ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Учебно-методическое пособие

по дисциплине

«Программирование мобильных устройств»

Автор
Кузин А.П.

Ростов-на-Дону, 2018

Аннотация

Учебно-методическое пособие предназначено для студентов очной формы обучения направления 09.03.04 «Программная инженерия».

Авторы

Старший преподаватель
каф. ПОВТиАС
Кузин А.П.

Оглавление

1. ЛАБОРАТОРНАЯ РАБОТА №1: ЗНАКОМСТВО СО СРЕДОЙ РАЗРАБОТКИ

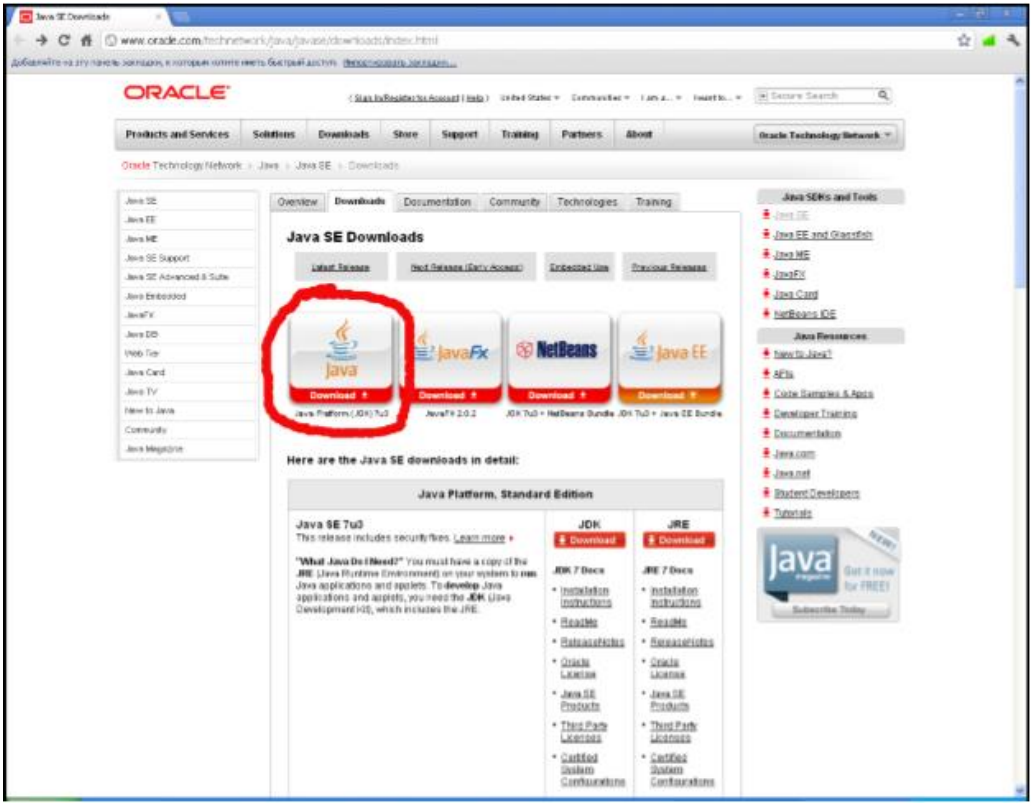
1.1. Установка и настройка компонентов среды разработки

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.

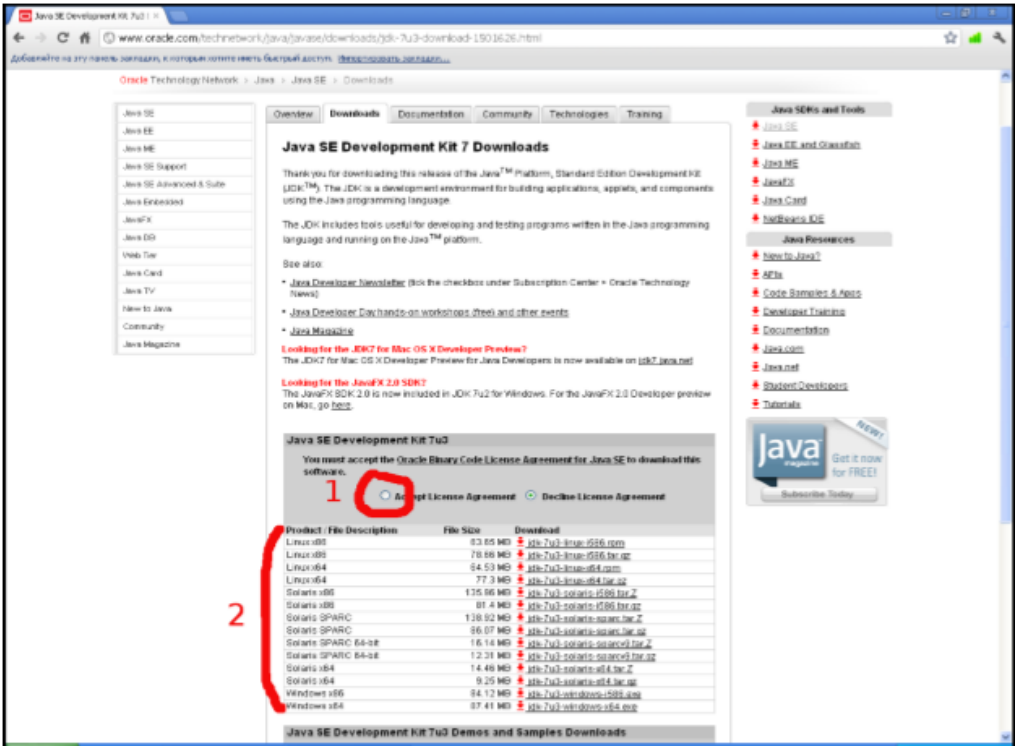
Установка JDK

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:

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

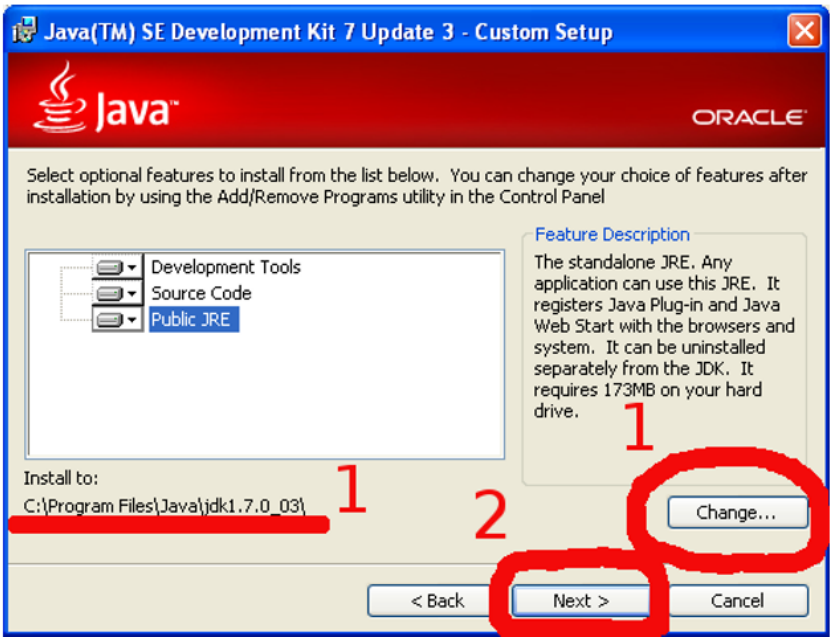


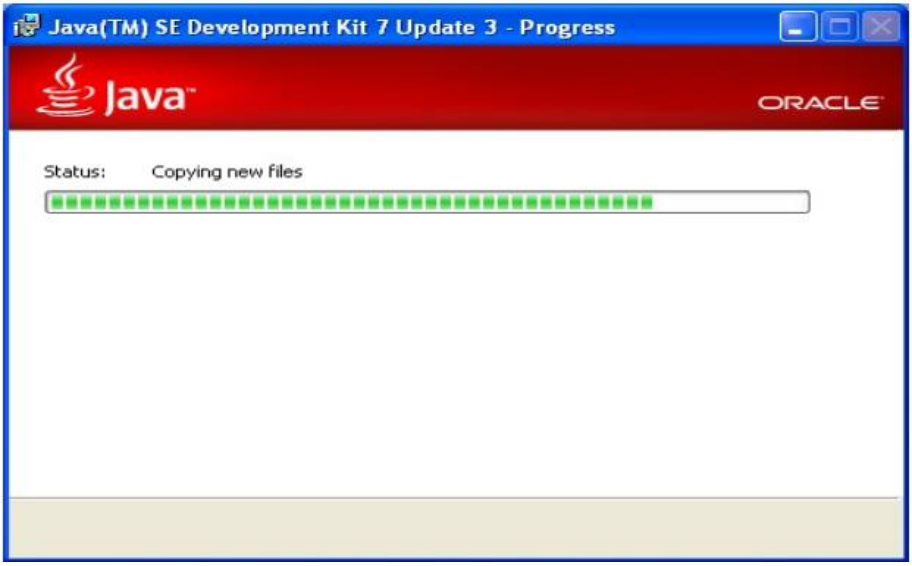
После принятия условий лицензионного соглашения Oracle Binary Code License Agreement for Java SE (1) можно выбрать подходящую для вашей платформы ссылку(2):



Установка загруженного JDK достаточно проста:

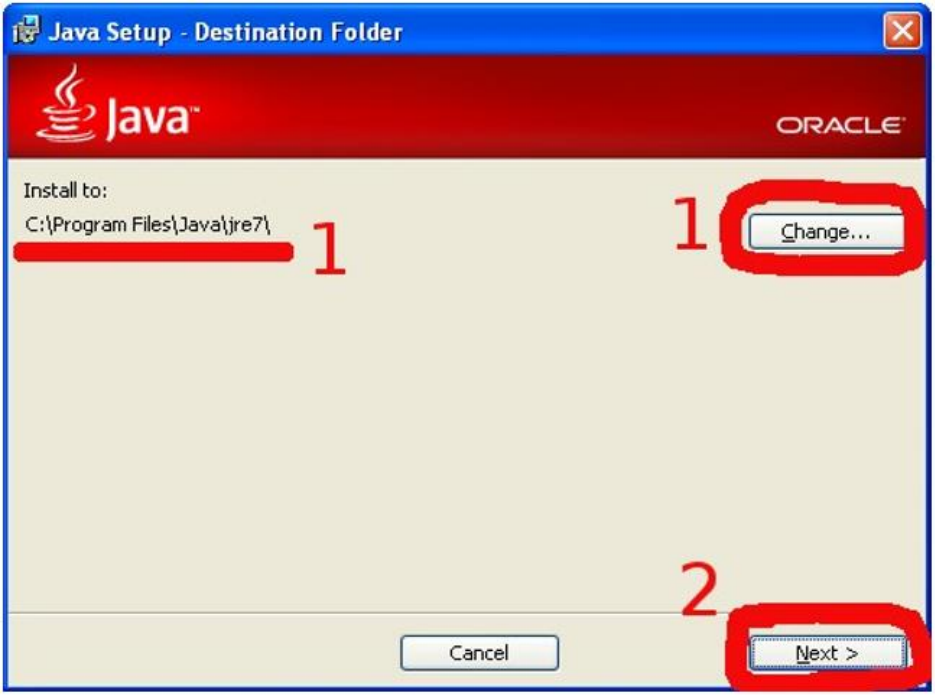
Если каталог для установки по умолчанию (1) не подходит, его можно изменить (2), а затем продолжить установку:





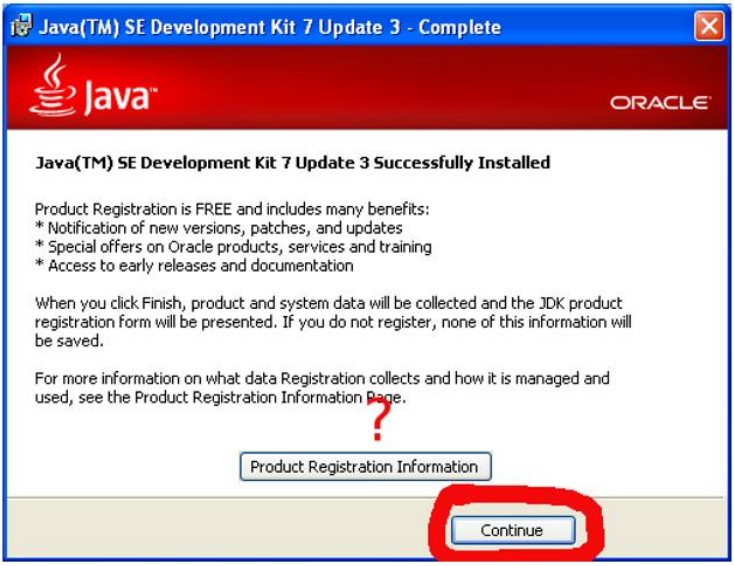
Далее все идет вполне ожидаемо:

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

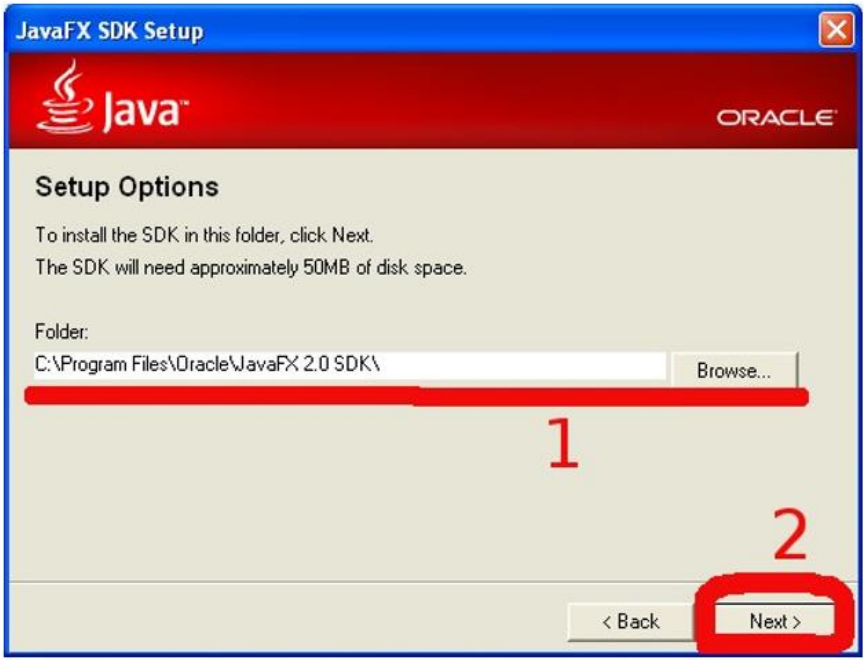


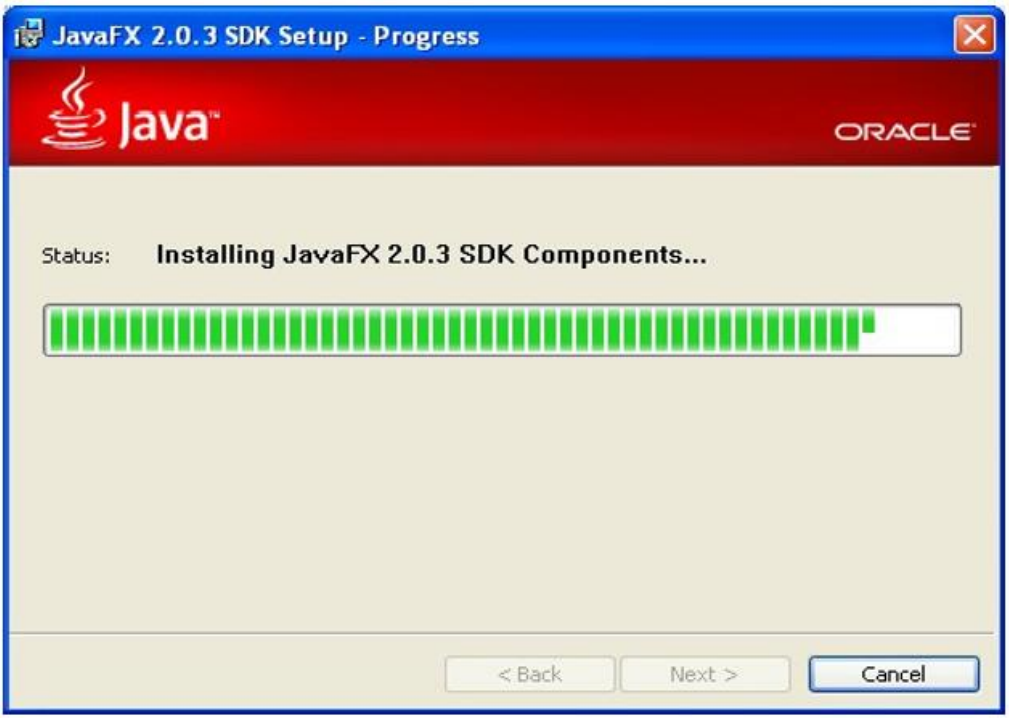
и продолжить установку:

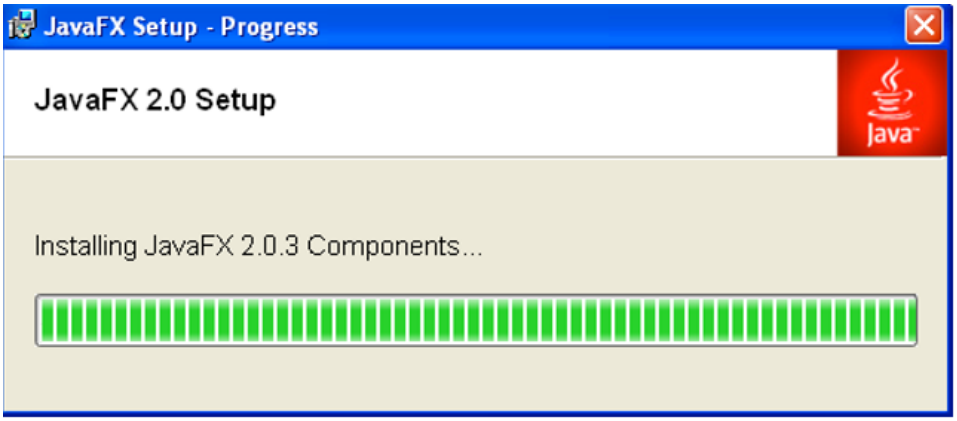
Далее при желании можно зарегистрировать установленный продукт:



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

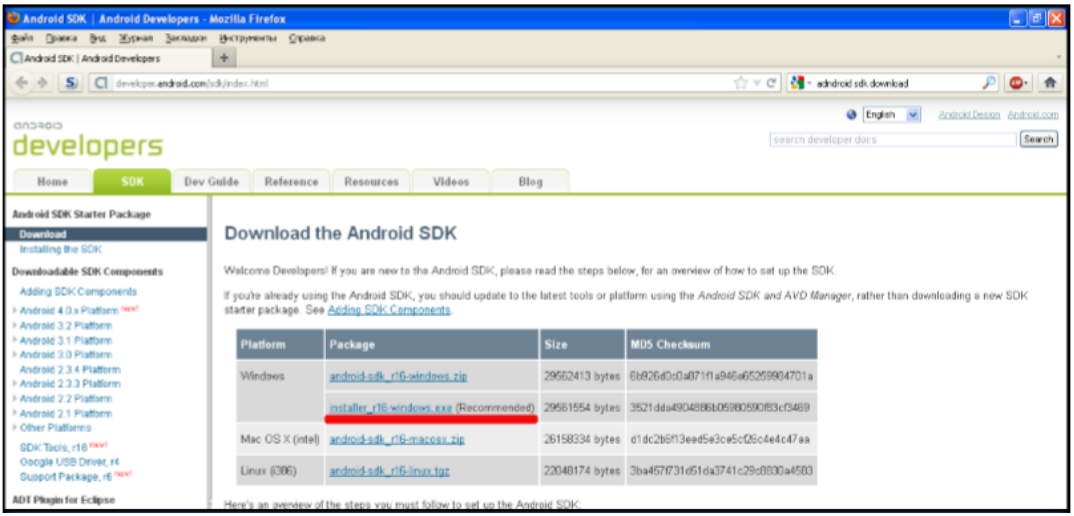


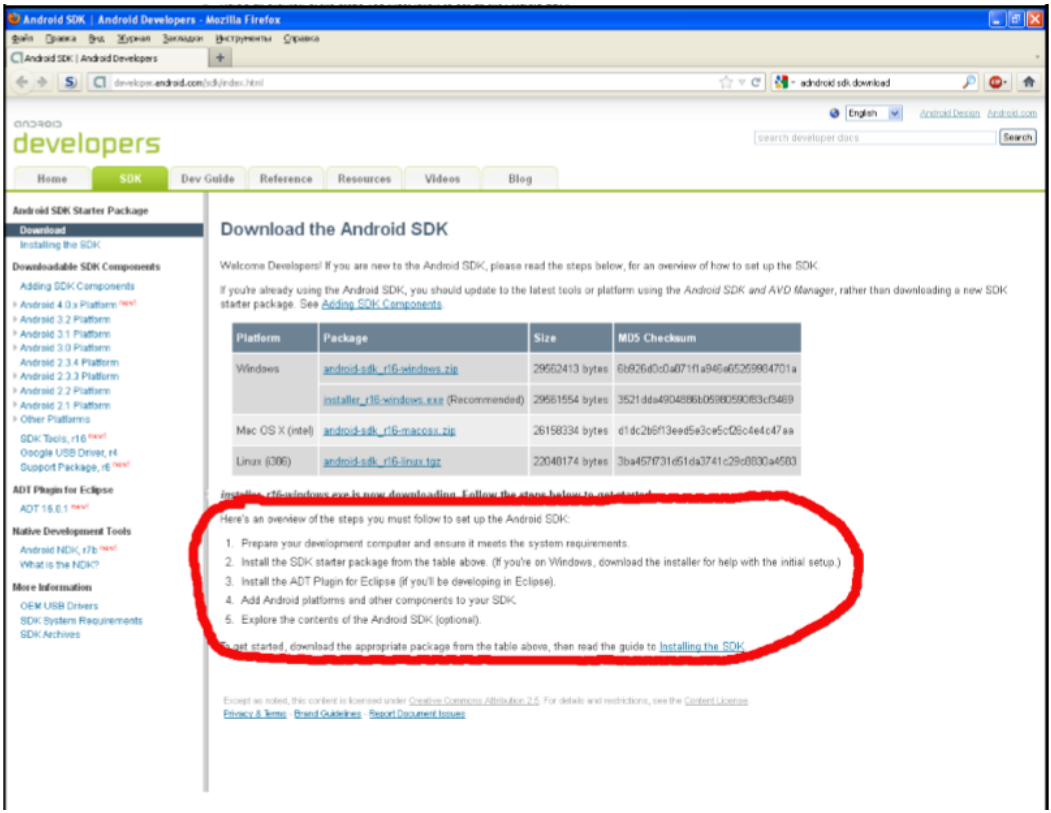




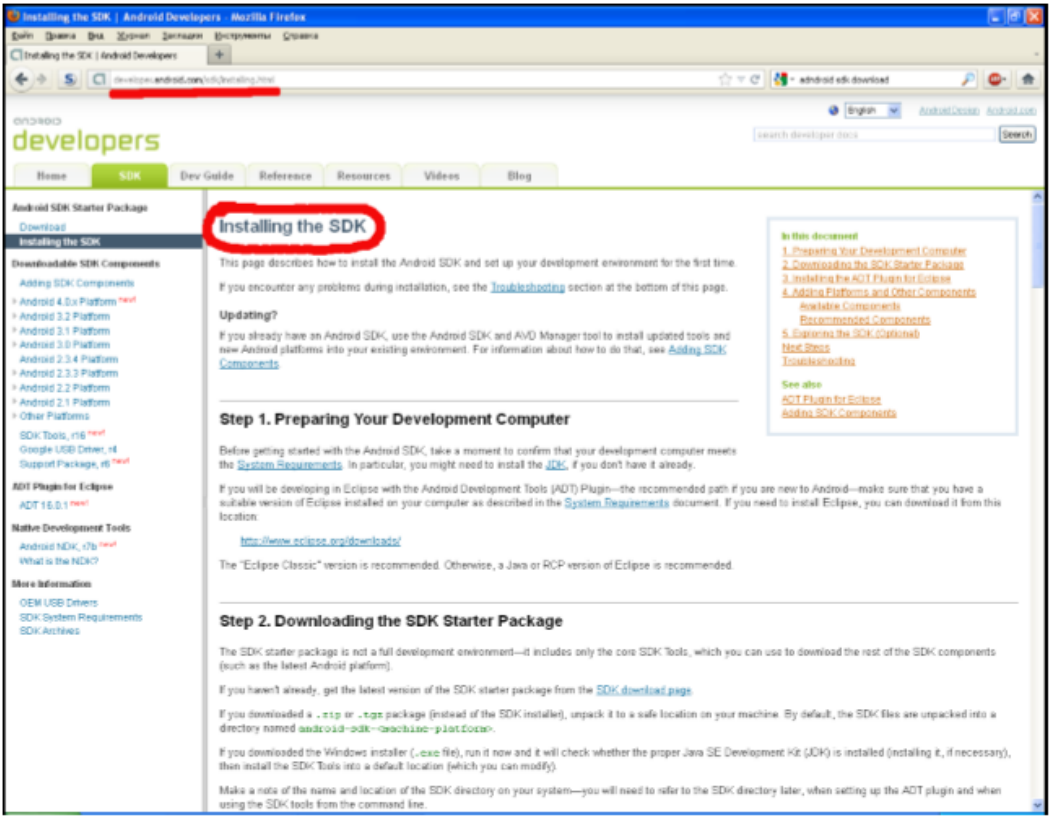
И, наконец, установка JDK закончена:

