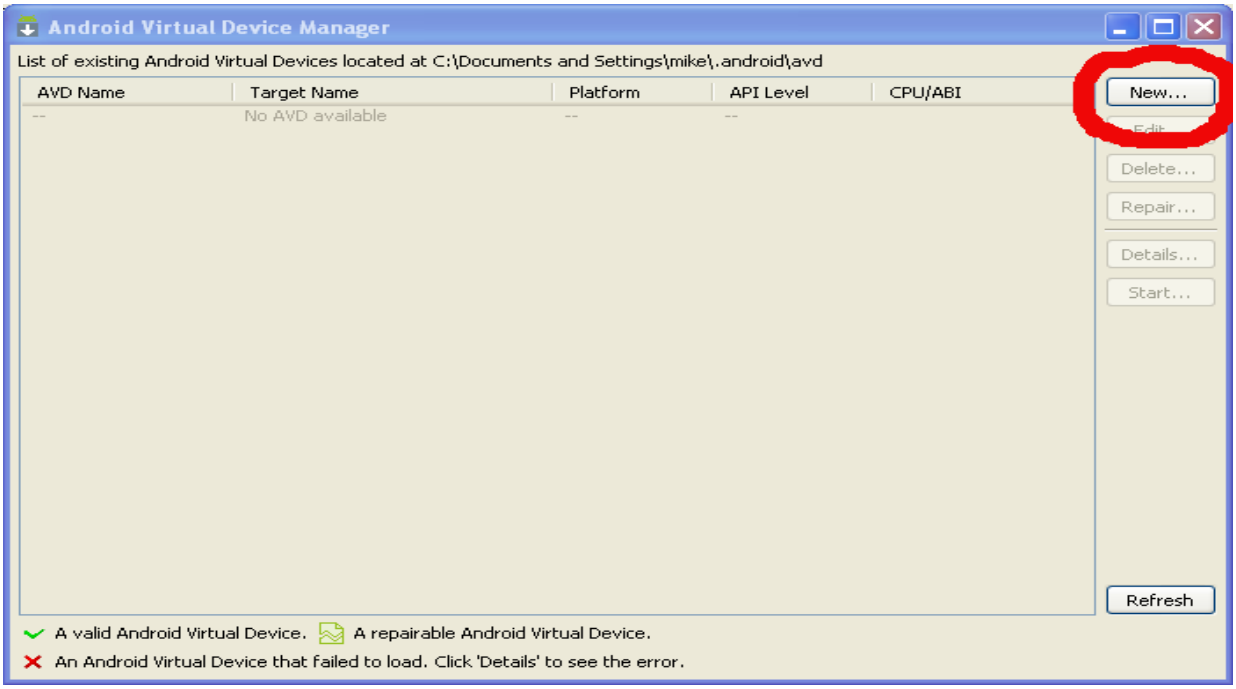




Далее мы запустим менеджер AVD (Android Virtual Devices) и сконфигурируем эмулятор на использование Android версии 2.1 с расширением Google API (GAPI). GAPI нужен, как правило, для приложений, использующих картографические возможности Android.

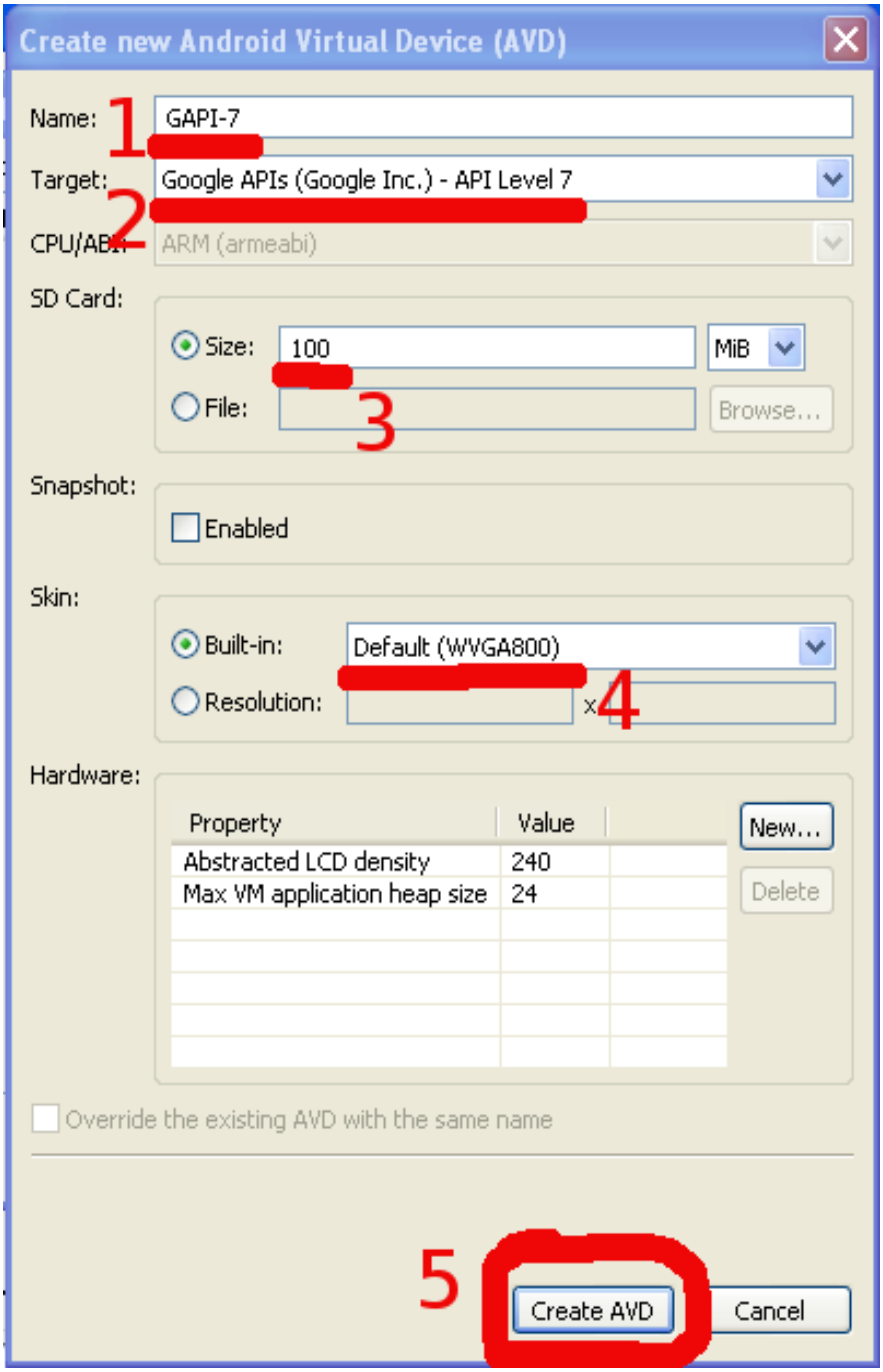


Создадим новое виртуальное устройство:

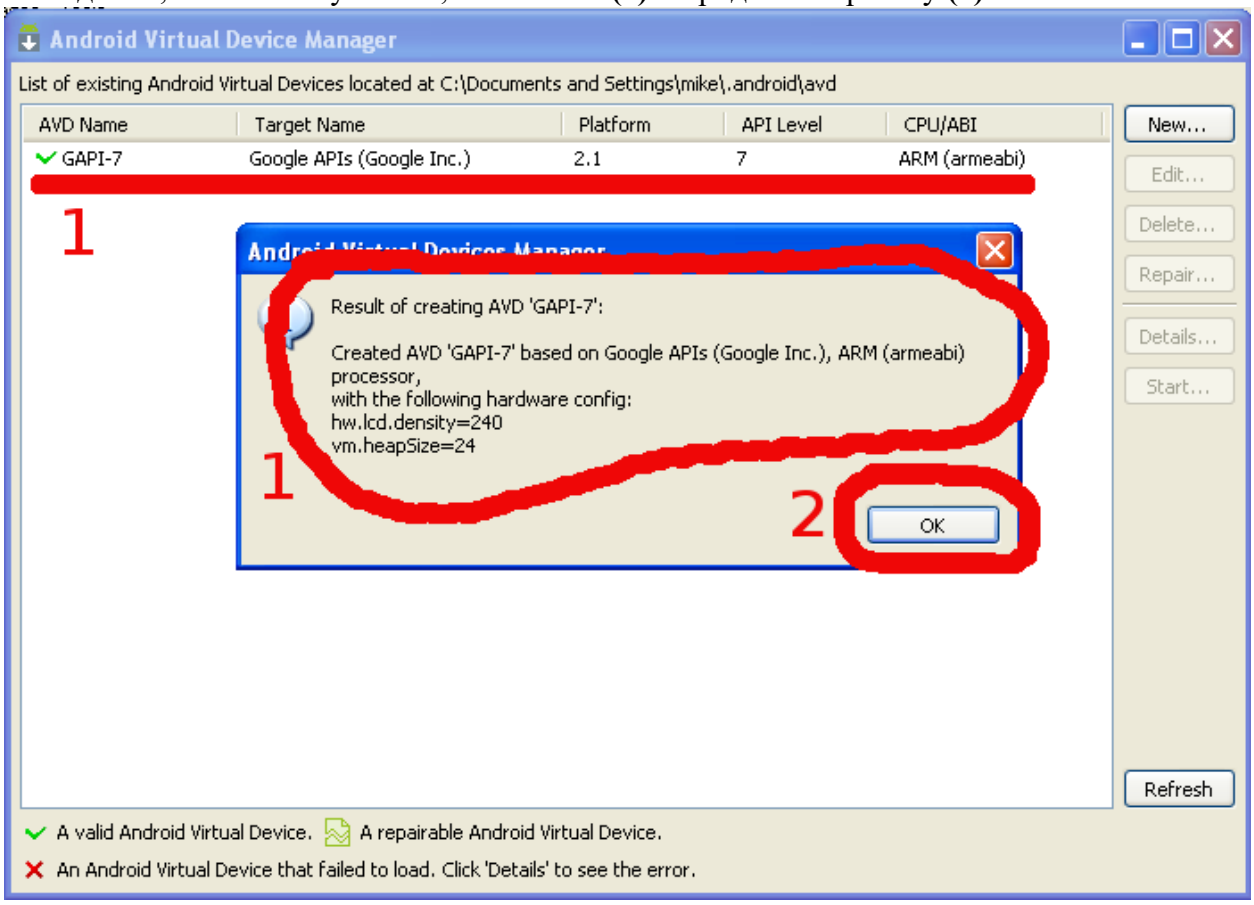


При создании нового виртуального устройства выбираем его название**(1)**, целевой API **(2)**, размер (или файл с образом) виртуальной SD-карты **(3)**, одно из стандартных или собственное разрешение экрана **(4)** и, наконец, создаем устройство**(5)**.

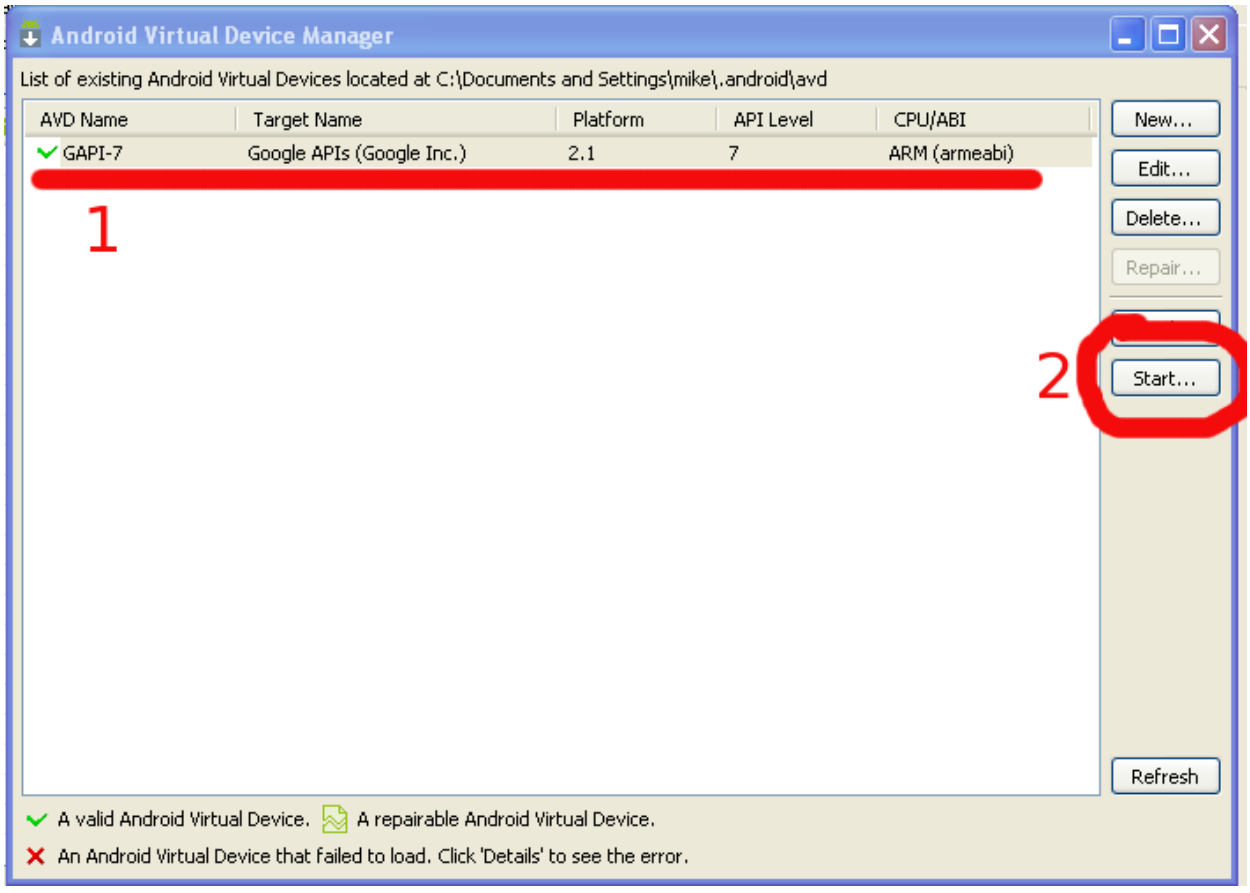
При необходимости можно указать плотность пикселей на экране, максимальный размер кучи для приложений внутри виртуальной машины, а также другие параметры:



Убедимся, что все получилось, как хотели **(1)** и продолжим работу **(2)**:

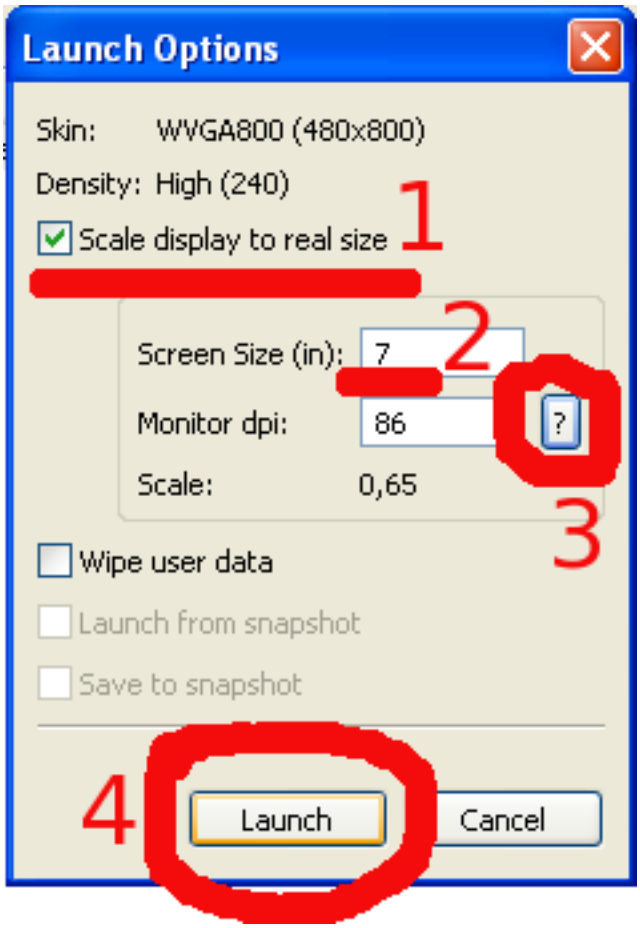


Пришло время запустить эмулятор:

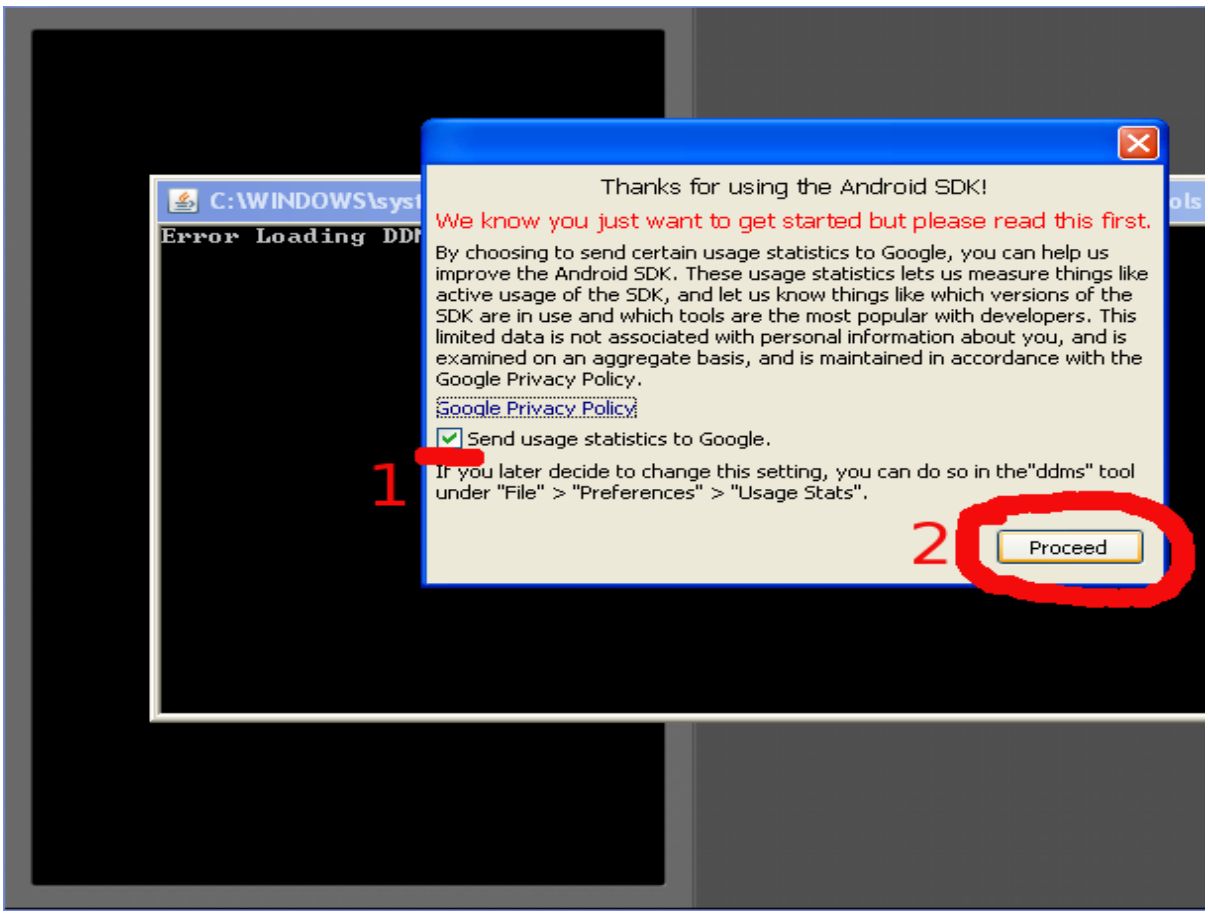


Важно: при использовании *русскоязычных* версий ОС Windows путь к рабочему каталогу AVD может включать русские буквы, что приведет к невозможности запуска эмуляторов. В этом случае можно создать каталог для AVD в корневом каталоге диска (например, **C:\AVD**) и присвоить переменной окружения **ANDROID_SDK_HOME** значение этого пути.

