Android Application Development

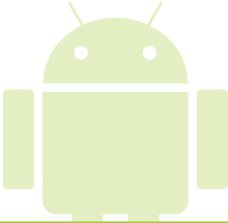
M.Sc. Riste Stojanov
M.Sc.Tomche Delev

Faculty of computer science and engineering - Skopje

October 11, 2011

Outline Day 1

Outline Day 1



What is Android?

- Android represents an exciting new opportunity to write innovative applications for mobile devices.
- Android is an **open-source software stack** that includes the **operating system**, **middleware**, and **key mobile applications** along with a set of **API libraries** for writing mobile applications that can shape the look, feel, and function of mobile handsets.

Android devices

Small, stylish, and versatile, modern mobile devices have become powerful tools that incorporate **cameras, media players, GPS systems, touchscreens** and other **sensors**.

MOTOROLA Sony Ericsson htc quietly brilliant

SAMSUNG LG ZTE中兴

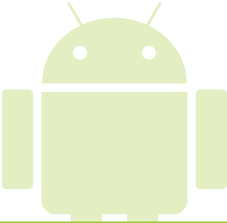
Resourses

- Android Developers
  - http://developer.android.com
- Download SDK
  - http://developer.android.com/sdk/index.html
- Android Market
  - http://market.android.com
  - https://market.android.com/publish/

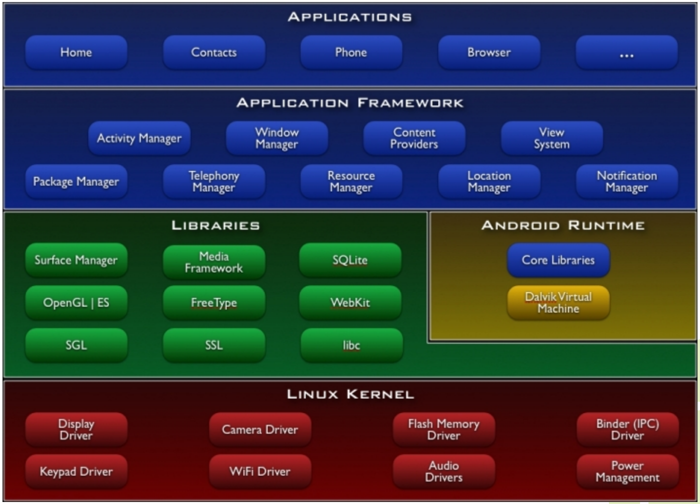
Application development

- Android native application
  - Build using the Android SDK and Java programming language
- Browser application
  - Android has browser with full web capabilities
  - All standard web application
  - jQuery

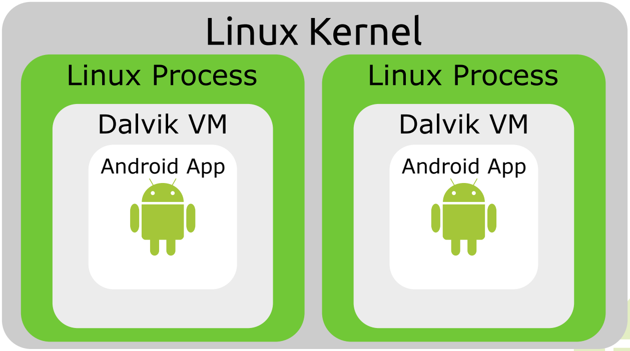
Outline Day 1



Android Architecture



Android Application Model

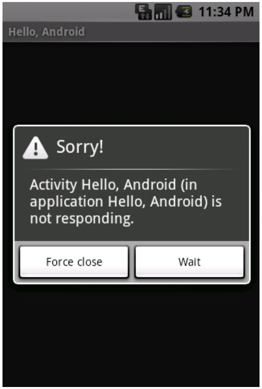


The Dalvik Virtual Machine

- Custom VM
- Ensures that multiple instances run efficiently on a single device
- Uses the device's underlying Linux kernel to handle low-level functionality
  - The Dalvik VM executes Dalvik executable files Format optimized to ensure minimal memory footprint
  - Creates .dex executables

Responsiveness (ANR)

Application not responding

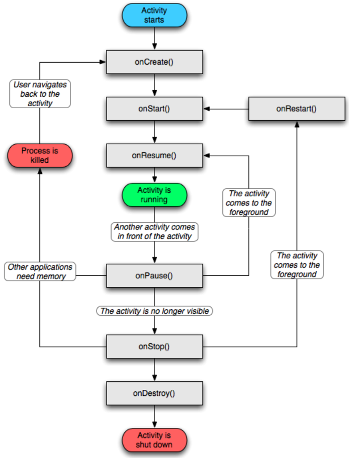


Activity Hello, Android (in application Hello, Android) is not responding.