Установка Android SDK На странице http://developer.android.com/sdk выбираем installer_r16-windows.exe



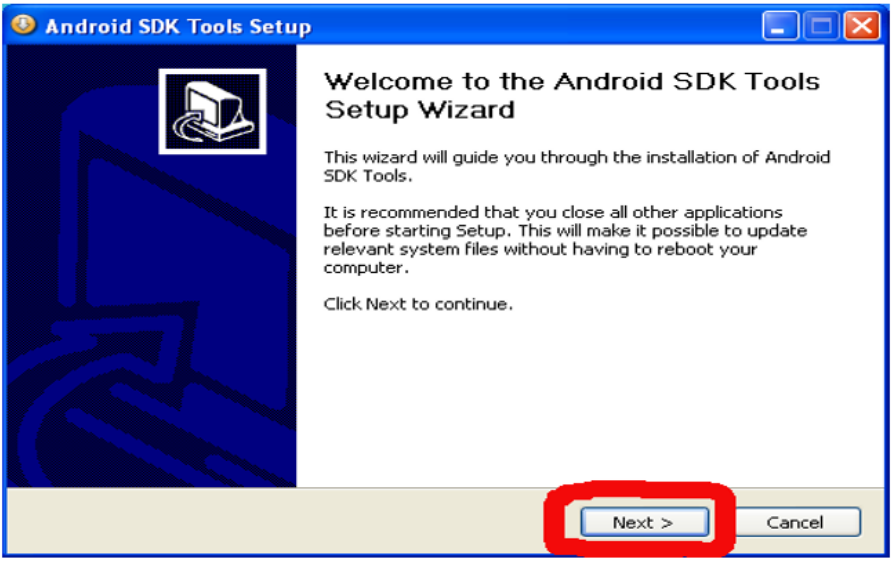


Здесь же имеются дополнительные инструкции по установке, а также ссылка на пошаговую инструкцию по установке SDK на все поддерживаемые платформы: http://developer.android.com/sdk/installing.html

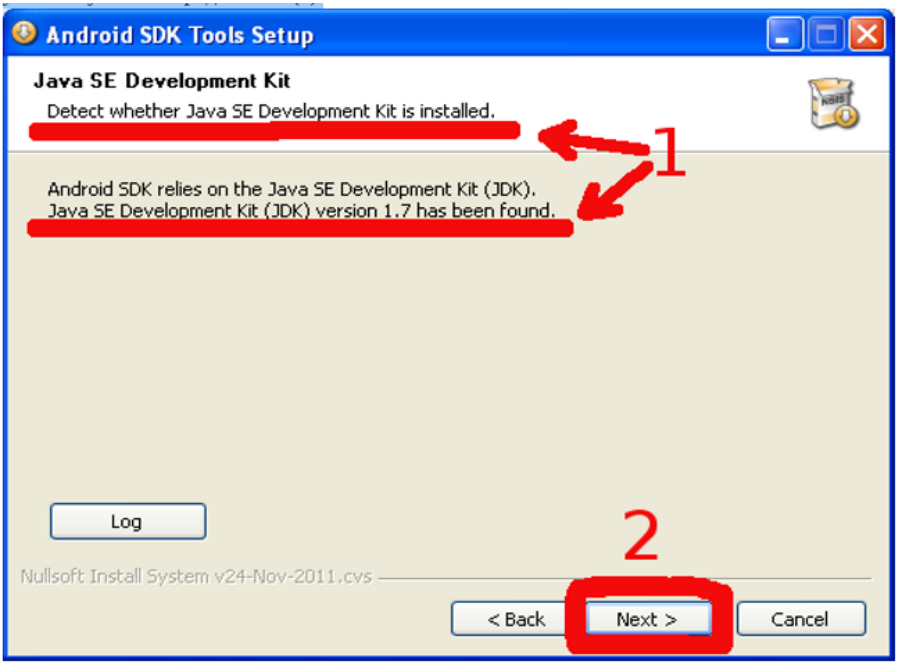


Ресурс http://developer.android.com является основным источником информации о платформе Android «из первых рук». В нем содержится огромное количество самых разнообразных сведений, необходимых разработчику, начиная с описания API и кончая такими вещами, как рекомендации по дизайну приложений и повышению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

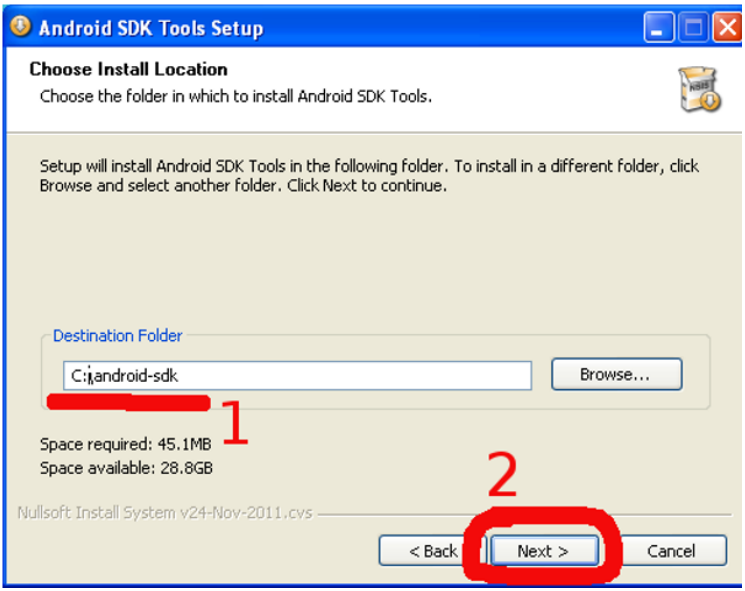
Установка загруженного Android SDK также не отличается особой сложностью:

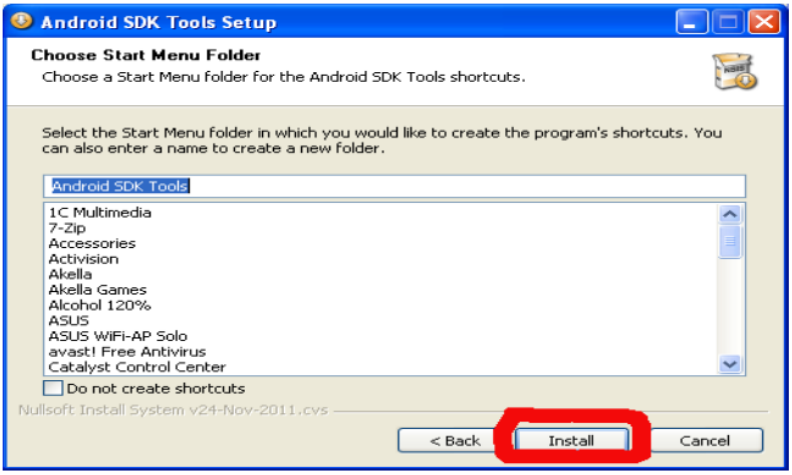


Мастер установки обнаружит (если сможет) установленную версию JDK (1), после чего установку можно продолжить (2):

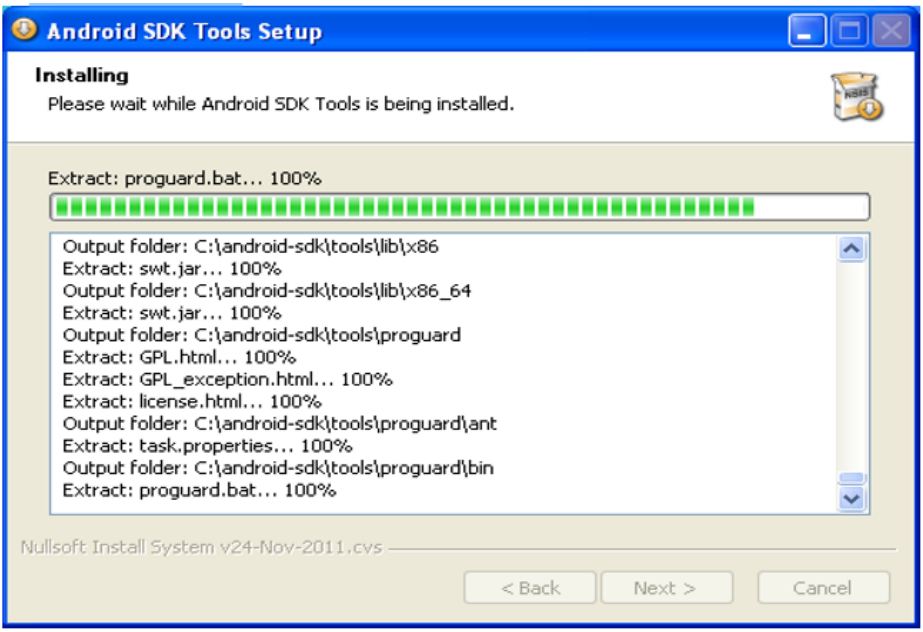


Если планируется запускать некоторые инструменты, входящие в Android SDK, из командной строки, имеет смысл выбрать каталог для установки ближе к корню файловой системы:

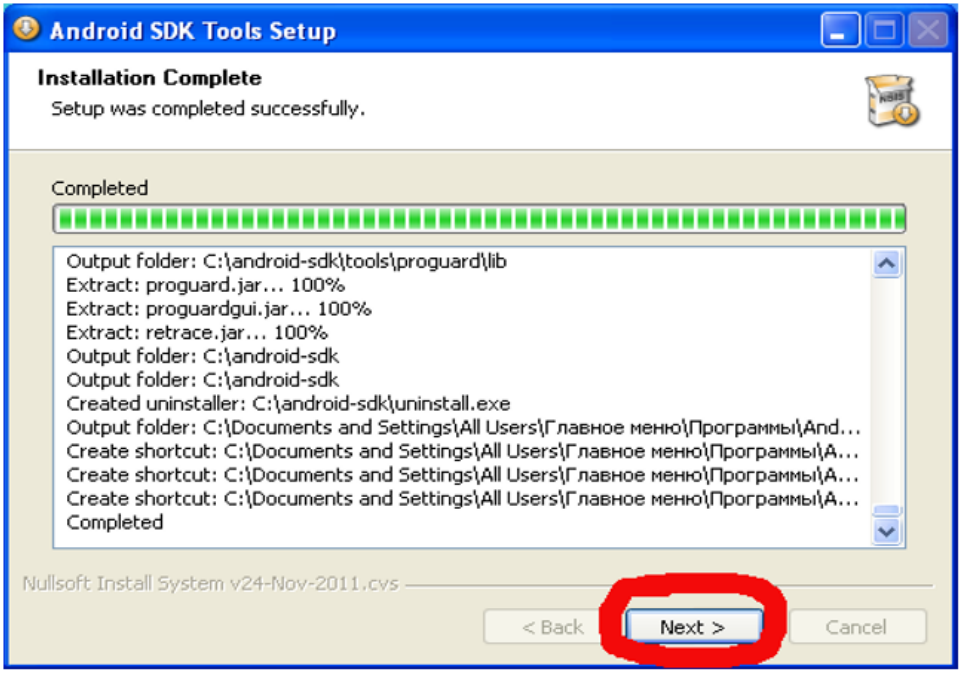


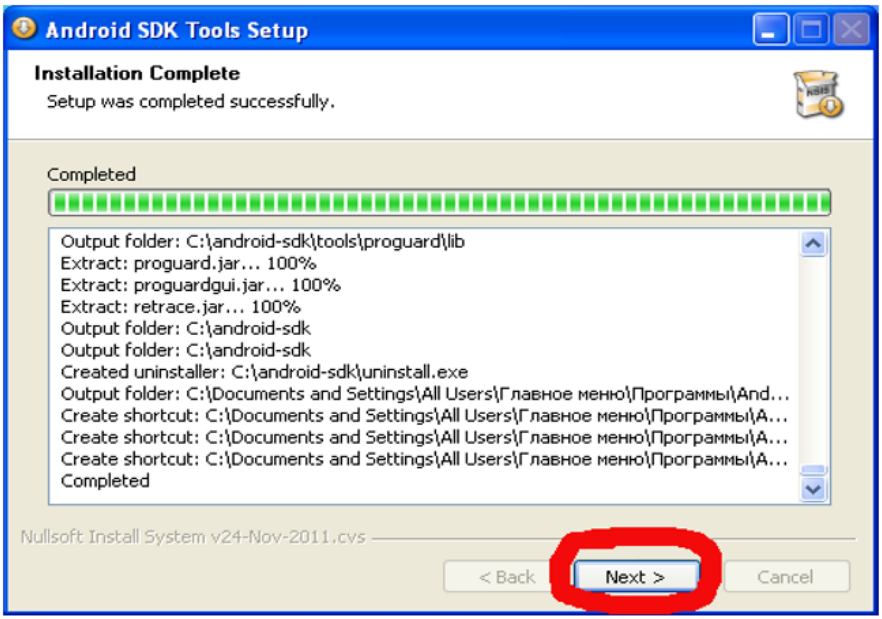


Установка начинается

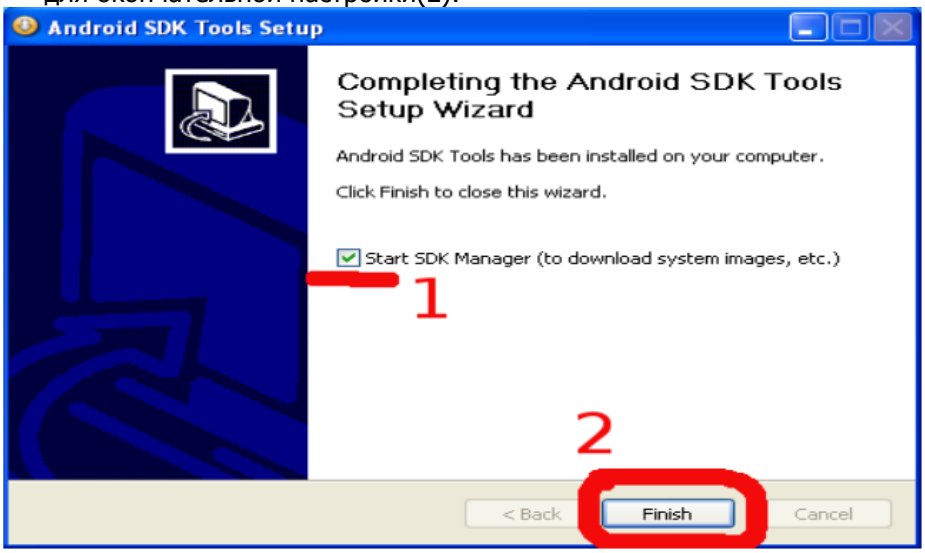


и заканчивается:

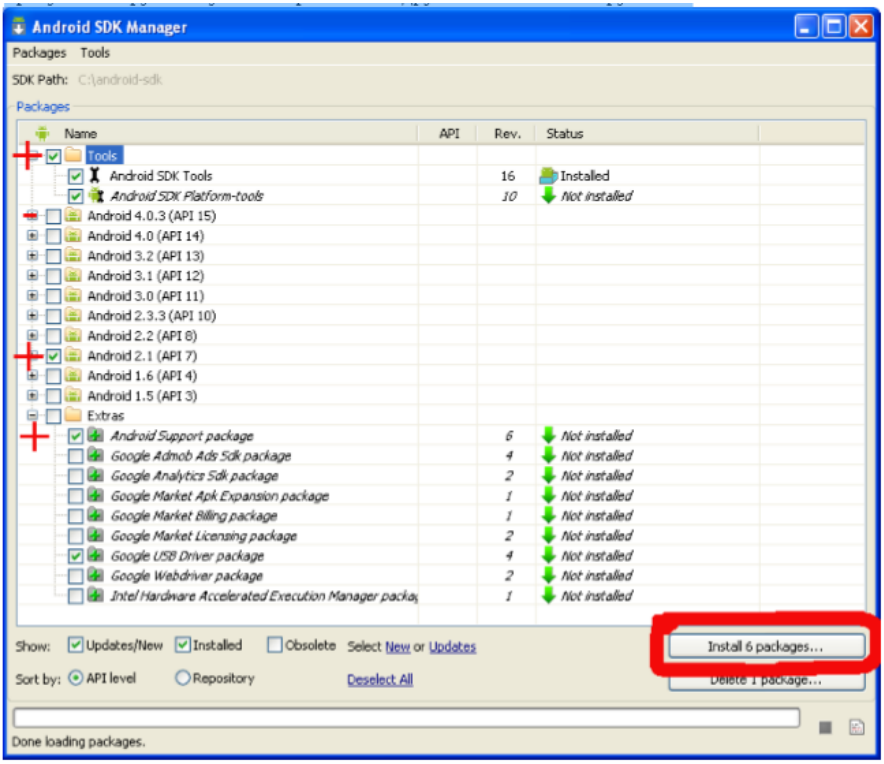


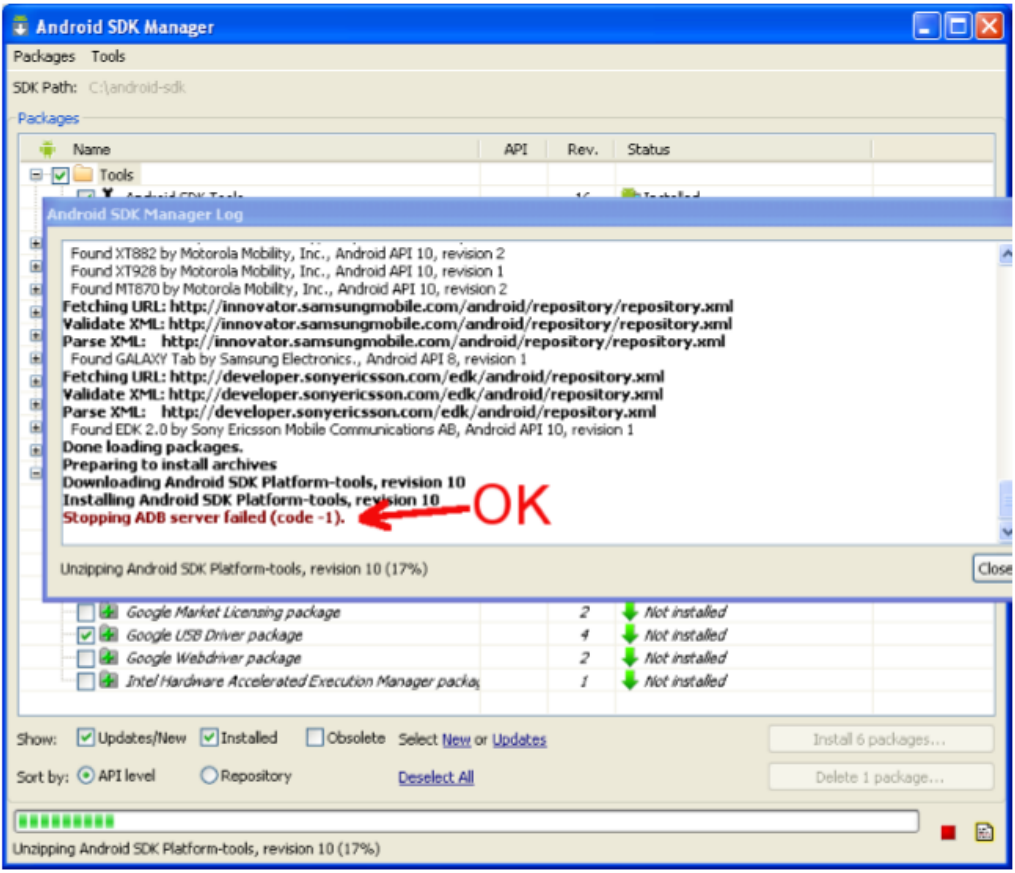


Можно сразу поставить галку(1) и запустить SDK Manager для окончательной настройки(2):

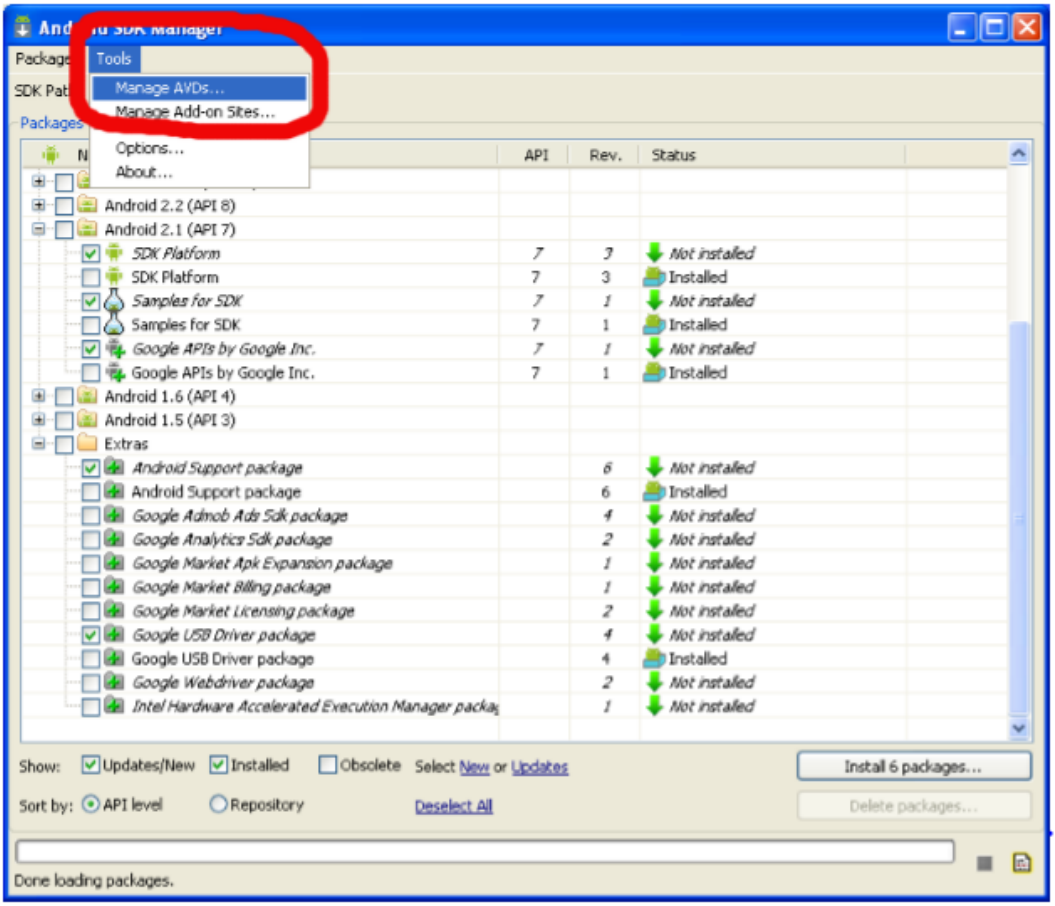


Требуется загрузить нужные версии API и другие полезные инструменты:

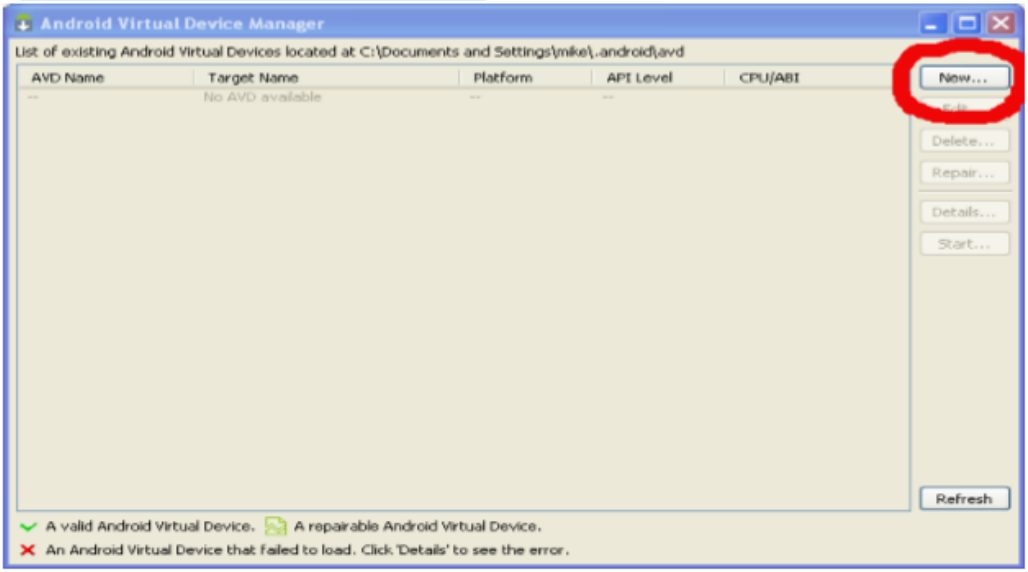

Android SDK Manager
Packages Tools
SDK Path: C:\android-sdk
Packages
Name API Rev. Status
Tools
Android SDK Tools 16 Installed
Android SDK Platform-tools 10 Not installed
Android 4.0.3 (API 15)
Android 4.0 (API 14)
Android 3.2 (API 13)
Android 3.1 (API 12)
Android 3.0 (API 11)
Android 2.3.3 (API 10)
Android 2.2 (API 8)
Android 2.1 (API 7)
Android 1.6 (API 4)
Android 1.5 (API 3)
Extras
Android Support package 6 Not installed
Google Admob Ads Sdk package 4 Not installed
Google Analytics Sdk package 2 Not installed
Google Market Apk Expansion package 1 Not installed
Google Market Billing package 1 Not installed
Google Market Licensing package 2 Not installed
Google USB Driver package 4 Not installed
Google Webdriver package 2 Not installed
Intel Hardware Accelerated Execution Manager packa 1 Not installed
Show: Updates/New Installed Obsolete Select New or Updates
Sort by: API level Repository Deselect All
Install 6 packages...
Delete 1 package...
Done loading packages.