При запуске эмулятора из менеджера AVD можно дополнительно указать некоторые параметры, особенно полезно управление размерами или плотностью пикселей экрана виртуального устройства: поставив галку **(1)**, можно указать желаемый размер экрана **(2)** в дюймах, вычислить, при необходимости, плотность пикселей **(3)** на используемом мониторе ПК:

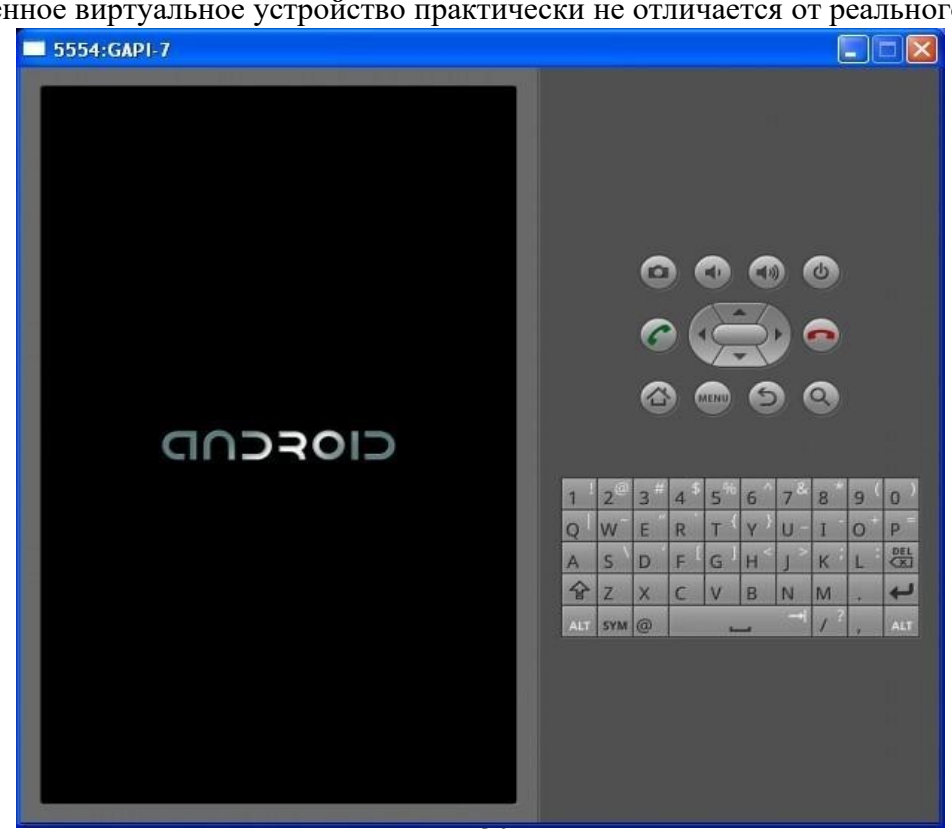


При первом запуске вы можете указать, следует ли отправлять анонимную статистику использования SDK в Google:



Эмулятор начнет загрузку выбранной при создании виртуального устройства версии ОС Android (в первый раз чуть дольше):

Загруженное виртуальное устройство практически не отличается от реального:



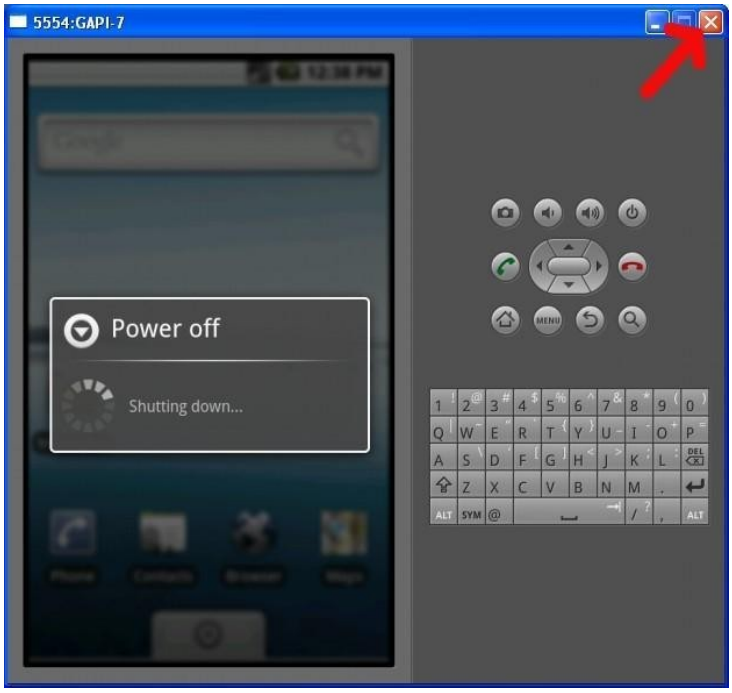
Более того, «пощупать» самые новые версии ОС Android и оценить их функциональные возможности можно ~~безвозмездно, то есть даром~~ просто выбрав нужную версию API при создании виртуального устройства.

> Одна из неприятных особенностей платформы Android с точки зрения разработчика заключается в большом количестве выпущенных версий, значительно различающихся по возможностям. В настоящий момент Open Handset Alliance (т. е. Google) несколько затормозила этот процесс, и на данный момент флагманской версией является 4.0.x. Тем не менее, при выборе уровня API (API level) для нового ПО следует учитывать наличие у пользователей большого количество относительно старых устройств, так что для охвата максимального сегмента рынка оптимальным выбором является Android 2.1 (поддерживается 97% процентов устройств с Android).

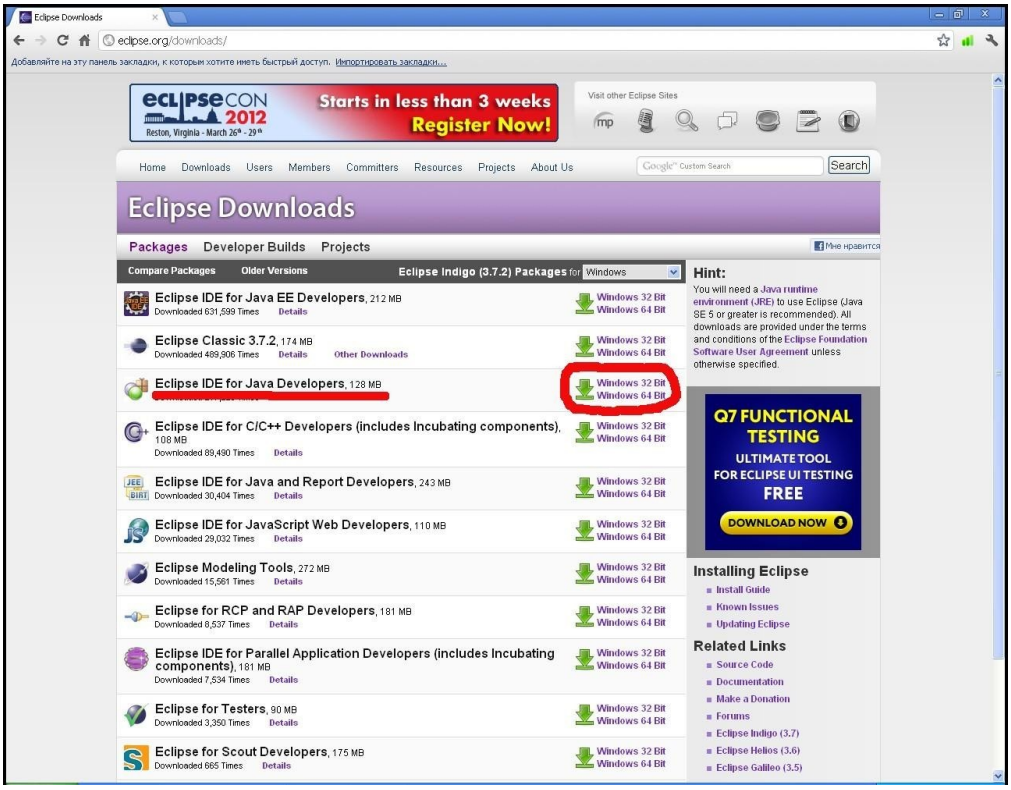
Так же, как и в реальном , в виртуальном устройстве существует опасность разрушить файловую систему при неправильном прекращении работы (сбое питания и т. п.). Далее показан штатный способ выключения виртуального устройства:


5554:GAPI-7
12:36 PM
Phone options
Silent mode
Sound is ON
Airplane mode
Airplane mode is OFF
Power off
1
5554:GAPI-7
Power off
Your phone will shut down.
OK
Cancel

Через несколько секунд окно можно закрыть:



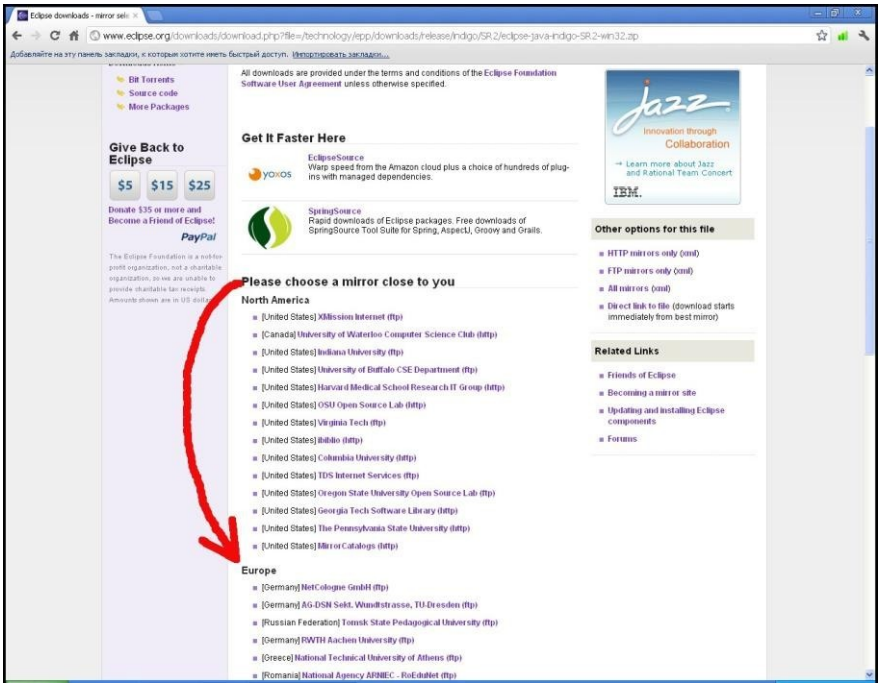
Установка IDE Eclipse



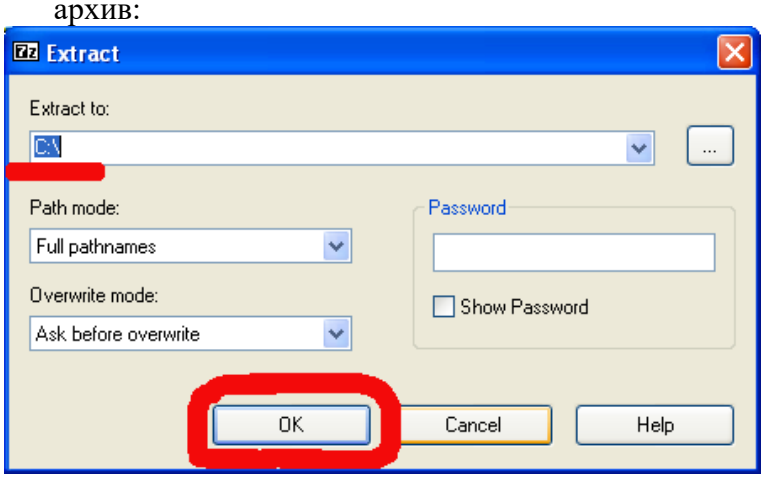
Интегрированная среда разработки Eclipse (довольно странное название, по-английски означает, в том числе помутнение рассудка) доступна для загрузки на официальном сайте по адресу http://eclipse.org/downloads/ и распространяется в виде ZIP-архива:

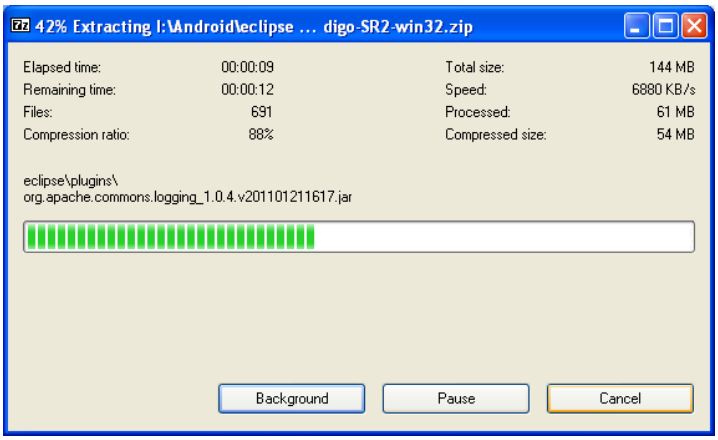
В нашем случае подходящим вариантом является **Eclipse IDE for Java Developers**.

После выбора подходящей для используемой на ПК операционной системы версии IDE, для уменьшения времени загрузки имеет смысл выбрать *европейское зеркало* сайта:



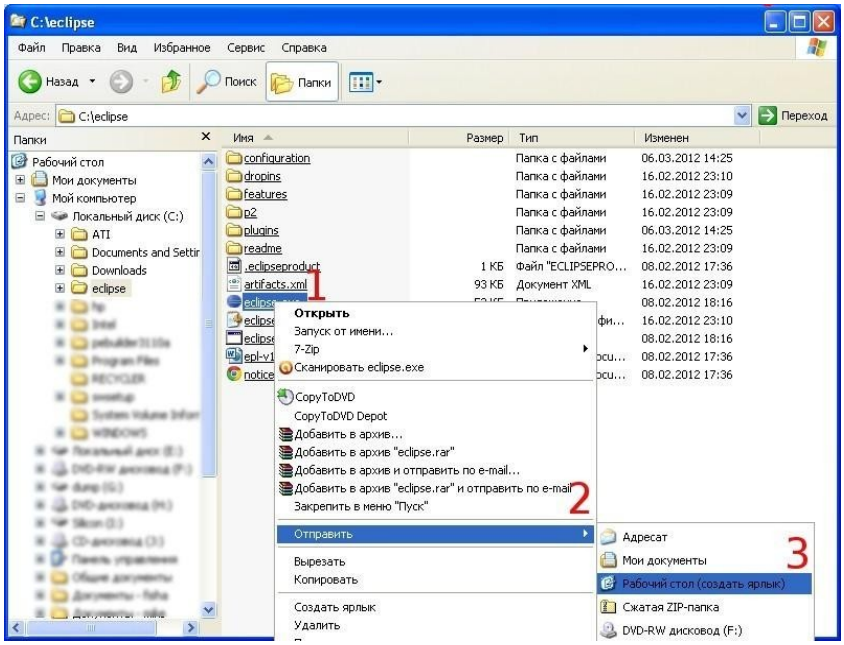
Получив нужный архив, выбираем подходящий путь для установки и распаковываем архив:





После распаковки IDE для удобства запуска создаем ярлык на рабочем столе:

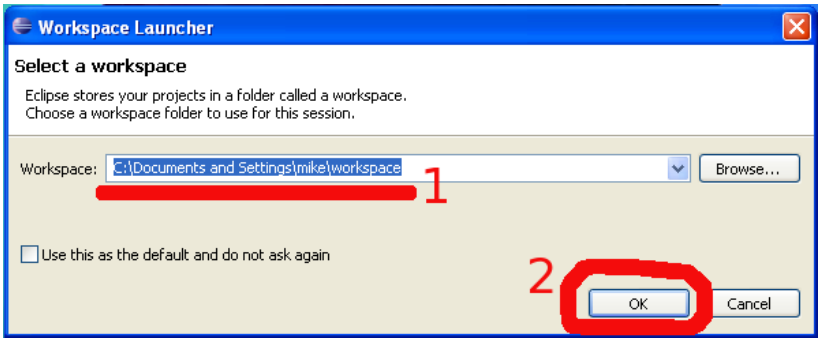
Voila! Установка Eclipse закончена. Остался последний шаг – установка плагина ADT.



