Mobile Device Platforms

Mensch-Maschine-Interaktion 2, WS 2010/2011

Michael Rohs
michael.rohs@ifi.lmu.de
MHCI Lab, LMU München

Lectures & Exercises

Lecture	Date	Topic
1	12.1.	Introduction to Mobile Platforms & Mobile Interaction
2	19.1.	Mobile Input and Output Technologies
3	26.1.	Mobile Communication, Location & Context
4	2.2.	Evaluation of Mobile Systems
5	9.2.	User Interface Design for Small Displays

Exercise	Date	Topic
0		Developing countries + Android-Eclipse
1	10.1.	Recipe input
2	17.1.	Touch input, gestures
3	24.1.	Location-based Audio
4	31.1.	Evaluation of mobile LMU Web portal

Mobile Human-Computer Interaction

- General topic
  - Human interaction with mobile and wearable devices
  - Human interaction while “on the move”
- Goals of this section
  - Learn about how to create mobile user interfaces
    - Mobile technologies
  - Understand that mobile interaction is interaction in context
    - “Human factors”
  - Learn about specific design considerations for mobile user interfaces
    - Small displays

More about this in WS in course on Mobile HCI

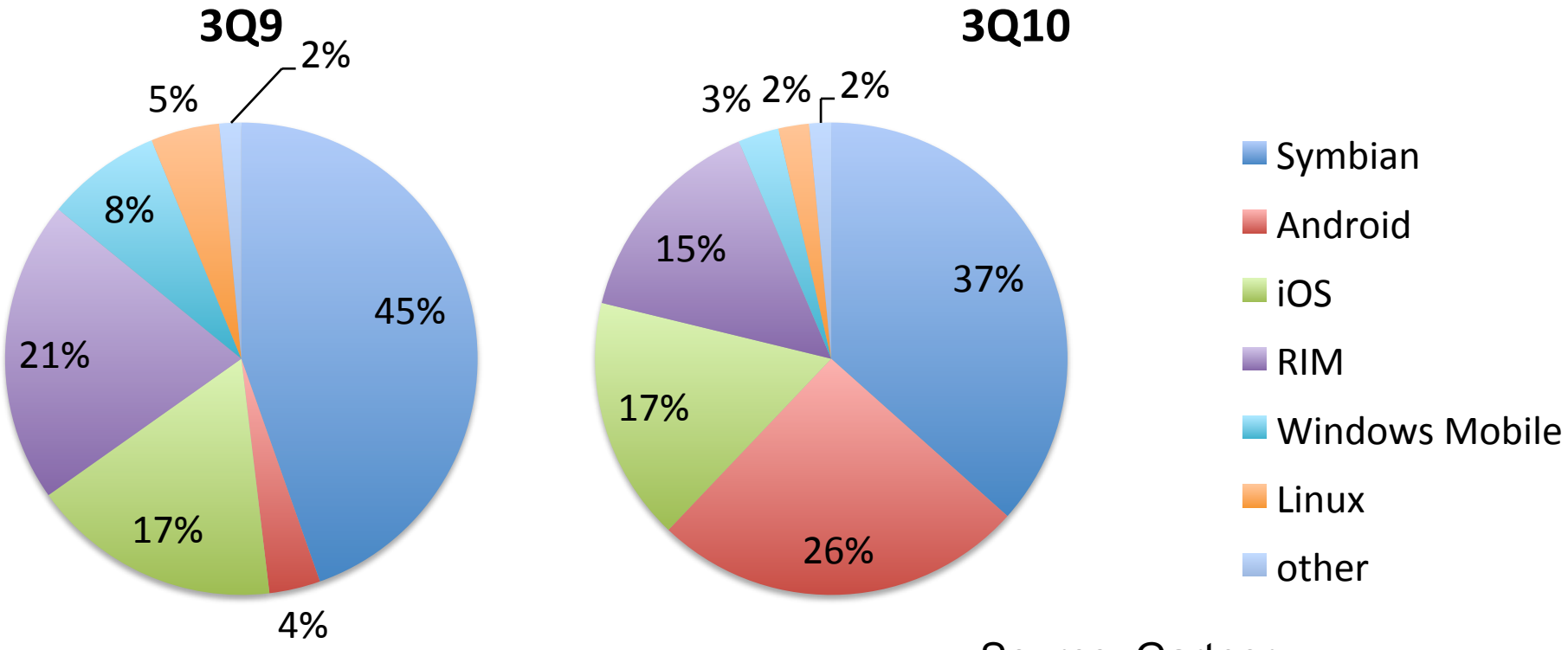
Mobile Device Development

- Platform specific
  - BREW (C, C++)
  - Symbian OS (C++, Python, Qt)
  - Apple iOS (Objective C)
  - Android (Java)
  - Windows CE / Windows Phone 7 (C#, C++)
- Platform independent
  - Java ME (Java Micro Edition)
  - Flash Lite / Flash
  - HTML 5, WAP / WML, cHTML (i-mode)
  - SMS, MMS

Mobile Application Development

- Who of you owns a mobile phone?
  - Is it possible to develop applications for this device?
  - Which platforms are supported?
  - Which programming languages and tools can be used?
  - How to install the programs on the device?