Android SDK Manager
Packages Tools
SDK Path: C:\android-sdk
Packages
Name API Rev. Status
Tools
Android SDK Manager Log
Found XT882 by Motorola Mobility, Inc., Android API 10, revision 2
Found XT928 by Motorola Mobility, Inc., Android API 10, revision 1
Found MT870 by Motorola Mobility, Inc., Android API 10, revision 2
Fetching URL: http://innovator.samsungmobile.com/android/repository/repository.xml
Validate XML: http://innovator.samsungmobile.com/android/repository/repository.xml
Parse XML: http://innovator.samsungmobile.com/android/repository/repository.xml
Found GALAXY Tab by Samsung Electronics., Android API 8, revision 1
Fetching URL: http://developer.sonyericsson.com/edk/android/repository.xml
Validate XML: http://developer.sonyericsson.com/edk/android/repository.xml
Parse XML: http://developer.sonyericsson.com/edk/android/repository.xml
Found EDK 2.0 by Sony Ericsson Mobile Communications AB, Android API 10, revision 1
Done loading packages.
Preparing to install archives
Downloading Android SDK Platform-tools, revision 10
Installing Android SDK Platform-tools, revision 10
Stopping ADB server failed (code -1).
OK
Unzipping Android SDK Platform-tools, revision 10 (17%)
Close
Google Market Licensing package 2 Not installed
Google USB Driver package 4 Not installed
Google Webdriver package 2 Not installed
Intel Hardware Accelerated Execution Manager packa 1 Not installed
Show: Updates/New Installed Obsolete Select New or Updates
Install 6 packages...
Sort by: API level Repository Deselect All
Delete 1 package...
Unzipping Android SDK Platform-tools, revision 10 (17%)

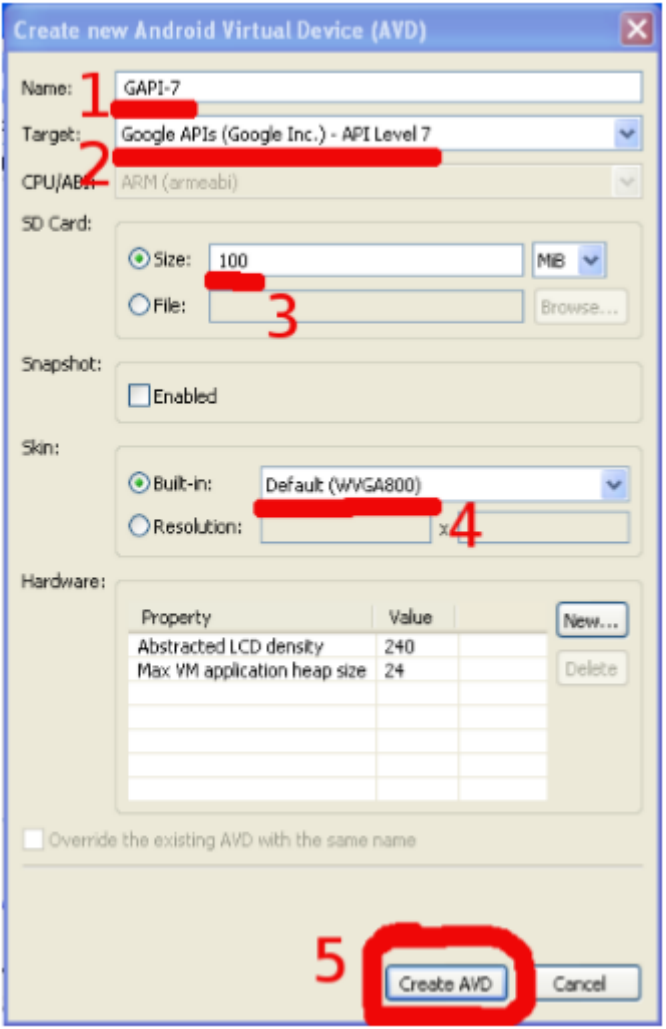
Далее мы запустим менеджер AVD (Android Virtual Devices) и сконфигурируем эмулятор на использование Android версии 2.1 с расширением Google API (GAPI). GAPI нужен, как правило, для приложений, использующих картографические возможности Android



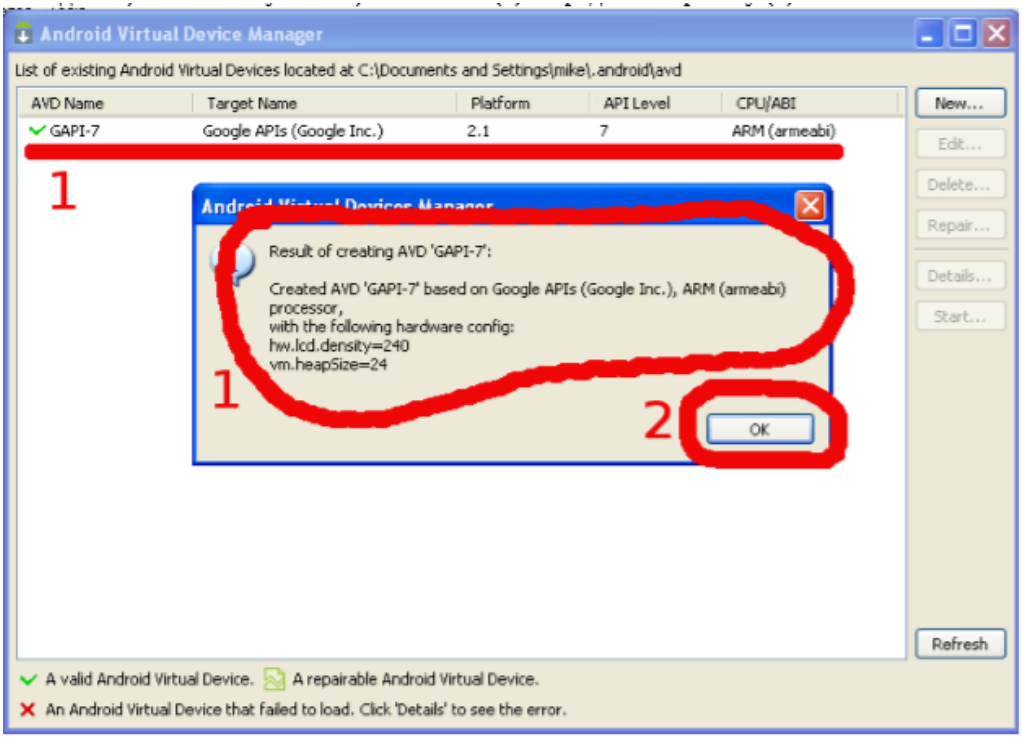
Создадим новое виртуальное устройство:



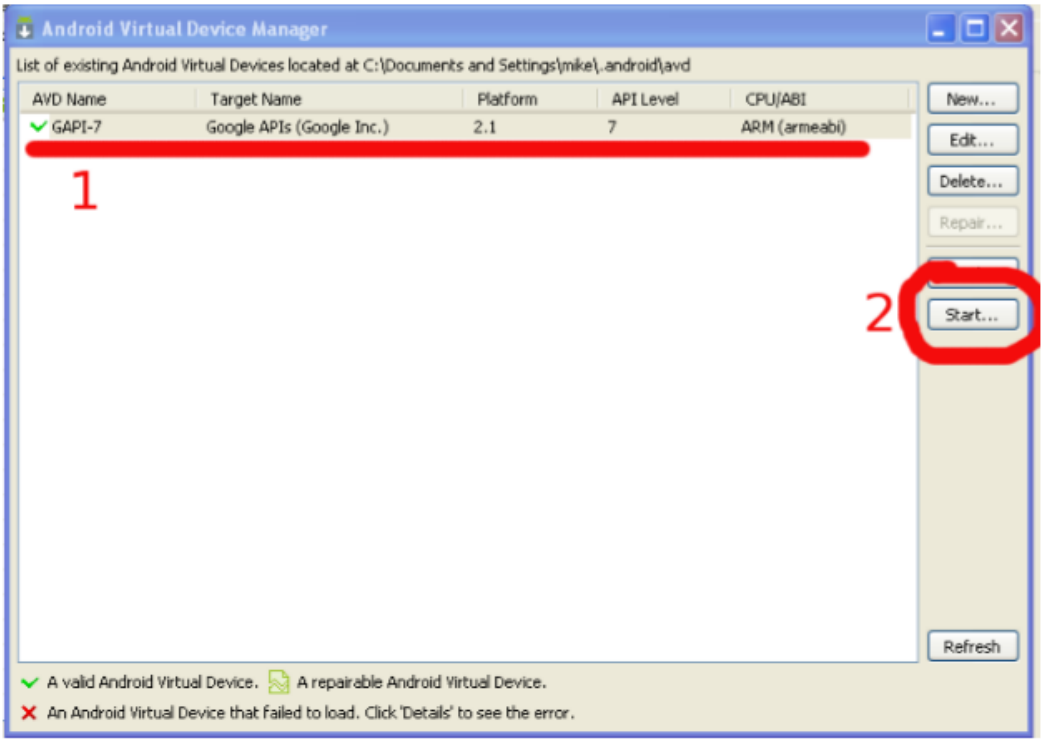
При создании нового виртуального устройства выбираем его название(1), целевой API (2), размер (или файл с образом) виртуальной SD-карты (3), одно из стандартных или собственное разрешение экрана (4) и, наконец, создаем устройство(5). При необходимости можно указать плотность пикселей на экране, максимальный размер кучи для приложений внутри виртуальной машины, а также другие параметры:



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

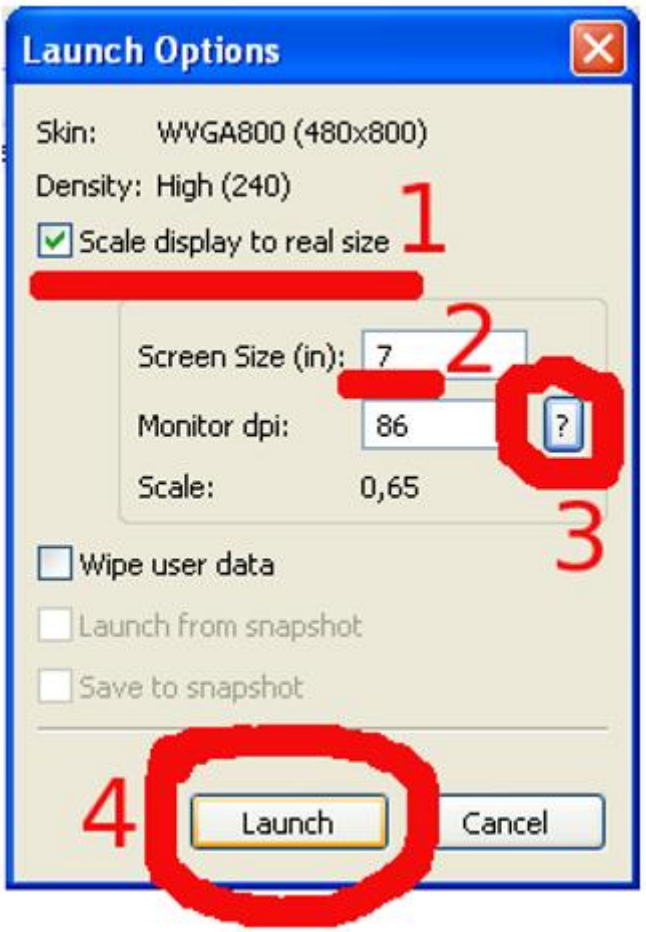


Пришло время запустить эмулятор:

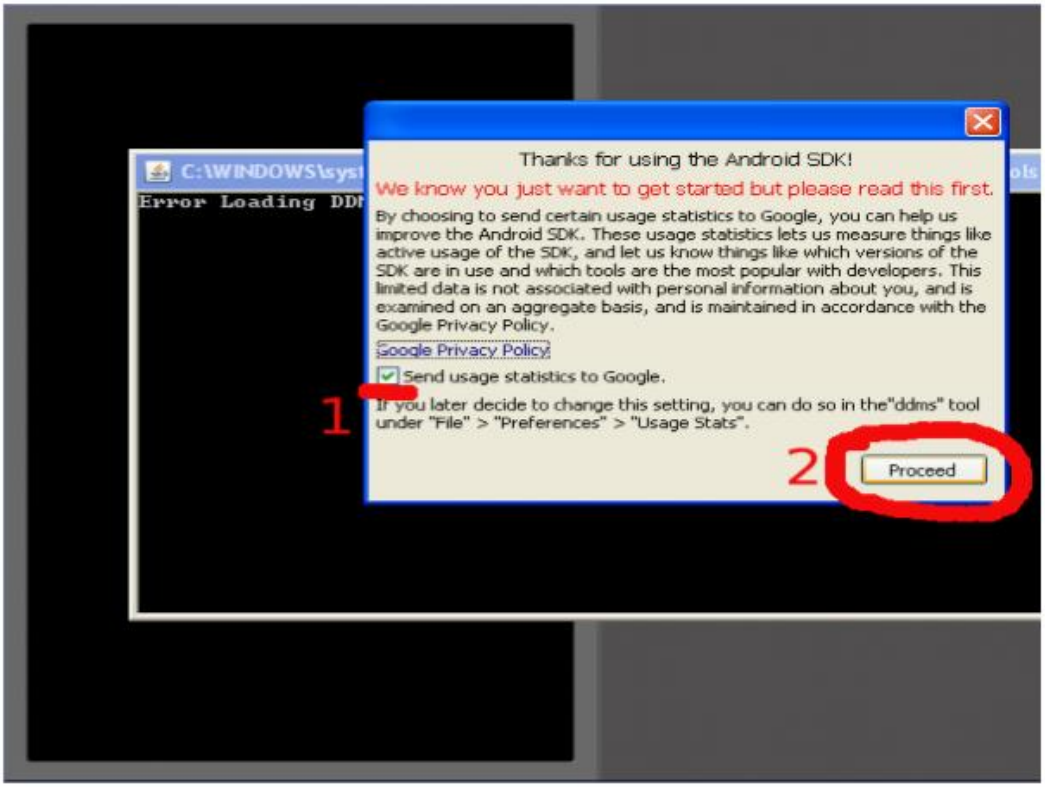


Важно: при использовании русскоязычных версий ОС Windows путь к рабочему каталогу AVD может включать русские буквы, что приведет к невозможности запуска эмуляторов. В этом случае можно создать каталог для AVD в корневом каталоге диска (например, C:\AVD) и присвоить переменной окружения ANDROID_SDK_HOME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

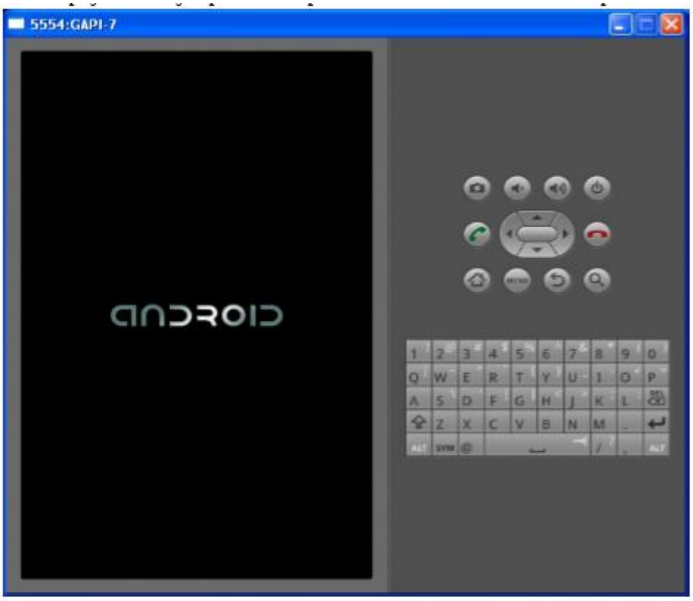


При первом запуске вы можете указать, следует ли отправлять анонимную статистику использования SDK в Google:



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

Загруженное виртуальное устройство практически не отличается от реального:

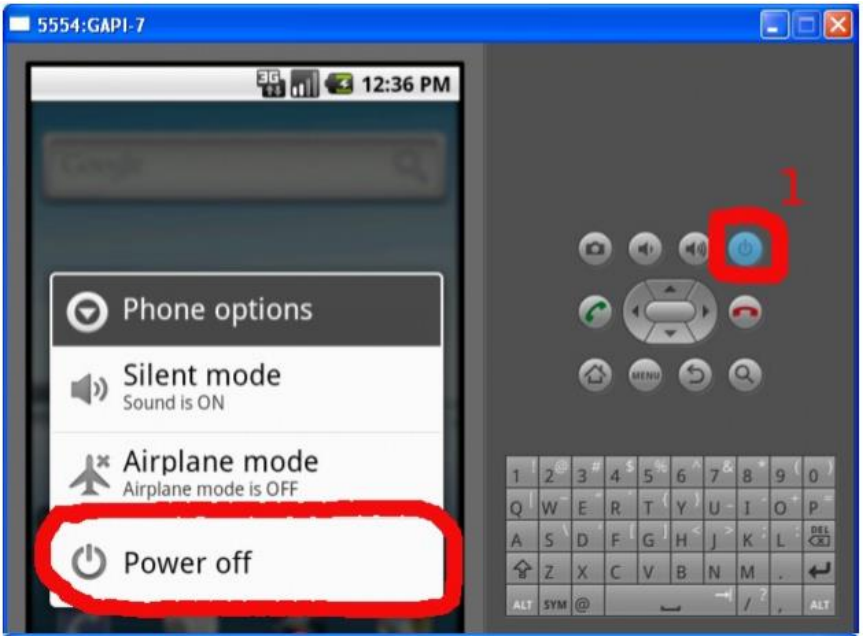


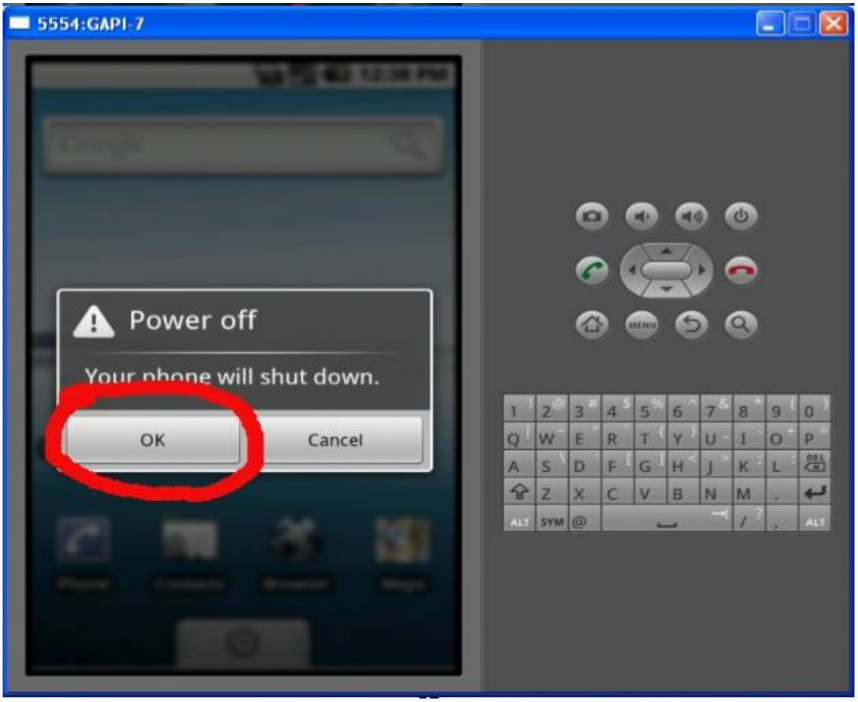
Более того, «пощупать» самые новые версии ОС Android и

оценить их функциональные возможности можно безвозмездно, то есть даром просто выбрав нужную версию API при создании виртуального устройства.