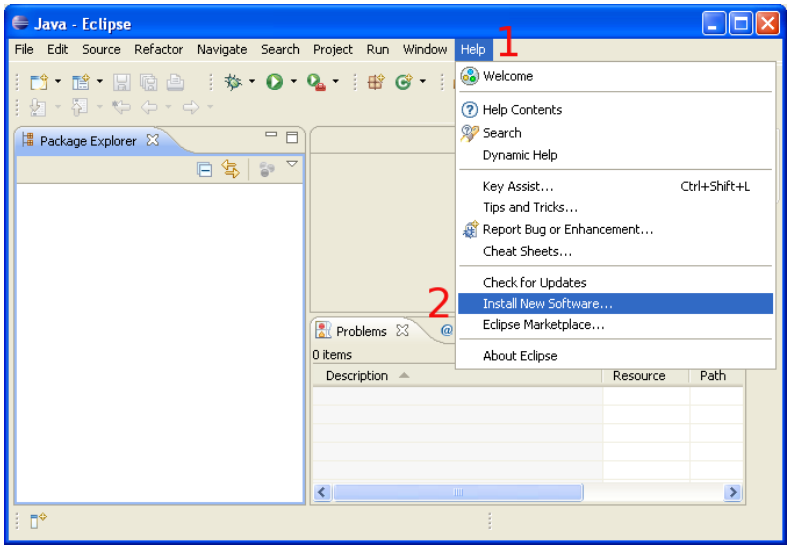
Установка плагина ADT

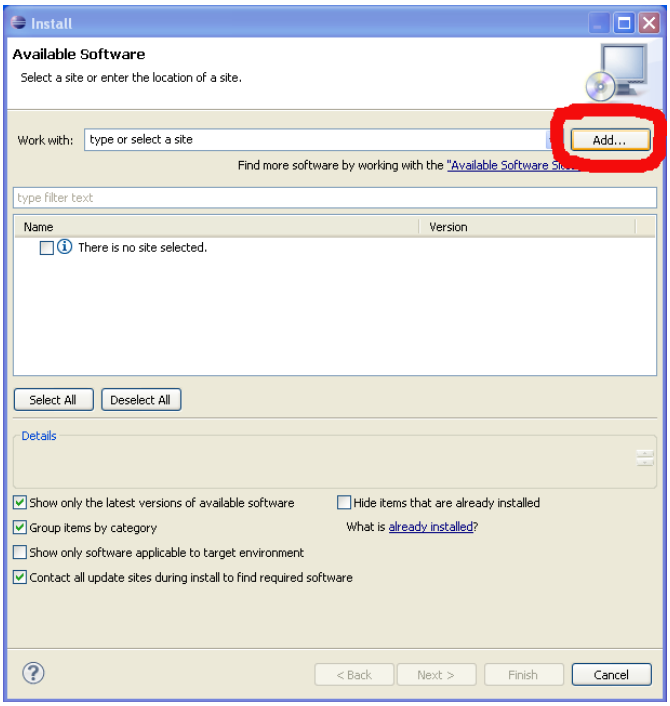
Для установки плагина ADT не требуется ручная загрузка, он будет установлен системой управления плагинами Eclipse. Тем не менее, мы рассмотрим установку подробно.

Запустим Eclipse

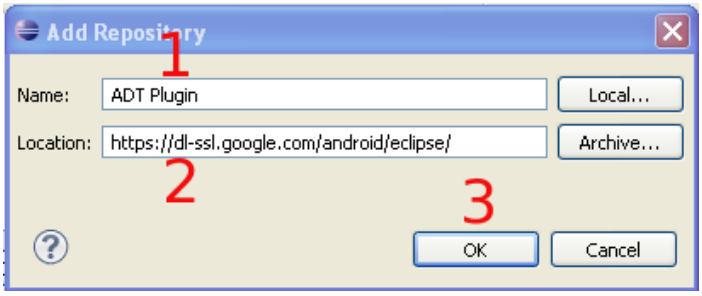


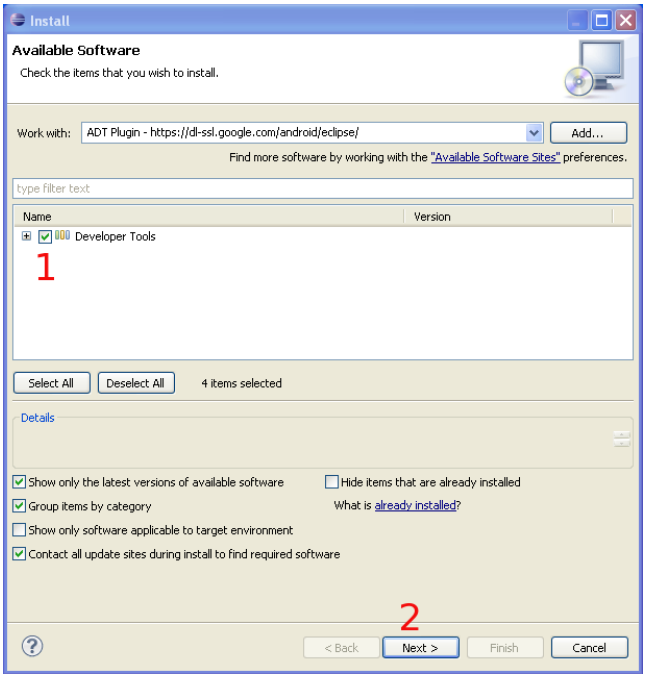
Выберем нужный пункт меню



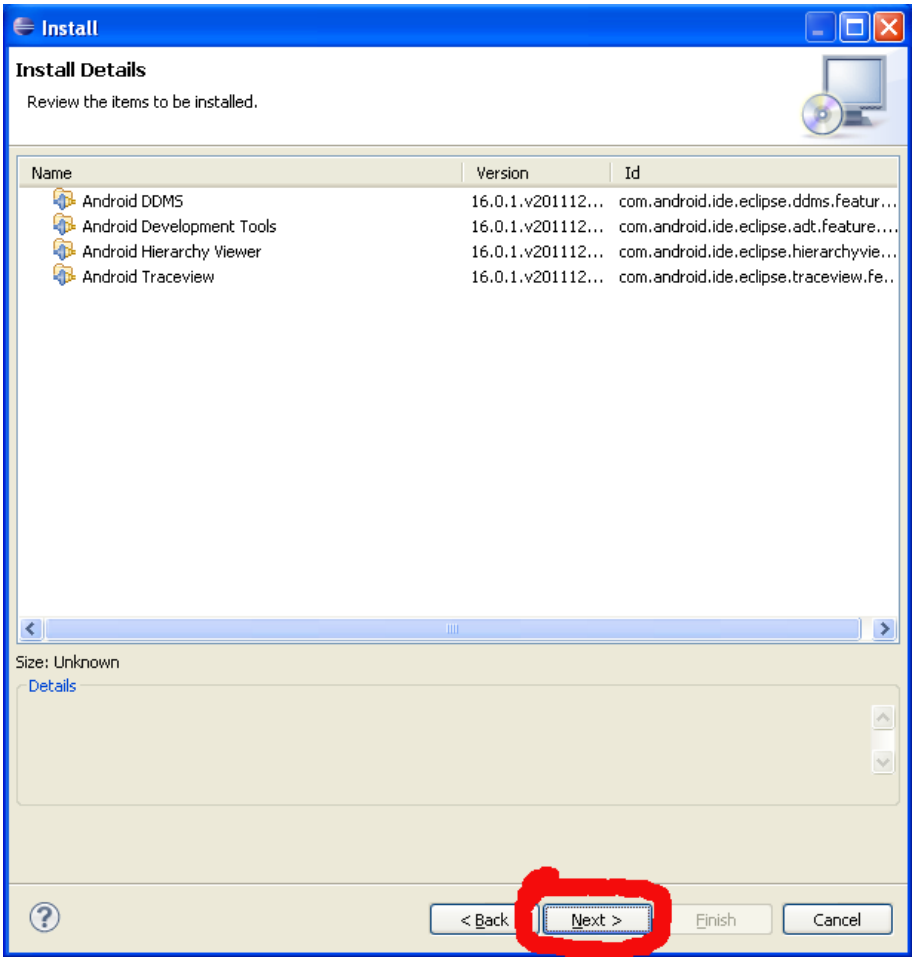


Добавим нужный источник ПО

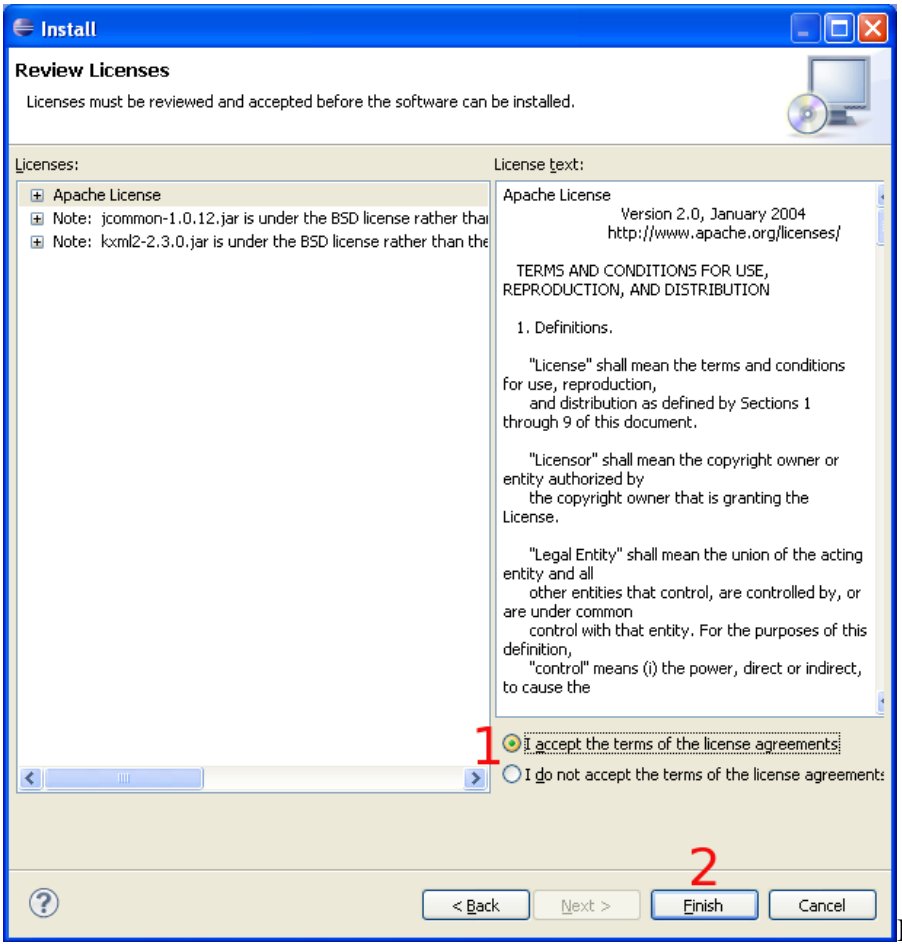




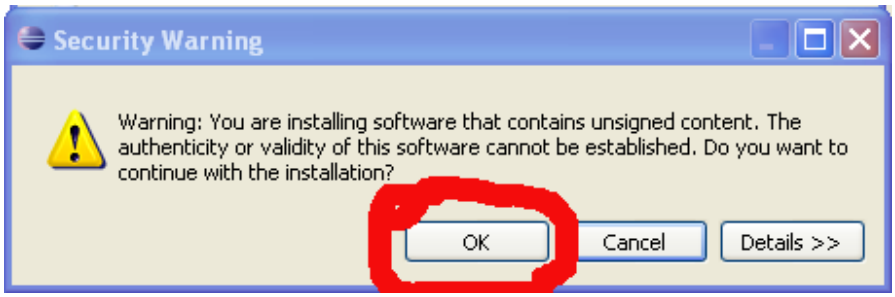
Выберем нужные (все) компоненты для загрузки и установки

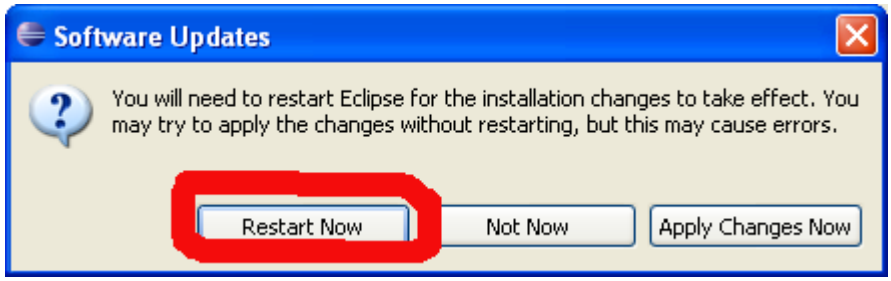


Убедимся в этом

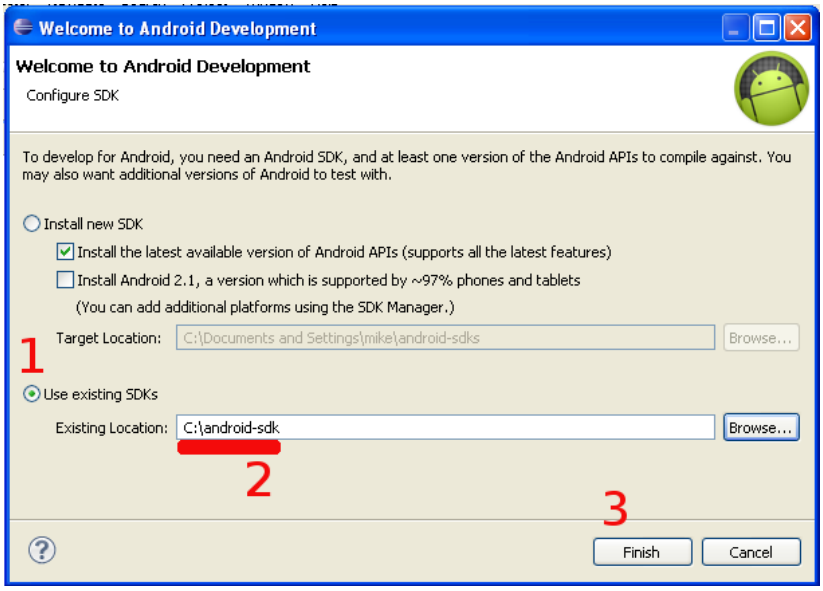


Примем все лицензионные соглашения и продолжим установку:





После установки выбранных компонентов перезапустим Eclipse
И настроим плагин ADT на использование уже установленного Android SDK.

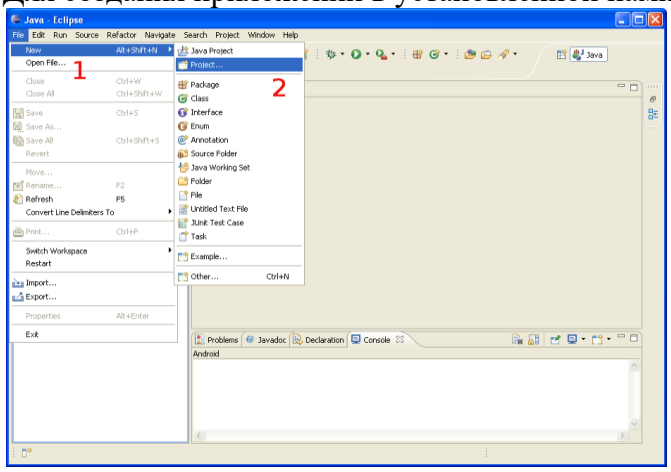


На снимке экрана (выше) можно увидеть, что ADT в состоянии сам загрузить и установить Android SDK. Мы использовали ручную установку для ускорения процесса (не тратили время на ожидание загрузки SDK).

Среда разработки установлена, пришло время создать первое приложение для платформы Android и запустить его в эмуляторе.

Создание первого приложения под Android

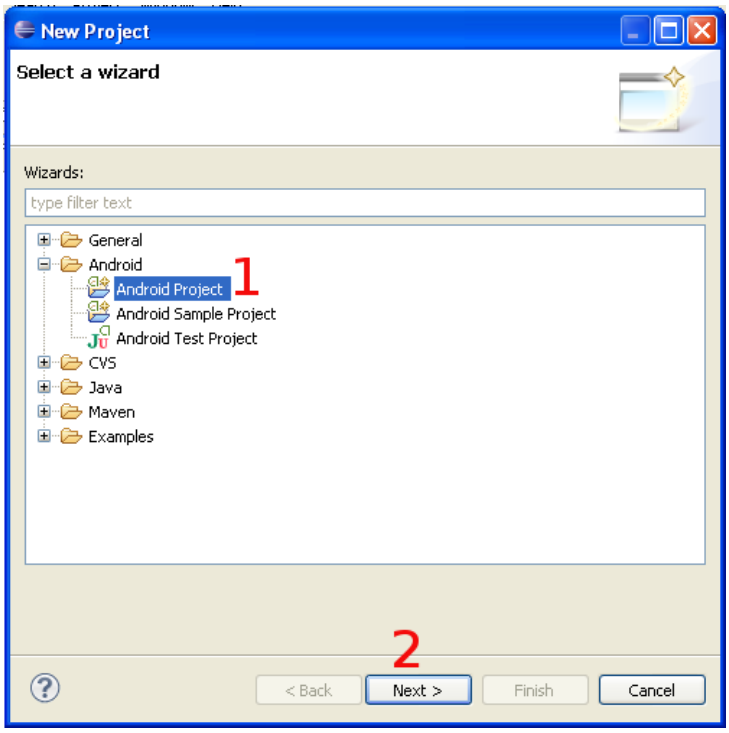
Для создания приложений в установленной нами среде разработки удобно использовать



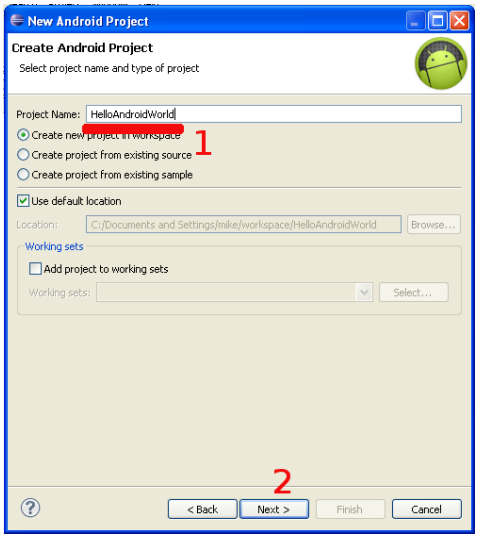
«Мастер создания нового

проекта»: Выбираем «**Android**

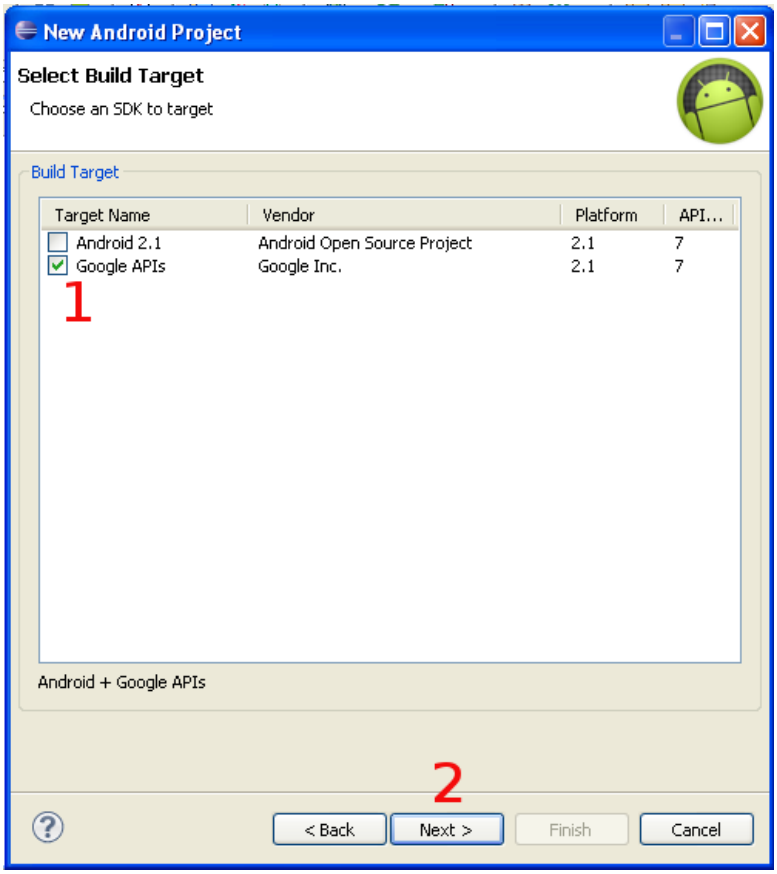
Project»