Market Shares: Smartphones

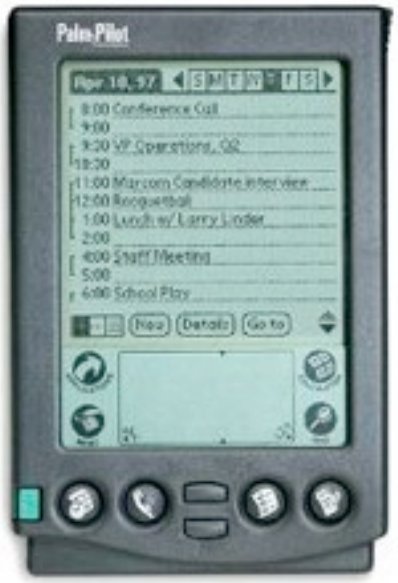


Source: Gartner
http://www.gartner.com/it/page.jsp?id=1466313

- Android market share grows rapidly
- Apple & Android share 81% mobile Web traffic
- Regional differences
  - USA: Apple, Android, RIM; Asia: Symbian

PDA Platforms: Palm OS / Windows Mobile

- Palm OS (historical, now webOS)
  - Operating system for PDAs
  - Initial release: 1996
  - Development: CodeWarrior Development Studio for Palm OS
  - Languages: C/C++



- Windows Mobile (Windows CE)
  - Operating system for PDAs and smart phones
  - Initial release: 1996
  - Development: Microsoft Visual Studio
  - Languages: C++, C#, Visual Basic .NET
  - http://www.microsoft.com/windowsmobile/

Windows Phone 7

en.wikipedia.org/wiki/Windows_Phone_7
windowsphone7.com
msdn.microsoft.com/en-us/windowsmobile

- Successor of Windows Mobile
  - Launched October 21, 2010
- “Metro” design language
  - “Typography is beautiful”, no decoration
  - “Start screen” made up of “tiles”
    - “Tiles” show live application data
  - “Hubs” integrate local and online content
    - “Pictures hub” = local photos & Facebook photo album
    - “People hub” = contacts from Gmail, Facebook, etc.
    - “Music and Video hub” integrates with Zune
    - “Games hub” integrates with Xbox Live
- Windows Phone Marketplace
- Advertising platform
- Programming: C#, Silverlight

Symbian

Symbian History

1984 Psion 1
(First PDA)

- 1997: Psion's EPOC OS
- 1998: Symbian consortium and Symbian OS
  - Ericsson, Nokia, Motorola, Psion (founders)
  - Sony Ericsson, Siemens, etc. (shareholders)
- 2000: First Symbian OS phone
  - Ericsson R380

- 2008: Nokia buys Symbian Ltd.
- 2009: Symbian made open source
  - July 2009: 250 million Symbian phones shipped
- Current version: Symbian^3
- 2010: Motorola, Samsung, LG, Sony-Ericsson will abandon the platform

Symbian Characteristics

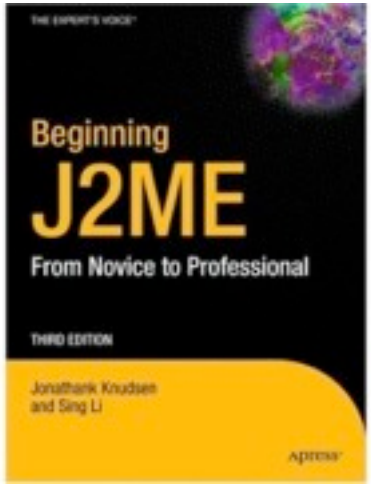
- OS for resource-constrained handheld devices
  - Pre-emptive multitasking, multithreading, memory protection, client-server architecture
  - Special features: conserving memory, reliability, CPU switched off when applications are not dealing with events
- From the developer's point of view
  - User interface framework
  - APIs (was Symbian C++; now C++, Qt)
  - SDK, Tools
- From the user's point of view
  - Consistent user interface
  - Extensible (third party applications)