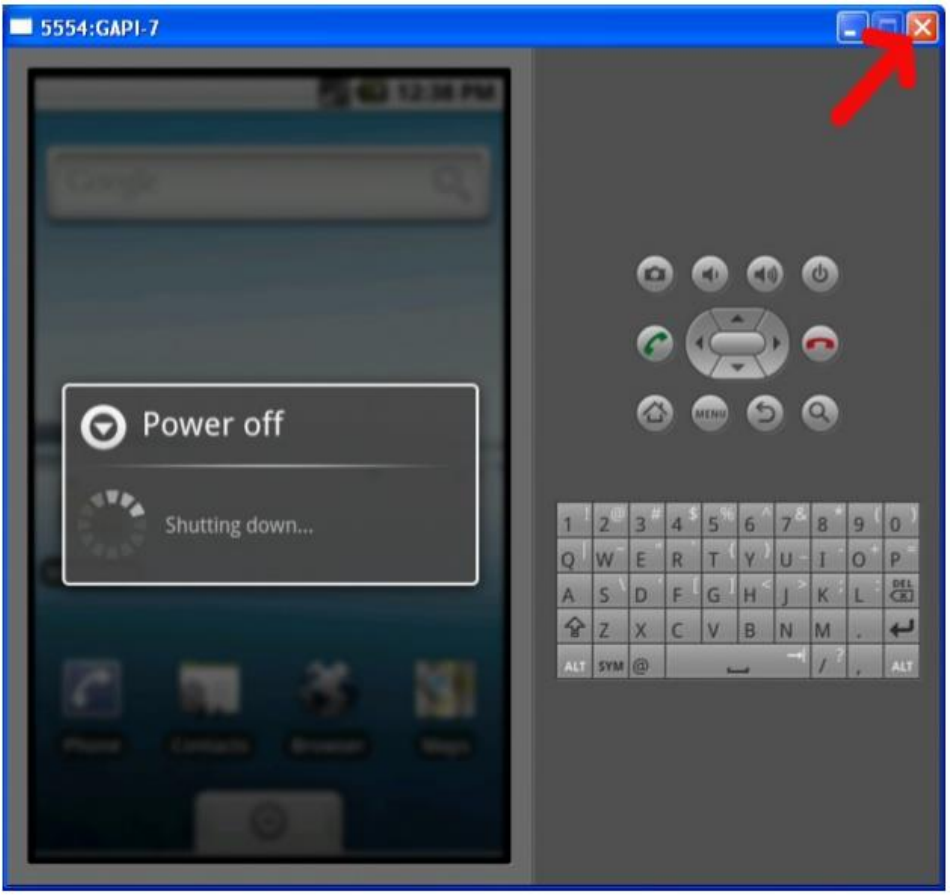
Одна из неприятных особенностей платформы Android с точки зрения разработчика заключается в большом количестве выпущенных версий, значительно различающихся по возможностям. В настоящий момент Open Handset Alliance (т. е. Google) несколько затормозила этот процесс, и на данный момент флагманской версией 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:

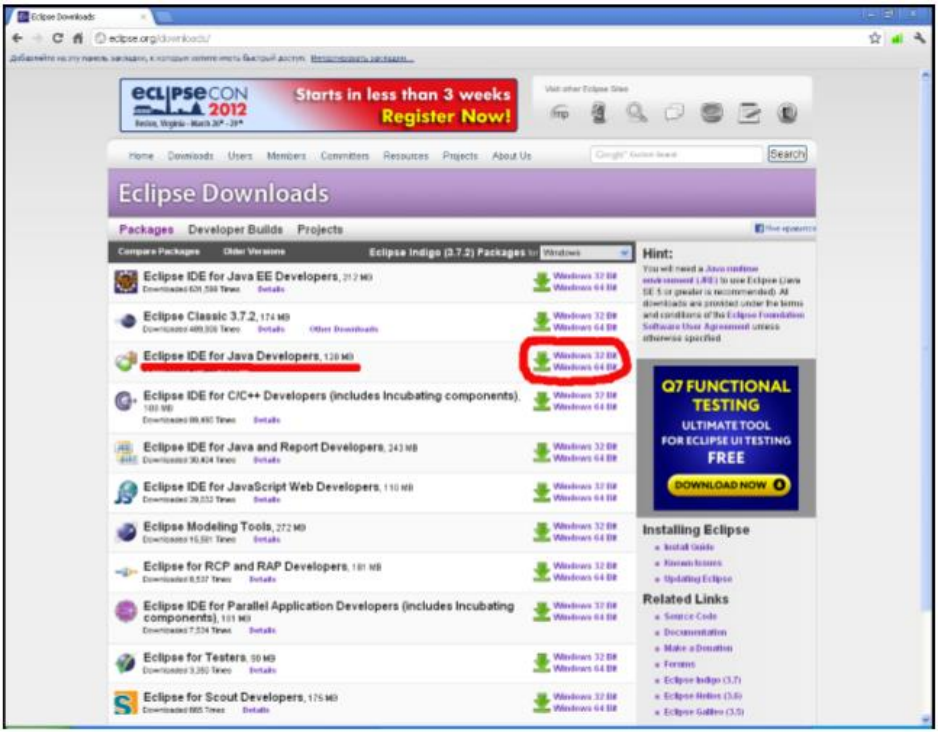




Через несколько секунд окно можно закрыть:

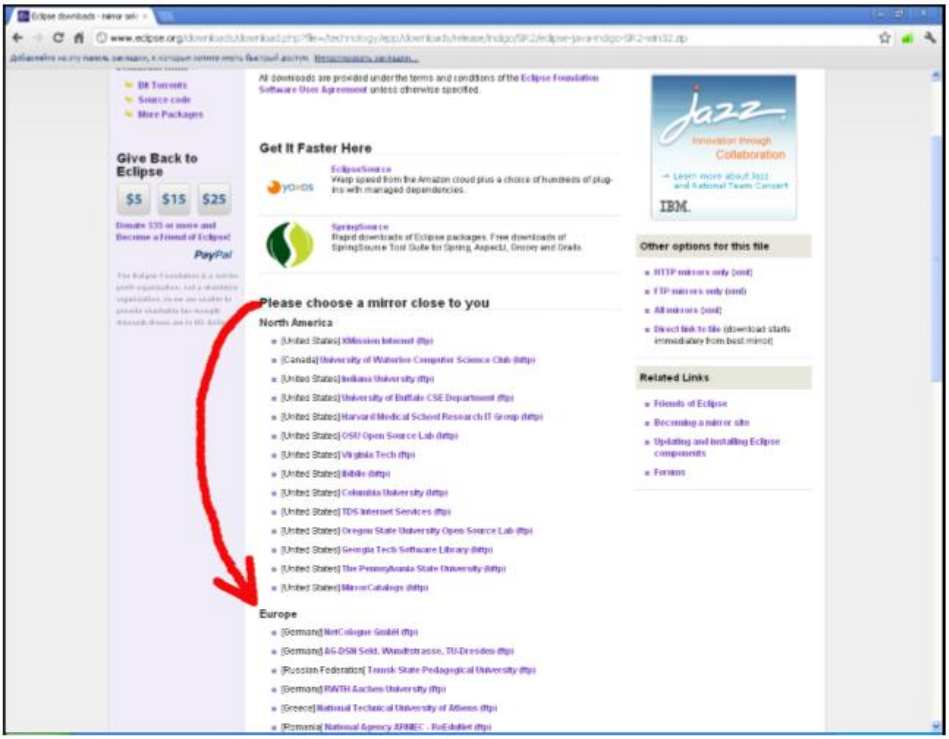


Установка IDE Eclipse Интегрированная среда разработки Eclipse (довольно странное название, по-английски означает, в том числе помутнение рассудка) доступна для загрузки на официальном сайте по адресу http://eclipse.org/downloads/ и распространяется в виде ZIP-архива:

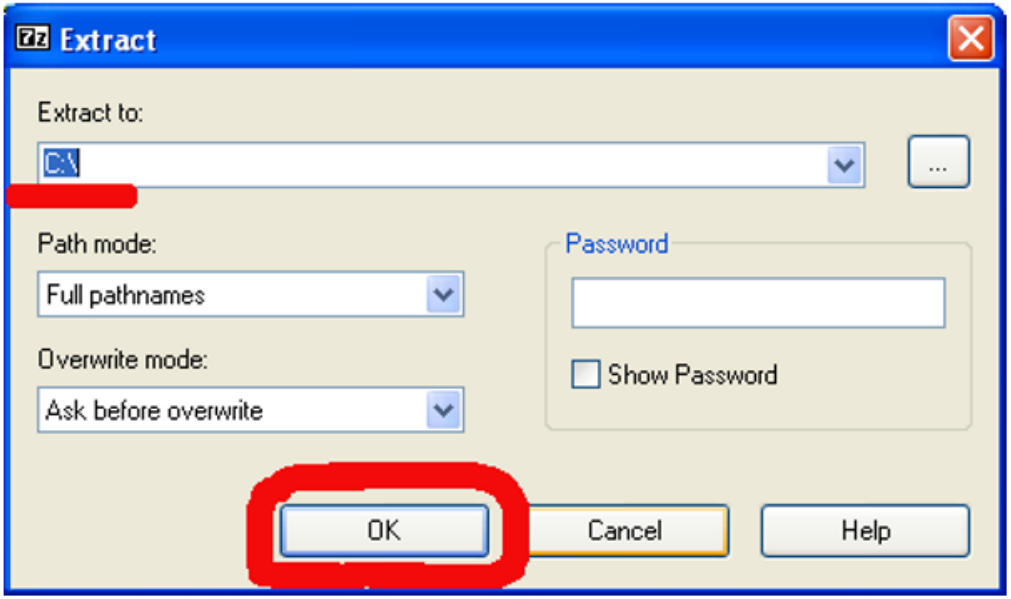


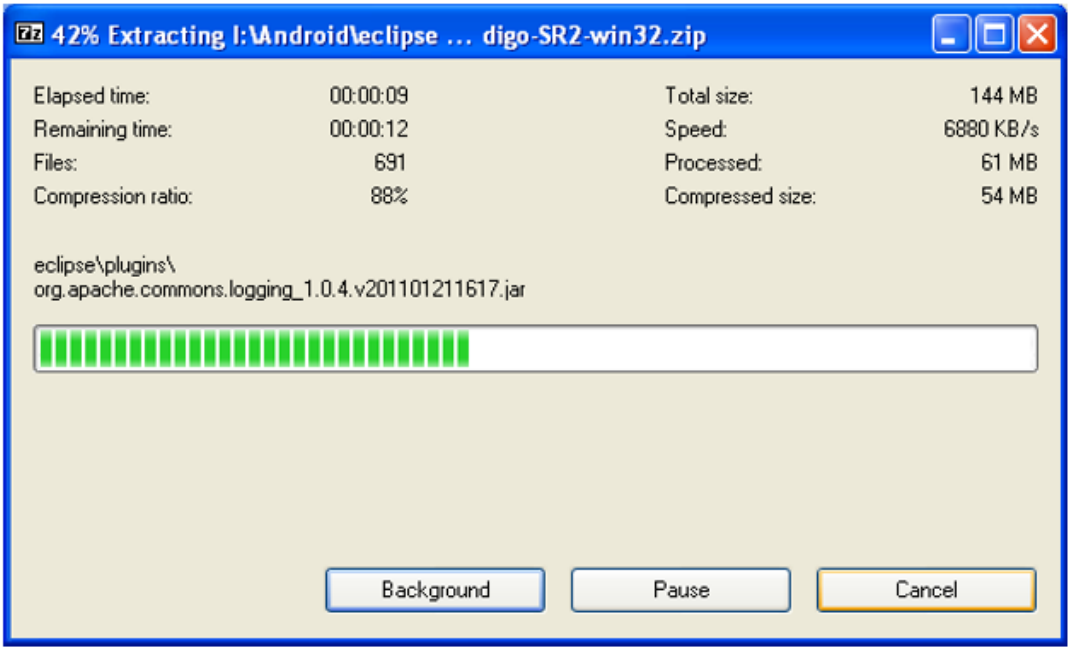
В нашем случае подходящим вариантом является Eclipse IDE for Java Developers.

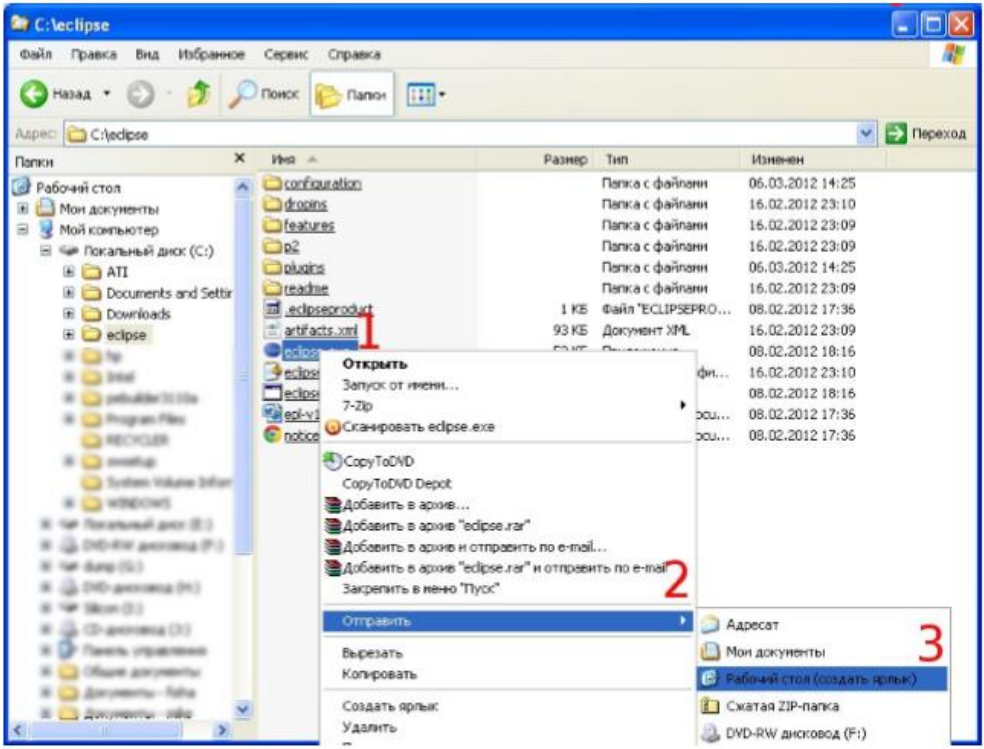
После выбора подходящей для используемой на ПК операционной системы версии IDE, для уменьшения времени загрузки имеет смысл выбрать европейское зеркало сайта:



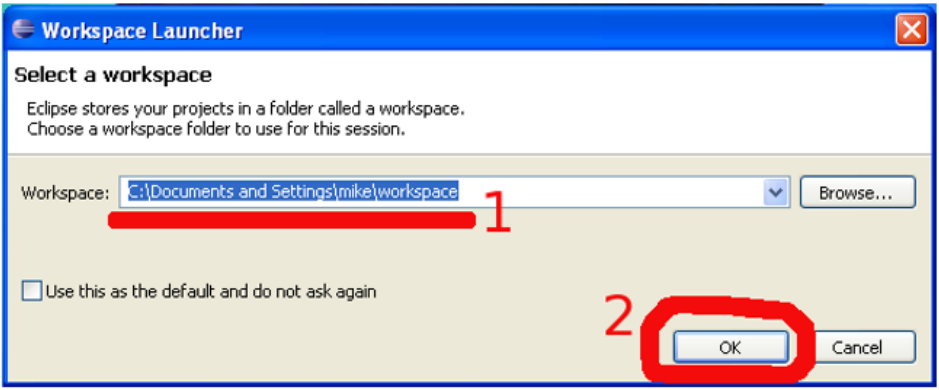
Получив нужный архив, выбираем подходящий путь для установки и распаковываем архив:


Extract
Extract to:
C:\
...
Path mode:
Full pathnames
Overwrite mode:
Ask before overwrite
Password
Show Password
OK
Cancel
Help


42% Extracting I:\Android\eclipse ... digo-SR2-win32.zip
Elapsed time: 00:00:09
Remaining time: 00:00:12
Files: 691
Compression ratio: 88%
Total size: 144 MB
Speed: 6880 KB/s
Processed: 61 MB
Compressed size: 54 MB
eclipse\plugins\
org.apache.commons.logging_1.0.4.v201101211617.jar
Background
Pause
Cancel



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



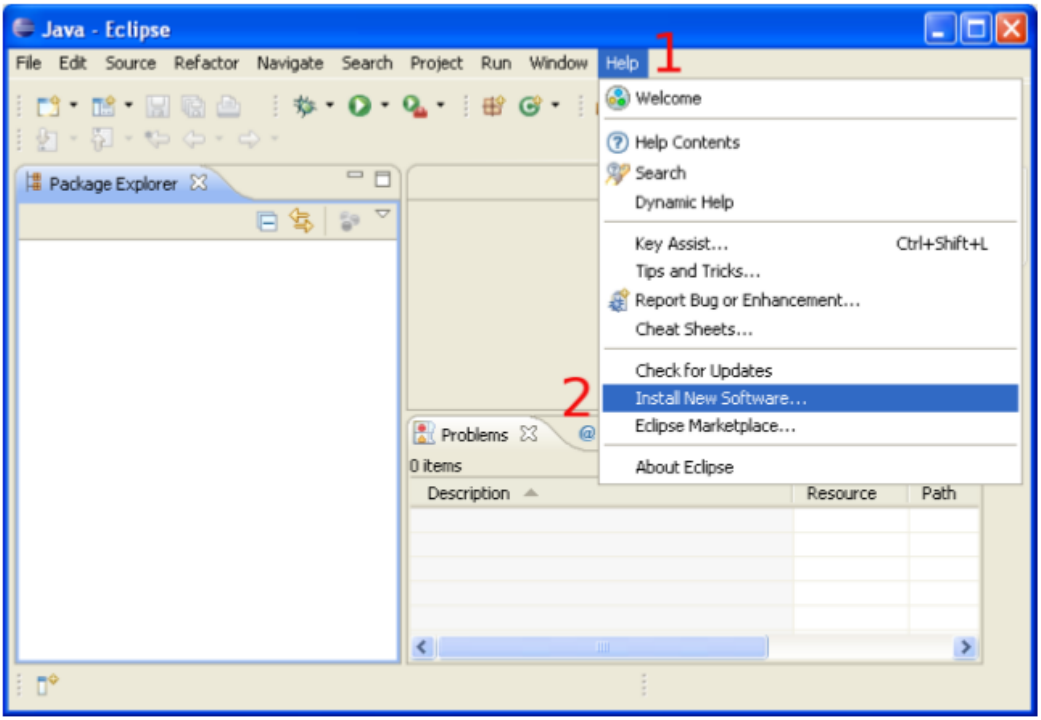
Установка Eclipse закончена. Остался последний шаг – установка плагина ADT.

Установка плагина ADT

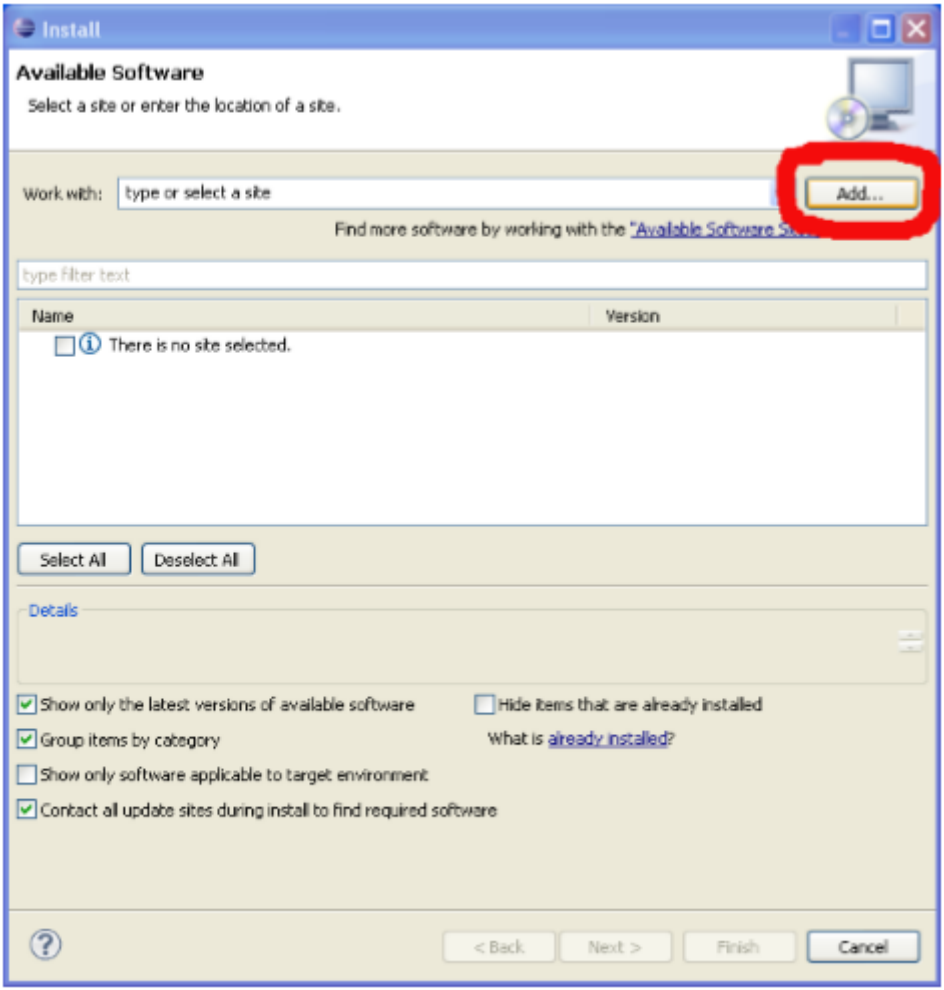
Для установки плагина ADT не требуется ручная загрузка, он будет установлен системой управления плагинами Eclipse. Тем не менее, мы рассмотрим установку подробно.

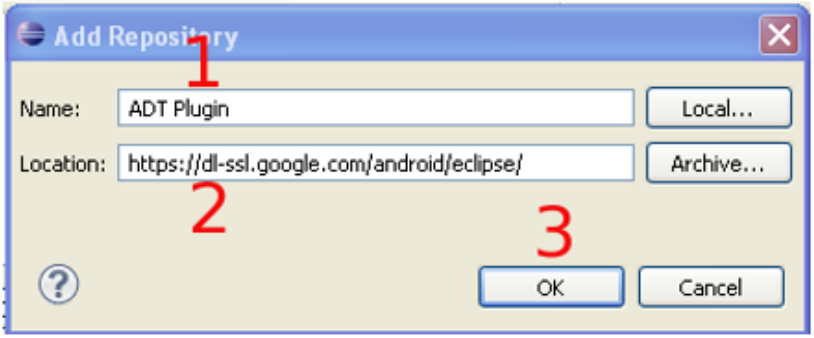
Запустим Eclipse

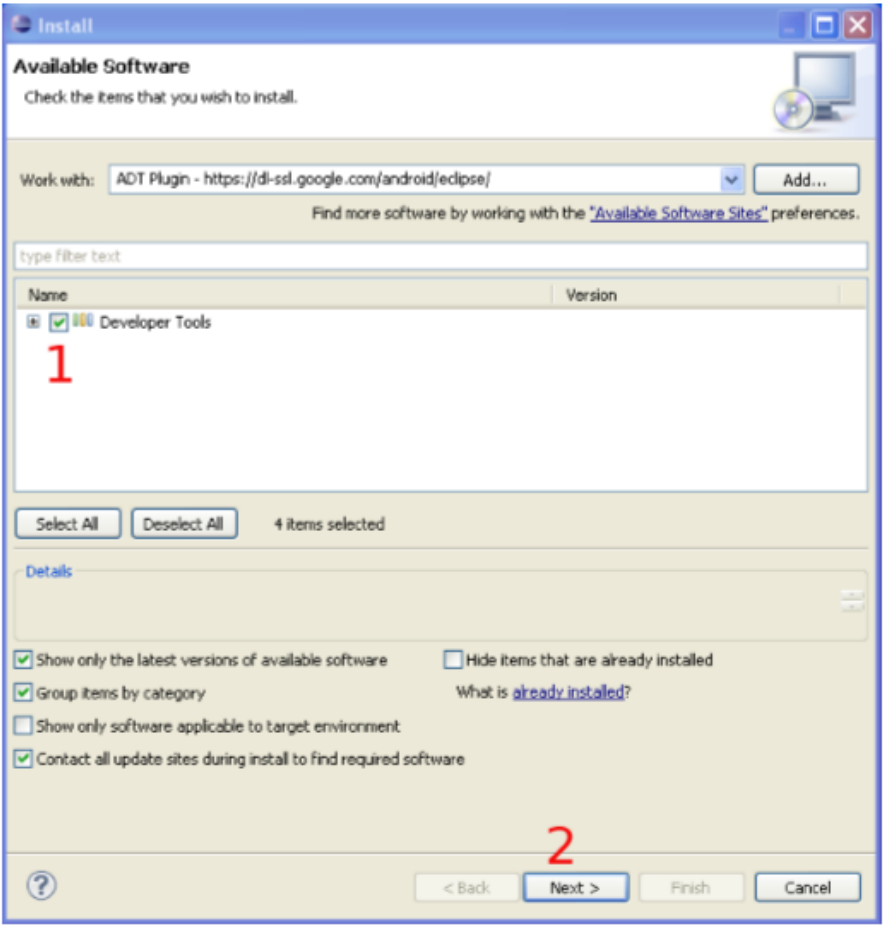
Выберем нужный пункт меню



Добавим нужный источник ПО


Install
Available Software
Select a site or enter the location of a site.
Work with:
type or select a site
Add...
Find more software by working with the "Available Software Sites"
type filter text
Name
Version
There is no site selected.
Select All
Deselect All
Details
Show only the latest versions of available software
Hide items that are already installed
Group items by category
What is already installed?
Show only software applicable to target environment
Contact all update sites during install to find required software
< Back
Next >
Finish
Cancel