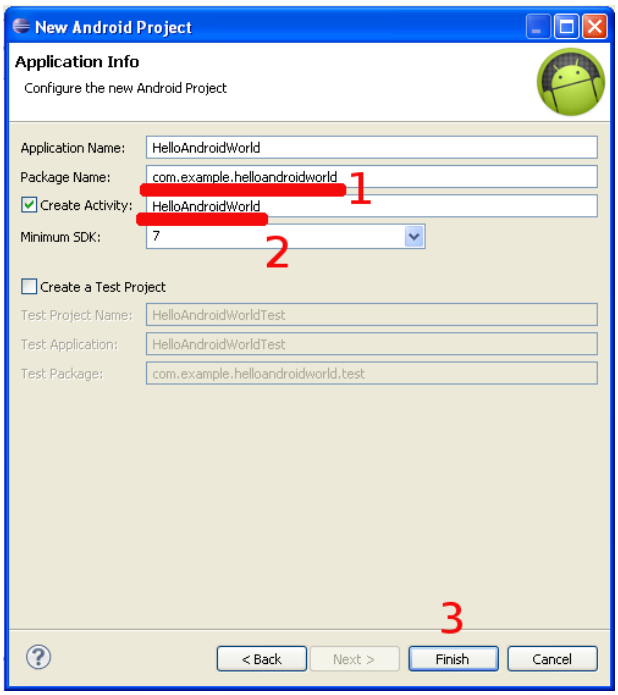

New Project
Select a wizard
Wizards:
type filter text
General
Android
Android Project
Android Sample Project
Android Test Project
CVS
Java
Maven
Examples
1
2
< Back
Next >
Finish
Cancel

Вводим **имя проекта**



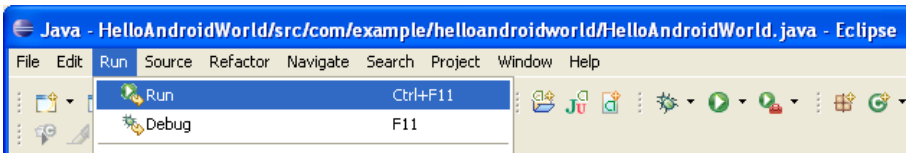
Выбираем **целевую платформу**



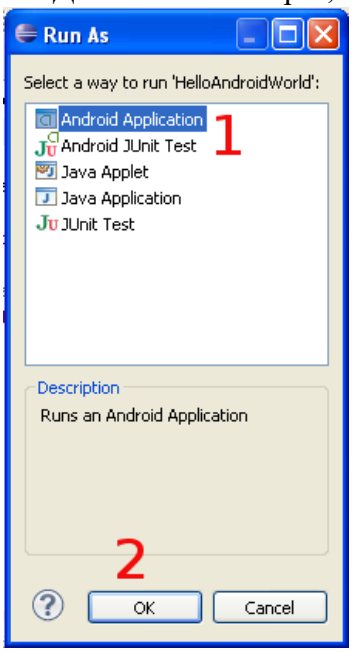


Вводим **имя пакета (1)** и имя класса **Активности(2)**, который будет для нас автоматически создан мастером:

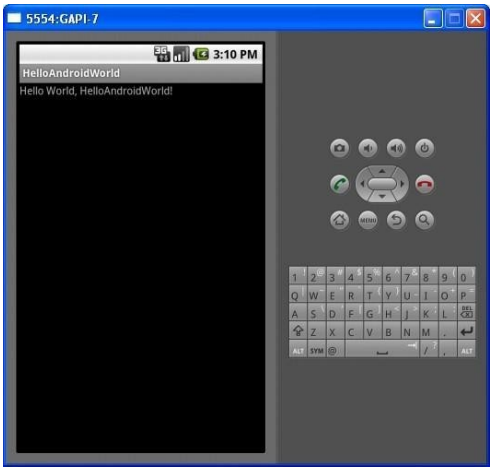
Созданный проект сразу ~~пока мы не внесли в него несовместимы с жизнью изменения~~ является работоспособный приложением, так что мы его можем запустить:



Даем понять Eclipse, как именно мы хотим запустить на выполнение наш проект:



И наслаждаемся результатом:



Практическая работа №2 Установка среды разработки мобильных приложений с применением виртуальной машины

Практическая работа №3 Создание эмуляторов и подключение устройств»

Цель работы:

Знакомство с интерфейсом среды программирования. Изучение структуры проекта

Теоретические сведения.

Рассмотрите основные понятия Android проекта.

Структура проекта

- src—«исходный код» приложения (java-классы)
- assets —пустая директория. Может использоваться для сохранения raw-файлов.
- gen—хранилище генерируемых системных файлов. В частности, здесь располагается файл R.java, в котором хранятся идентификаторы всех ресурсов, создаваемых в проекте (строковые ресурсы и т.п.).
- libs—различные библиотеки, используемые приложением
- res—ресурсы проекта.
- AndroidManifest.xml—файл описания проекта (поддерживаемые версии SDK, версия приложения и т.п.)
- project.properties—файл, включающий настройки проекта, такие как build target.

Ресурсы проекта

- anim/ Содержит XML файлы, компилируемые в анимационные объекты.
- color/ Содержит XML файлы описывающие цвета.
- drawable/ Содержит растровые файлы (PNG, JPEG, orGIF), 9-Patch файлы, и XML файлы, описывающие Drawableshapes или Drawableobjects включающие множественные состояния(нормальное, нажатое, состояние фокуса).
- layout/ Содержит XML файлы описывающие макеты экрана
- menu/ Содержит XML файлы, определяющие меню приложения.
- raw/ Для хранения произвольных файлов.
- values/ Содержит XML файлы, компилируемые во множество видов ресурсов (strings.xmlи т.д).
- xml/ СодержитXML файлы конфигурирующие компоненты приложения. Например, XML файл определяющий экран настроек.

Пример простейшего файла AndroidManifest.xml

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

```
android:versionCode="1"
```

```
android:versionName="1.0">
<uses-sdkandroid:minSdkVersion="19"/>
<application                                    android:label="@string/app_name"
android:icon="@drawable/ic_launcher">
<activity android:name="MyActivity"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
</application>
</manifest>
```

Для начала работы требуется:

- Java Development Kit
- Android Software Development Kit

Android Virtual Device Manager

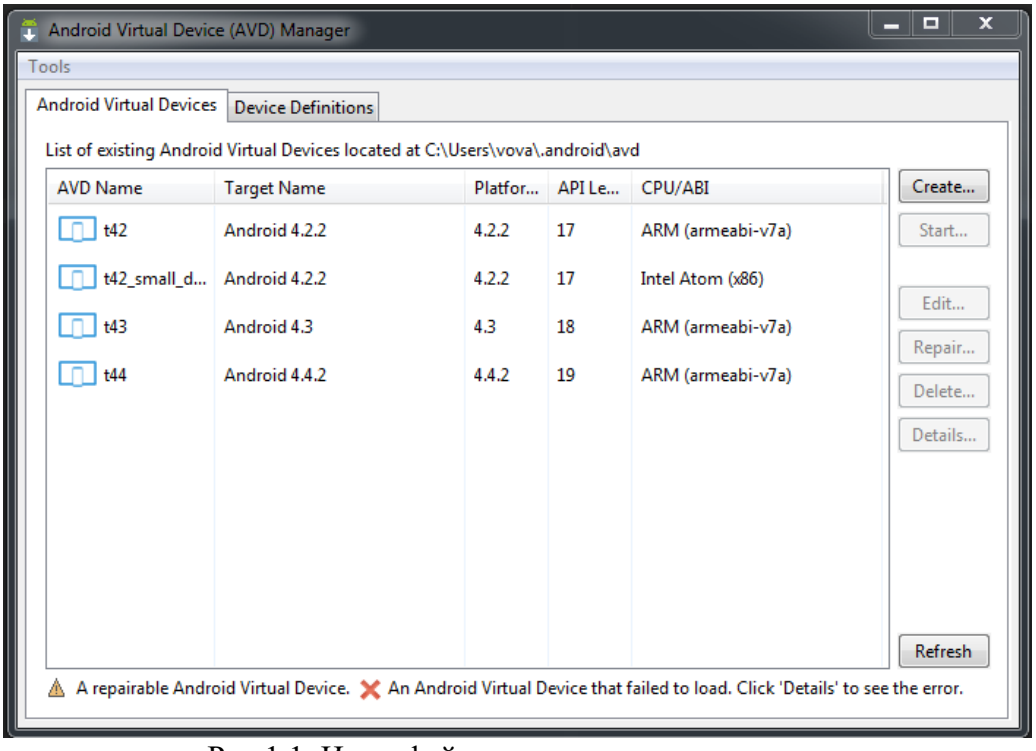


Рис.1.1. Интерфейс виртуального менеджера

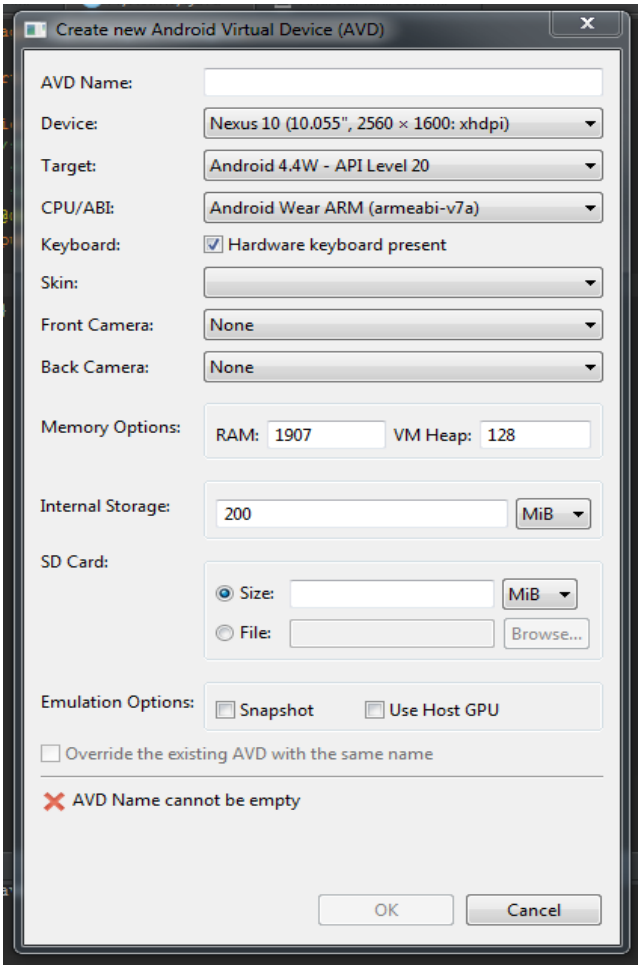


Рис. 1.2. Создание нового виртуального устройства

Задания лабораторной работы

Задание 1. Создание проекта приложения

Запустите среду программирования в IDE InelliJ Idea

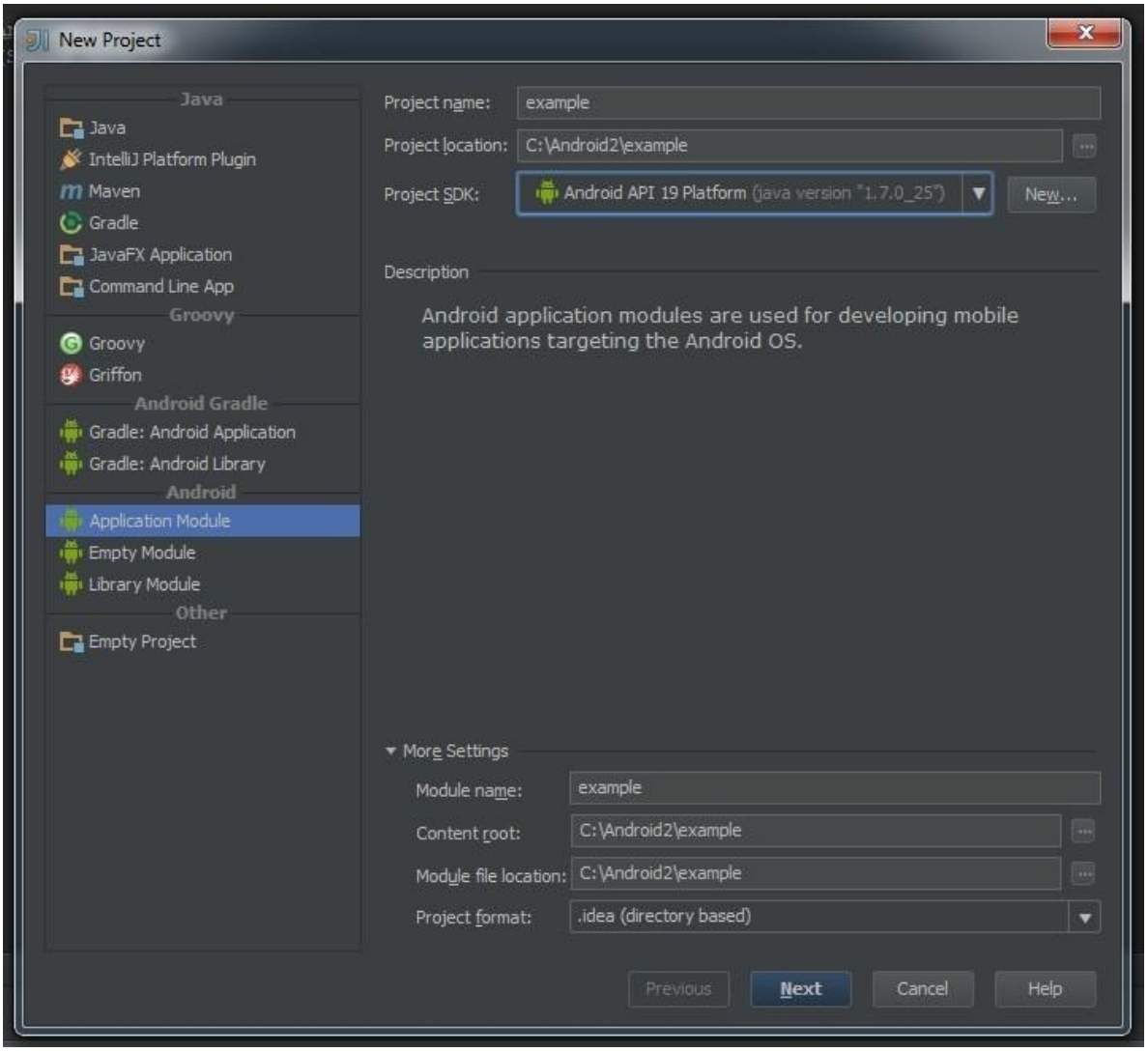


Рис. 1.3. Создание нового проекта в среде IntellyJ Idea

Введите данные проекта. Сохраните проект.

Задание 2. Создание приложений с одним экраном (Activity) Необходимо создать два **activity** и организовать переход между ними. Содержимое Activity 1

- кнопка с именем btn1

Содержимое Activity 2:

- TextView с текстом «Переданный параметр: *значение_параметра*». *значение_параметра* – принятый параметр из Activity 1.

При запуске приложения пользователь должен попадать на экран с **Activity 1**. После нажатия на кнопку btn1 необходимо осуществить переход к **Activity 2** и передавать параметр из **Activity 1**. В качестве значения передаваемого параметра использовать свою фамилию.

Ключевые классы: Activity, Intent, Button, TextView.

Лабораторная работа №2. Основы верстки

Цель работы:

Изучить основы верстки. Научиться управлять интерфейсом мобильного устройства при разработке программного приложения.

Теоретические сведения

Просмотрите основные сведения о классах, которые понадобятся при разработке приложения.

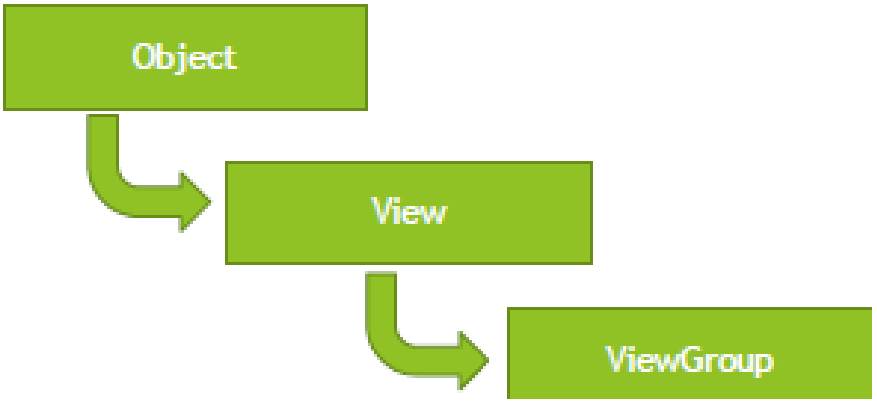


Рис.2.1. Иерархия классов View и ViewGroup

Дерево представлений для Activity представлено на рисунке 2.2.