Nokia N95

Java ME

Java ME: Java on Mobile Devices

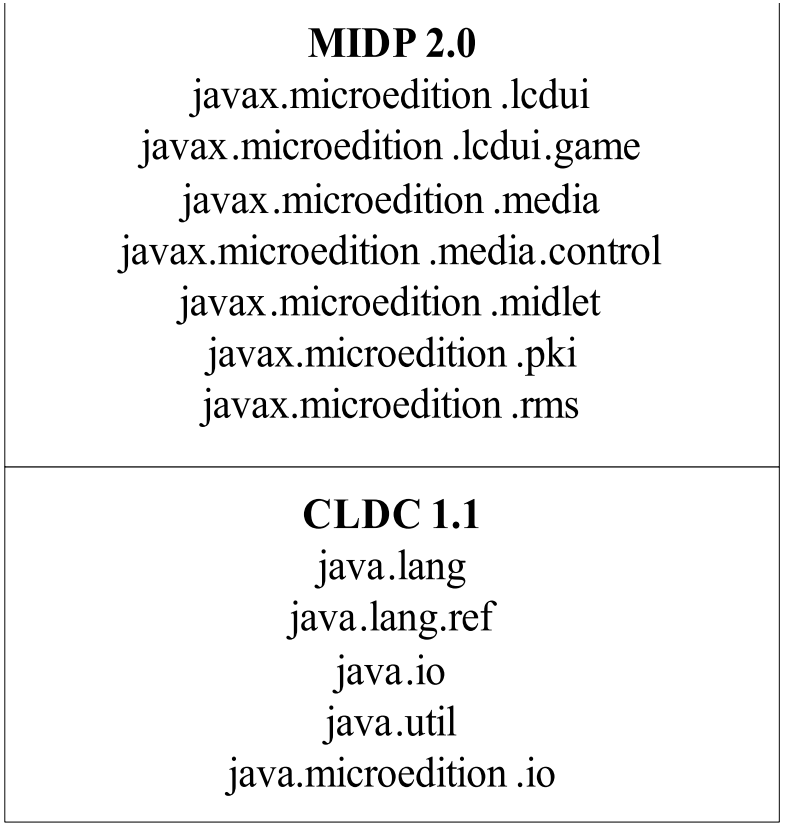
- "The most ubiquitous application platform for mobile devices" (SUN)
  - http://java.sun.com/javame/ (moved to Oracle)
  - More than 2 billion phones shipped supporting Java ME
  - Not supported by newest mobile operating systems



- Jonathan Knudsen and Sing Li. Beginning J2ME: From Novice to Professional. 2005.
- Source code: http://www.apress.com/resource/bookfile/2080