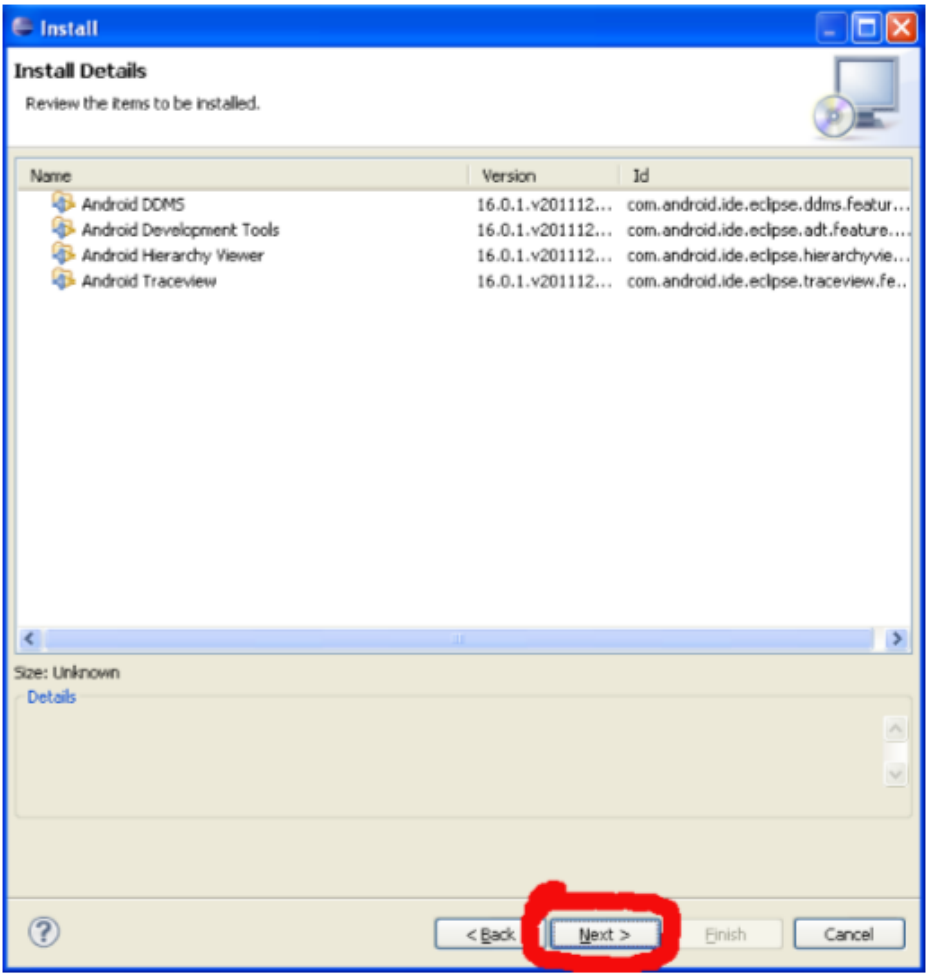


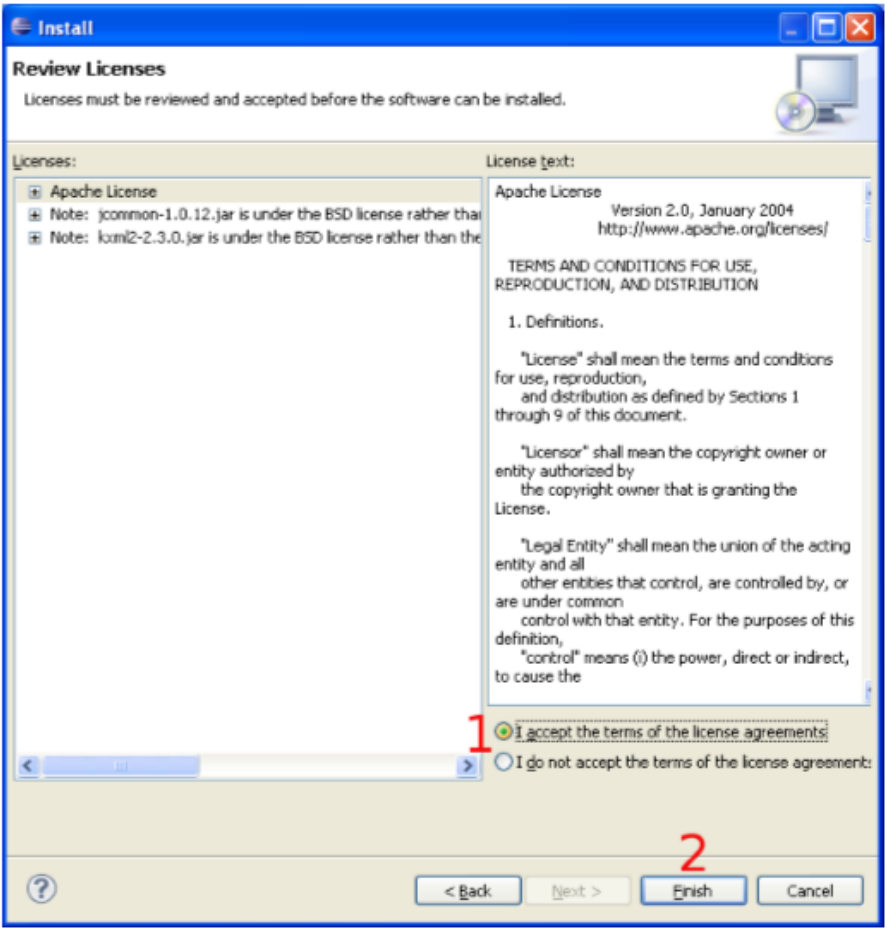
Выберем нужные (все) компоненты для загрузки и установ-

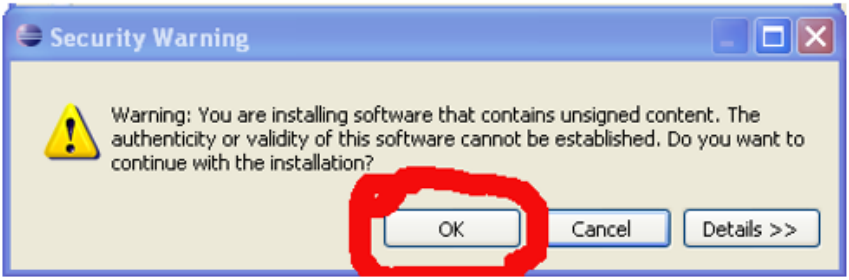
ки

Убедимся в этом

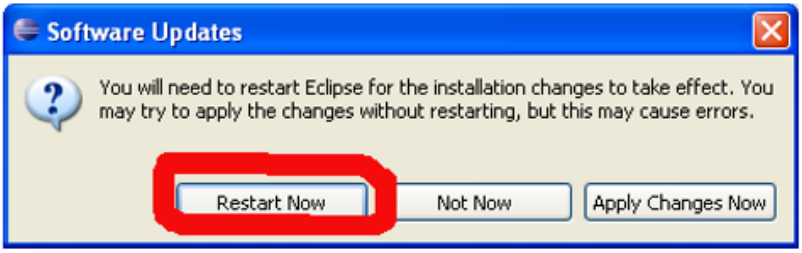
Примем все лицензионные соглашения и продолжим установку



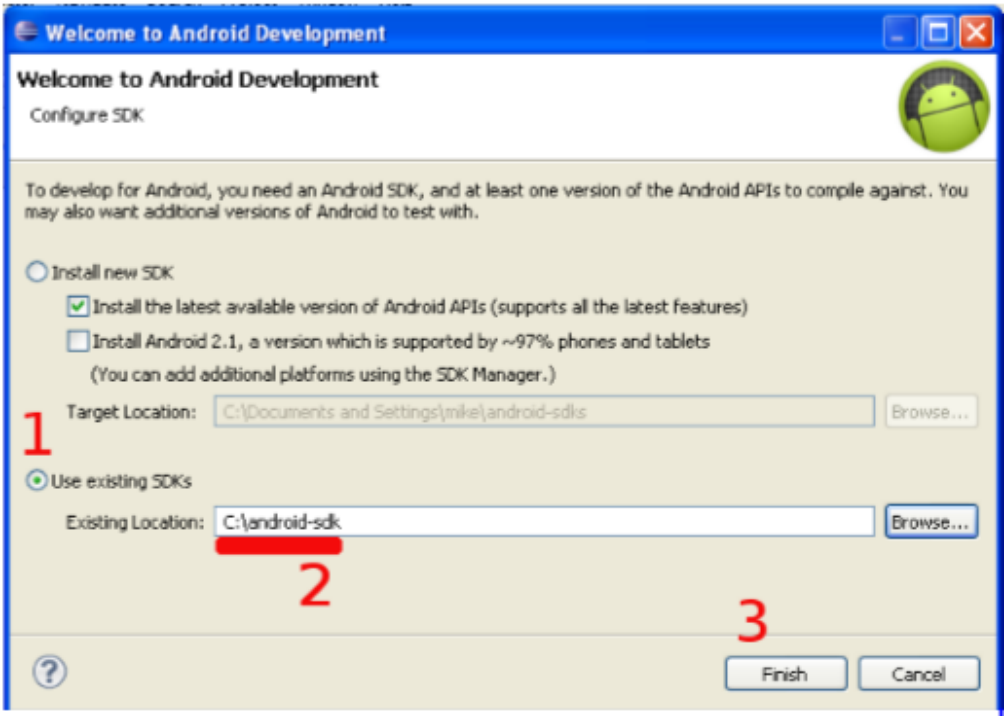

Install
Review Licenses
Licenses must be reviewed and accepted before the software can be installed.
Licenses:
Apache License
Note: jcommon-1.0.12.jar is under the BSD license rather thai
Note: lxml2-2.3.0.jar is under the BSD license rather than the
License text:
Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/
TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
1. Definitions.
"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.
"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.
"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
1
I accept the terms of the license agreements
I do not accept the terms of the license agreement
2
< Back
Next >
Finish
Cancel



После установки выбранных компонентов перезапустим Eclipse



И настроим плагин ADT на использование уже установленного Android SDK.

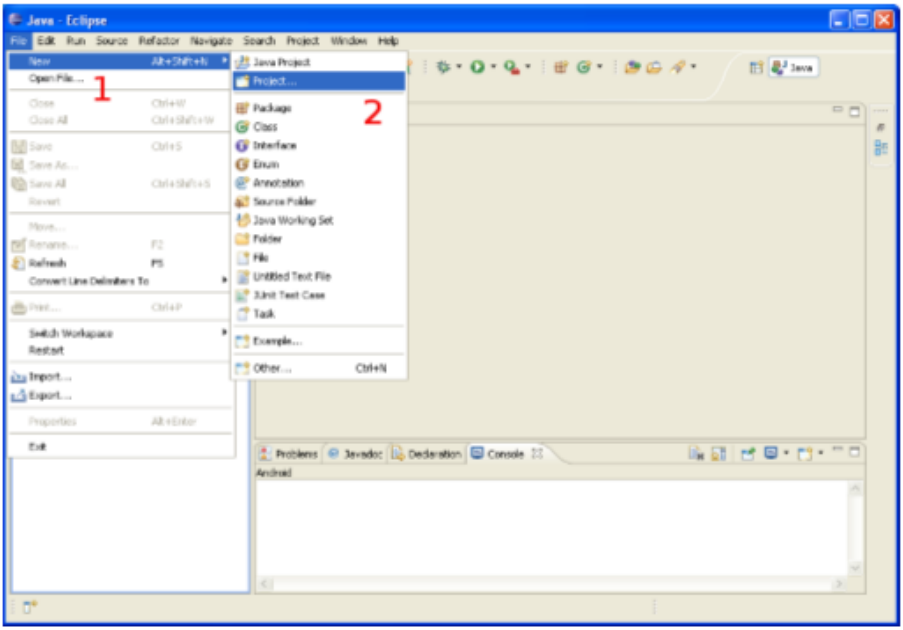


На снимке экрана (выше) можно увидеть, что ADT в состоянии сам загрузить и установить Android SDK. Мы использовали ручную установку для ускорения процесса (не тратили время на ожидание загрузки SDK).

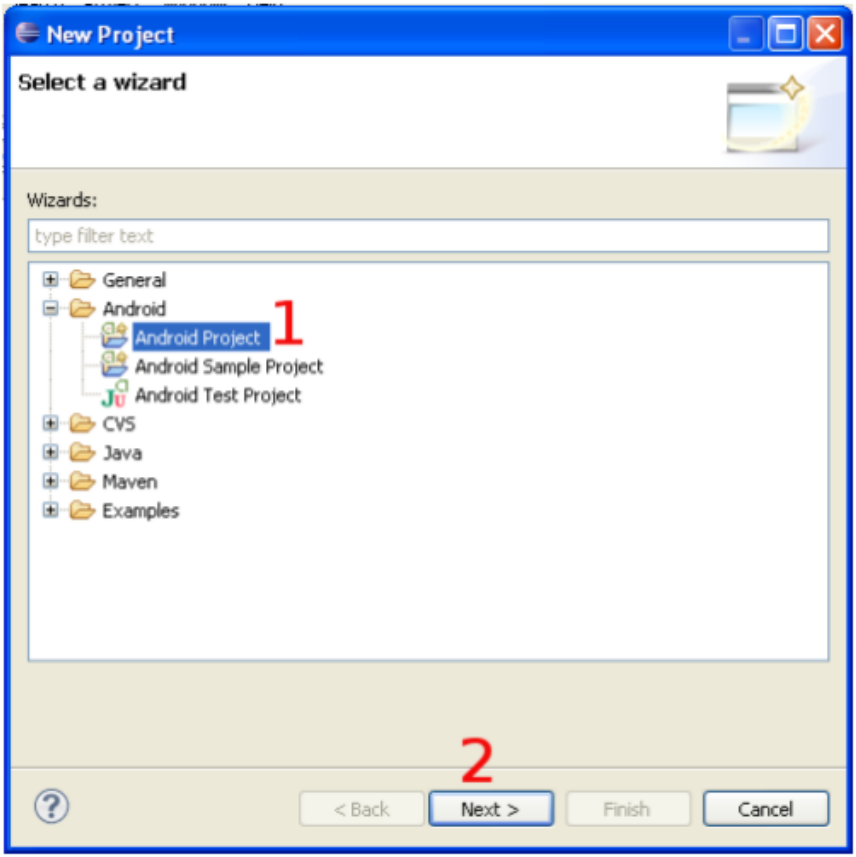
Среда разработки установлена, пришло время создать первое приложение для платформы Android и запустить его в эмуляторе.

1.2. Создание первого приложения под Android

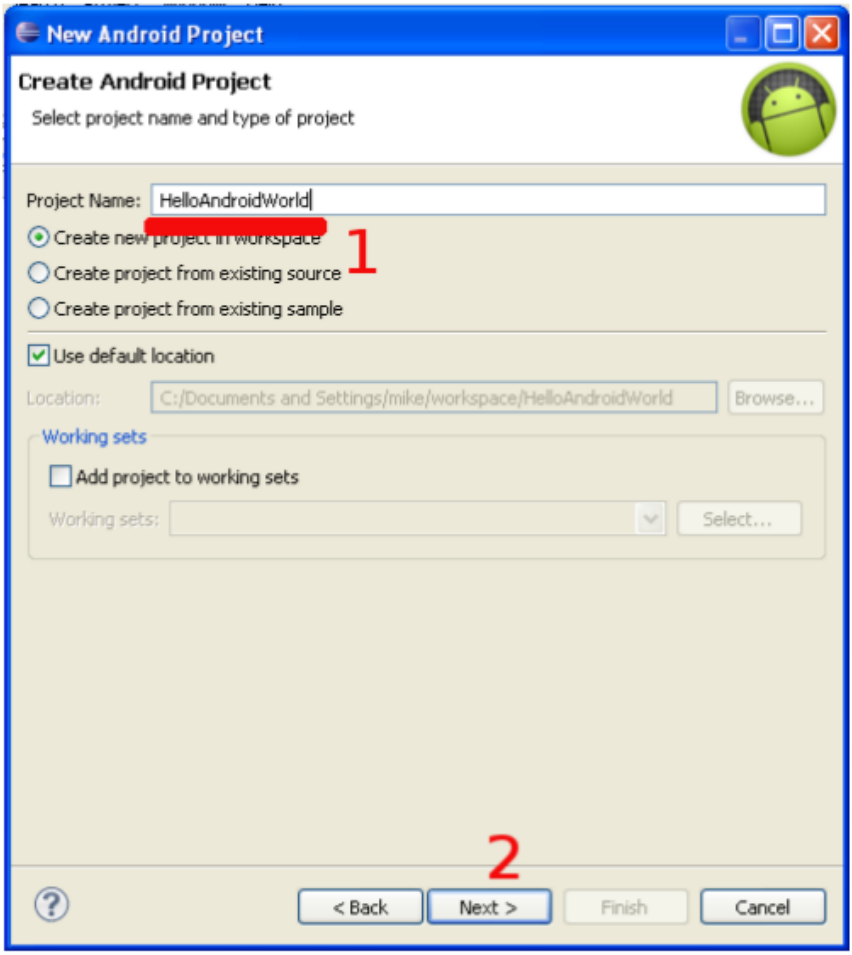
Для создания приложений в установленной нами среде разработки удобно использовать «Мастер создания нового проекта»:



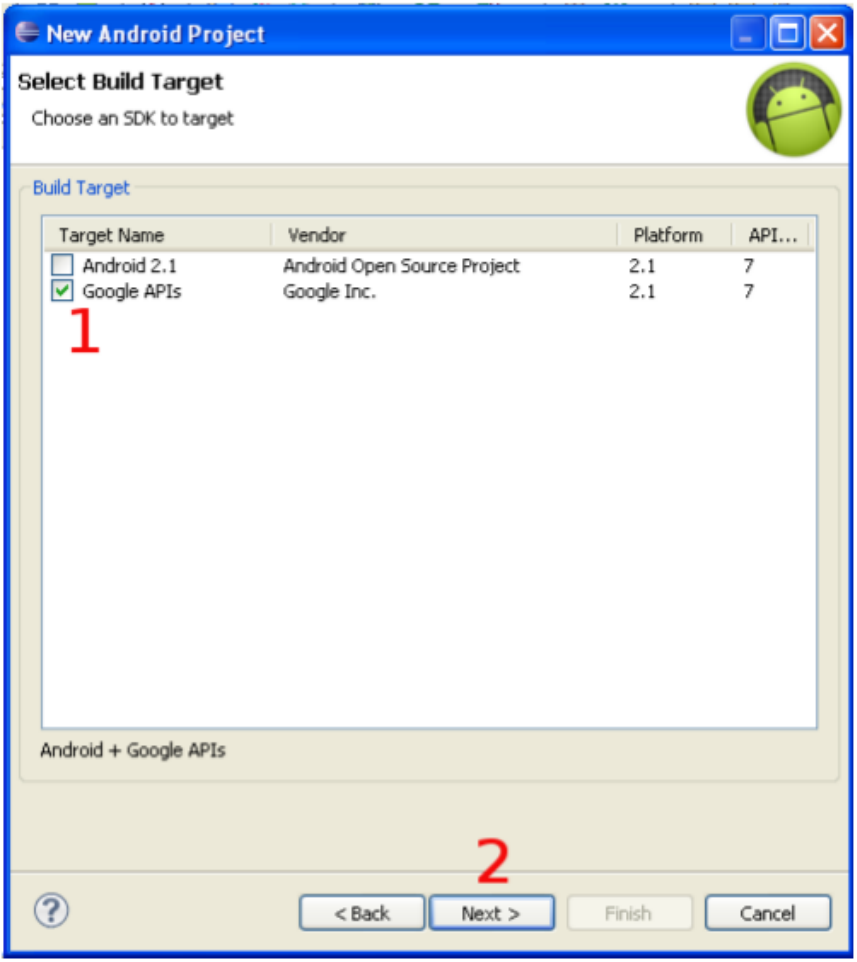
Выбираем «Android Project»



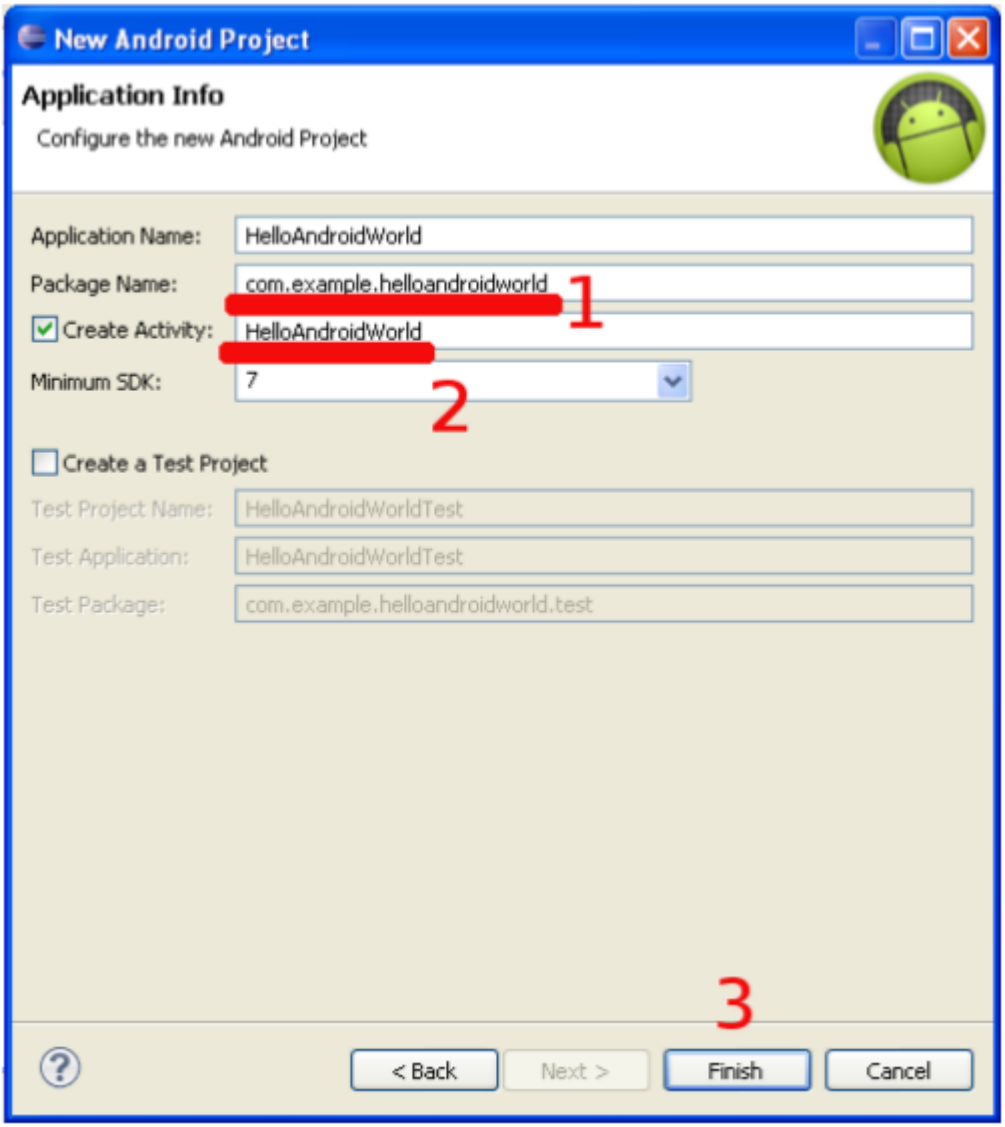
Вводим имя проекта



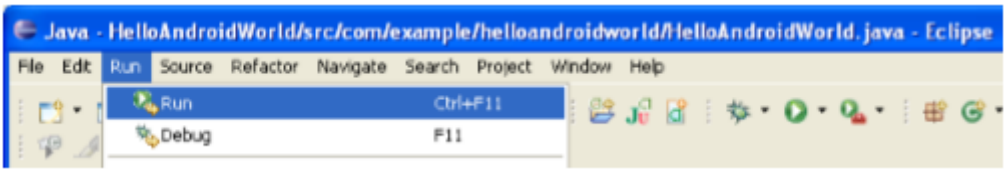
Выбираем целевую платформу



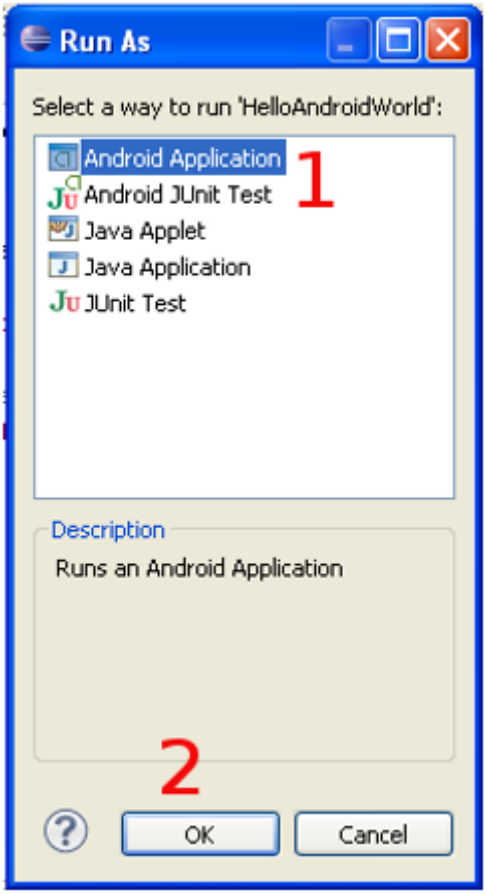
Вводим имя пакета (1) и имя класса Активности(2), который будет для нас автоматически создан мастером:



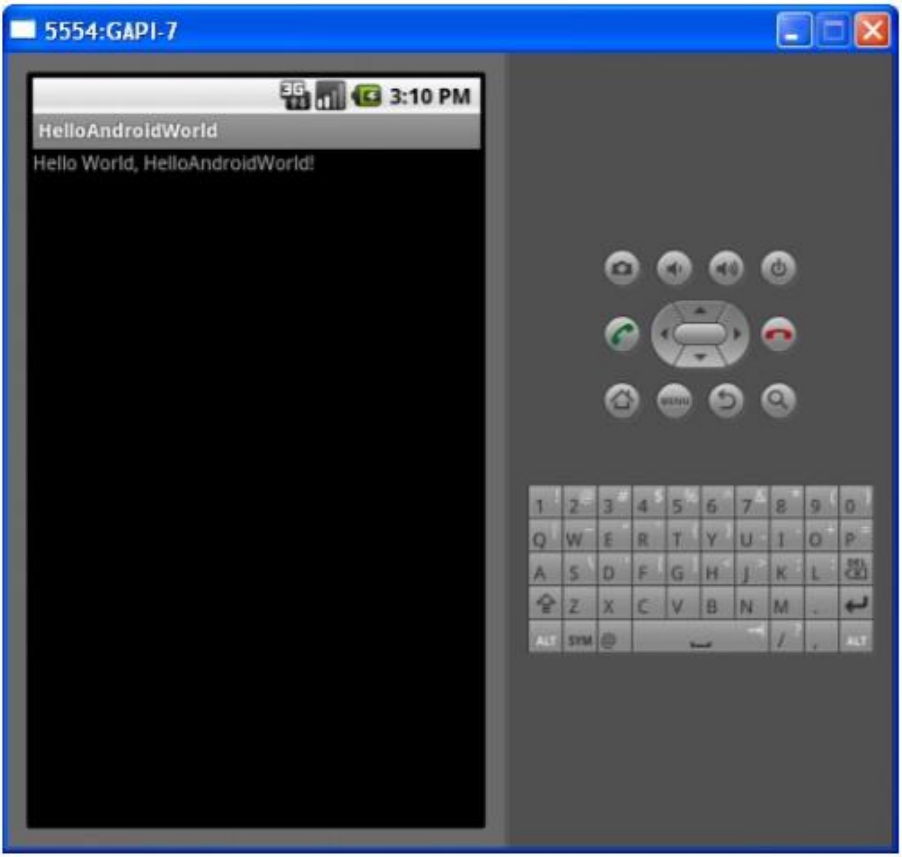
Созданный проект сразу пока мы не внесли в него несовместимы с жизнью изменения является работоспособный приложением, так что мы его можем запустить:



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



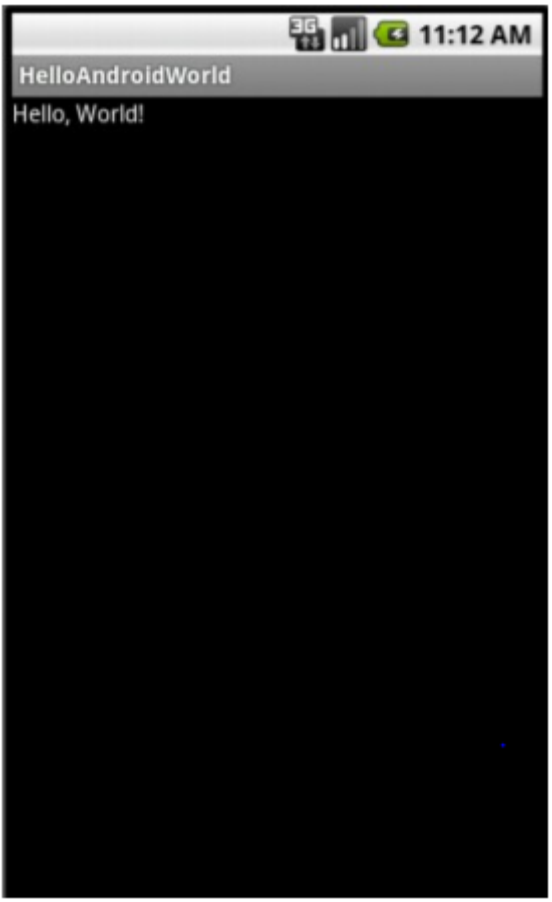
И наслаждаемся результатом:



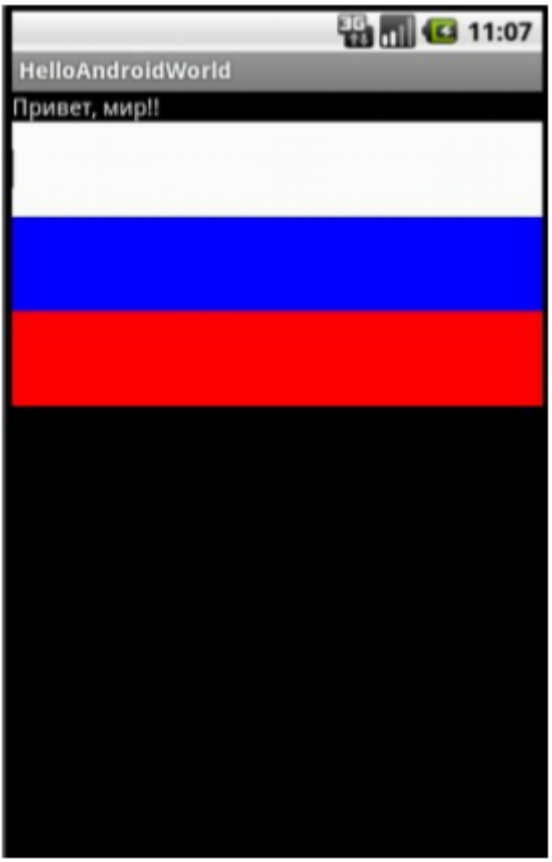
2. ЛАБОРАТОРНАЯ РАБОТА №2: ЛОКАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

2.1. Локализация приложения

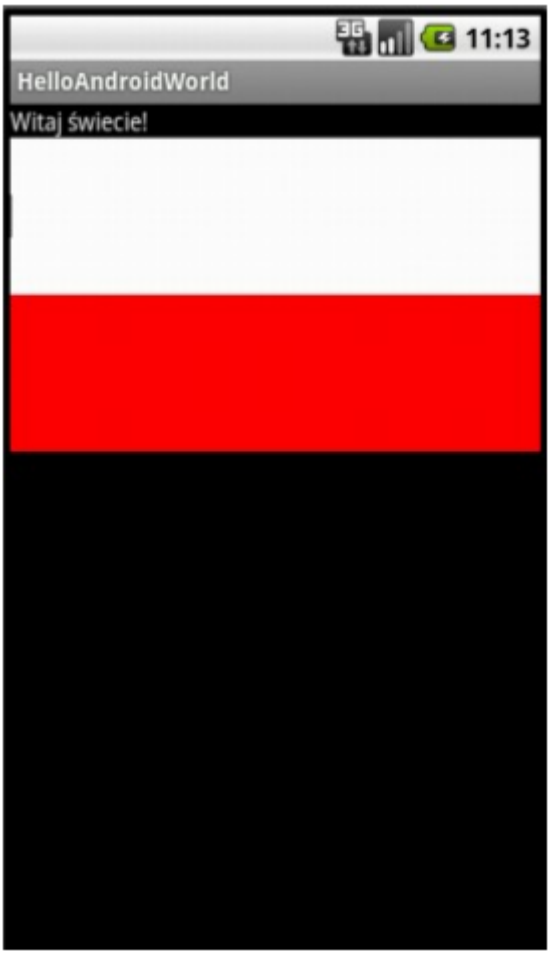
1. Измените ресурсы приложения HelloAndroidWorld так, чтобы оно выглядело следующим образом (на виртуальном устройстве должен быть установлен английский язык):



2. Добавьте нужные подкаталоги и ресурсы в каталог res проекта HelloAndroidWorld, чтобы при настройке русского языка приложение меняло свой вид на следующий:



3. Проделайте те же действия для польского языка:



4. Восстановите языковые настройки на виртуальном устройстве.

2.2. Использование LinearLayout

1. Продолжите локализацию приложения HelloAndroidWorld, теперь для французского языка. Для рисования флага используйте вложенный LinearLayout с вертикальной ориентацией

дочерних элементов:

2. Обратите внимание, что при размещении элементов внутри вложенного LinearLayout удобно указывать атрибут android:layout_weight="1", в этом случае дочерние виджеты будут размещены по горизонтали равномерно.

3. После получения нужного результата верните стандартные языковые настройки в виртуальном устройстве.

2.3. Использование анимации

1. Создайте новый проект AnimSample.
2. В каталоге res создайте каталог anim, а в нем файл с

именем ship_anim.xml, описывающий анимацию. Отредактируйте файл, чтобы он имел следующее содержимое:

```
<?xml version="1.0" encoding="utf-8"?>
<set
xmlns:android="http://schemas.android.com/apk/res/android" >
    <rotate
                android:duration="3333"
                android:fromDegrees="0"
                android:pivotX="50%"
        android:pivotY="50%"
        android:repeatCount="infinite"
        android:repeatMode="reverse"
        android:toDegrees="1080" />

    <translate
        android:duration="1900"
        android:fromXDelta="-50%p"
        android:repeatCount="infinite"
        android:repeatMode="reverse"
        android:toXDelta="50%p" />

     <translate
        android:duration="1300"
        android:fromYDelta="-50%p"
        android:repeatCount="infinite"
        android:repeatMode="reverse"
        android:startOffset="123"
        android:toYDelta="50%p" />
    <alpha
        android:duration="500"
        android:fromAlpha="1.0"
        android:repeatCount="infinite"
        android:repeatMode="reverse"
        android:toAlpha="0.3" /
    <scale
        android:duration="10000"
        android:fromXScale="0.0"
        android:fromYScale="0.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:repeatCount="infinite"
```

```
            android:repeatMode="reverse"
            android:toXScale="2.5"
            android:toYScale="2.5" />
    </set>
```

3. Добавьте рисунок, к которому будет применяться анимация (файл lander_plain.png), в каталог res/drawable-mdpi.
4. В файле разметки res/layout/main.xml замените элемент TextView на ImageView со следующими атрибутами:

```
    <ImageView
            android:id="@+id/shipView"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:src="@drawable/lander_plain" />
```

5. В конце метода onCreate Активности (она в этом проекте одна) добавьте следующие строки:

```
ImageView ship = (ImageView) findViewById(R.id.shipView);
Animation shipAnim = AnimationUtils.loadAnimation(this,
        R.anim.ship_anim);
ship.startAnimation(shipAnim);
```

6. Запустите проект.

3. ЛАБОРАТОРНАЯ РАБОТА №3: ИСПОЛЬЗОВАНИЕ RELATIVELAYOUT

3.1. Использование RelativeLayout

RelativeLayout является очень полезным вариантом разметки, позволяющим создавать пользовательский интерфейс без избыточного количества вложенных элементов ViewGroup.

1. Создайте новый проект с именем RelativeLayout-Sample.
2. Добавьте нужные строки в файл res/values/strings.xml и удалите строку с именем hello:

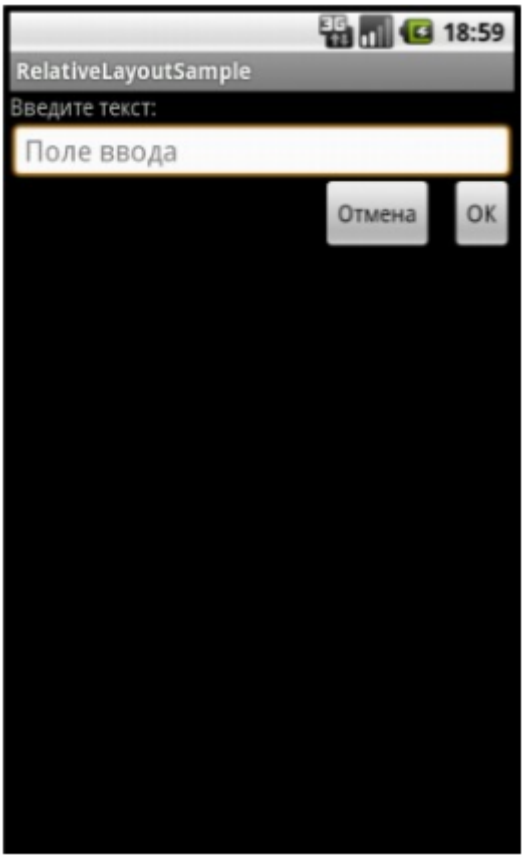
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="label_text">Введите текст:</string>
  <string name="entry_hint">Поле ввода</string>
```

```
    <string name="app_name">RelativeLayoutSample</string>
</resources>
```

3. Отредактируйте файл разметки res/layout/main.xml так, чтобы он имел следующее содержимое:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent" >
<TextView
  android:id="@+id/label"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:text="@string/label_text" />
<EditText
  android:id="@+id/entry"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:layout_below="@id/label"  andro-
id:background="@android:drawable/editbox_background"  an-
droid:hint="@string/entry_hint" />
<Button
  android:id="@+id/ok"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignParentRight="true"
  android:layout_below="@id/entry"
  android:layout_marginLeft="10dip"
  android:text="@android:string/ok" />
 <Button
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignTop="@id/ok"
  android:layout_toLeftOf="@id/ok"
  android:text="@android:string/cancel" />
</RelativeLayout>
```

4. Запустите приложение:



5. Отредактируйте файл AndroidManifest.xml, чтобы изменить тему, используемую приложением. Для это в узел <application> внесите следующие изменения:

```
  . . .
<application
  android:icon="@drawable/ic_launcher"
  android:label="@string/app_name"
  android:theme="@android:style/Theme.Light" >
  . . .
```

6. Запустите приложение с новой темой:

7. Мы использовали для текста кнопок в разметке стандартные строковые значения, которые предоставляет Android. Поэкспериментируйте с настройками языка в виртуальном устройстве и обратите внимание, как при запуске приложения меняются надписи на кнопках. Очевидно, что использование стандартных строковых значений позволяет минимизировать затраты на локализацию приложений.

4. ЛАБОРАТОРНАЯ РАБОТА №4: ИСПОЛЬЗОВАНИЕ TABWIDGET

4.1. Использование TabWidget

«Табулированная» разметка позволяет создавать UI, содержание вкладки. В этом случае используются такие виджеты,

как TabHost и TabWidget.

TabHost является корневым узлом в разметке, содержащим TabWidget для отображения «вкладок», и FrameLayout для соответствующего им контента.

Отображение контента вкладок можно реализовать двумя способами:

- описав Представление (View) для содержимого каждой вкладки внутри одной и той же Активности;
- используя вкладки для переключения между различными Активностями.

Выбор конкретной реализации зависит от потребностей разработчика, но обычно более предпочтительным является второй вариант, так в этом случае разные вкладки обрабатываются разными Активностями, а не одной (достаточно громоздкой), что позволяет делать код более ясным и управляемым. В данной лабораторной работе мы используем вариант реализации с независимыми Активностями, поэтому для реализации трех вкладок нам потребуются четыре Активности (три для вкладок и одна «главная).

1. Создайте новый проект с именем TabWidgetSample.
2. Опишите нужные строки в файле res/values/strings.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">TabWidgetSample</string>
  <string name="tab1_indicator">Students</string>
  <string name="tab2_indicator">Teachers</string>
  <string name="tab3_indicator">Classes</string>
  <string name="tab1_content">This is Students tab</string>
  <string name="tab2_content">This is Teachers tab</string>
  <string name="tab3_content">This is Classes tab</string>
</resources>
```

3. Отредактируйте файл res/layout/main.xml:

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

```
xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@android:id/tabhost"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
      <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:padding="5dp">
      <TabWidget
        android:id="@android:id/tabs"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
      <FrameLayout
        android:id="@android:id/tabcontent"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:padding="5dp" />
      </LinearLayout>
      </TabHost>
```

4. Создайте три Активности с именами StudentsActivity, TeachersActivity и ClassesActivity.

5. 5. В каждой из созданных Активностей переопределите метод onCreate следующим образом (внеся соответствующие изменения в имя ресурса tabX_content):

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  Resources res = getResources();
  String contentText = res.getString(R.string.tab2_content);

  TextView textView = new TextView(this);
  textView.setText(contentText);
  setContentView(textView);
}
```

Обратите внимание на то, что в каждой Активности используются собственные строковые ресурсы.

6. Замените базовый класс для TabWidgetSampleActi-

vity с Activity на TabActivity и переопределите метод onCreate следующим образом:

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  Resources res = getResources();
  String tab1Indicator = res.getString(R.string.tab1_indicator);
  String tab2Indicator = res.getString(R.string.tab2_indicator);
  String tab3Indicator = res.getString(R.string.tab3_indicator);

  TabHost tabHost = getTabHost();
  TabHost.TabSpec spec;
  Intent intent;

  intent = new Intent().setClass(this, StudentsActivity.class);
  spec = tabHost.newTabSpec("students").setIndicator(tab1Indicator)
.setContent(intent); tabHost.addTab(spec); intent = new Intent().setClass(this, TeachersActivity.class);
  spec = tabHost.newTabSpec("teachers").setIndicator(tab2Indicator)
.setContent(intent); tabHost.addTab(spec);

  intent = new Intent().setClass(this, ClassesActivity.class);
  spec = tabHost.newTabSpec("class").setIndicator(tab3Indicator)
.setContent(intent); tabHost.addTab(spec);

  tabHost.setCurrentTab(1);
}
```

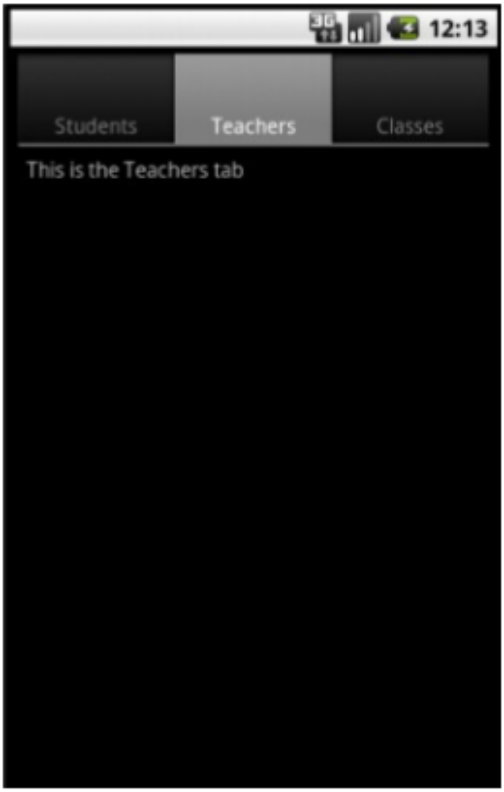
В этом методе мы используем для запуска нужных Активностей так называемые явные Намерения (Intent). Намерения и их использование будут рассмотрены позже.

7. При попытке запуска проекта на выполнение произойдет аварийное завершение работы приложения, так как Активности, созданные для отображения контента вкладок неизвестны системе, потому что не описаны в Манифесте приложения. Требуется отредактировать файл AndroidManifest.xml так, чтобы все используемые Активности были в нем описаны. Кроме того, для улучшения внешнего вида интерфейса имеет

смысл изменить используемую тему оформления на такую, где нет строки заголовка : Узел <application> должен выглядеть следующим образом:

```
<application android:icon="@drawable/ic_launcher"
 android:label="@string/app_name" andro-
id:theme="@android:style/Theme.NoTitleBar" >
<activity
 android:name=".TabWidgetSampleActivity"
 android:label="@string/app_name" >
<intent-filter>
 <action android:name="android.intent.action.MAIN" />
 <category andro-
id:name="android.intent.category.LAUNCHER" />
 </intent-filter>
</activity>
 <activity android:name=".StudentsActivity" />
 <activity android:name=".TeachersActivity" />
 <activity android:name=".ClassesActivity" />
</application>
```

8. Запустите приложение и поэкспериментируйте с вкладками:



9. Локализуйте приложение с помощью индивидуальных для разных языков строковых ресурсов.

5. ЛАБОРАТОРНАЯ РАБОТА №5: ИСПОЛЬЗОВАНИЕ WEBVIEW

5.1. Использование WebView

В данной лабораторной работе рассматривается использование виджета web-браузера и применяется итеративный подход к созданию приложения.

1. Создайте новый проект с именем WebViewSample.
2. Отредактируйте файл res/layout/main.xml:

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

```
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" />
```

3. Добавьте в конец метода onCreate Активности WebViewSampleActivity следующие строки:

```
WebView webView = (WebView) findViewById(R.id.webview);
        webView.getSettings().setJavaScriptEnabled(true);
  webView.loadUrl("http://www.ya.ru");
```

4. Запустите приложение:

5. Очевидно, что в супе чего-то не хватает у приложения отсутствуют полномочия на доступ к сети. Добавим нужные полномочия в Манифест приложения, добавив

элемент <uses-permission> внутри корневого узла <mainfest>:

<uses-permission android:name="android.permission.INTERNET"/>

6. Запустим проект (нижняя часть картинки отрезана):

7. Продолжим улучшать приложение. Для увеличения полезной площади экрана запретим показ заголовка, для этого укажем соответствующую тему в файле Манифеста:

```
<activity
 android:name=".WebViewSampleActivity"
 android:label="@string/app_name"
        android:theme="@android:style/Theme.NoTitleBar" >
```

8. Запустим проект и убедимся том, что (первая) цель достигнута.

9. 9. В настоящий момент ссылки, ведущие за пределы сайта www.ya.ru обслуживаются стандартным web-браузером, а не нашим WebView, так как оно не в состоянии обработать эти запросы и виджет webView автоматически посылает системе соответствующее Намерение (Intent), обрабатываемое стандартным браузером. У этой проблемы есть два решения:

• Добавить в Манифест нужный Фильтр Намерений (Intent Filter)

• Переопределить внутри нашей Активности класс Web-ViewClient, чтобы приложение могло обрабатывать свои собственные запросы на отображение web-ресурсов с помощью имеющегося виджета WebView.

Второй вариант является более приемлемым еще и потому, что в этом случае не будет рассматриваться системой как альтернативный web-браузер, что произошло бы в случае добавления Фильтра Намерений в Манифест.

10. Вынесем объект webView из метода onCreate и сделаем его членом класса, после чего добавим внутри Активности новый класс WebViewSampleClient, расширяющий класс WebViewClient , после чего установим свой обработчик запросов на отображение web-ресурсов:

```
public class WebViewSampleActivity extends Activity {
WebView webView;
@Override
  public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  webView = (WebView) findViewById(R.id.webview);
  webView.getSettings().setJavaScriptEnabled(true);
  webView.loadUrl("http://www.ya.ru");
  webView.setWebViewClient(new WebViewSampleClient());
  }
  private class WebViewSampleClient extends WebViewClient {
  @Override
  public boolean shouldOverrideUrlLoading(WebView view,
String url){
  view.loadUrl(url);       return true;
} } }
```

11. Запустим приложение и убедимся, что оставшийся серьезный недостаток – «неправильная» обработка нажатия кнопки «назад» устройства, приложение при этом заканчивает свою работу. Решение этой проблемы простое и прямолинейное: сделаем свой обработчик событий нажатия на кнопки, в котором будет реализовано только одно действие: если была нажата

кнопка «назад», виджет WebView получит команду вернуться на предыдущую страницу (если у него есть такая возможность). Переопределим метод onKeyDown в Активности:

```
@Override public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK) && web-View.canGoBack()){
        webView.goBack();
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
```

12. Запустим приложение и убедимся, что цель достигнута.

6. ЛАБОРАТОРНАЯ РАБОТА №6: ИСПОЛЬЗОВАНИЕ LISTVIEW

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 такого события, 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.

7. ЛАБОРАТОРНАЯ РАБОТА №7: ИСПОЛЬЗОВАНИЕ УПРАВЛЯЮЩИХ ЭЛЕМЕНТОВ В ПОЛЬЗОВАТЕЛЬСКОМ ИНТЕРФЕЙСЕ

7.1. Цель работы

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

7.2. 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 andro-id: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();
}
```

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

7.3. Использование виджета CheckBox

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.

7.4. 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.

7.5. 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.