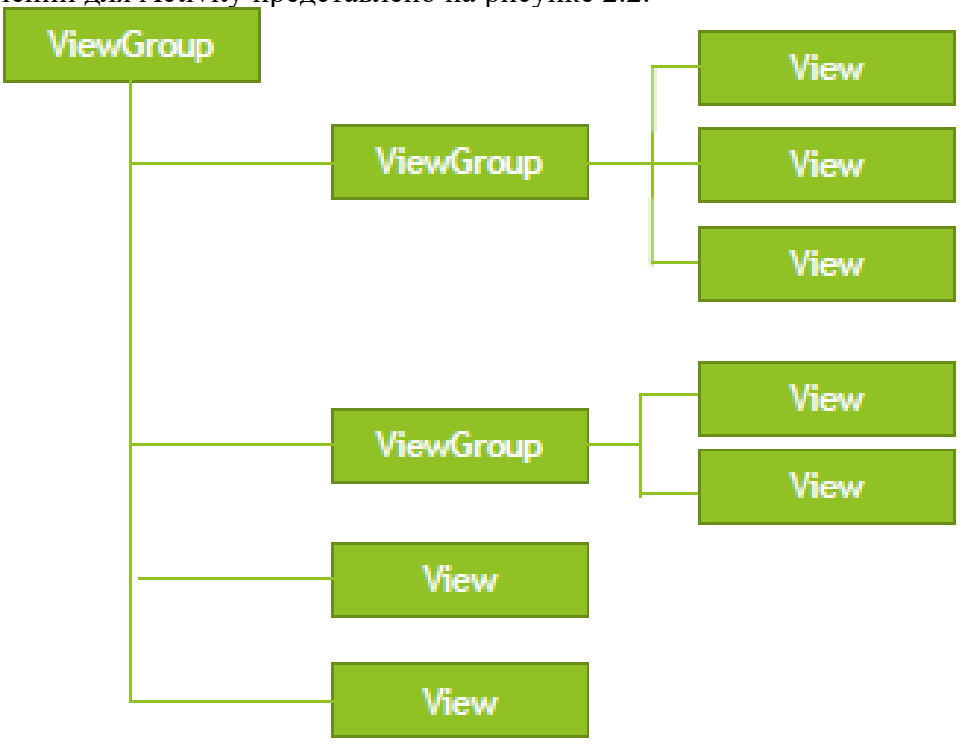


Рис.2.2. Дерево представлений для Activity

Файл разметки имеет следующую структуру

```
<?xml version="1.0" encoding="utf-8"?>
        <LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
```

```
>
<LinearLayout android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="0.25" android:padding="5dp">
<Button android:layout_width="0dp"
android:layout_weight="0.33"
android:layout_height="wrap_content"
android:text="Button1"
android:id="@+id/button3" android:layout_gravity="right"/>
<Button android:layout_width="0dp"
android:layout_weight="0.33"
android:layout_height="wrap_content"
android:text="Button2" android:id="@+id/button"/>
<Button android:layout_width="0dp"
android:layout_weight="0.33"
android:layout_height="wrap_content"
android:text="Button3"
android:id="@+id/button2" android:layout_gravity="center_horizontal"/>
</LinearLayout>
<LinearLayout android:layout_width="match_parent"
android:layout_height="0dp"
android:paddingLeft="40dp"
android:paddingRight="40dp"
android:layout_weight="0.5"
android:gravity="center_vertical">
<Button android:layout_width="0dp"
android:layout_weight="0.33"
android:layout_height="wrap_content"
android:text="Button4" android:id="@+id/button3"/>
<Button android:layout_width="0dp"
android:layout_weight="0.33"
```

```
android:layout_height="wrap_content"
android:text="Button5" android:id="@+id/button"/>
</LinearLayout>
<LinearLayout android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="0.25" android:padding="5dp"
android:gravity="bottom">
…
</LinearLayout>
</LinearLayout>
```

Распространенные виды макетов

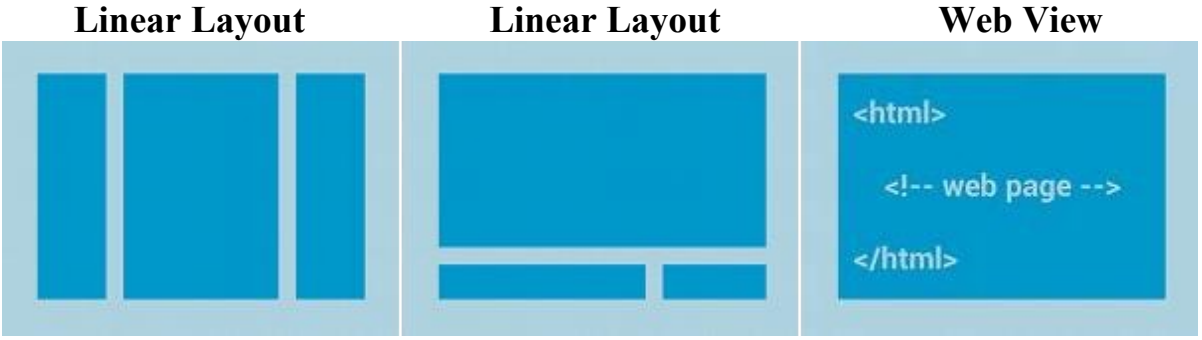


Рис. 2.3. Распространенные виды макетов

Макеты с адаптером

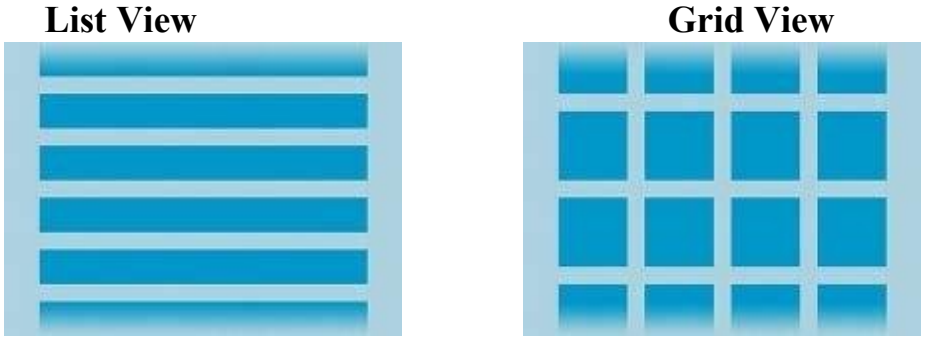


Рис.2.4. Интерфейс макетов с адаптером

Атрибуты LinearLayout

Attribute Name	*Related Method*	*Description*
android:baselineAligned	setBaselineAligned(boolean)	When set to false, prevents the layout from aligning its children's baselines.
android:baselineAlignedChildIndex	setBaselineAlignedChildIndex(int)	When a linear layout is part of another layout that is baseline aligned, it can specify which of its children to baseline align to (that is, which child TextView).
android:divider	setDividerDrawable(Drawable)	Drawable to use as a vertical divider between buttons.
android:gravity	setGravity(int)	Specifies how an object should position its content, on both the X and Y axes, within its own bounds.
android:measureWithLargestChild	setMeasureWithLargestChildEnabled(boolean)	When set to true, all children with a weight will be considered having the minimum size of the largest child.
android:orientation	setOrientation(int)	Should the layout be a column or a row? Use "horizontal" for a row, "vertical" for a column.
android:weightSum		Defines the maximum weight sum.

Задания лабораторной работы

Задание 1. Разработать мобильное приложение, состоящее из четырех activity.

После запуска приложения пользователь должен попадать на экран с activity1. На данном экране должно быть представлено меню, состоящее из четырех кнопок. Высота кнопок должна составлять 20% от высоты экрана. Расстояние между кнопками – 2%. Первая и последняя кнопка должны быть на равном расстоянии от краев экрана. Ширина кнопок 75%, выравнивание посередине.

После нажатия на первую кнопку пользователь должен переходить к activity2, его внешний вид представлен на рисунке 1. Верстка должна осуществляться с использованием LinearLayout, ширина кнопок должна задаваться в процентах от ширины экрана.

Рис. 2.5. Внешний вид экрана для первого задания

После нажатия на вторую кнопку в activity1 пользователь должен переходить к activity3, его внешний вид представлен на рисунке 2. Верстка должна осуществляться с использованием RelativeLayout (не использовать LinearLayout).

Рис. 2.6. Результат первого этапа выполнения задания

Третья кнопка в activity1 должна создавать activity3. Внешний вид activity3 представлен на рис. 2.7.

Рис.2.3. Интерфейс приложения на этапе activity3.

Кнопка должна быть выровнена по центру экрана. Цвет обводки кнопки #505050. Толщина обводки в соответствии с месяцем вашего рождения (от 1 до 12). Радиус скругления 24dp. Цвет фона экрана #FFFFFF. При нажатии на кнопку ее цвет должен изменяться на светло-зеленый.

Нажатие на четвертую кнопку в activity1 должно приводить к закрытию приложения.

Практическая работа №4 Настройка режима терминала»

Практическая работа №5 Создание нового проекта»

Цель работы

Изучить работу Android приложения с базой данных.

Теоретические сведения.

SQLite

- SQLite—компактная встраиваемая реляционная база данных с открытым исходным кодом. Поддерживает динамическое типизированиеданных.
- Возможные типы полей: INTEGER, REAL, TEXT, BLOB.

SQLiteOpenHelper

Абстрактные методы

- onCreate( ) -вызывается при первом создании базы данных;
- onUpgrade( ) -вызывается при модификации базы данных

Реализация метода onCreate

```
public void onCreate( SQLiteDatabasedb) {
dЬ.execSQL( "CREATE TABLE"+ TABLE_NAME
+ "(_id INTEGER PRIMARY KEYAUTOINCREMENT ,"
+ COL NAME + " TEXT ,"+ COL PHONE + " TEXT ) ; " ) ;
```

Реализация метода onUpgrade

```
@ Override
publicvoid onUpgrade( SQLiteDatabasedb, intoldVersion, intnewVersion)
{
db.execSQL( "DROP TABLE IF EXISTS"+ TABLE_NAME ) ;
onCreate( db) ;
}
```

Чтение из базы данных

```
Cursor query( String table, String [ ] columns, String
selection, String [ ] selectionArgs,
String groupBy, String having , String sortOrder)
```

Позиционирование курсора

- moveToFirst( ) -перемещает курсор в первую запись в выборке;
- moveToLast( ) -перемешает курсор в последнюю запись в выборке;
- moveToNext( ) -перемешает курсор в следующую запись и одновременноопределяет, существует ли эта запись. Метод moveToNext( ) возвращаеttrue, если курсор указывает на другую строку после перемещения, и false,если текущая запись была последней в выборке;
- moveToPrevious( ) -перемешает курсор в предыдущую запись;

- moveToPosition( ) -перемешает курсор в указанную позицию;
- getPosition( ) -возвращает текущий индекс позиции курсора.

- isFirst( ) ;
- isLast( ) ;
- isBeforeFirst( ) ;
- isAfterLast( ) .

Обновление и удаление записей

- intupdate(String table, ContentValuesvalues, String whereClause, String [ ] whereArgs)
- intdelete (String table, String whereClause, String [ ] whereArgs)

Задания на лабораторную работу

Задание 1. Необходимо создать приложение, взаимодействующее с базой данных. Первое активити должно содержать три кнопки. При нажатии на первую кнопку должно открываться новое активити, выводящее информацию из таблицы «Одногруппники» в удобном для восприятия формате.

При запуске приложения необходимо:

1. Создать БД, если ее не существует.
2. Создать таблицу «Одногруппники», содержащую поля:
    - ID;
    - ФИО;
    - Время добавления записи.
3. Удалять все записи из БД, а затем вносить 5 записей об одногруппниках.

При нажатии на вторую кнопку необходимо внести еще одну запись в таблицу.

При нажатии на третью кнопку необходимо заменить ФИО в последней внесенной записи на Иванов Иван Иванович.

Задание 2. Создать новое отдельное приложение на основе приложения, созданного в части 1.

Переопределить функцию onUpgrade. При изменении версии БД необходимо удалить таблицу «Одногруппники», создать таблицу «Одногруппники» содержащую следующие поля:

- ID;
- Фамилия;
- Имя;
- Отчество;
- Время добавления записи.

Изменить версию базы данных.

Практическая работа №6 Изучение и комментирование кода»

1. Создайте новый проект **MetroPicker**.
2. Добавьте вспомогательную Активность **ListViewActivity** для отображения и выбора станций метро, в качестве заготовки используйте результаты лабораторной работы **«Использование ListView»** .
3. Отредактируйте файл разметки **res/layout/main.xml**: добавьте кнопку выбора станции метро, присвоив идентификаторы виджетам *TextView* и *Button* для того, чтобы на них можно было ссылаться в коде.
4. Установите обработчик нажатия на кнопку в главной Активности для вызова списка станции и выбора нужной станции.
5. Напишите нужный обработчик для установки выбранной станции метро в виджет *TextView* родительской Активности (метод *setText* виджета *TextView* позволяет установить отображаемый текст). Не забудьте обработать ситуацию, когда пользователь нажимает кнопку «Назад» (в этом случае «никакой станции не выбрано» и главная Активность должна известить об этом пользователя).
6. Убедитесь в работоспособности созданного приложения, проверив реакцию различные действия потенциальных пользователей.

Неявные намерения

Неявные намерения используются для запуска Активностей для выполнения заказанных действий в условиях, когда неизвестно (или безразлично), какая именно Активность (и из какого приложения) будет использоваться.

При создании Намерения, которое в дальнейшем будет передано методу *startActivity*, необходимо назначить действие (*action*), которое нужно выполнить, и, возможно, указать *URI данных*, которые нужно обработать. Также можно передать дополнительную информацию с помощью свойства *extras* Намерения. Android сам найдет подходящую Активность (основываясь на характеристиках Намерения) и запустит ее. Пример неявного вызова телефонной «звонилки»:

```
Intent intent = new Intent(Intent.ACTION_DIAL,
            Uri.parse("tel:(495)502-99-11"));

startActivity(intent);
```

Для определения того, какой именно компонент должен быть запущен для выполнения действий, указанных в Намерениях, Android использует *Фильтры Намерений* (*Intent Filters*). Используя *Фильтры Намерений*, приложения сообщают системе, что они могут выполнять определенные *действия* (*action*) с определенными данными (*data*) при определенных условиях (*category*) по заказу других компонентов системы.

Для регистрации компонента приложения (Активности или Сервиса) в качестве потенциального обработчика Намерений, требуется добавить элемент **<intent-filter>** в качестве дочернего элемента для нужного компонента в Манифесте приложения. У элемента **<intent-filter>** могут быть указаны следующие дочерние элементы (и соответствующие атрибуты у них):

- **<action>**. Атрибут *android:name* данного элемента используется для указания названия действия, которое может обслуживаться. Каждый Фильтр Намерений должен содержать не менее одного вложенного элемента *<action>*. Если не указать*действие*, ни одно Намерение не будет «проходить» через этот Фильтр. У главной Активности приложения в Манифесте должен быть указан Фильтр Намерений с *действием* ***android.intent.action.MAIN***
- **<category>**. Сообщает системе, при каких обстоятельствах должно обслуживаться *действие* (с помощью атрибута *android:name).* Внутри *<intent-filter>* может быть указано несколько категорий. Категория ***android.intent.category.LAUNCHER*** требуется Активности, которая желает иметь «иконку» для запуска. Активности, запускаемые с помощью метода startActivity, обязаны иметь категорию ***android.intent.category.DEFAULT***
- **<data>**. Дает возможность указать тип данных, которые может обрабатывать компонент. *<intent-filter>* может содержать несколько элементов *<data>*. В этом элементе могут использоваться следующие атрибуты:
  - *android:host* : имя хоста (например, *www.specialist.ru*)
  - *android:mimetype* : обрабатываемый тип данных (например, *text/html*)
  - *android:path* : «путь» внутри URI (например, */course/android*)
  - *android:port* : порт сервера (например, *80*)
  - *android:scheme* : схема URI (например, *http*)

Пример указания Фильтра Намерений:

```
<activity android:name=".MyActivity"
    android:label="@string/app_name" >

    <intent-filter>

      <action android:name="android.intent.action.SEND" />

      <category android:name="android.intent.category.DEFAULT" />

      <data android:mimeType="text/plain" />

    </intent-filter>

</activity>
```

При запуске Активности с помощью метода *startActivity* неявное Намерение обычно подходит только одной Активности. Если для данного Намерения подходят несколько Активностей, пользователю предлагается список вариантов.

Определение того, какие Активности подходят для Намерения, называется *Intent*

Resolution. Его задача – определить наиболее подходящие Фильтры Намерений, принадлежащие компонентам установленных приложений. Для этого используются следующие проверки в указанном порядке:

- **Проверка действий**. После этого шага остаются только компоненты приложений,у которых в Фильтрах Намерений указано *действие* Намерения. В случае, если *действие* в Намерении отсутствует, совпадение происходит для всех Фильтров Намерений, у которых указано хотя бы одно *действие*.
- **Проверка категорий**. Все категории, имеющиеся у Намерения, должны присутствовать в Фильтре Намерений. Если у Намерения нет категорий, то на данном этапе ему соответствуют все Фильтры Намерений, за одним исключением, упоминавшимся выше: Активности, запускаемые с помощью метода startActivity, обязаны иметь категорию ***android.intent.category.DEFAULT***, так как Намерению, использованному в этом случае, по умолчанию присваивается данная категория, даже если разработчик не указал ничего явно. Из этого исключения, в свою очередь, есть исключение: если у Активности присутствуют действие ***android.intent.action.MAIN*** и категория ***android.intent.category.LAUNCHER***, ему не требуется иметь категорию ***android.intent.category.DEFAULT***.
- **Проверка данных**. Здесь применяются следующие правила:
  - Намерение, не содержащее ни *URI*, ни *типа данных*, проходит через Фильтр,если он тоже ничего перечисленного не содержит.
  - Намерение, которое имеет *URI*, но не содержит *тип данных* (и *тип данных* невозможно определить по *URI*), проходит через Фильтр, если *URI* Намерения совпадает с *URI* Фильтра. Это справедливо только в случае таких URI, как *mailto:* или *tel:*, которые не ссылаются на реальные данные.
  - Намерение, содержащее *тип данных*, но не содержащее *URI* подходит толькодля аналогичных Фильтров Намерений.
  - Намерение, содержащее и *тип данных*, и *URI* (или если тип данных может быть вычислен из *URI*), проходит этот этап проверки, только если его *тип данных* присутствует в Фильтре. В этом случае *URI* должен совпадать, либо(!) у Намерения указан URI вида *content:* или *file:*, а у Фильтра *URI* не указан. То есть, предполагается, что если у компонента в Фильтре указан только *тип данных*, то он поддерживает URI вида *content:* или *file:*.

В случае, если после всех проверок остается несколько приложений, пользователю предлагается выбрать приложение самому. Если подходящих приложений не найдено, в выпустившей Намерение Активности возникает Исключение.

Практическая работа №7 Лабораторная работа «Изменение элементов дизайна»

Модифицируйте проект **MetroPicker** следующим образом:

1. Добавьте главное меню в Активность, отображающую список станций метро. В меню должен быть один пункт: «вернуться». Меню создайте динамически в коде,без использования строковых ресурсов.
2. Динамически создайте контекстное меню для Представления *TextView,* отображающего выбранную станцию метро главной Активности. Выбор пункта меню должен сбрасывать выбранную станцию.
3. Для главной Активности создайте основное меню из двух пунктов: «сбросить» и «выйти». Реализуйте нужные функции при выборе этих пунктов.
4. Повторите реализацию п.п. 1, 2 и 3 с помощью ресурсов, описывающих меню.