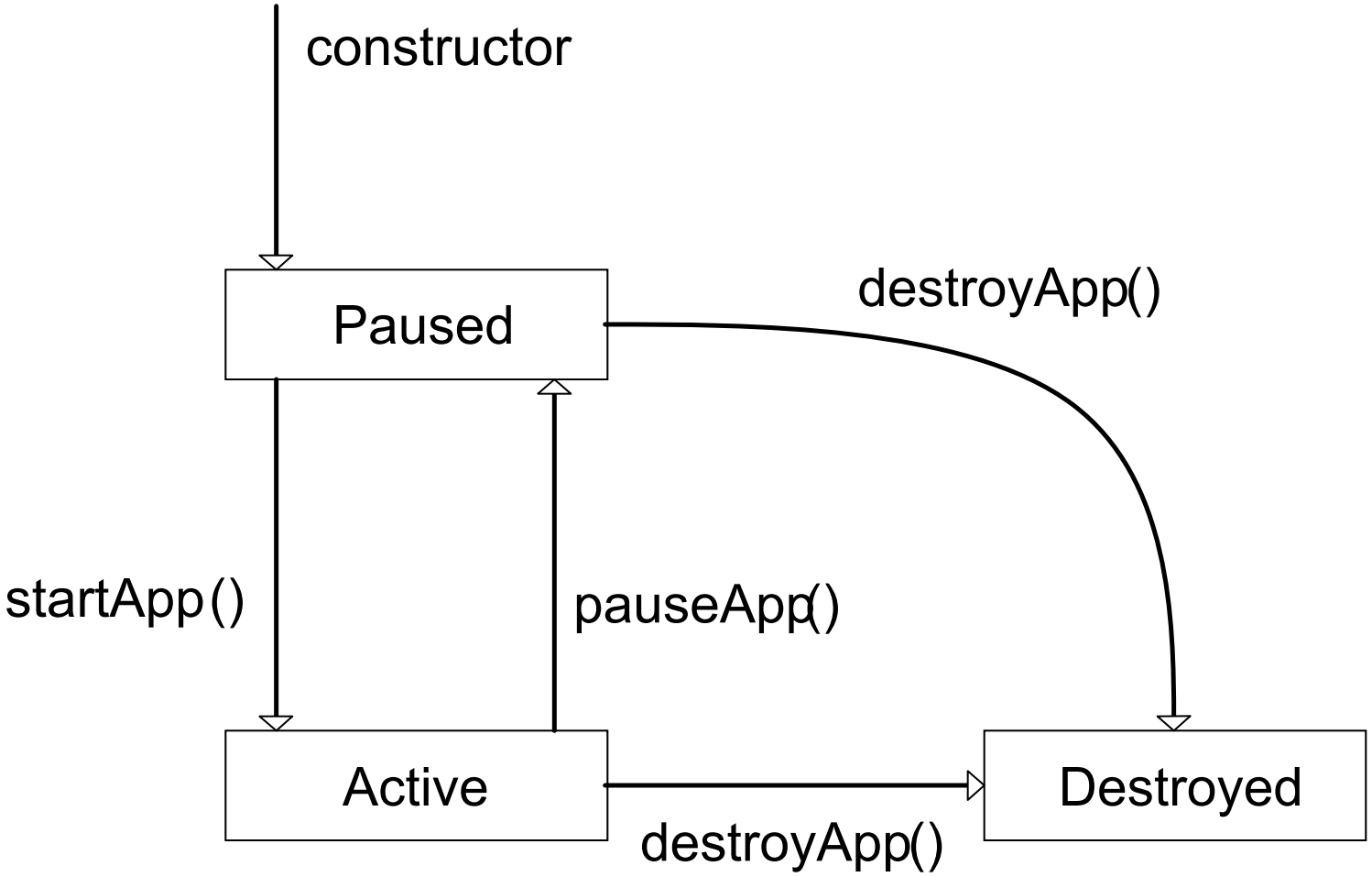
CLDC & MIDP

- Specify set of libraries for a particular device class
  - Mobile Information Device Profile (MIDP)
  - Connected Limited Device Configuration (CLDC)
- Device profiles for handheld devices, set-top boxes, etc.