7.6. 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"/>
```

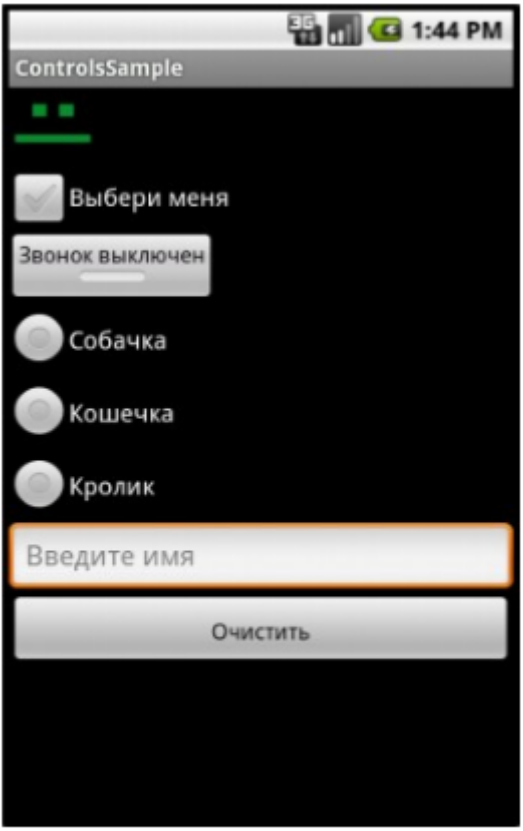
2. Для обработки введенного пользователем текста добавьте следующий код в конце метода onCreate. Обратите внимание, этот обработчик, в отличие от предыдущих, использованных нами, возвращает значение true или false. Семантика этих значений традиционна: true означает, что событие (event) обработано и больше 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) find-
ViewById(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. 4. Добавьте кнопку «Очистить» в разметку и напишите обработчик, очищающий текстовое поле (используйте метод setText виджета EditText):



5. Проверьте работу приложения.

8. ЛАБОРАТОРНАЯ РАБОТА №8: ВЫЗОВ АКТИВНОСТИ С ПОМОЩЬЮ ЯВНОГО НАМЕРЕНИЯ И ПОЛУЧЕНИЕ РЕЗУЛЬТАТОВ РАБОТЫ

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.

8.1. 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 только для 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.

Использование неявных Намерений

1. Измените проект MetroPicker так, чтобы для запуска Активности ListViewActivity использовалось неявное Намерение с действием (action) , определенным в вашем приложении и имеющем значение "com.example.metropicker.intent.action.PICK_METRO_STATION".
2. Проверьте работу приложения.

Определение Намерения, вызвавшего запуск Активности

Поскольку объекты типа Intent служат, в том числе, для передачи информации между компонентами одного или нескольких приложений, может возникнуть необходимость в работе с объектом Намерения, вызвавшим Активность к жизни.

Для получения доступа к этому объекту используется метод getIntent.

Пример: Intent intent = getIntent();

Имея данный объект, можно получить доступ к информации, содержащейся в нем:

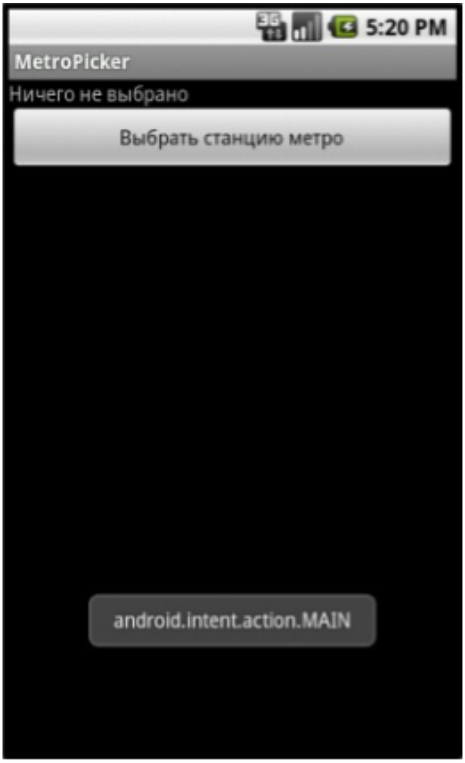
- метод getAction возвращает действие Намерения
- метод getData возвращает данные Намерения (обычно URI)
- набор методов для разных типов вида getТИПExtra позволяет получить доступ к типизированным значениям, хранящимся в свойстве extras Намерения.

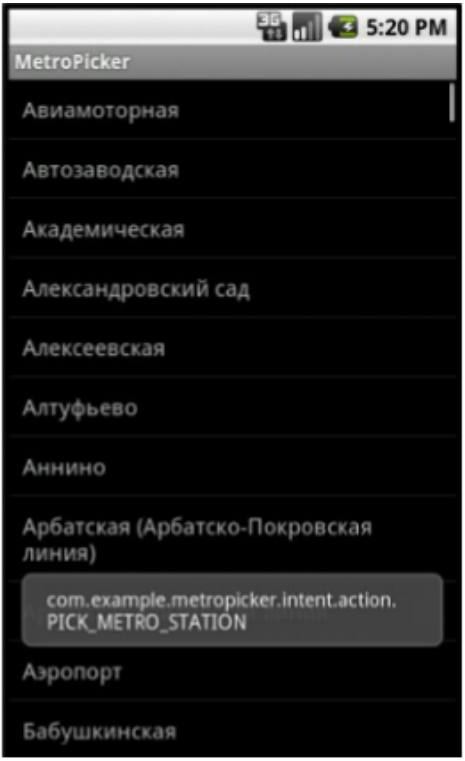
Примеры:

```
String action = intent.getAction();
Uri data = intent.getData();
```

Получение данных из Намерения

1. Модифицируйте методы onCreate двух ваших Активностей из предыдущей лабораторной работы так, чтобы с помощью Toast они показывали действие вызвавшего их Намерения .
2. Проверьте работу приложения:





Сохранение состояния и настроек приложения

Поскольку жизненный цикл приложений и Активностей в Android может прерваться в любой момент, для повышения привлекательности и удобства пользовательского интерфейса желательно иметь возможность сохранять (и восстанавливать!) состояния Активностей не только при выходе из активного состояния, но и между запусками. Android предлагает для этого следующие механизмы:

- Общие Настройки (Shared preferences): простой способ сохранения состояния UI и настроек приложения, использующий пары ключ/значение для хранения примитивных типов данных и обеспечения доступа к ним по имени.
- Сохранение состояния приложения: Для Активностей существуют обработчики событий, позволяющие сохранять и восстанавливать текущее состояние UI, когда выходит и

возвращается в активный режим. Для сохранения состояния используется объект класса Bundle, который передается методам onSaveInstanceState (для сохранения состояния), onCreate и on-RestoreInstanceState (для восстановления). Для доступа к данным в этом случае также используются пары ключ/значение. Обработчики событий из суперклассов берут на себя основную работу по сохранению и восстановлению вида UI, фокуса полей и т. д.

• Прямая работа с файлами. Если не подходят описанные выше варианты, приложение может напрямую читать и писать данные из файла. Для этого можно использовать как стандартные классы и методы Java, обеспечивающие ввод/вывод, так и методы openFileInput и openFileOutput, предоставленные Android, для упрощения чтения и записи потоков, относящихся к локальным файлам.

8.2. Общие Настройки (Shared Preferences)

Класс SharedPreferences предлагает методы для хранения и получения данных, записываемых в файлы, доступные по умолчанию только конкретному приложению. Такой способ хранения обеспечивает сохранность этих данных не только в течении жизненного цикла приложения, но и между запусками и даже перезагрузкой ОС.

Для сохранения данных в файле используется транзакционный механизм: вначале требуется получить объект класса SharedPreferences.Editor для конкретного файла настроек, после чего с помощью методов вида putТип этого объекта установить нужные значения. Запись значений производится методом commit. Примеры:

Сохранение данных:

```
private static final String PREFS = "PREFS";
static final String KEY_STATION = "selectedStation";
private SharedPreferences prefs;
private static final String NOTHING_SELECTED = "Ничего не выбрано";
private String selectedStation;
prefs = getSharedPreferences(PREFS, MODE_PRIVATE);
Editor editor = prefs.edit();
```

```
editor.putString(KEY_STATION, selectedStation);
editor.commit();
```

Получение данных:

```
prefs = getSharedPreferences(PREFS, MODE_PRIVATE);
selectedStation = prefs.getString(KEY_STATION, NOTHING_SELECTED); tv.setText(selectedStation);
```

Использование SharedPreferences для сохранения состояния

1. Модифицируйте методы onCreate и onActivityResult проекта MetroPicker для сохранения выбранной станции метро между запусками приложения.
2. Проверьте работоспособность приложения.

Использование SharedPreferences для сохранения настроек

1. Модифицируйте проект ControlsSample так, чтобы состояние управляющих элементов сохранялось и восстанавливалось между запусками приложения.
2. Проверьте работоспособность приложения.

9. ЛАБОРАТОРНАЯ РАБОТА №9: СОЗДАНИЕ И ИСПОЛЬЗОВАНИЕ МЕНЮ

9.1. Теоретическая часть

Меню в Android

Использование меню в приложениях позволяет сохранить ценное экранное пространство, которое в противном случае было бы занято относительно редко используемыми элементами пользовательского интерфейса.

Каждая Активность может иметь меню, реализующие специфические только для нее функции. Можно использовать также контекстные меню, индивидуальные для каждого Представления на экране.

Основы использования меню

В Android реализована поддержка трехступенчатой системы меню, оптимизированную, в первую очередь, для небольших экранов:

• Основное меню возникает внизу на экране при нажатии на кнопку «menu» устройства. Оно может отображать текст и иконки для ограниченного (по умолчанию, не больше шести) числа пунктов. Для этого меню рекомендуется использовать иконки с цветовой гаммой в виде оттенков серого с элементами рельефности. Это меню не может содержать радиокнопки и чекбоксы. Если число пунктов такого меню превышает максимально допустимое значение, в меню автоматически появляется пункт с надписью «еще» («more»). При нажатии на него отобразится Расширенное меню.

• Расширенное меню отображает прокручиваемый список, элементами которого являются пункты, не вошедшие в основное меню. В этом списке не могут отображаться иконки, но есть возможность отображения радиокнопок и чекбоксов. Поскольку не существует способа отобразить расширенное меню вместо основного, об изменении состояния каких-то компонентов приложения или системы рекомендуется уведомлять пользователя с помощью изменения иконок или текста пунктов меню.

• Дочернее меню (меню третьего уровня) может быть вызвано из основного или расширенного меню и отображается во всплывающем окне. Вложенность не поддерживается, и попытка вызвать из дочернего еще одно меню приведет к выбросу исключения.

9.2. Создание меню

При обращении к меню вызывается метод onCreateOptionsMenu Активности и для появления меню на экране его требуется переопределить. Данный метод получает в качестве параметра объект класса Menu, который в дальнейшем используется для манипуляций с пунктами меню.

Для добавления новых пунктов в меню используется метод add объекта Menu со следующими параметрами:

• Группа: для объедине- ния пунктов меню для группо-

вой обработки

104

• Идентификатор: уникальный идентификатор пункта меню. Этот идентификатор передается обработчику нажатия на пункт меню – методу onOptionsItemSelected.

• Порядок: значение, указывающее порядок, в котором пункты меню будут выводиться.

• Текст: надпись на данном пункте меню.

После успешного создания меню метод onCreateOptionsMenu должен вернуть значение true. Пример показывает создание меню из трех пунктов с использованием строковых ресурсов:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  super.onCreateOptionsMenu(menu);
  menu.add(0, Menu.FIRST, Menu.NONE,
R.string.menu_item1);
  menu.add(0, Menu.FIRST+1, Menu.NONE,
R.string.menu_item2);
  menu.add(0, Menu.FIRST+2, Menu.NONE,
R.string.menu_item3);

  return true;
  }
```

Для поиска пунктов меню по идентификатору можно использовать метод findItem объекта Menu.

Параметры пунктов меню

Наиболее полезными параметрами пунктов меню являются следующие:

• Краткие заголовки: используются в случае, если пункт может отобразиться в основном меню. Устанавливается методом setTitleCondensed объекта класса MenuItem: menuItem.setTitleCondensed("заголовок");

• Иконки: идентификатор Drawable, содержащий нужную картинку: menuItem.setIcon(R.drawable.menu_item_icon);

• Обработчик выбора пункта меню: установить можно, но

не рекомендуется из соображений повышения производительности, лучше использовать обработчик всего меню (onOptionsItemSelected). Тем не менее, пример:

```
menuItem.setOnMenuItemClickListener(new OnMenuItemClickListener() {
  public boolean onMenuItemClick(MenuItem _menuItem) {
  // обработать выбор пункта return true;
  }
});
```

• Намерение: это намерение автоматически передается методу startActivity, если нажатие на пункт меню не было обработано обработчиками onMenuItemClickListener и onOptionsItemSelected:

```
menuItem.setIntent(new Intent(this, MyOtherActivity.class));
```

Динамическое изменение пунктов меню

Непосредственно перед выводом меню на экран вызывается метод onPrepareOptionsMenu текущей Активности, и переопределяя его можно динамически изменять состояние пунктов меню: разрешать/запрещать, делать невидимым, изменять текст и т. д.

Для поиска пункта меню, подлежащего модификации можно использовать метод findItem объекта Menu, передаваемого в качестве параметра:

```
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
  super.onPrepareOptionsMenu(menu);
  MenuItem menuItem = menu.findItem(MENU_ITEM);
  //модифицировать пункт меню....
  return true;
}
```

Обработка выбора пункта меню

Android позволяет обрабатывать все пункты меню (выбор их) одним обработчиком onOptionsItemSelected . Выбранный пункт меню передается этому обработчику в качестве объекта класса

MenuItem.

Для реализации нужной реакции на выбор пункта меню требуется определить, что именно было выбрано. Для этого используется метод getItemId переданного в качестве параметра объекта, а полученный результат сравнивается с идентификаторами, использованными при добавлении пунктов в меню в методе onCreateOptionsMenu.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
  // Проверить каждый известный пункт
  case (MENU_ITEM):
  // сделать что-то...
            return true;
  }
  // Если пункт не обработан, отдаем обработку дальше
  return super.onOptionsItemSelected(item);
}
```

Дочерние и контекстные меню

При появлении на экране дочерние и контекстные меню выглядят одинаково, в виде плавающих окон, но при этом создаются по-разному.

Создание дочерних меню

Для создания дочерних меню используется метод addSubMenu объекта класса Menu:

```
SubMenu sub = menu.addSubMenu(0, 0, Menu.NONE, "дочернее меню"); sub.setHeaderIcon(R.drawable.icon);
sub.setIcon(R.drawable.icon);
MenuItem submenuItem = sub.add(0, 0, Menu.NONE, "пункт дочернего меню");
```

Как уже было сказано выше, вложенные дочерние меню Android не поддерживает.

Создание контекстных меню

Наиболее часто используемым способом создания контекстного меню в Android является переопределение метода onCreateContextMenu Активности:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenu.ContextMenuInfo menuInfo) {
  super.onCreateContextMenu(menu, v, menuInfo);
  menu.setHeaderTitle("Конекстное меню");
  menu.add(0, Menu.FIRST, Menu.NONE, "Пункт 1");
  menu.add(0, Menu.FIRST+1, Menu.NONE, "Пункт 2");
  menu.add(0, Menu.FIRST+2, Menu.NONE, "Пункт 3");
}
```

Регистрация обработчика контекстного меню для нужных Представлений осуществляется с помощью метода Активности registerForContextMenu:

```
@Override
public void onCreate(Bundle savedInstanceState) { s
  uper.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  tv = (TextView) findViewById(R.id.text_view);
  registerForContextMenu(tv);
}
```

Пример обработчика:

```
@Override
public boolean onContextItemSelected(MenuItem item) {
  switch (item.getItemId()) {
  case DELETE_ID:
        AdapterContextMenuInfo info = (AdapterContextMen-
uInfo) item .getMenuInfo();          db.deleteItem(info.id);
        populate();
        return true;
  }
  return super.onContextItemSelected(item);
}
```

Описание меню с помощью XML

Часто бывает удобнее всего описывать меню, в том числе иерархические, в виде ресурсов. О преимуществах такого подхода говорилось выше.

Меню традиционно описываются и хранятся в каталоге res/menu проекта. Все иерархии меню (если есть иерархические меню) должны находиться в отдельных файлах, а имя файла будет использовать как идентификатор ресурса. Корневым элементом файла должен быть тэг <menu>, а пункты меню описываются тэгом <item>. Свойства пунктов меню описываются соответствующими атрибутами:

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

<item
  android:id="@+id/item01"
  android:icon="@drawable/menu_item"
  android:title="Пункт 1">
</item>
<item
  android:id="@+id/item02"
  android:checkable="true"
  android:title="Пункт 2">
</item>
<item
  android:id="@+id/item03"
  android:title="Пункт 3">
</item>
<item
  android:id="@+id/item04"
  android:title="Дочернее меню 1">
  <menu>
        <item
              android:id="@+id/sub1item01"
              android:title="Пункт дочернего меню 1">
        </item>
  </menu>
</item>
<item
  android:id="@+id/item05"
```

```
  android:title="Дочернее меню 2">
  <menu>
        <item
              android:id="@+id/sub2item01"
              android:title="Пункт дочернего меню 2">
        </item>
  </menu>
</item>
</menu>
```

Для создания объектов Menu из ресурсов в событий onCreateOptionsMenu и onCreateContextMenu используется метод inflate объекта типа MenuInflater:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  super.onCreateOptionsMenu(menu);
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.menu1, menu);
   return true;
}
```

или так:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
        ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);

        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu2, menu);

        menu.setHeaderTitle("Контекстное меню");
}
```

9.3. Задание к лабораторной работе

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

10. ЛАБОРАТОРНАЯ РАБОТА №10: РАБОТА С SQLITE БЕЗ КЛАССА-АДАПТЕРА И С КЛАССОМ-АДАПТЕРОМ

Цель данной лабораторной работы – освоить взаимодействие с СУБД SQLite без применения специальных классов-адаптеров и создание простого приложения, позволяющего создавать, хранить и удалять короткие заметки.

10.1. Теоретическая часть

Работа с базами данных в Android

Механизм работы с базами данных в Android позволяет хранить и обрабатывать структурированную информацию. Любое приложение может создавать свои собственные базы данных, над которыми оно будет иметь полный контроль.

В Android используется библиотека SQLite, представляющую из себя реляционную СУБД, обладающую следующими характерными особенностями: свободно распространяемая (open source), поддерживающая стандартный язык запросов и транзакции, легковесная, одноуровневая (встраиваемая), отказоустойчивая.

Курсоры (Cursor) и ContentValues

Запросы к СУБД возвращают объекты типа Cursor. Для экономии ресурсов используется подход, когда при извлечении данных не возвращаются копии их значений из СУБД, а создается Cursor, обеспечивающий навигацию и доступ к запрошенному набору исходных данных. Методы объекта Cursor предоставляют

различные возможности навигации, назначение которых, как правило, понятно из названия:

- moveToFirst
- moveToNext
- moveToPrevious
- getCount
- getColumnIndexOrThrow
- getColumnName
- getColumnNames
- moveToPosition
- getPosition

При добавлении данных в таблицы СУБД применяются объекты класса ContentValues. Каждый такой объект содержит данные одной строки в таблице и, по сути, является ассоциативным массивом с именами столбцов и соответствующими значениями.

Работа с СУБД SQLite

При создании приложений, использующих СУБД, во многих случаях удобно применять инструменты, называющиеся ORM (Object-Relationship Mapping), отображающие данные из одной или нескольких таблиц на объекты используемого языка программирования. Кроме того, ORM позволяют абстрагироваться от конкретной реализации и структуры таблиц и берут на себя обязанности по взаимодействию с СУБД. К сожалению, в силу ограниченности ресурсов мобильной платформы ORM в настоящий момент в Android практически не применяется. Тем не менее, разумным подходом при разработке приложения будет инкапсуляция всех взаимодействий с СУБД в одном классе, методы которого будут предоставлять необходимые услуги остальным компонентам приложения.

Хорошей практикой является создание вспомогательного класса, берущего на себя работу с СУБД. Данный класс обычно инкапсулирует взаимодействия с базой данных, предоставляя интуитивно понятный строго типизированный способ удаления, добавления и изменения объектов. Такой Адаптер базы данных также должен обрабатывать запросы к БД и переопределять методы для открытия, закрытия и создания базы данных. Его

также обычно используют как удобное место для хранения статических констант, относящихся к базе данных, таких, например, как имена таблиц и полей. Ниже показан пример каркаса для реализации подобного Адаптера:

```
public class SampleDBAdapter {
  private static final String DATABASE_NAME = "SampleDatabase.db";
  private static final String DATABASE_TABLE = "SampleTable";
  private static final int DATABASE_VERSION = 1;

  // Имя поля индекса для
        public static final String KEY_ID = "_id";
  // Название и номер п/п (индекс) каждого поля
        public static final String KEY_NAME = "name";
        public static final int NAME_COLUMN = 1;
  // Для каждого поля опишите константы аналогичным образом...
  // SQL-запрос для создания БД
        private static final String DATABASE_CREATE = "create table "
DATABASE_TABLE + " (" + KEY_ID
+ " integer primary key autoincrement, " + KEY_NAME
+ " text not null);";
// Переменная для хранения объекта БД
private SQLiteDatabase db;

  // Контекст приложения для
  private final Context context;
  // Экземпляр вспомогательного класса для открытия и обновления БД
  private myDbHelper dbHelper;
  // Конструктор public SampleDBAdapter(Context _context) {
  context = _context;
  dbHelper = new myDbHelper(context, DATABASE_NAME, null,
        DATABASE_VERSION);
}

  // «Открывашка» БД
  public SampleDBAdapter open() throws SQLException {
  try {
        db = dbHelper.getWritableDatabase();
```

```
        } catch (SQLiteException e) {
                db = dbHelper.getReadableDatabase();
        } return this;
    }

        // Метод для закрытия БД
        public void close() {
        db.close();
        }

        // Метод для добавления данных, возвращает индекс
        // свежевставленного объекта
        public long insertEntry(SampleObject _SampleObject) {
        // Здесь создается объект ContentValues, содержащий
        // нужные поля и  производится вставка return index;
        }

        // Метод для удаления строки таблицы по индексу
        public boolean removeEntry(long _rowIndex) {
        return db.delete(DATABASE_TABLE, KEY_ID + "=" +
_rowIndex, null) > 0;
        }

        // Метод для получения всех данных.
        // Возвращает курсор, который можно использовать для
        // привязки к адаптерам типа SimpleCursorAdapter
        public Cursor getAllEntries() {
        return db.query(DATABASE_TABLE, new String[] { KEY_ID,
KEY_NAME },
                                                null, null, null,
null, null);
    }

        // Возвращает экземпляр объекта по индексу
        public SampleObject getEntry(long _rowIndex) {
        // Получите курсор, указывающий на нужные данные из
БД
        // и создайте новый объект, используя этими данными
        // Если ничего не найдено, верните null
        return objectInstance;
    }
        // Изменяет объект по индексу
```

```
        // Увы, это не ORM :(
                public boolean updateEntry(long _rowIndex,
SampleObject _SampleObject) {
                // Создайте объект ContentValues на основе
свойств SampleObject
                // и используйте его для обновления строки в
таблице return true;
                // Если удалось обновить, иначе false :)
        }
                // Вспомогательный класс для открытия и
обновления БД
                private static class myDbHelper extends SQLiteOpen-
Helper {
                        public myDbHelper(Context context, String
name,
                        CursorFactory factory, int version) {
                                super(context, name, factory, versi-
on);
                        }

                        // Вызывается при необходимости создания
БД
                        @Override
                        public void onCreate(SQLiteDatabase _db) {
                                _db.execSQL(DATABASE_CREATE);
                        }

                        // Вызывается для обновления БД, когда
текущая версия БД
                        // в приложении новее, чем у БД на диске
                        @Override
                        public void onUpgrade(SQLiteDatabase _db,
int _oldVersion,
                                int _newVersion) {
                        // Выдача сообщения в журнал, полезно
при отладке
          Log.w("TaskDBAdapter", "Upgrading from version " +
_oldVersion
        + " to " + _newVersion
        + ", which will destroy all old data");

                        // Обновляем БД до новой версии.
```

```
                    // В простейшем случае убиваем старую БД
                    // и заново создаем новую.
                    // В реальной жизни стоит подумать о
пользователях
                    // вашего приложения и их реакцию на
потерю старых данных.
        _db.execSQL("DROP TABLE IF EXISTS " +
DATABASE_TABLE);                    onCreate(_db);
                }
        }
    }
```

Работа с СУБД без адаптера

При нежелании использовать класс SQLiteOpenHelper (и вообще адаптер БД) можно воспользоваться методом openOrCreateDatabase контекста приложения:

```
    private static final String DATABASE_NAME = "myDatabase.db";
    private static final String DATABASE_TABLE = "mainTable";
    private static final String DATABASE_CREATE = "create table "
    DATABASE_TABLE + " ( _id integer primary key autoincrement,"
    + "column_one text not null);";
    SQLiteDatabase myDatabase;

    private void createDatabase() {
      myDatabase = openOrCreateDatabase(DATABASE_NAME,
MODE_PRIVATE,          null);
      myDatabase.execSQL(DATABASE_CREATE);
    }
```

В этом случае можно работать с базой данных с помощью, например, метода execSQL экземпляра БД, как показано на примере выше.

Особенности работы с БД в Android

При работе с базами данных в Android следует избегать хранения BLOB'ов с таблицах изза резкого падения эффективности работы.

Как показано в примере адаптера БД, для каждой таблицы рекомендуется создавать автоикрементное поле _id, которое будет уникальным индексом для строк. Если же планируется делегировать доступ к БД с помощью контент-провайдеров, такое поле является обязательным.

Выполнение запросов для доступа к данным

Для повышения эффективности использования ресурсов мобильной платформы запросы к БД возвращают объект типа Cursor, используемый в дальнейшем для навигации и получения значений полей. Выполнение запросов осуществляется с помощью метода query экземпляра БД, параметры которого позволяют гибко управлять критериями выборки:

```
// Получить поля индекса, а также первое и третье из таблицы,
// без дубликатов String[] result_columns = new String[] {KEY_ID, KEY_COL1, KEY_COL3}; Cursor allRows = myDatabase.query(true, DATABASE_TABLE, result_columns,
null, null, null, null, null, null);

// Получить все поля для строк, где третье поле равно требуемому
// значению,
// результат отсортировать по пятому полю
String where = KEY_COL3 + "=" + requiredValue;
String order = KEY_COL5;
Cursor myResult = myDatabase.query(DATABASE_TABLE, null, where, null, null, null, order);
```

Доступ к результатам с помощью курсора

Для получения результатов запроса необходимо установить курсор на нужную строку с помощью методов вида moveToМестоположение, перечисленных выше. После этого используются типизированные методы getТип, получающие в качестве параметра индекс (номер) поля в строке. Как правило, эти значения являются статическими константами адаптера БД:

```
String columnValue = myResult.getString(columnIndex);
```

В примере ниже показано, как можно просуммировать все поля (типа float) из результатов выполнения запроса (и получить среднюю сумму счета):

```
int KEY_AMOUNT = 4;
Cursor myAccounts = myDatabase.query("my_bank_accounts", null, null, null, null, null, null); float totalAmount = 0f;

// Убеждаемся, что курсор непустой
if (myAccounts.moveToFirst()) {
  // Проходимся по каждой строке
        do {
        float amount = myAccounts.getFloat(KEY_AMOUNT);
        totalAmount += amount;
        } while(myAccounts.moveToNext());
}

float averageAmount = totalAmount / myAccounts.getCount();
```

Изменение данных в БД

В классе SQLiteDatabase, содержащем методы для работы с БД, имеются методы insert, update и delete, которые инкапсулируют операторы SQL, необходимые для выполнения соответствующих действий. Кроме того, метод execSQL позволяет выполнить любой допустимый код SQL(если вы, например, захотите увеличить долю ручного труда при создании приложения). Следует помнить, что при наличии активных курсоров после любого изменения данных следует вызывать метод refreshQuery для всех курсоров, которые имеют отношение к изменяемым данным (таблицам).

Вставка строк

Метод insert хочет получать (кроме других параметров) объект ContentValues, содержащий значения полей вставляемой строки и возвращает значение индекса:

```
ContentValues newRow = new ContentValues();
// Выполним для каждого нужного поля в строке
newRow.put(COLUMN_NAME, columnValue);
db.insert(DATABASE_TABLE, null, newRow);
```

Обновление строк

Также используется ContentValues, содержащий подлежащие изменению поля, а для указания, какие именно строки нужно изменить, используется параметр where, имеющий стандартный для SQL вид:

```
ContentValues updatedValues = new ContentValues();

// Повторяем для всех нужных полей
updatedValues.put(COLUMN_NAME, newValue);

// Указываем условие
String where = COLUMN_NAME + "=" + "'Бармаглот'";

// Обновляем
db.update(DATABASE_TABLE, updatedValues, where, null);
```

Метод update возвращает количество измененных строк. Если в качестве параметра where передать null, будут изменены все строки таблицы.

Удаление строк

Выполняет похожим на update образом:

```
db.delete(DATABASE_TABLE, KEY_ID + "=" + rowId, null);
```

Лабораторная работа «работа с SQLite без класса-адаптера»

Цель данной лабораторной работы – освоить взаимодействие с СУБД SQLite без применения специальных классов-адаптеров.

Задание 1

1. Создайте новый проект SQLTest, главная Активность которого будет расширять класс ListActivity.
2. В методе onCreate главной Активности сделайте открытие или создание БД, используя в качестве основы информацию из пункта «Работа с СУБД без адаптера» и метода onUpgrade вспомогательного класса из предшествующего ему

пункта. После создания БД создайте таблицу с любым полем (плюс индекс), в которую добавьте 3..5 уникальных записей со строковым полем. Индекс должен инкрементироваться автоматически.

3. В этом же методе сделайте запрос к таблице, получающий все строки и с помощью имеющегося курсора запишите данные в массив строк.

4. С помощью дополнительной разметки и адаптера ArrayAdapter отобразите полученные из СУБД данные на экране Активности.

5. Добавьте обработчик клика на элемент списка, чтобы при клике запрашивался из БД индекс элемента с этой строкой и отображался с помощью Toast.

6. Добавьте контекстное меню к элементам списка, содержащее пункт «удалить» и реализуйте удаление. После удаления должны производится действия из п.п. 3 и 4 для изменения состава отображаемых данных.

Использование SimpleCursorAdapter

SimpleCursorAdapter позволяет привязать курсор к ListView. используя описание разметки для отображения строк и полей. Для однозначного определения того, в каких элементах разметки какие поля, получаемые через курсор, следует отображать, используются два массива: строковый с именами полей строк, и целочисленный с идентификаторами элементов разметки:

```
Cursor cursor = [ . . . запрос к БД . . . ];

String[] fromColumns = new String[] {KEY_NAME, KEY_NUMBER};
int[] toLayoutIDs = new int[] { R.id.nameTextView, R.id.numberTextView};

SimpleCursorAdapter myAdapter;
myAdapter = new SimpleCursorAdapter(this, R.layout.item_layout, cursor, fromColumns, toLayoutIDs);

myListView.setAdapter(myAdapter);
```

работа с SQLite с классом-адаптером

Задание 2

1. Создайте новый проект NotesSample.
2. Для простоты использования все действия с записями будут производиться в рамках одной Активности, поэтому используйте максимально упрощенный интерфейс, как показано на следующей странице. Для реализации можно воспользоваться такой разметкой в файле res/layout/main.xml:

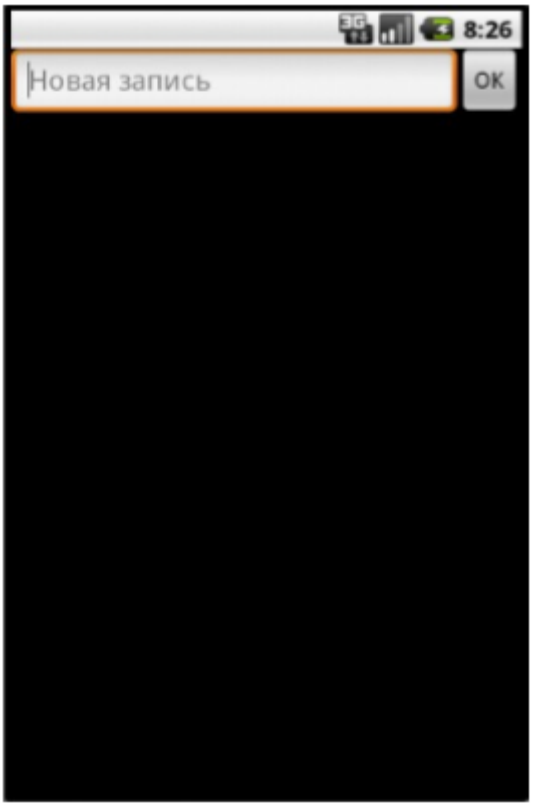
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >

  <RelativeLayout
      android:layout_width="fill_parent"
      android:layout_height="wrap_content" >

      <Button
          android:id="@+id/save_button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_alignParentRight="true"
          android:text="@android:string/ok" />

      <EditText
          android:id="@+id/edit_text"
          android:layout_width="fill_parent"
          android:layout_toLeftOf="@id/save_button"
          android:layout_height="wrap_content"
          android:hint="Новая запись"/>
  </RelativeLayout>
  <ListView
          android:id="@+id/myListView"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content" />

  </LinearLayout>
```



3. Реализуйте сохранение записей по нажатию на кнопку, а удаление через контекстное меню. Работа с СУБД должна осуществляться с использованием адаптера. Пример обработчика выбора пункта контекстного меню приведен в соответствующем разделе данного методического руководства.

10.2. Создание контент-провайдера

Цель данной лабораторной работы – получить навыки создания собственных контентпровайдеров. В качестве основы возьмите упражнение «работа с SQLite с классомадаптером» и замените прямое обращение к СУБД на взаимодействие с контентпровайдером, который будет инкапсулировать все взаимодействие с «настоящими» данными в отдельном классе.

Теория

Контент-провайдеры

Контент-провайдеры предоставляют интерфейс для публикации и потребления структурированных наборов данных, основанный на URI с использованием простой схемы content://. Их использование позволяет отделить код приложения от данных, делая программу менее чувствительной к изменениям в источниках данных.

Для взаимодействия с контент-провайдером используется уникальный URI, который обычно формируется следующим образом:

content://<домен-разработчика-наоборот>.provider.<имя-приложения>/<путь-к-даным>

Классы, реализующие контент-провайдеры, чаще всего имеют статическую строковую константу CONTENT_URI, которая используется для обращения к данному контентпровайдеру.

Контент-провайдеры являются единственным способом доступа к данным других приложений и используются для получения результатов запросов, обновления, добавления и удаления данных. Если у приложения есть нужные полномочия, оно может запрашивать и модифицировать соответствующие данные, принадлежащие другому приложению, в том числе данные стандартных БД Android. В общем случае, контент-провайдеры следует создавать только тогда, когда требуется предоставить другим приложениям доступ к данным вашего приложения. В остальных случаях рекомендуется использовать СУБД (SQLite). Тем не менее, иногда контент-провайдеры используются внутри одного приложения для поиска и обработки специфических запросов к данным.

Использование контент-провайдеров

Для доступа к данным какого-либо контент-провайдера используется объект класса ContentResolver, который можно получить с помощью метода getContentResolver контекста приложения для связи с поставщиком в качестве клиента:

ContentResolver cr = getApplicationContext().getContentResolver();

Объект ContentResolver взаимодействует с объектом контент-провайдера, отправляя ему запросы клиента. Контент-провайдер обрабатывает запросы и возвращает результаты обработки.

Контент-провайдеры представляют свои данные потребителям в виде одной или нескольких таблиц подобно таблицам реляционных БД. Каждая строка при этом является отдельным «объектом» со свойствами, указанными в соответствующих именованных полях. Как правило, каждая строка имеет уникальный целочисленный индекс и именем «_id», который служит для однозначной идентификации требуемого объекта.

Контент-провайдеры, обычно предоставляют минимум два URI для работы с данными: один для запросов, требующих все данные сразу, а другой – для обращения к конкретной «строке». В последнем случае в конце URI добавляется /<номер-строки> (который совпадает с индексом «_id»).

Запросы на получение данных

Запросы на получение данных похожи на запросы к БД, при этом используется метод query объекта ContentResolver. Ответ также приходит в виде курсора, «нацеленного» на результирующий набор данных (выбранные строки таблицы):

```
ContentResolver cr = getContentResolver();

// получить данные всех контактов Cursor c =
cr.query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

// получить все строки, где третье поле имеет конкретное
// значение и отсортировать по пятому полю
String where = KEY_COL3 + "=" + requiredValue;
String order = KEY_COL5;
```

Cursor someRows = cr .query(MyProvider.CONTENT_URI, null, where, null, order);

Извлечение данных-результатов запроса с помощью курсора было рассмотрено ранее.

Изменение данных

Для изменения данных применяются методы insert, update и delete объекта ContentResolver. Для массовой вставки также существует метод bulkInsert.

Пример добавления данных:

```
ContentResolver cr = getContentResolver();

ContentValues newRow = new ContentValues();
// повторяем для каждого поля в строке:

newRow.put(COLUMN_NAME, newValue);

Uri myRowUri = cr.insert(SampleProvider.CONTENT_URI, newRow);

// Массовая вставка:
ContentValues[] valueArray = new ContentValues[5];

// здесь заполняем массив

// делаем вставку
int count = cr.bulkInsert(MyProvider.CONTENT_URI, valueArray);
```

При вставке одного элемента метод insert возвращает URI вставленного элемента, а при массовой вставке возвращается количество вставленных элементов.

Пример удаления:

```
ContentResolver cr = getContentResolver();
// удаление конкретной строки cr.delete(myRowUri, null, null);
```

```
// удаление нескольких строк
String where = "_id < 5";
cr.delete(MyProvider.CONTENT_URI, where, null);
```

Пример изменения:

```
ContentValues newValues = new ContentValues();
 newValues.put(COLUMN_NAME, newValue);
String where = "_id < 5";
getContentResolver().update(MyProvider.CONTENT_URI,
newValues, where, null);
```

Задание

Для чтения информации о контактах используется контент-провайдер ContactsContract, точнее, один из его подклассов. Для этой лабораторной работы воспользуемся провайдером ContactsContract.Contacts. Для чтения контактов приложению требуются полномочия READ_CONTACTS.

1. Добавьте несколько контактов в эмуляторе (поскольку требуется только отображаемое имя контакта, остальные поля можно не заполнять :).
2. Создайте новый проект ContactsSample.
3. Выведите имена всех контактов (с помощью ListView), используя для получения информации URI ContactsContract.Contacts.CONTENT_URI. Необходимое имя поля для привязки адаптера найдите среди статических констант класса ContactsContract.Contacts.

11. ЛАБОРАТОРНАЯ РАБОТА №11: ИСПОЛЬЗОВАНИЕ СЕТЕВЫХ СЕРВИСОВ

11.1. Цель работы

Целью данной работы является создание простого приложения, получающего курсы иностранных валют по отношению к рублю с сайта ЦБ РФ в формате XML и отображающего данные в виде списка (ListView).

11.2. Создание контент-провайдеров

Для создания собственного контент-провайдера требуется расширить класс ContentProvider и переопределить метод onCreate, чтобы проинициализировать источник данных, который требуется опубликовать. Остальные методы этого класса будут, по сути, обертками вокруг методов работы с исходным источником данных. Каркас для класса показан ниже:

```
public class NewProvider extends ContentProvider {
  public final static Uri CONTENT_URI=Uri.parse("URI провайдера");
  @Override
  public int delete(Uri uri, String selection, String[] selectionArgs) {
  // Удаление данных return 0;
  }

  @Override
  public String getType(Uri uri) {
        // Возвращает тип MIME для указанных объектов(объекта)
        return null;
  }

  @Override
  public Uri insert(Uri uri, ContentValues values) {
        // Добавление данных
        return null;
  }

  @Override
  public boolean onCreate() {
        // Инициализация источника данных
        return false;
  }

  @Override
  public Cursor query(Uri uri, String[] projection,
        String selection, String[] selectionArgs, String sortOrder) {
        // Стандартный запрос return null;
        }
```

```
    @Override
    public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
        // Обновление данных
        return 0;
    }
}
```

Как говорилось выше, URI обычно выглядят примерно так:

content://<домен-разработчика-наоборот>.provider.<имя-приложения>/<путь-к-даным>

Традиционно URI должны быть представлены двумя способами, одним – для доступа ко всем значениям определенного типа, другой – для указания на конкретный экземпляр данных.

Например, URI content://com.android.provider.metropicker/stations мог бы использоваться для получения списка всех станций (метро), а

content://com.android.provider.metropicker/stations/17 – для станции с индексом 17.

При создании контент-провайдера обычно применяется статический объект класса UriMatcher, который служит для определения деталей запроса к контент-провайдеру. Использование UriMatcher позволяет «разгрузить» логику программы и избежать множественного сравнения строковых объектов за счет централизованного «отображения» URI разных видов на целочисленные константы. Особенно он полезен в случаях, когда создаваемый контент-провайдер обслуживает разные URI для доступа к одному и тому же источнику данных. Использовать UriMatcher очень просто: внутри класса контентпровайдера можно добавить такой код:

```
// Константы для разных типов запросов
private static final int ALL_STATIONS = 1;
private static final int SINGLE_STATION = 2;
private static final UriMatcher uriMatcher;
// Заполнение UriMatcher'a
```

```
// Если URI оканчивается на /stations - это запрос про все станции
// Если на stations/[ID] - про конкретную станцию

static {
  uriMatcher = new UriMatcher(UriMatcher.NO_MATCH); uriMatcher.addURI("com.example.provider.metropicker", "stations", ALL_STATIONS); uriMatcher.addURI("com.example.provider.metropicker", "stations/#",SINGLE_STATION); }
```

В дальнейшем полученный в запросе к контент-провайдеру URI проверяется в методах класса следующим образом:

```
@Override
public Cursor query(Uri uri, String[] projection, String selection,
  String[] selectionArgs, String sortOrder) {
  switch (uriMatcher.match(uri)) {
  case ALL_STATIONS:
        // Вернуть курсор, указывающий на выборку со
        // всеми станциями
  case SINGLE_STATION:
        // Вытащить ID станции из URI:
        String _id = uri.getPathSegments().get(1);

        // Вернуть курсор, указывающий на выборку с одной
        // станцией.
  }
  return null;
}
```

При наполнении объекта UriMatcher шаблонами могут применять в «#» и «*» в качестве специальных символов: # в шаблоне совпадает с любым числом, а * - с любым текстом.

Кроме уникальных URI, используемым создаваемым контент-провайдером, для определения клиентами типа возвращаемых по запросу данных используются уникальные типы MIME. Метод getType класса ContentProvider обычно возвращает один тип данных для массовой выборки, а другой – для одиночных записей. В нашем случае это могло бы выглядеть так:

```
@Override
public String getType(Uri uri) {
  switch (uriMatcher.match(uri)) {
  case ALL_STATIONS:
        return "vnd.com.example.cursor.dir/station";
  case SINGLE_STATION:
        return "vnd.com.example.cursor.item/station";
  default:
        throw new IllegalArgumentException("Unsupported URI: " + uri);
  }
}
```

Для того, что бы Android знал о существовании и мог использовать (через ContentResolver) ваш контент-провайдер, его нужно описать в Манифесте приложения. Минимальное описание выглядит так:

```
<provider
  android:name=".NewProvider"
  android:authorities="com.android.provider.metropicker" >
</provider>
```

В данном случае указаны только обязательные атрибуты элемента <provider>: имя класса контент-провайдера и его область ответственности. Более полную информацию о возможных атрибутах можно получить на сайте developer.android.com: http://developer.android.com/guide/topics/manifest/provider-element.html

11.3. Использование интернет-сервисов

Помимо простого отображения web-контента с помощью виджета WebView, разработчик имеет возможность «низкоуровневой работы с разнообразными сетевыми сервисами. Для этого всего лишь требуется создать сетевое подключение (запрос к серверу), получить, обработать и отобразить данные к нужном виде. Традиционно наиболее удобными форматами для сетевого обмена данными считаются XML и JSON.

Разумеется, любое приложение, использующее сетевые подключения, должно иметь в Манифесте соответствующие полномочия:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Для создания потока данных от сервера можно использовать класс HttpUrlConnection, являющийся расширением класса UrlConnection из пакета java.net. Пакеты java.net и android.net содержат классы, позволяющие управлять сетевыми соединениями. Более подробную информацию о классе HttpUrlConnection с примерами использования можно получить здесь: http://developer.android.com/reference/java/net/HttpURLConnection.html

Простой пример создания соединения:

```
private static final String some_url = "....";
. . . .
try {
  // Создаем объект типа URL
  URL url = new URL(getString(R.string.rates_url));

  // Соединяемся HttpURLConnection httpConnection = (HttpURLConnection) url .openConnection();

  // Получаем код ответа
  int responseCode = httpConnection.getResponseCode();

  // Если код ответа хороший, парсим поток(ответ сервера)
  if (responseCode == HttpURLConnection.HTTP_OK) {
        // Если код ответа хороший, обрабатываем ответ
        InputStream in = httpConnection.getInputStream();
  } else {
        // Сделать извещения об ошибках, если код ответа нехороший
  }

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

```
            e.printStackTrace();
        }
    }
```

Для получения данных будет использоваться URL http://www.cbr.ru/scripts/XML_daily.asp

Ответ сервера выглядит примерно так: <ValCurs Date="04.04.2012" name="Foreign Currency Market"> <Valute ID="R01010"> <NumCode>036</NumCode> <CharCode>AUD</CharCode> <Nominal>1</Nominal> <Name>Австралийский доллар</Name> <Value>30,4632</Value> </Valute> . . . . . </ValCurs>

11.4. Задание

Для каждой валюты (элемент Valute) потребуется извлечь значения дочерних элементов CharCode, Nominal, Name и Value. Значение атрибута Date корневого элемента (ValCurs) будет использоваться для изменения заголовка приложения.

1. Создайте новый проект CurrencyRates. Главная (и единственная) Активность с таким же именем (CurrencyRates) должна расширять класс ListActivity.
2. Отредактируйте Манифест приложения, добавив в него необходимые полномочия и тему для Активности (android:theme="@android:style/Theme.Light").
3. Файл строковых ресурсов strings.xml (в каталоге res/values) отредактируйте следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">Курсы ЦБ РФ</string>
  <string
name="rates_url">http://www.cbr.ru/scripts/XML_daily.asp</string>
</resources>
```

4. Для отображения информации требуется создать разметку для элементов списка. В каталоге res/layout создайте файл item_view.xml со следующим содержимым:

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

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="horizontal" >
 <TextView
  android:id="@+id/charCodeView"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:background="#FF8"
  android:minWidth="45sp"
  android:padding="4dp"
  android:textColor="#00F"
  android:textStyle="bold"
  android:gravity="center"
  android:shadowDx="8"
  android:shadowDy="8"
  android:shadowColor="#000"
  android:shadowRadius="8"/>
 <TextView
  android:id="@+id/valueView"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:textColor="#008"
  android:background="#FFE"
  android:minEms="3"
  android:padding="3dp" />
 <TextView
  android:id="@+id/nominalView"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:padding="3dp" />
<TextView
  android:id="@+id/nameView"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:ellipsize="marquee"
  android:singleLine="true" />
</LinearLayout>
```

5. Вся логика приложения будет сосредоточена в классе CurrencyRates, поэтому остальные изменения будут

касаться только этого класса. Добавьте необходимые константы:

```
private final static String KEY_CHAR_CODE = "CharCode";
private final static String KEY_VALUE = "Value";
private final static String KEY_NOMINAL = "Nominal";
private final static String KEY_NAME = "Name";
```

6. Тело метода onCreate должно содержать только две строки: super.onCreate(savedInstanceState);
populate();

Поскольку CurrencyRates является наследником ListActivity, вызов setContentView не требуется. Метод populate будет наполнять ListView содержимым с помощью адаптера (SimpleAdapter), заполнив его данными, полученными от метода getData.

7. Добавьте метод populate, в котором создается и настраивается адаптер:

```
private void populate() {
  ArrayList<Map<String, String>> data = getData();

  String[] from = { KEY_CHAR_CODE, KEY_VALUE, KEY_NOMINAL, KEY_NAME };
  int[] to = { R.id.charCodeView, R.id.valueView, R.id.nominalView,
               R.id.nameView };

  SimpleAdapter sa = new SimpleAdapter(this, data, R.layout.item_view, from, to);

  setListAdapter(sa);
}
```

8. Добавьте метод getData. Именно в нем будет создаваться и обрабатываться соединение с сервером, а также анализироваться XML-данные и заполняться список, который будет отображаться адаптером. Метод getData объемнее остальных, но ничего сложного в нем нет (стоит отметить, что интерфейсы Document, Element и NodeList должны импортироваться из пакета org.w3c.dom):

```
private ArrayList<Map<String, String>> getData() {
  ArrayList<Map<String, String>> list =
  new ArrayList<Map<String, String>>();
```

```
Map<String, String> m;
try { // Создаем объект URL
        URL url = new URL(getString(R.string.rates_url));
        // Соединяемся
        HttpURLConnection httpConnection =
                (HttpURLConnection) url.openConnection();
                // Получаем от сервера код ответа
                        int responseCode = httpConnec-
tion.getResponseCode();
                // Если код ответа хороший, парсим
поток(ответ сервера),
                // устанавливаем дату в заголовке
приложения и
                // заполняем list нужными Map'ами
        if (responseCode == HttpURLConnection.HTTP_OK) {
                InputStream in = httpConnec-
tion.getInputStream();
                DocumentBuilderFactory dbf = DocumentBuil-
derFactory .newInstance();

        DocumentBuilder db = dbf.newDocumentBuilder();
        Document dom = db.parse(in);
        Element docElement = dom.getDocumentElement();
        String date = docElement.getAttribute("Date");
        setTitle(getTitle() + " на " + date);
        NodeList nodeList = docElement
.getElementsByTagName("Valute");

        int count = nodeList.getLength();
        if (nodeList != null && count > 0) {
                for (int i = 0; i < count; i++) {
                        Element entry = (Element) node-
List.item(i);
                        m = new HashMap<String, String>();

                        String charCode = entry

        .getElementsByTagName(KEY_CHAR_CODE)

.item(0).getFirstChild().getNodeValue();

                        String value = entry
```

```
        .getElementsByTagName(KEY_VALUE)

        .item(0).getFirstChild().getNodeValue();

                                String nominal = "за " + entry

        .getElementsByTagName(KEY_NOMINAL)

        .item(0).getFirstChild().getNodeValue();

                                String name = entry

        .getElementsByTagName(KEY_NAME)

        .item(0).getFirstChild().getNodeValue();

                                m.put(KEY_CHAR_CODE, charCode);
                                m.put(KEY_VALUE, value);
                                m.put(KEY_NOMINAL,      nominal);
m.put(KEY_NAME, name);
                                list.add(m);
                                }
                        }
                } else {
                        // Сделать извещения об ошибках, если код
ответа
                        // нехороший
                }
        } catch (MalformedURLException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        } catch (ParserConfigurationException e) {
                e.printStackTrace();
        } catch (SAXException e) {
                e.printStackTrace();
        }
        return list;
      };
```

9. Проверьте работоспособность приложения. В реальной

программе следует отслеживать наличие подключения устройства к сети, отслеживать ошибки соединения, а также получать данные из сети в отдельном потоке, чтобы не «замораживать» интерфейс пользователя. Эти темы будут рассмотрены во втором курсе.