Практическая работа №8Обработка событий: подсказки»

1. Создайте новый Android проект **ListViewSample**.
2. В каталоге **res/values** создайте файл **arrays.xml** со следующим содержимым:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string-array name="stations">
        <item>Авиамоторная</item>
        <item>Автозаводская</item>
        <item>Академическая</item>
        <item>Александровский сад</item>
        <item>Алексеевская</item>
        <item>Алтуфьево</item>
        <item>Аннино</item>
        <item>Арбатская (Арбатско-Покровская линия)</item>
        <item>Арбатская (Филевская линия</item>
        <item>Аэропорт</item>
        <item>Бабушкинская</item>
        <item>Багратионовская</item>
        <item>Баррикадная</item>
        <item>Бауманская</item>
```

<item>Беговая</item>

<item>Белорусская</item>

<item>Беляево</item>

<item>Бибирево</item>

<item>Библиотека имени Ленина</item>

<item>Битцевский парк</item>

<item>Борисовская</item>

<item>Боровицкая</item>

<item>Ботанический сад</item>

<item>Братиславская</item>

<item>Бульвар адмирала Ушакова</item>

<item>Бульвар Дмитрия Донского</item>

<item>Бунинская аллея</item>

<item>Варшавская</item>

<item>ВДНХ</item>

<item>Владыкино</item>

<item>Водный стадион</item>

<item>Войковская</item>

<item>Волгоградский проспект</item>

<item>Волжская</item>

<item>Волоколамская</item>

<item>Воробьевы горы</item>

<item>Выставочная</item>

<item>Выхино</item>

<item>Деловой центр</item>

<item>Динамо</item>

<item>Дмитровская</item>

<item>Добрынинская</item>

<item>Домодедовская</item>

<item>Достоевская</item>

<item>Дубровка</item>

<item>Жулебино</item>

<item>Зябликово</item>

<item>Измайловская</item>

<item>Калужская</item>

<item>Кантемировская</item>

<item>Каховская</item>

<item>Каширская</item>

<item>Киевская</item>

<item>Китай-город</item>

<item>Кожуховская</item>

<item>Коломенская</item>

<item>Комсомольская</item>

<item>Коньково</item>

<item>Красногвардейская</item>

<item>Краснопресненская</item>

<item>Красносельская</item>

<item>Красные ворота</item>

<item>Крестьянская застава</item>

<item>Кропоткинская</item>

<item>Крылатское</item>

<item>Кузнецкий мост</item>

<item>Кузьминки</item>

<item>Кунцевская</item>

<item>Курская</item>

<item>Кутузовская</item>

<item>Ленинский проспект</item>

<item>Лубянка</item>

<item>Люблино</item>

<item>Марксистская</item>

<item>Марьина роща</item>

<item>Марьино</item>

<item>Маяковская</item>

<item>Медведково</item>

<item>Международная</item>

<item>Менделеевская</item>

<item>Митино</item>

<item>Молодежная</item>

<item>Мякинино</item>

<item>Нагатинская</item>

<item>Нагорная</item>

<item>Нахимовский проспект</item>

<item>Новогиреево</item>

```
<item>Новокузнецкая</item>
<item>Новослободская</item>
<item>Новоясеневская</item>
<item>Новые Черемушки</item>
<item>Октябрьская</item>
<item>Октябрьское поле</item>
<item>Орехово</item>
<item>Отрадное</item>
<item>Охотныйряд</item>
<item>Павелецкая</item>
<item>Парк культуры</item>
<item>Парк Победы</item>
<item>Партизанская</item>
<item>Первомайская</item>
<item>Перово</item>
<item>Петровско-Разумовская</item>
<item>Печатники</item>
<item>Пионерская</item>
<item>Планерная</item>
<item>Площадь Ильича</item>
<item>Площадь Революции</item>
<item>Полежаевская</item>
<item>Полянка</item>
<item>Пражская</item>
<item>Преображенская площадь</item>
<item>Пролетарская</item>
<item>Проспект Вернадского</item>
<item>Проспект Мира</item>
<item>Профсоюзная</item>
```

```
<item>Пушкинская</item>
<item>Речной вокзал</item>
<item>Рижская</item>
<item>Римская</item>
<item>Рязанский проспект</item>
<item>Савеловская</item>
<item>Свиблово</item>
<item>Севастопольская</item>
<item>Семеновская</item>
<item>Серпуховская</item>
<item>Славянский бульвар</item>
<item>Смоленская (Арбатско-Покровская линия)</item>
<item>Смоленская (Филевская линия)</item>
<item>Сокол</item>
<item>Сокольники</item>
<item>Спортивная</item>
<item>Сретенский бульвар</item>
<item>Строгино</item>
<item>Студенческая</item>
<item>Сухаревская</item>
<item>Сходненская</item>
<item>Таганская</item>
<item>Тверская</item>
<item>Театральная</item>
<item>Текстильщики</item>
<item>Теплый Стан</item>
<item>Тимирязевская</item>
<item>Третьяковская</item>
<item>Трубная</item>
<item>Тульская</item>
<item>Тургеневская</item>
```

```
        <item>Тушинская</item>

        <item>Улица 1905года</item>

        <item>Улица Академика Янгеля</item>

        <item>Улица Горчакова</item>

        <item>Улица Подбельского</item>

        <item>Улица Скобелевская</item>

        <item>Улица Старокачаловская</item>

        <item>Университет</item>

        <item>Филевский парк</item>

        <item>Фили</item>

        <item>Фрунзенская</item>

        <item>Царицыно</item>

        <item>Цветной бульвар</item>

        <item>Черкизовская</item>

        <item>Чертановская</item>

    </string-array>

</resources>
```

3. В каталоге res/layout создайте файл list_item.xml со следующим содержимым:

```
<?xml version="1.0" encoding="utf-8"?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"

    android:padding="10dp"
    android:textSize="16sp" >

</TextView>
```

4. Модифицируйте метод onCreate вашей Активности:

```
@Override

public void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);

      Resources r = getResources();

      String[] stationsArray = r.getStringArray(R.array.stations);
```

```
        ArrayAdapter<String> aa = new ArrayAdapter<String>(this,
                    R.layout.list_item, stationsArray);


        setListAdapter(aa); ListView

        lv = getListView();


    }
```

5. Измените базовый класс Активности с **Activity** на **ListActivity**.
6. Запустите приложение.
7. Для реакции на клики по элементам списка требуется добавить обработчик такогособытия, с помощью метода *setOnItemClickListener*. В качестве обработчика будет использоваться анонимный объект класса *OnItemClickListener*. Добавьте следующий код в *нужное место*:

```
        lv.setOnItemClickListener(new OnItemClickListener() {

              public void onItemClick(AdapterView<?> parent, View v,

                          int position, long id) {


                    CharSequence text = ((TextView) v).getText();

                    int duration = Toast.LENGTH_LONG;

                    Context context = getApplicationContext();
                    Toast.makeText(context, text, duration).show();

              }

        });
```

8. Запустите приложение и «покликайте» по станциям метро.

«Использование управляющих элементовв пользовательском интерфейсе»

Цель лабораторной работы – научиться использовать в интерфейсе пользователя различные управляющие элементы: кнопки с изображениями, радиокнопки, чекбоксы и пр.

Подготовка

1. Создайте новый проект **ControlsSample**.
2. Отредактируйте файл **res/layout/main.xml** так, чтобы остался только корневой элемент **LinearLayout**. В него в дальнейшем будут добавляться необходимые дочерние элементы:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"

    android:orientation="vertical" >


</LinearLayout>
```

Использование графической кнопки

Для использования изображения вместо текста на кнопке потребуются три изображения для трех состояний кнопки: обычное, выбранное («в фокусе») и нажатое. Все эти три изображения с соответствующими состояниями описываются в одном XML файле, который используется для создания такой кнопки.

1. Скопируйте нужные изображения кнопки в каталог **res/drawable-mdpi**, для обновления списка содержимого каталога в **Eclipse** можно использовать кнопку **F5**.
2. В этом же каталоге создайте файл smile_button.xml, описывающий, какие изображения в каких состояниях кнопки нужно использовать:

```
<?xml version="1.0" encoding="utf-8"?>

<selector xmlns:android="http://schemas.android.com/apk/res/android">


    <item android:drawable="@drawable/smile_pressed"
      android:state_pressed="true"/>

    <item android:drawable="@drawable/smile_focused"
      android:state_focused="true"/>

    <item android:drawable="@drawable/smile_normal" />


</selector>
```

3. Добавьте элемент **Button** внутри **LinearLayout** в файле разметки **res/layout/main.xml**:

```
    <Button

        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/smile_button"
        android:onClick="onButtonClicked"
        android:padding="10dp" />
```

4. Обратите внимание на атрибут `android:onClick="onButtonClicked"`, указывающий, какой метод из Активности будет использоваться как обработчик нажатия на данную кнопку. Добавьте этот метод в Активность:

```
public void onButtonClicked(View v) {
```

```
        Toast.makeText(this, "Кнопка нажата", Toast.LENGTH_SHORT).show();
    }
```

5. Запустите приложение и посмотрите, как изменяется изображение кнопки в разныхсостояниях, а также как функционирует обработчик нажатия на кнопку.

Использование виджета CheckBox

1. Добавьте элемент **CheckBox** внутри **LinearLayout** в файле разметки **res/layout/main.xml**:

```
<CheckBox

    android:id="@+id/checkbox"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onCheckboxClicked"
    android:text="Выбери меня" />
```

2. Атрибут `android:onClick="onCheckboxClicked"` определяет, какой метод из Активности будет использоваться как обработчик нажатия на виджет. Добавьте этот метод в Активность:

```
public void onCheckboxClicked(View v) {
    if (((CheckBox) v).isChecked()) {

        Toast.makeText(this, "Отмечено", Toast.LENGTH_SHORT).show();

    } else {

        Toast.makeText(this, "Не отмечено", Toast.LENGTH_SHORT).show();

    }

}
```

3. Запустите приложение и посмотрите на поведение чекбокса в разных ситуациях.

Использование виджета ToggleButton

Данный виджет хорошо подходит в качестве альтернативы радиокнопкам и чекбоксам, когда требуется переключаться между двумя взаимоисключающими состояниями, например, *Включено/Выключено*.

1. Добавьте элемент **ToggleButton** внутри **LinearLayout** в файле разметки **res/layout/main.xml**:

```
<ToggleButton android:id="@+id/togglebutton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="Звонок включен"
    android:textOff="Звонок выключен"
    android:onClick="onToggleClicked"/>
```

2. Атрибут `android:onClick="onToggleClicked"` определяет, какой метод из Активности будет использоваться как обработчик нажатия на виджет. Добавьте этот метод в Активность:

```
public void onToggleClicked(View v) {
    if (((ToggleButton) v).isChecked()) {
        Toast.makeText(this, "Включено", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(this, "Выключено", Toast.LENGTH_SHORT).show();
    }
}
```

3. Запустите приложение и проверьте его функционирование.

Использование виджета RadioButton

Радиокнопки используются для выбора между различными взаимоисключающими вариантами. Для создания *группы радиокнопок* используется элемент *RadioGroup*, внутри которого располагаются элементы *RadioButton*.

1. Добавьте следующие элементы разметки внутри **LinearLayout** в файле **res/layout/main.xml**:

```
<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >

    <RadioButton
        android:id="@+id/radio_dog"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="Собачка" />

    <RadioButton
        android:id="@+id/radio_cat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="Кошечка" />

    <RadioButton
        android:id="@+id/radio_rabbit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:onClick="onRadioButtonClicked"
        android:text="Кролик" />

</RadioGroup>
```

2. Добавьте метод **onRadioButtonClicked** в Активность:

```
public void onRadioButtonClicked(View v) {
      RadioButton rb = (RadioButton) v;

      Toast.makeText(this, "Выбрано животное: " + rb.getText(),
                  Toast.LENGTH_SHORT).show();

}
```

3. Проверьте работу приложения.

Использование виджета EditText

Виджет *EditText* используется для ввода текста пользователем. Установленный для этого виджета обработчик нажатий на кнопки будет показывать введенный текст с помощью *Toast*.

1. Добавьте элемент **EditText** внутри **LinearLayout** в файле разметки **res/layout/main.xml**:

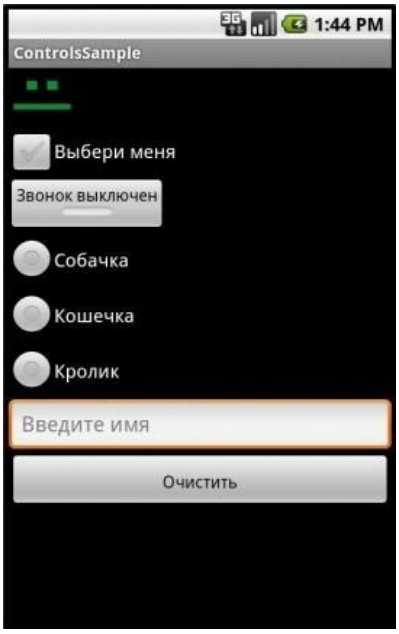
```
<EditText

    android:id="@+id/user_name"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="Введите имя"/>
```

2. Для обработки введенного пользователем текста добавьте следующий код в конце метода **onCreate**. Обратите внимание, этот обработчик, в отличие от предыдущих, использованных нами, *возвращает* значение **true** или **false**. Семантика этих значений традиционна: *true* означает, что событие (*event*) обработано и больше никаких действий не требуется, *false* означает, что событие *не обработано этим обработчиком* и будет передано следующим обработчикам в цепочке. В нашем случае реагирование происходит только на нажатие (*ACTION_DOWN*) кнопки Enter (*KEYCODE_ENTER*):

```
final EditText userName = (EditText) findViewById(R.id.user_name);

userName.setOnKeyListener(new View.OnKeyListener() {

      @Override

      public boolean onKey(View v, int keyCode, KeyEvent event) {

            if ((event.getAction() == KeyEvent.ACTION_DOWN)

                        && (keyCode == KeyEvent.KEYCODE_ENTER)) {
```

```
                    Toast.makeText(getApplicationContext(),
                          userName.getText(),
                          Toast.LENGTH_SHORT).show();


                    return true;

                }
                return false;

            }

        });
```

3. Запустите приложение и проверьте его работу.
4. Добавьте кнопку **«Очистить»** в разметку и напишите обработчик, очищающий текстовое поле (используйте метод *setText* виджета *EditText*):



5. Проверьте работу приложения.

Практическая работа №9 Обработка событий: цветовая индикация»

Задание 1. Разработать мобильное приложение, состоящее из четырех activity.

После запуска приложения пользователь должен попадать на экран с activity1. На данном экране должно быть представлено меню, состоящее из четырех кнопок. Высота кнопок должна составлять 20% от высоты экрана. Расстояние между кнопками – 2%. Первая и последняя кнопка должны быть на равном расстоянии от краев экрана. Ширина кнопок 75%, выравнивание посередине.

После нажатия на первую кнопку пользователь должен переходить к activity2, его

внешний вид представлен на рисунке 1. Верстка должна осуществляться с использованием LinearLayout, ширина кнопок должна задаваться в процентах от ширины экрана.

Рис. 2.5. Внешний вид экрана для первого задания

После нажатия на вторую кнопку в activity1 пользователь должен переходить к activity3, его внешний вид представлен на рисунке 2. Верстка должна осуществляться с использованием RelativeLayout (не использовать LinearLayout).

Рис. 2.6. Результат первого этапа выполнения задания

Третья кнопка в activity1 должна создавать activity3. Внешний вид activity3 представлен на рис. 2.7.

Рис.2.3. Интерфейс приложения на этапе activity3.

Кнопка должна быть выровнена по центру экрана. Цвет обводки кнопки #505050. Толщина обводки в соответствии с месяцем вашего рождения (от 1 до 12). Радиус скругления 24dp. Цвет фона экрана #FFFFFF. При нажатии на кнопку ее цвет должен изменяться на светло-зеленый.

Нажатие на четвертую кнопку в activity1 должно приводить к закрытию приложения.

Практическая работа №10 Подготовка стандартных модулей

Цель работы:

Изучить работу с потоками. Научиться работать с мультимедиа файлами. Изучить работу с классом AsyncTask

Теоретические сведения

Главный и обычный потоки

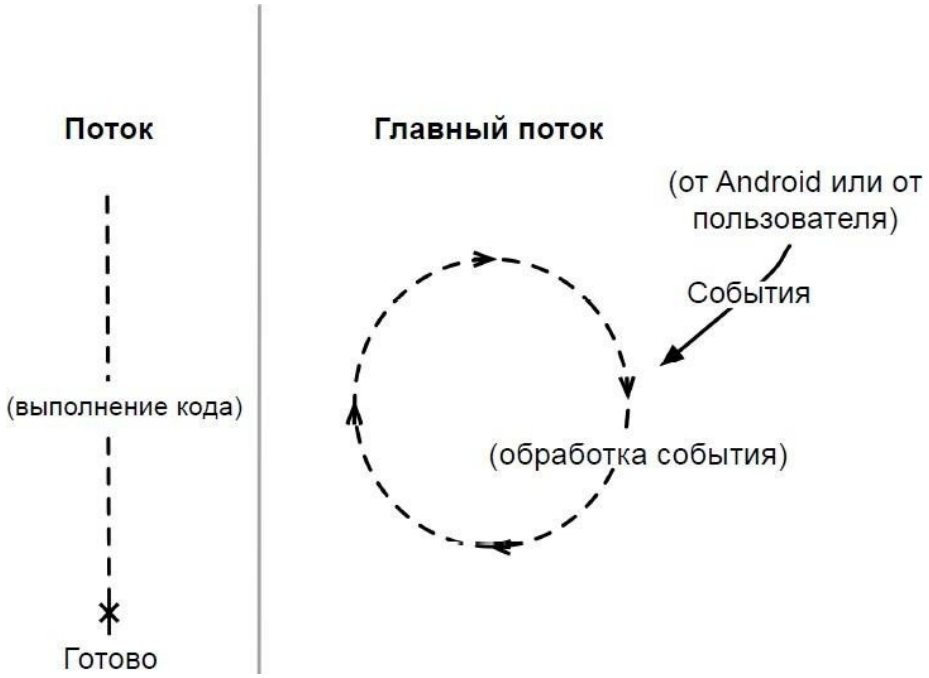


Рис.3.1. Графическое представление потоков

Асинхронные потоки в Android

AsyncTask-это специальный абстрактный класс, предоставляющий набор методов для реализации:

- onPreExecute-для размещения инициализирующего кода (UI поток)
- doInBackground-для размещения тяжелого кода, который будет выполняться в другом потоке
- onProgressUpdate-для информирования о прогрессе (UI поток)
- onPostExecute-для обработки результата, возвращенного doInBackground(UI поток) и вспомогательных методов
- isCancelled-для получения информации об отмене задачи
- publishProgress-для перевода сообщения о прогрессе в UI поток с последующим вызовом onProgressUpdate