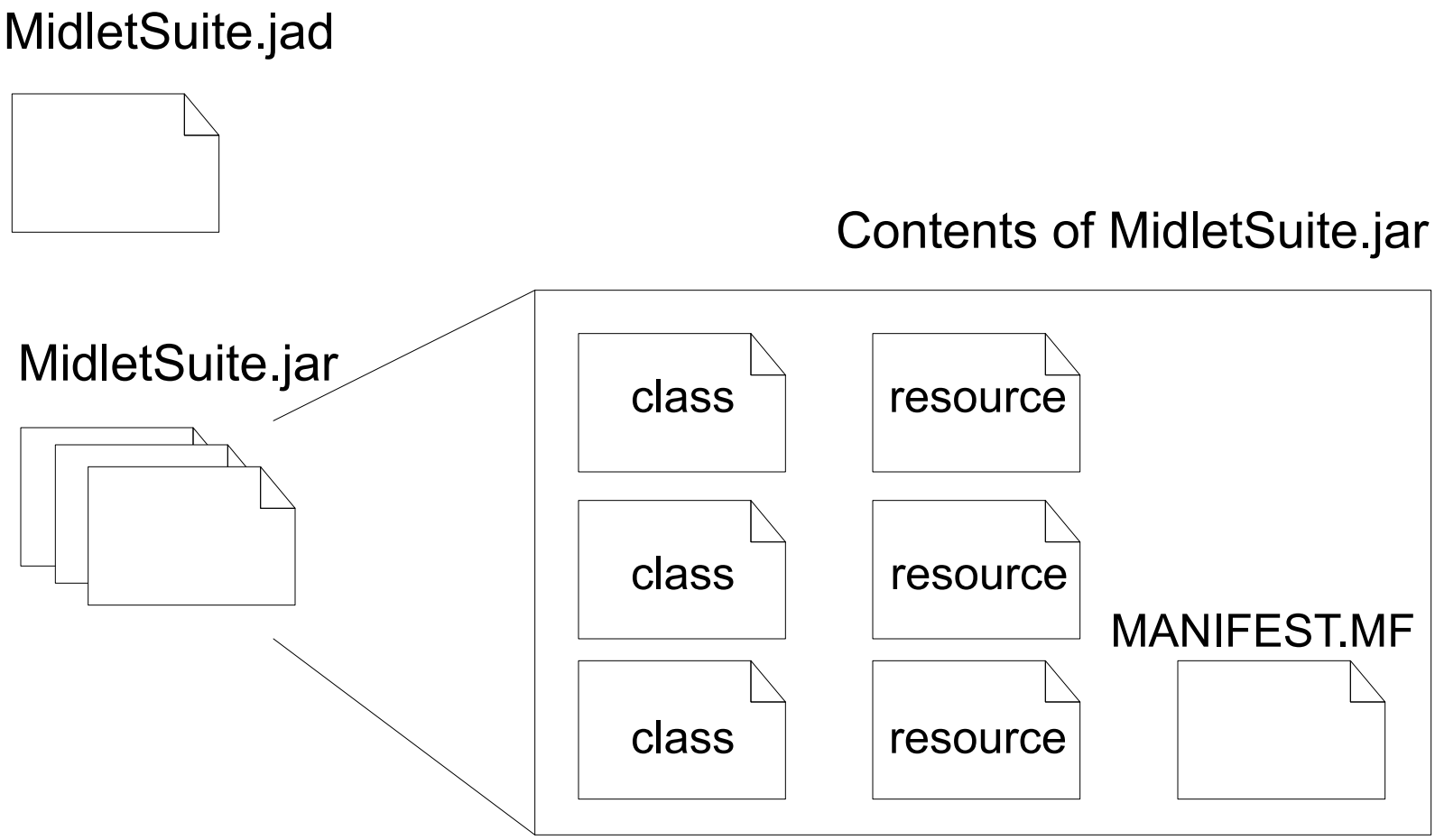


APIs restricted in comparison to J2SE

MIDlet (MIDP Application): Life Cycle



Anatomy of a MIDlet Suite

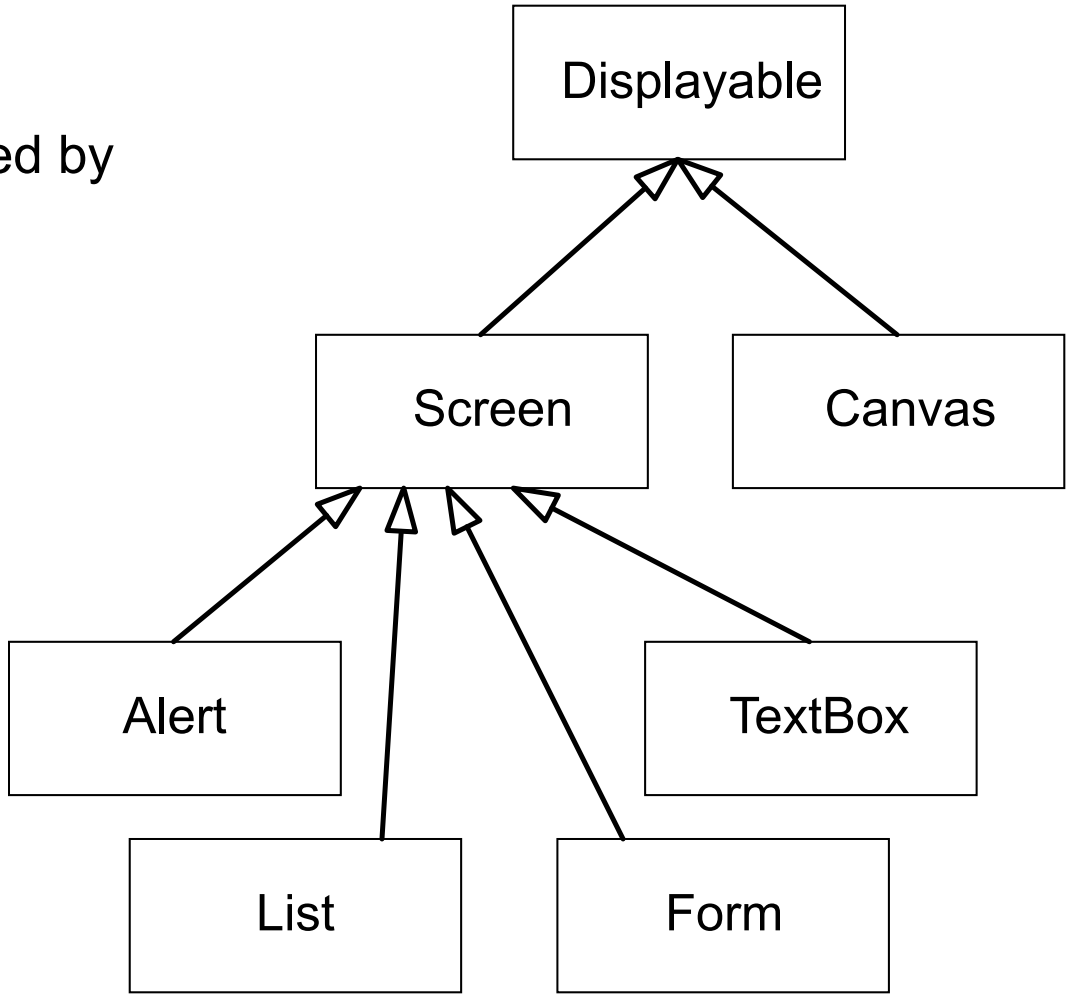


User Interface: View from the Top

- User-interface classes *javax.microedition.lcdui*
- Device display represented by *Display (getDisplay())*

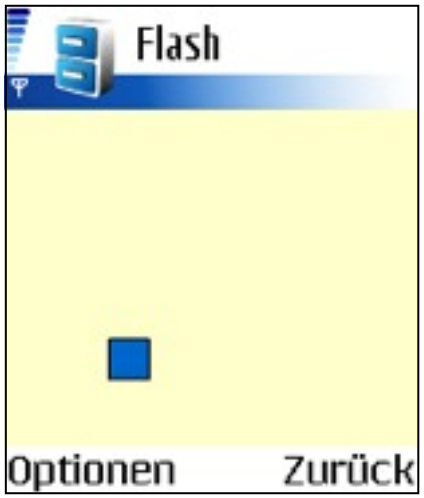
- *Display: easel*
- *Displayable: canvas on easel*