Platform Components

- Activities
  - Your applications presentation layer
- Services
  - The invisible workers of your application
- Content Providers
  - Shareable data stores
- Intents
  - An inter-application message-passing framework
- Broadcast Receivers
  - Intent broadcast consumers
- Widgets
  - Visual application components that can be added to the home screen
- Notifications
  - A user notification framework

The Activity Life Cycle


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Outline Day 1

Development Environment

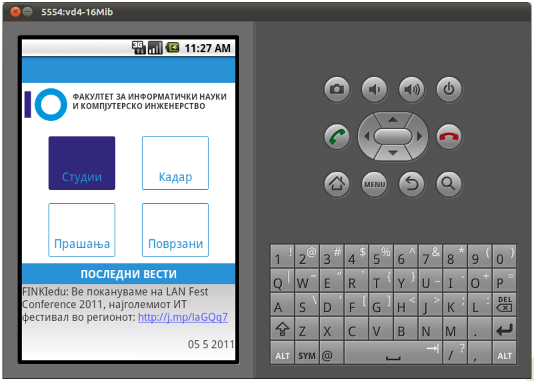
- Preparing Your Development Computer
  - JDK (Java Development Kit)
  - Eclipse
- Downloading the SDK Starter Package
- Installing the ADT Plugin for Eclipse
- Adding Platforms and Other Components
- Exploring the SDK (Optional)

Hello world application

Toolchains

- Android SDK
  - Emulator
- DDMS
- Hierarchy Viewer
- Trace View
- Nine Patch Editor
- Eclipse IDE plugin
  - Integrated with the android SDK tools

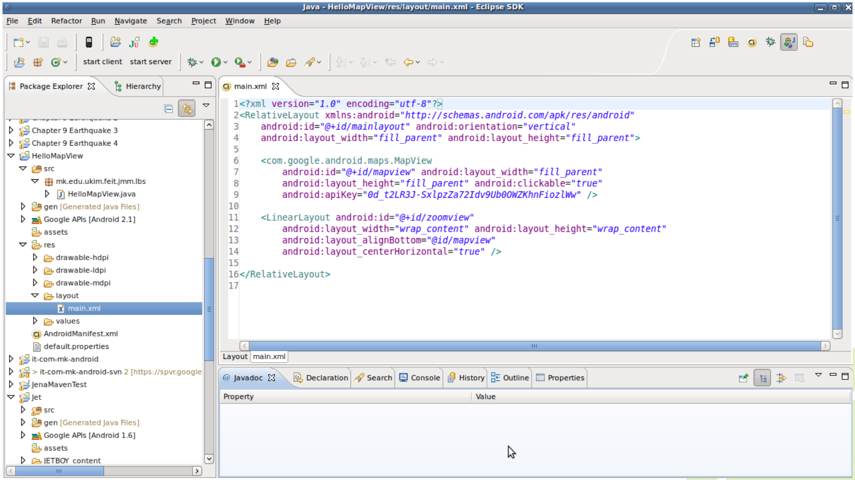
Emulator



FINKIedu: Ве покануваме на LAN Fest Conference 2011, најголемиот ИТ фестивал во регионот: http://j.mp/laGQq7

05 5 2011

Eclipse Plugin


Java - HelloMapView/res/layout/main.xml - Eclipse SDK
File Edit Refactor Run Navigate Search Project Window Help
start client start server
Package Explorer
Hierarchy
Chapter 9 Earthquake 3
Chapter 9 Earthquake 4
HelloMapView
src
mk.edu.ukim.feit.jmm.lbs
HelloMapView.java
gen [Generated Java Files]
Google APIs [Android 2.1]
assets
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
layout
main.xml
values
AndroidManifest.xml
default.properties
it-com-mk-android
> it-com-mk-android-svn 2 [https://spvr.google
JenaMavenTest
Jet
src
gen [Generated Java Files]
Google APIs [Android 1.6]
assets
JETBOY_content
main.xml
Layout main.xml
Javadoc Declaration Search Console History Outline Properties
Property Value