Последовательность выполнения методов AsyncTask

onPreExecute
doInBackground
onPostExecute

publishProgress

onProgressUpdate

Правила использования AsyncTask:

Объект AsyncTaskдолжен быть создан в UI-потоке

- Метод execute должен быть вызван в UI-потоке
- Метод executeможет быть запущен только один раз
- Не вызывайте методы onPreExecute, doInBackground, onPostExecuteи onProgressUpdate

Передача данных в AsyncTask

Объявляем класс

```
Class MyAsyncTaskextends AsyncTask<String, Integer, Long>{
…
}
```

- Первый параметр используется методом doInBackground protectedLong doInBackground(String... urls)
- Второй параметр используется методом onProgressUpdate protectedvoidonProgressUpdate(Integer… progress)
- Третий параметр используется методом onPostExecute protectedvoidonPostExecute(Longresult)

Промежуточные данные

Последовательность действий для передачи промежуточных данных в основной поток программы:

- В методе doInBackgroundвызываем метод publishProgress
- В методе onProgressUpdate обрабатываем переданный в publishProgress параметр и выводим прогресс

Метод get

- Возвращает результат выполнения метода doInBackground
- Вызывается из UI потока

```
MyAsyncTaskat = newMyAsyncTask();
…
result= at.get();
```

Задание на лабораторную работу

Задание 1. Рассмотрите пример передачи данных. MyAsyncTaskat = newMyAsyncTask();
at.execute(“url1”, “url2”);

doInBackground(String... urls)

Задание 2. Рассмотрите пример вывода промежуточных данных.

```
@Override
Protected void doInBackground(String... urls) { try{
                int cnt= 0;
                for(Stringurl: urls) {
                        // обрабатываем первый параметр
                                …
                        // выводим промежуточные результаты
```

```
                    cnt++;
                    publishProgress(cnt);
                }
                TimeUnit.SECONDS.sleep(1);
            } catch(InterruptedExceptione) {
                e.printStackTrace();
                }
                return null;
}
@Override
                protected void onProgressUpdate(Integer... values) {
                    super.onProgressUpdate(values); tv.setText("обработано " +
                    values[0] + " параметров");
}
```

Задание 3. Проверьте пример создания простой асинхронной задачи

```
public class MainActivity extends Activity{ MyAsyncTask at;
TextView tv;

        public void onCreate(Bundle savedInstance State) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            tv= (TextView) findViewById(R.id.tv);
            MyAsyncTask at = new MyAsyncTask();
            at.execute();
        }
        }
```

Задание 4. На основании изученных примеров разработать приложение сохраняющее статистику проигрываемых песен на радио Мегабайт. Для сохранения песни и названия необходимо создать базу данных, содержащую таблицу со следующими полями:

1) ID
2) Исполнитель
3) Название трека
4) Время внесения записи

При включении приложения необходимо производить проверку подключения к Интернету. В случае если подключение отсутствует – выводить всплывающее сообщение (Toast) с предупреждением о запуске в автономном режиме (доступен только просмотр внесенных ранее записей).

После включения приложение должно производить асинхронный опрос сервера с интервалом 20 секунд. Если название трека не совпадает с последней записью в таблице необходимо произвести запись в БД.

URL адрес, по которому можно получить информацию о текущем треке и исполнителе:

http://media.ifmo.ru/api_get_current_song.php

Формат возвращаемых данных – JSON.

В случае успешного выполнения запроса результат будет иметь вид:

{“result”: “success”, “info” : “*Исполнитель – Название трека*” }

В случае ошибки API вернет следующую строку:

{“result”: “error”, “info” : “*Информация об ошибке*” }

Для успешного взаимодействия с API при обращении к странице необходимо передавать логин (login) и пароль(password) как POST- параметры.

login**: 4707login**

password**: 4707pass**

Приложение должно содержать активити, позволяющее просматривать внесенные в базу данных записи.

Практическая работа №11 Обработка событий: переключение между экранами

Цель работы

Изучить инструменты хранения данных, а также работу с внешними файлами.

Теоретические сведения

Варианты взаимодействия с сервером

1. Сокеты

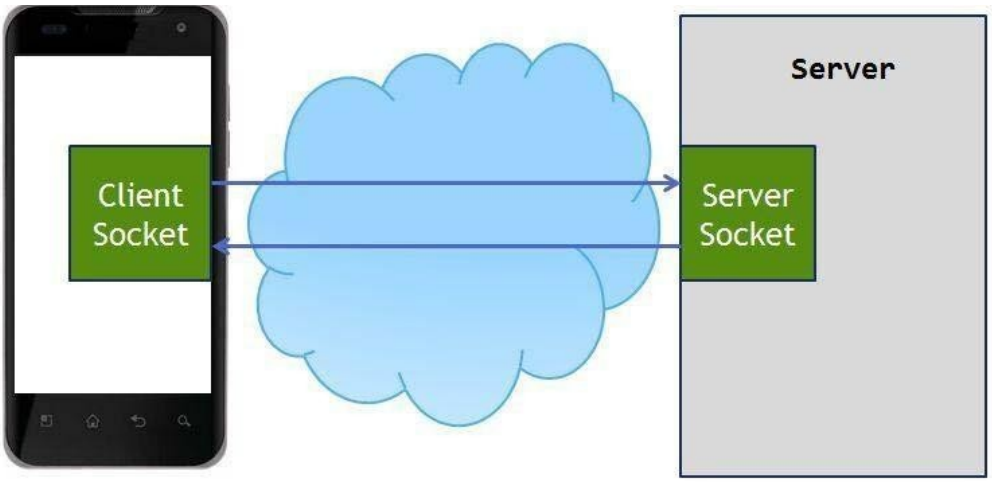


Рис. 5.1. Схема взаимодействия клиентского сокета с серверным Используется в приложениях, где важна скорость доставки сообщения,

важен порядок доставки сообщений и необходимо держать стабильное соединение с сервером. Такой способ зачастую реализуется в мессенджерах и играх.

2. Частые опросы

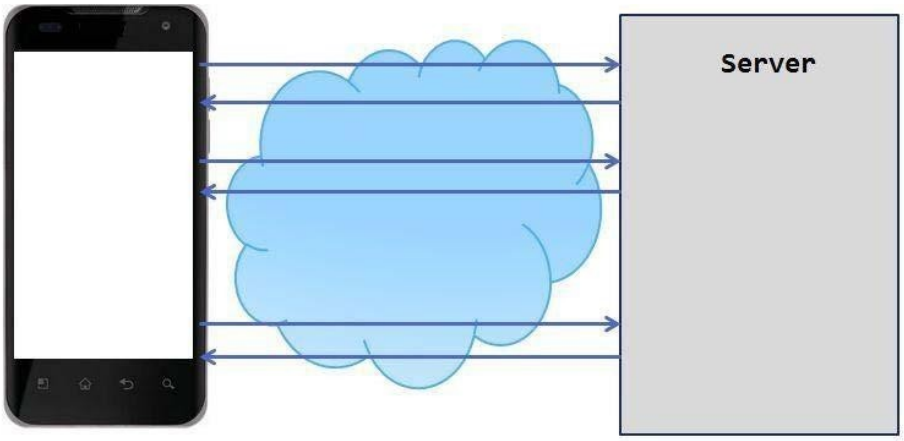


Рис. 5.2. Схема реализации частых опросов

Клиент посылает запрос на сервер и запрашивает новые данные. Сервер отвечает на запрос клиента и отдает все, что у него накопилось к этому моменту.

3. Длинные опросы

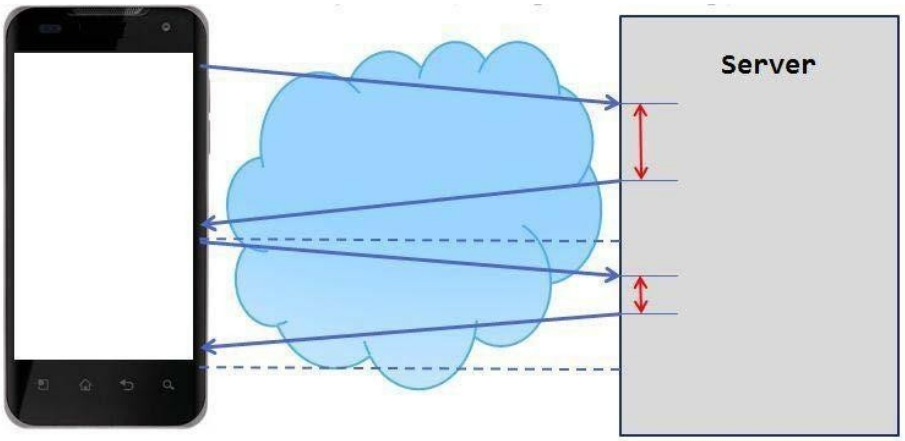


Рис. 5.3. Схема реализации длинных опросов

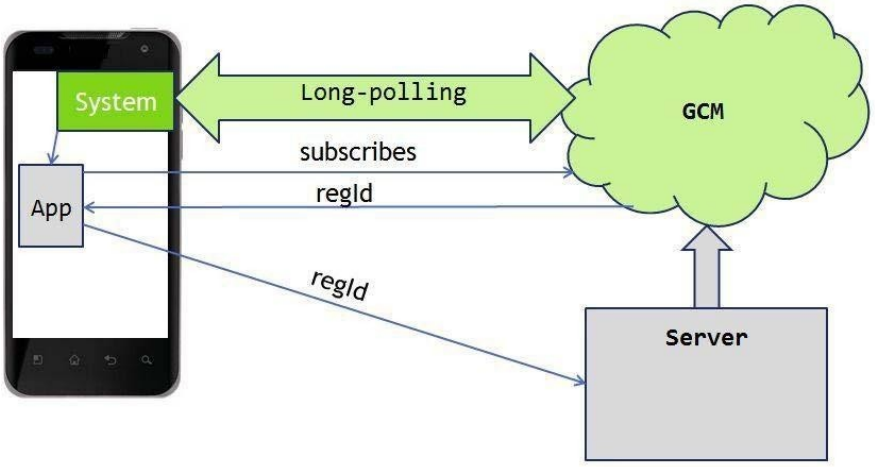


Рис. 5.4. Механизм реализации длинных опросов Метод длинных опросов

- Реализация этого подхода достаточно сложна на мобильном клиенте в первую очередь из-за нестабильности мобильного соединения.
- При данном подходе расходуется меньше трафика, чем при обычном polling’е, т.к. сокращается количество установок соединений с сервером.
- Механизм long polling, или пуш-уведомлений (push notifications), реализован в самой платформе Android. (ваше приложение подписывается у сервиса Google Cloud Messaging (GCM) на получение пуш-уведомлений)

Библиотеки для работы с сетью

java.net	Содержит классы, связанные с сетевыми функциями, в том числе сокеты потоков и датаграмм, протокол IP, а также общие средства для работы с HTTP. Это многоцелевой ресурс для работы с сетями.

java.io	Пакет не относится непосредственно к сетям. Его классы используются сокетами и соединениями, содержащимися в других пакетах Java. Они используются также для обмена с локальными файлами (что часто происходит при взаимодействии с сетью).

java.nio	Содержит классы, которые служат буфером для определенных типов данных. Удобен для организации сетевой связи между двумя конечными точками средствами Java.
org.apache.*	Набор пакетов, которые обеспечивают точный контроль и функции для HTTP-коммуникаций на основе Apache - популярного веб-сервера с открытым исходным кодом.
android.net	Содержит дополнительные сокеты доступа к сети в дополнение к основным классам java.net.*. Этот пакет включает в себя класс URI, который часто используется в разработке приложений Android, не связанных с сетью.
android.net.http	Содержит классы для работы с сертификатами SSL.
android.net.wifi	Содержит классы для реализации всех аспектов WiFi (802.11 Wireless Ethernet) на платформе Android.

Разрешения

- android.permission.INTERNET
- android.permission.ACCESS_NETWORK_STATE

Задание на лабораторную работу

Задание 1. Изучите пример подключения к сети.

```
public void myClickHandler(View view) {
...
ConnectivityManager connMgr = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
 NetworkInfo networkInfo = connMgr.getActiveNetworkInfo(); if
(networkInfo != null && networkInfo.isConnected()) {
        // fetch data

} else {
        // display error
        }
        ...
}
```

Задание 2. Изучите код приложений

HttpURLConnection

```
private String downloadUrl(String myurl) throws IOException {
        InputStream is = null;
        // Only display the first 500 characters of the retrieved
        // web page content. int
        len = 500;
        try {
                URL url = new URL(myurl);
                HttpURLConnection conn = (HttpURLConnection)
        url.openConnection();
```

```
                conn.setReadTimeout(10000 /* milliseconds */);
                conn.setConnectTimeout(15000 /* milliseconds */);
                conn.setRequestMethod("GET"); conn.setDoInput(true);
                // Starts the query conn.connect();
                int response = conn.getResponseCode();
                Log.d(DEBUG_TAG, "The response is: " + response); is =
                conn.getInputStream();
                // Convert the InputStream into a string String
                contentAsString = readIt(is, len); return
                contentAsString;
        // Makes sure that the InputStream is closed after the app is
        // finished using it.
        } finally {
                if (is != null) {
                is.close();
                }
        }
}
```

Преобразование полученной информации к типу Srting

```
public String readIt (InputStream stream, int len) throws IOException,
UnsupportedEncodingException {
        Reader reader = null;
        reader = new InputStreamReader(stream, "UTF-8"); char[]
        buffer = new char[len];
        reader.read(buffer); return
        new String(buffer);
        }
```

Http GET запрос

- Создаем HttpClient

```
HttpClient client = new DefaultHttpClient();
```

- Создаем объект HttpGet

```
HttpGet request = new HttpGet("http://www.example.com");
```

- Выполняем HTTP запрос

```
HttpResponse response;
try {
        response = client.execute(request);
        Log.d("Response of GET request", response.toString());
        } catch (ClientProtocolException e) {
                // TODO Auto-generated catch block
        e.printStackTrace();
        } catch (IOException e) {
```

```
            // TODO Auto-generated catch block
        e.printStackTrace();
        }
```

Взаимодействие с сервером через сокеты

```
public class Requester extends Thread {
        Socket requestSocket;
        String message;
        StringBuilder returnStringBuffer = new StringBuilder();
        Message lmsg;
        int ch;
        @Override public void run() { try {
            this.requestSocket = new Socket("remote.servername.com",
        13);
            InputStreamReader isr = new
        InputStreamReader(this.requestSocket. getInputStream(), "ISO-8859- 1");
            while ((this.ch = isr.read()) != -1) {
            this.returnStringBuffer.append((char) this.ch);
                }
            this.message = this.returnStringBuffer.toString();
            this.lmsg = new Message();
            this.lmsg.obj = this.message;
            this.lmsg.what = 0;
            h.sendMessage(this.lmsg);
            this.requestSocket.close();
        }
        catch (Exception ee) {
            Log.d("sample application", "failed to read data" +
        ee.getMessage());
            }
        }
}
```

Задание 3. Работа с внешними файлами.

Разработать мобильное приложение, позволяющее пользователю асинхронно скачивать файлы журнала Научно-технический вестник. Файлы хранятся на сервере в формате PDF и расположены по адресу:

http://ntv.ifmo.ru/file/journal/идентификатор_журнала.pdf

Не для всех ID имеются журналы, поэтому необходимо предусмотреть сообщение об отсутствии файла. В случае если файл не найден, ответ от сервера будет содержать главную страницу сайта.

Определить существует ли файл можно по возвращаемому сервером заголовку (параметр content-type).

Примеры ссылок:

http://ntv.ifmo.ru/file/journal/1.pdf – возвращен PDF файл

http://ntv.ifmo.ru/file/journal/2.pdf – файл не найден, возвращена главная страница сайта

Файлы должны храниться на устройстве в папке, создаваемой при первом запуске приложения (путь до папки и ее название определите самостоятельно).

После окончания загрузки файла должна становиться доступной кнопка «Смотреть» и кнопка «Удалить».

При нажатии на кнопку «Смотреть» должно происходить открытие сохраненного на устройстве файла. Предусмотреть ошибку, если на устройстве не установлено приложение, открывающее PDF файлы.

При нажатии на кнопку «Удалить» загруженный файл должен удаляться с устройства.

Рис.5.5. Интерфейс приложения

Задание 4. Хранение и чтение настроек.

При запуске приложения пользователю должно выводиться всплывающее полупрозрачное уведомление (popupWindow), с краткой инструкцией по использованию приложения (можете написать случайный текст), чекбоксом «Больше не показывать» и кнопкой «ОК».

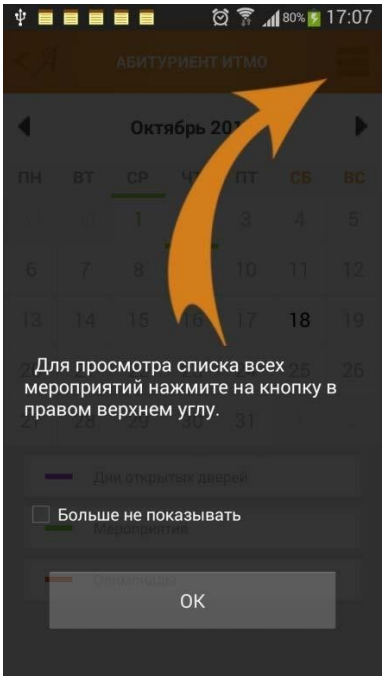


Рис.5.6. Результат работы приложения

Если чекбокс был отмечен и нажата кнопка ОК, необходимо произвести сохранение данного параметра используя SharedPreferences. При следующем запуске приложения производить проверку параметра, и не выводить всплывающее сообщение, если чекбокс был отмечен.

Практическая работа №12 Передача данных между модулями

Цель работы

Изучить инструменты хранения данных, а также работу с внешними файлами.