- *Canvas: Discovery*
- *Screen: Abstraction*



Prototyping: Adobe Flash / Flash Lite

- Rapid development of interactive content
- Cross-device development
  - Android, Windows Mobile, Windows Phone 7, Symbian, BREW, Palm webOS, Blackberry
  - Not iOS!
- Features
  - Language: ActionScript
  - Input: text entry; multitouch and gestures; accelerometer
  - Media handling (images, sound, video)
  - Loading and parsing of XML
  - Persistent data



http://www.adobemediaplayer.com/devnet/devices.html

Android

Android

- Free, open, mobile platform
  - “Software stack for mobile devices that includes an operating system, middleware and key applications.”
  - Apache v2 open-source license
- Open Handset Alliance
  - Group of 34 companies that develop Android platform
- Linux as hardware abstraction layer
  - Kernel, device drivers
- Java
  - Application framework
  - Applications (and C/C++ native SDK)

Android

- No distinction between core and 3rd party applications
  - All components can be replaced
- Integration of data on the phone with data on the Web
- T-Mobile G1
  - First commercial Android handset
  - Google Maps Street View, Gmail, YouTube
  - Available in the USA since October 23rd, 2008

http://code.google.com/android/dev-devices.html
http://en.wikipedia.org/wiki/T-Mobile_G1
http://www.htc.com/www/product/g1/overview.html

Outline

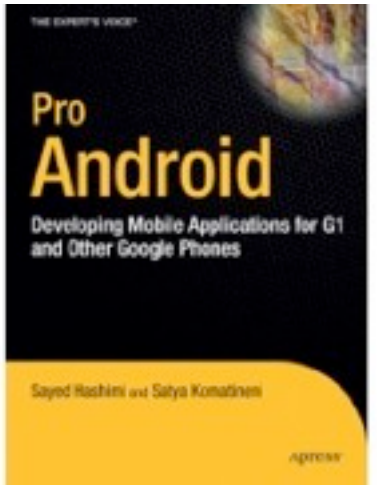
- The Android Platform
- Installing Android & Hello World
- Activity Lifecycle & Intents
- Resources & UI Components

Next lectures:

- Basic Graphics & Touch Input
- Location & Maps
- Bluetooth

Android Books

- Sayed Y. Hashimi, Satya Komatineni:
  Pro Android.
  Apress, 2009. ISBN 1430215968.
  - General introduction into the concepts of Android



- Many other books on Android available
  - http://tinyurl.com/399ksu3

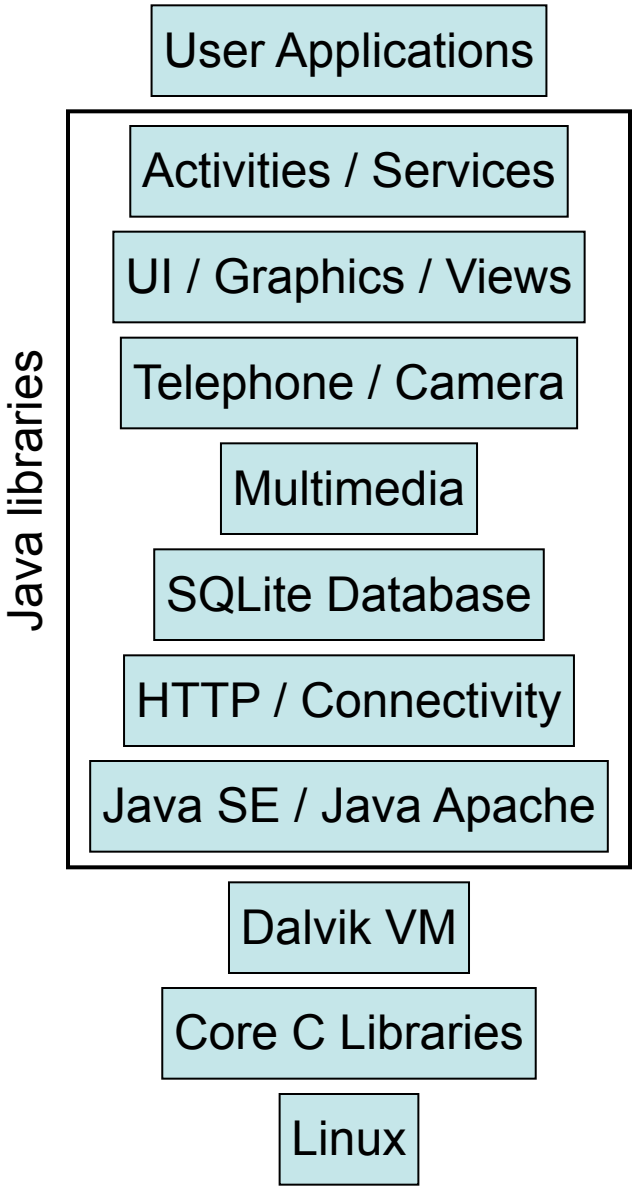
- Android developer pages
  (platform documentation)
  - http://developer.android.com

very good!