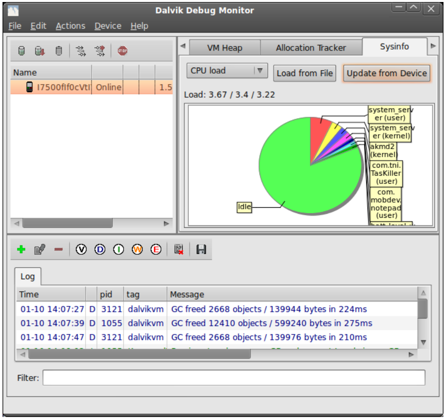
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mainlayout" android:orientation="vertical"
    android:layout_width="fill_parent" android:layout_height="fill_parent">

    <com.google.android.maps.MapView
        android:id="@+id/mapview" android:layout_width="fill_parent"
        android:layout_height="fill_parent" android:clickable="true"
        android:apiKey="0d_t2LR3J-SxlpzZa72Idv9Ub0OWZKhnFiozlWw" />

    <LinearLayout android:id="@+id/zoomview"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_alignBottom="@id/mapview"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Dalvik Debug Monitor Server

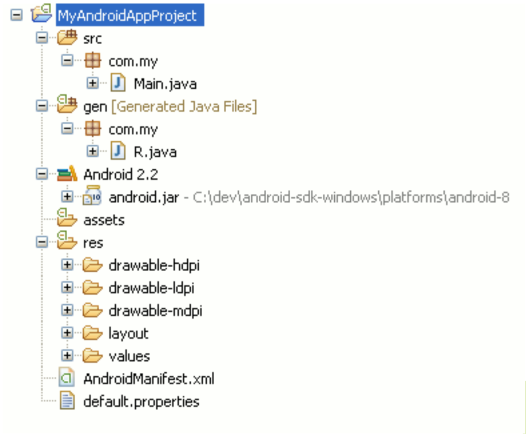


I7500flf0cVtl Online 1.5

Load: 3.67 / 3.4 / 3.22

```
Time            pid  tag       Message
01-10 14:07:27 D 3121 dalvikvm  GC freed 2668 objects / 139944 bytes in 224ms
01-10 14:07:39 D 1055 dalvikvm  GC freed 12410 objects / 599240 bytes in 275ms
01-10 14:07:47 D 3121 dalvikvm  GC freed 2668 objects / 139976 bytes in 210ms
```

Project Directory Layout



```
MyAndroidAppProject
  src
    com.my
      Main.java
  gen [Generated Java Files]
    com.my
      R.java
  Android 2.2
    android.jar - C:\dev\android-sdk-windows\platforms\android-8
  assets
  res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    layout
    values
  AndroidManifest.xml
  default.properties
```

Activity

- An Activity is an application component that provides a screen with which users can interact.
- Each activity is given a window in which to draw its user interface.
- Typically, one activity in an application is specified as the "**main**" activity, which is presented to the user when launching the application for the first time

Creating an Activity

Part 1

- To create an activity, you must create a subclass of **Activity**.
- In your subclass, you need to implement **callback methods** that the system calls when the activity transitions between various states of its lifecycle, such as when the activity is being **created, stopped, resumed**, or **destroyed**.

Creating an Activity

Part 2

- **`onCreate()`**
  - The system calls this when creating your activity. Within your implementation, you should initialize the essential components of your activity.
  - Most importantly, this is where you must call setContentView() to define the layout for the activity's user interface.
- **`onPause()`**
  - The system calls this method as the first indication that the user is leaving your activity.
  - This is usually where you should commit any changes that should be persisted beyond the current user session (because the user might not come back).

Implementing a user interface

Views and widgets

- The user interface for an activity is provided by a hierarchy of viewsobjects derived from the View class.
- Each view controls a particular rectangular space within the activity's window and can respond to user interaction.
- For example, a view might be a button that initiates an action when the user touches it.
- "Widgets" are views that provide a visual (and interactive) elements for the screen, such as a button, text field, checkbox, or just an image.