Теоретические сведения

Варианты хранения данных на устройстве

- Preferences (аналог INI-файлам в ОС Windows)
- SQLite - база данных, таблицы
- обычные файлы - внутренние и внешние (на SD карте)

Preferences

Значения сохраняются в виде пары: имя, значение

- Для хранения используется файл формата XML
- Путь к сохраняемым настройкам /data/data/YOUR_PACKAGE_NAME/shared_prefs/YOUR_PREFS_NAM E.xml

Отличие методов getPreferences и getSharedPreferences

```
public SharedPreferences getPreferences(int mode) {
        return getSharedPreferences(getLocalClassName(), mode);
}
```

Метод getSharedPreferences

public abstract SharedPreferences getSharedPreferences (String name, int mode)

- Name – имя файла с настройками

- Mode – режим
    - MODE_PRIVATE – режим по умолчанию. Файл доступен из только из текущего приложения.
    - MODE_WORLD_READABLE (deprecated in API level 17) – разрешено чтение всеми приложениями.
    - MODE_WORLD_WRITEABLE - (deprecated in API level 17) – разрешена запись всеми приложениями.
    - MODE_MULTI_PROCESS – режим для взаимодействия с файлом из нескольких процессов

Создание экрана настроек

pref.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">
        <CheckBoxPreference
                android:key="notif"
                android:summary="Enable notifications"
                android:title="Notifications">
        </CheckBoxPreference>
```

```
                <EditTextPreference
                        android:key="address" android:summary="Address
                        for notifications" android:title="Address">
</EditTextPreference>
    </PreferenceScreen>
```

Создание экрана настроек

PrefActivity.java

```
public class PrefActivity extends PreferenceActivity { @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                addPreferencesFromResource(R.xml.pref);
                }
}
```

Создание экрана настроек

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
                 xmlns:android=http://schemas.android.com/apk/res/android
                android:layout_width="fill_parent" android:layout_height="fill_parent"
                android:orientation="vertical">
                <TextView
                        android:id="@+id/tvInfo"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="">
                </TextView>
</LinearLayout>
```

Создание экрана настроек

MainActivity.java

```
        public class MainActivity extends Activity {
                TextView tvInfo;
                SharedPreferences sp;
                /** Called when the activity is first created. */ public void
                onCreate(Bundle savedInstanceState) {
                        super.onCreate(savedInstanceState);
                        setContentView(R.layout.main);
                        tvInfo = (TextView) findViewById(R.id.tvInfo);
                        sp = PreferenceManager.getDefaultSharedPreferences(this);
```

```
                // полная очистка настроек
                // sp.edit().clear().commit();
}
protected void onResume() {
                Boolean notif = sp.getBoolean("notif", false); String
                address = sp.getString("address", "");
        String text = "Notifications are " + ((notif) ? "enabled, address = " + address : "disabled");
                tvInfo.setText(text);
                super.onResume();
                }
                public boolean onCreateOptionsMenu(Menu menu) { MenuItem
                        mi = menu.add(0, 1, 0, "Preferences");
                mi.setIntent(new Intent(this, PrefActivity.class)); return
                        super.onCreateOptionsMenu(menu);
                }
}
```

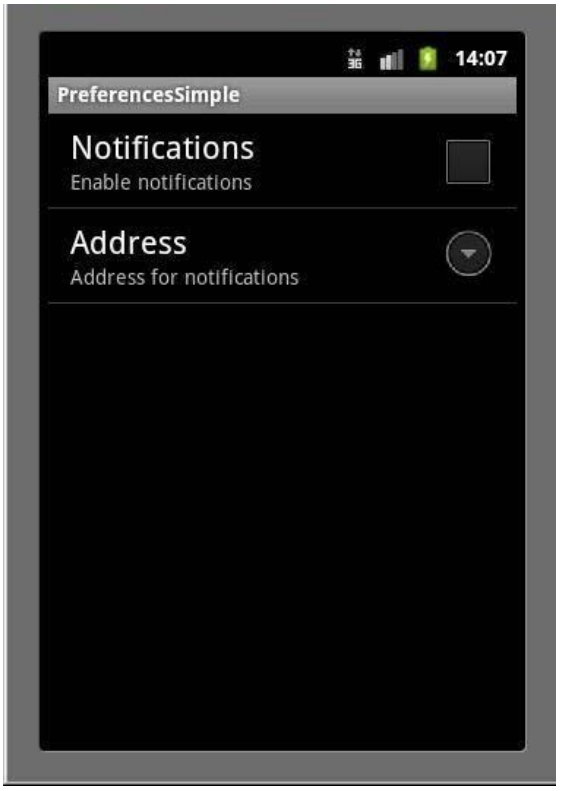


Рис.6.1. Интерфейс приложения

Работа с файлами

Основные используемые классы

- BufferedReader, BufferedWriter
- InputStreamReader, OutputStreamWriter
- openFileInput, openFileOutput\
- FileReader, FileWriter

- File

Пример

```
public class MainActivity extends Activity { final
        String LOG_TAG = "myLogs"; final String
        FILENAME = "file";
        final String DIR_SD = "MyFiles";
        final String FILENAME_SD = "fileSD";
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
        }
        public void onclick(View v) {
                switch (v.getId()) { case
                R.id.btnWrite:
                        writeFile();
                        break;
                case R.id.btnRead:
                        readFile();
                        break;
                case R.id.btnWriteSD:
                        writeFileSD();
                        break;
                case R.id.btnReadSD:
                        readFileSD();
                        break;
        }
}
```

Запись файла в память устройства

```
void writeFile() { try {
// отрываем поток для записи
        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(
                openFileOutput(FILENAME, MODE_PRIVATE)));
                // пишем данные
                bw.write("Содержимое файла");
                // закрываем поток
                bw.close();
                Log.d(LOG_TAG, "Файл записан");
                } catch (FileNotFoundException e) {
                e.printStackTrace(); }
```

```
                catch (IOException e) {
                e.printStackTrace();
                }
}
```

Чтение файла из памяти устройства

```
void readFile() { try {
        // открываем поток для чтения BufferedReader br = new
        BufferedReader(new InputStreamReader(
        openFileInput(FILENAME)));
        String str = "";
        // читаем содержимое
        while ((str = br.readLine()) != null) {
                Log.d(LOG_TAG, str);
        }
        } catch (FileNotFoundException e) {
                e.printStackTrace();
        }
        catch (IOException e) {
                        e.printStackTrace();
                }
        }
```

Запись на SD карту

```
void writeFileSD() {
                // проверяем доступность SD
                if (!Environment.getExternalStorageState().equals(
                Environment.MEDIA_MOUNTED)) {
        Log.d(LOG_TAG, "SD-карта не доступна: " +
Environment.getExternalStorageState());
                return;
                }
// получаем путь к SD
File sdPath = Environment.getExternalStorageDirectory();
// добавляем свой каталог к пути
sdPath = new File(sdPath.getAbsolutePath() + "/" + DIR_SD);
// создаем каталог sdPath.mkdirs();
// формируем объект File, который содержит путь к файлу
File sdFile = new File(sdPath, FILENAME_SD); try {
                // открываем поток для записи
                BufferedWriter bw = new BufferedWriter(new FileWriter(sdFile));
```

```
                    // пишем данные bw.write("Содержимое
                    файла на SD");
                    // закрываем поток
                    bw.close();
                    Log.d(LOG_TAG, "Файл записан на SD: " +  sdFile.getAbsolutePath());
}
                    catch (IOException e) {
                    e.printStackTrace();
}
}
```

Чтение с SD карты

```
void readFileSD() {
        // проверяем доступность SD
        if (!Environment.getExternalStorageState().equals(
        Environment.MEDIA_MOUNTED)) {
        Log.d(LOG_TAG, "SD-карта не доступна: " +
    Environment.getExternalStorageState());
        return;
        }
        // получаем путь к SD
        File sdPath = Environment.getExternalStorageDirectory();
        // добавляем свой каталог к пути
        sdPath = new File(sdPath.getAbsolutePath() + "/" + DIR_SD);
        // формируем объект File, который содержит путь к файлу
        File sdFile = new File(sdPath, FILENAME_SD); try {
        // открываем поток для чтения
        BufferedReader br = new BufferedReader(new FileReader(sdFile)); String str
        = "";
        // читаем содержимое
        while ((str = br.readLine()) != null) {
        Log.d(LOG_TAG, str);
        }
        } catch (FileNotFoundException e) {
        e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
        }
```

Разрешение на работу с файлами на SD карте

- android.permission.READ_EXTERNAL_STORAGE
- android.permission.WRITE_EXTERNAL_STORAGE

Задание на лабораторную работу

Задание 1. Разработать мобильное приложение, позволяющее устанавливать напоминания.

Требования к приложению

1) Создание напоминаний и сохранение их в базу данных. В БД должны храниться следующие значения (Заголовок, Текст уведомления, дата уведомления)
2) Просмотр установленных уведомлений
3) Удаление уведомлений
4) Дата напоминания должна устанавливаться с помощью TimePickerDialog и DatePickerDialog
5) Стилизовать напоминание в Notification Center и Status bar (установить собственный лого)
6) При нажатии на уведомление в Notification Center переходить в активити приложения с полным текстом уведомления.

Классы для создания приложения: Notification, NotificationManager, PendingIntent, BroadcastReceiver, AlarmManager

Практическая работа №13 Тестирование и оптимизация мобильного приложения

пошаговый алгоритм тестирования мобильных приложений

Обеспечение качества (QA, от английского - *Quality Assurance*) является неотъемлемой частью жизненного цикла разработки любых приложений, включая мобильные. К сожалению, многие упускают из виду критические особенности тестирования мобильных приложений, которые часто приводят к сбоям, ошибкам в работе приложения и плохому качеству обслуживания клиентов.

Чтобы обеспечить успешную разработку любого приложения, специалист-тестировщик должен принимать участие во всех этапах разработки - от создания концепции и анализа требований, до создания спецификаций тестирования и выпуска готового продукта. Обеспечение качества также является ключевым элементом в последующих, после прохождения этапов разработки, обзорах программного продукта.

Однако часто бывает сложно определить, с чего начать организацию процесса тестирования мобильного приложения. Для беспроблемного тестирования мы рекомендуем просто выполнить девять указанных ниже шагов.

Давайте рассмотрим особенности тестирования мобильных приложений.

Цикл жизни спринтов

Этап 1: Планирование

Когда этап разработки приложения почти завершен, вы должны снова поставить перед собой вопрос - чего вы пытаетесь достичь разработкой данного приложения и какие у вас есть ограничения.

Вы должны определить следующее:

- Взаимодействует ли ваше приложение с другими приложениями?
- Насколько функциональны все возможности приложения?
- Является ли тестируемое мобильное приложение нативным, Mobile-web или гибридным?

- Ограничена ли задача тестирования приложения тестированием только внешнего интерфейса?
- Стоят ли задачи на тестирование бэкенда?
- Какова должна быть совместимость с различными беспроводными сетями?
- Как сильно данные приложения и свободное пространство, занимаемое им, зависят от особенностей использования приложения?
- Насколько быстро загружается ваше приложение, насколько быстро происходит серфинг по меню приложения и его функциям?
- Как будет обрабатываться возможное увеличение нагрузки на приложение?
- Влияют ли различные изменения в статусе и состоянии телефона на работу мобильного приложения?

Убедитесь, что вы договорились с командой тестировщиков о роли каждого из них и о ваших ожиданиях от процесса тестирования. В конце концов, общение является ключом к поддержанию правильной рабочей среды в команде.

Правильное понимание ролей и задач также относится и к моменту прописывания списка тест кейсов. Вся команда QA должна поддерживать и обновлять этот документ с отчетами по тестированию всех функций, реализованных на протяжении всего процесса разработки.

Этап 2. Определение необходимых типов тестирования мобильных приложений

Перед тестированием любых мобильных приложений определите, что именно в данном мобильном приложении вы хотите протестировать: набор функциональности, удобство использования, совместимость, производительность, безопасность и т. д. На этом же этапе имеет смысл выбрать методы тестирования мобильного приложения.

Определите, на какие целевые устройства направлено данное приложение, и какие требования к функционалу следует проверить.

Вы также должны определить, какие целевые устройства нужно включить в список тестирования.

Вы можете сделать это следующим образом:

- Выяснить, какие устройства будет поддерживать приложение;
- Определить, какая версия операционной системы будет самой ранней из тех, что поддерживаются приложением;
- Выявить наиболее популярные модели мобильных устройств у целевой аудитории;
- Определить набор не основных (дополнительных) устройств с экранами разных размеров, потенциально поддерживаемых приложением;
- Решить, будете ли вы использовать для тестирования физические устройства или их эмуляторы.

Этап 3: Тестовые случаи и разработка сценариев тестирования приложения

Подготовьте документ, описывающий тестовые случаи (test cases) для каждой тестируемой функции и функциональности.

В дополнение к функциональным тестовым случаям, также должны быть охвачены некоторые отдельные моменты (кейсы):

- Особенность использование батареи;
- Скорость работы приложения;
- Требования к данным;
- Объем используемой памяти.

Также перед началом тестирования важно определиться, какое сочетание ручного и автоматического тестирования вы будете применять.

При необходимости подготовьте отдельные наборы ручных тестовых случаев и сценариев для автоматического тестирования и адаптируйте их согласно требованиям проекта.

Этап 4: Ручное и автоматическое тестирование

Теперь пришло время для выполнения ручных и автоматизированных тестов. Ранее, на предыдущих этапах, вы уже определили, какие тесты и скрипты использовать и подготовили их. Теперь, на текущем этапе, вы выполняете запуск тестов для проверки механизмов основной функциональности, чтобы убедиться в отсутствии поломок.

Автоматизированное тестирование мобильных приложений хорошо экономит время и другие ресурсы тестировщиков.

Этап 5: Тестирование юзабилити и бета-тестирование

После того, как базовый функционал протестирован, настало время убедиться, что мобильное приложение является достаточно простым в использовании и обеспечивает удовлетворительный пользовательский опыт. На этом этапе необходимо поддерживать соответствие матрице кроссплатформенности, чтобы обеспечить охват пользователей различных платформ, достигнутый бета-тестерами.

Пример матрицы поддержки разных версий платформы iOs

После того, как приложение будет протестировано внутри компании, вы сможете выпустить бета-версию приложения на рынок.

Тестирование совместимости

Мобильные устройства различаются в зависимости от платформы, модели и версии их операционной системы. Важно выбрать такое подмножество устройств, которое будет соответствовать вашему приложению.

Тестирование пользовательского интерфейса

Пользовательский опыт является ключевым элементом, при тестировании приложения. Ведь наше приложение разрабатывается именно для конечных пользователей. Вам следует качественно проверить удобство использования приложения, навигацию по его элементам и контент. Тестируйте меню, опции, кнопки, закладки, историю, настройки и навигацию приложения.

Тестирование интерфейса

Тестирование пунктов меню, кнопок, закладок, истории, настроек и навигации по приложению.

Тестирование внешних факторов

Приложения для мобильных устройств не будут единственными приложениями на устройстве пользователя. Вместе с вашим приложением будут установлены приложения от сторонних разработчиков. Возможно десятки таких приложений. Следовательно, вашему приложению придётся взаимодействовать с этими сторонними приложениями и прерывать работу различных функций устройства, таких как различные типы сетевых подключений, обращение к SD-карте, телефонные звонки и другие функции устройства.

Тестирование доступности

Мобильными устройствами могут пользоваться различные люди с ограниченными возможностями. По этой причине важно протестировать возможность работы с приложением людей с дальтонизмом, нарушениями слуха, проблемами пожилого возраста и другими возможными проблемами. Такое тестирование является важной частью общего тестирования юзабилити.

Этап 6: Тестирование производительности

Мобильные устройства предоставляют для приложений меньший объем памяти и меньшую доступную мощность процессора, чем стационарные компьютеры и ноутбуки. По этой причине в работе мобильных приложений очень важна эффективность использования предоставляемых ресурсов. Вам следует проверить работоспособность тестируемого приложения, изменив соединение с 2G, 3G на WIFI, проверить скорость отклика, потребление заряда батареи, стабильность работы и т. д.

Рекомендуется проверять приложение на предмет масштабируемости применения и наличие возможных проблем с производительностью.

В рамках этого этапа важно пройти и нагрузочное тестирование мобильного приложения.

Функциональное тестирование

Функциональность приложения должна быть полностью протестирована. Особое внимание следует уделить установке, обновлениям, регистрации и входу в систему, обеспечению, работе со специфическими функциями устройства и сообщениям об ошибках.

Функциональное тестирование мобильного приложения, по большей части, может быть выполнено так же, как вы выполнили бы его для любого другого типа приложения. По этой причине мы не будем вдаваться в подробности этого типа тестирования. Однако следует указать области, которые имеют особое значение для мобильных приложений.

Имейте в виду, что функциональное тестирование должно включать в себя тестирование всех функций приложения и не должно быть излишне сосредоточено на какой-то одной функции.

В рамках функционального тестирования, вам следует выполнить следующие тесты:

- Тестирование процесса установки;
- Тестирование возможности обновлений;
- Эксплуатационное тестирование;
- Тестирование процесса регистрации и авторизации;
- Тестирование функций, специфических для устройства;
- Тестирование отправки и получения сообщений об ошибках;
- Низкоуровневое тестирование ресурсов: использование памяти, автоматическое освобождение ресурсов и т.д.
- Тестирование сервисов: функционирование как в режиме онлайн, так и в автономном режиме.

Этап 7: Аттестационное тестирование и тестирование безопасности приложения

Безопасность и конфиденциальность данных имеют огромное значение в наше время. Пользователи требуют, чтобы вся их информация хранилась безопасно и конфиденциально.

Убедитесь, что тестируемое приложение надежно защищено. Выполните проверку на возможность внедрения SQL инъекций, на возможность перехвата сеансов, анализа дампов данных, анализа пакетов и SSL трафика.

Очень важно проверить безопасность хранилища конфиденциальных данных вашего мобильного приложения и его поведение в соответствии с различными схемами разрешений для устройств.

Помимо проверки безусловного шифрования имен пользователей и паролей, задайте себе следующие вопросы:

- Есть ли у приложения сертификаты безопасности?
- Использует ли приложение безопасные сетевые протоколы?

• Существуют ли какие-либо ограничения, например количество попыток входа в систему до блокировки пользователей?

Этап 8: Тестирование устройства

Выполните тесты по тем алгоритмам, которые вы ранее прописали в тестовых случаях и сценариях тестирования на всех определенных для тестирования устройствах, в облаке и / или на физических устройствах.

Этап 9: контрольный этап и резюме

Этот этап включает в себя подробное и полное тестирование - от ранних итеративных этапов тестирования до регрессионных тестов, которые все еще могут потребоваться для стабилизации работы приложения и выявления незначительных дефектов.

На этом этапе тестирования вы можете добавить для проверки новые функции и изменить настройки на те, которых не будет в финальной версии.

После завершения тестирования приложения, дополнительные параметры и функции, добавленные для проверки на этом этапе, удаляются, и окончательная версия становится готовой для представления общественности.

Итоговый отчет о тестировании

Весь процесс тестирования мобильных приложений должен быть тщательно задокументирован. Проверьте дважды, сделаны ли нужные записи, и после этого сформируйте свой окончательный отчет о тестировании (test summary report).

Этот отчет должен включать:

• Важную информацию, выявленную в результате проведенных испытаний;
• Информацию о качестве проводимого тестирования;
• Сводную информацию о качестве тестируемого мобильного приложения;
• Статистику, полученную из отчетов об различных инцидентах;
• Информацию о видах тестирования и времени, затраченном на каждый из них. Следует также указать в отчете, что:
• данное мобильное приложение пригодно для использования в том качестве, в котором заявлено;
• соответствует всем критериям приемлемости функционала и качества работы.