The Android Platform

The Android Platform

- General-purpose computing platform for mobile devices
- Linux-based OS stack
  - Hardware abstraction, memory management, process management
  - Dalvik VM (register-based, smaller class files)
- Application framework
  - Activities, services, telephony, connectivity, graphics, UI, etc.
  - Applications can publish capabilities, replace components
- Android SDK
  - Java API (most of Java SE)
  - UI framework



History of Android

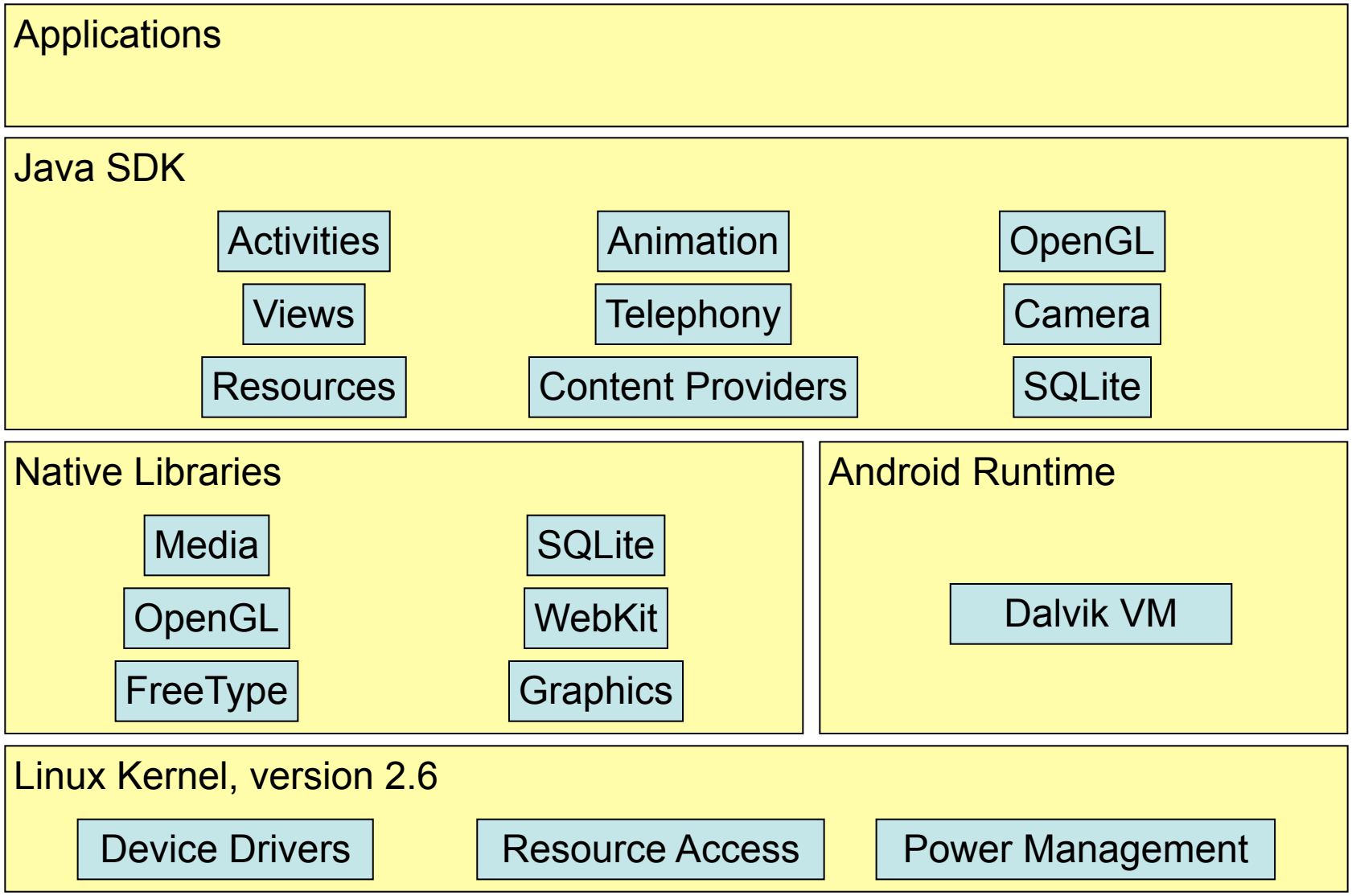
- 2005
  - Google buys startup company Android Inc.
  - Work on Dalvik VM starts
- 2007
  - Open Handset Alliance announced (http://www.openhandsetalliance.com)
  - “Early Look” SDK
- 2008
  - T-Mobile G1 announced
  - SDK 1.0 released
  - Android open sourced under Apache’s open source license
- 2009
  - SDKs 1.5 (Cupcake) and 1.6 (Donut) released

http://www.techradar.com/news/phone-and-communications/mobile-phones/a-complete-history-of-android-470327
http://www.wired.com/techbiz/media/magazine/16-07/ff_android?currentPage=all
http://en.wikipedia.org/wiki/Android_(operating_system)#History

Architectural Goals

- Encourage low-cost development of mobile applications
  - Openness, affordability, quality development framework
- Enable interaction between applications
  - Services, data exchange, UI
- Exploit cloud-computing model
  - Local data stores, backed up on the Web