Implementing a user interface

Layouts

- "Layouts" are views derived from ViewGroup that provide a unique layout model for its child views, such as a linear layout, a grid layout, or relative layout.
  - The most common way to define a layout using views is with an XML layout file saved in your application resources.
  - This way, you can maintain the design of your user interface separately from the source code that defines the activity's behavior.
  - You can set the layout as the UI for your activity with setContentView(), passing the resource ID for the layout.
- You can also create new Views in your activity code and build a view hierarchy by inserting new Views into a ViewGroup, then use that layout by passing the root ViewGroup to setContentView().

Declaring the activity in the manifest

You must declare your activity in the manifest file in order for it to be accessible to the system. To decalare your activity, open your manifest file and add an `<activity>` element as a child of the `<application>` element.

Example

```
<manifest ... >
  <application ... >
    <activity android:name=".ExampleActivity" />
        ...
  </application ... >
  ...
</manifest>
```

Using intent filters

An <activity> element can also specify various intent filtersusing the <intent-filter> elementin order to declare how other application components may activate it.

Example

```
<activity android:name=".ExampleActivity" android:icon="
    @drawable/app_icon">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.
            LAUNCHER" />
    </intent-filter>
</activity>
```

Starting an Activity

You can start another activity by calling **`startActivity()`**, passing it an **`Intent`** that describes the activity you want to start

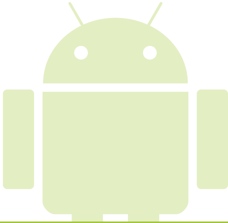
Example

```
Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent);
```

Shutting Down an Activity

You can shut down an activity by calling its **`finish()`** method.

You can also shut down a separate activity that you previously started by calling **`finishActivity()`**.



Implementing the lifecycle callbacks

- `onCreate`
- `onStart`
- `onResume`
- `onPause`
- `onStop`
- `onDestroy`

Outline Day 1

Activating components: intents

- Activities, services, and broadcast receivers are activated by asynchronous messages called **intents**
- An intent is an `Intent` object that holds the content of the message

Fundamental Aandroid UI Design

- Views
  - Views are the base class for all visual interface elements
  - Derived from the View class
- View Groups
  - Can contain multiple child Views
  - Extend the ViewGroup class
- Activities
  - Represent the window, or screen, being displayed.
  - To display a user interface you assign a View (usually a layout) to an Activity

Introducing Views

Inflating an Activity layout

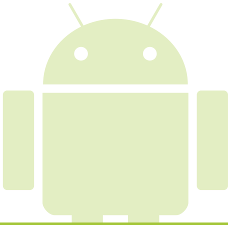
```java
@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(R.layout.main);
    TextView textView = (TextView)findViewById(R.id.textView);
}
```

Creating a UI layout in code

```java
@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    TextView textView = new TextView(this);
    setContentView(textView);
    textView.setText("Hello Android");
}
```

Event Listeners

- Java event handling mechanism
- Interfaces containing method that is invoked on some user action
- Examples Listeners:
  - View.OnClickListener
  - onClick(View v)
  - View.OnKeyListener
  - onKey(View v, int keyCode, KeyEvent event)
  - View.OnTouchListener
  - onTouch(View v, MotionEvent event)



Using event listeners

Anonimous implementation of OnClickListener

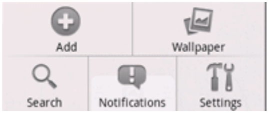
```
private OnClickListener btnClicked = new OnClickListener() {
    public void onClick(View v) {
        // do something when button is clicked
    }
};
protected void onCreate(Bundle bundle) {
    // Capture our button from layout
    Button btn = (Button) findViewById(R.id.btn);
    // Register the onClick listener with the implementation above
    btn.setOnClickListener(btnClicked);
}
```

In XML

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/btnNewGame"
    android:onClick="onBtnClick"

public void onBtnClick(View view) {
    // do something when button is clicked
}
```

Menus



- Options Menu
  - This is the primary set of menu items for an Activity
  - It is revealed by pressing the device MENU key
- Context Menu
  - This is a floating list of menu items that may appear when you perform a long-press on a View
- Submenu
  - This is a floating list of menu items that is revealed by an item in the Options Menu or a Context Menu

Options Menu

When the menu is opened for the first time, the Android system will call the Activity `onCreateOptionsMenu()` callback method

The menu xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/save_game"
        android:title="@string/menu_save_game" />
    <item android:id="@+id/end_game"
        android:title="@string/menu_end_game" />
</menu>
```

Inflating the menu

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