Android Software Stack



Android Characteristics

- Activity
  - Applications structured as Activities
  - A logical unit of user action
  - Typically represented by a screen containing views
  - Can also be viewless (e.g. a service)
- Framework manages lifecycle of activity
  - Hide, restore, stop, close activity windows
- Declarative UI definition (XML files)
  - Resources (view definitions, strings, bitmaps)

Installing Android and Hello World

Installing Android (1/2)

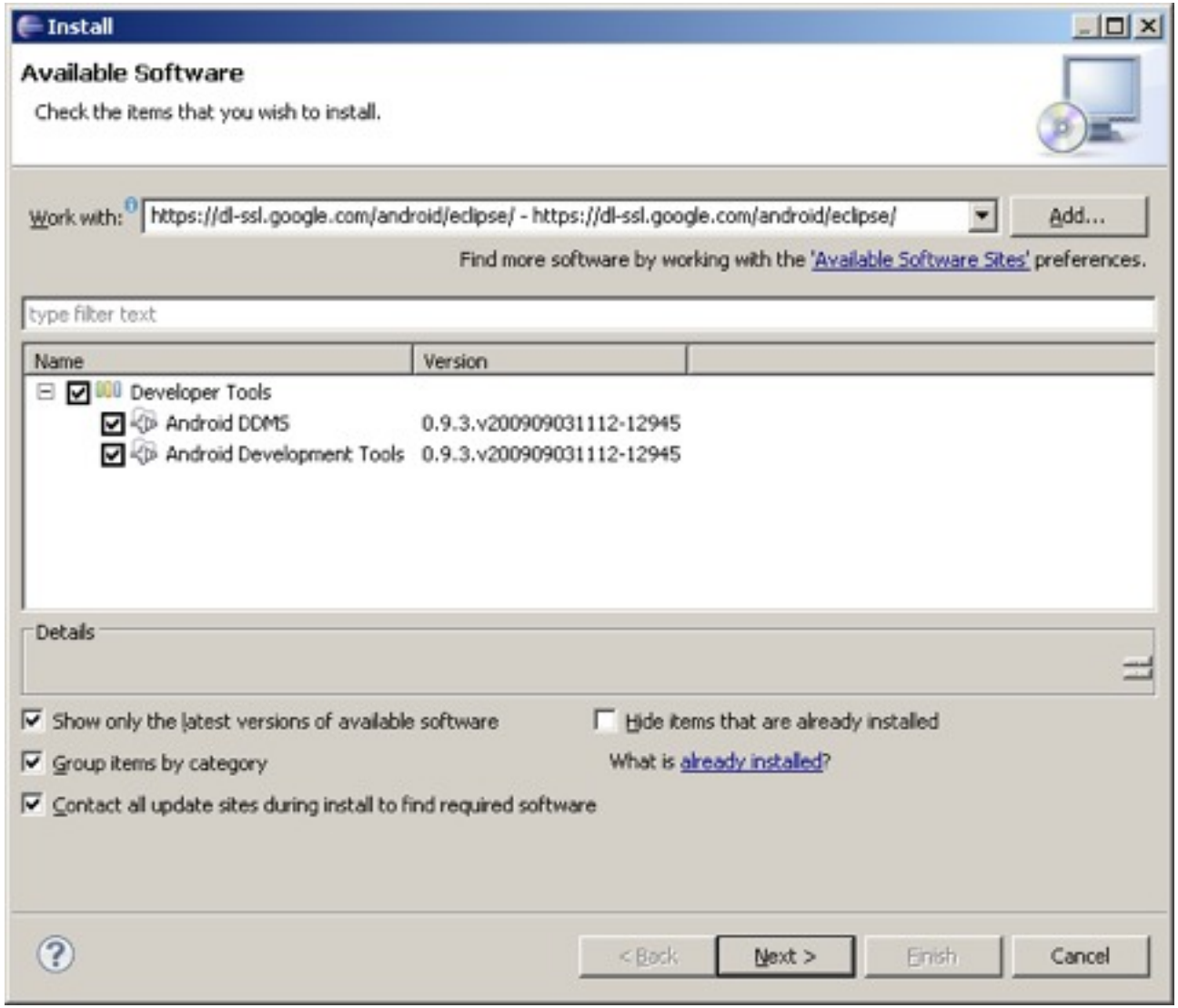
- Java JDK 6, Standard Edition (not only JRE)
  - http://java.sun.com/javase/downloads/index.jsp
- Eclipse IDE (3.4 or newer)
  - http://www.eclipse.org/downloads/
  - Eclipse IDE for Java Developers
- Android SDK starter package (depending on your platform)
  - http://dl.google.com/android/android-sdk_r08-windows.zip
  - http://dl.google.com/android/android-sdk_r08-mac_86.zip
  - http://dl.google.com/android/android-sdk_r08-linux_86.tgz
- See also: “Quick Steps”
  - http://developer.android.com/sdk/index.html

Installing Android in Eclipse (2/2)

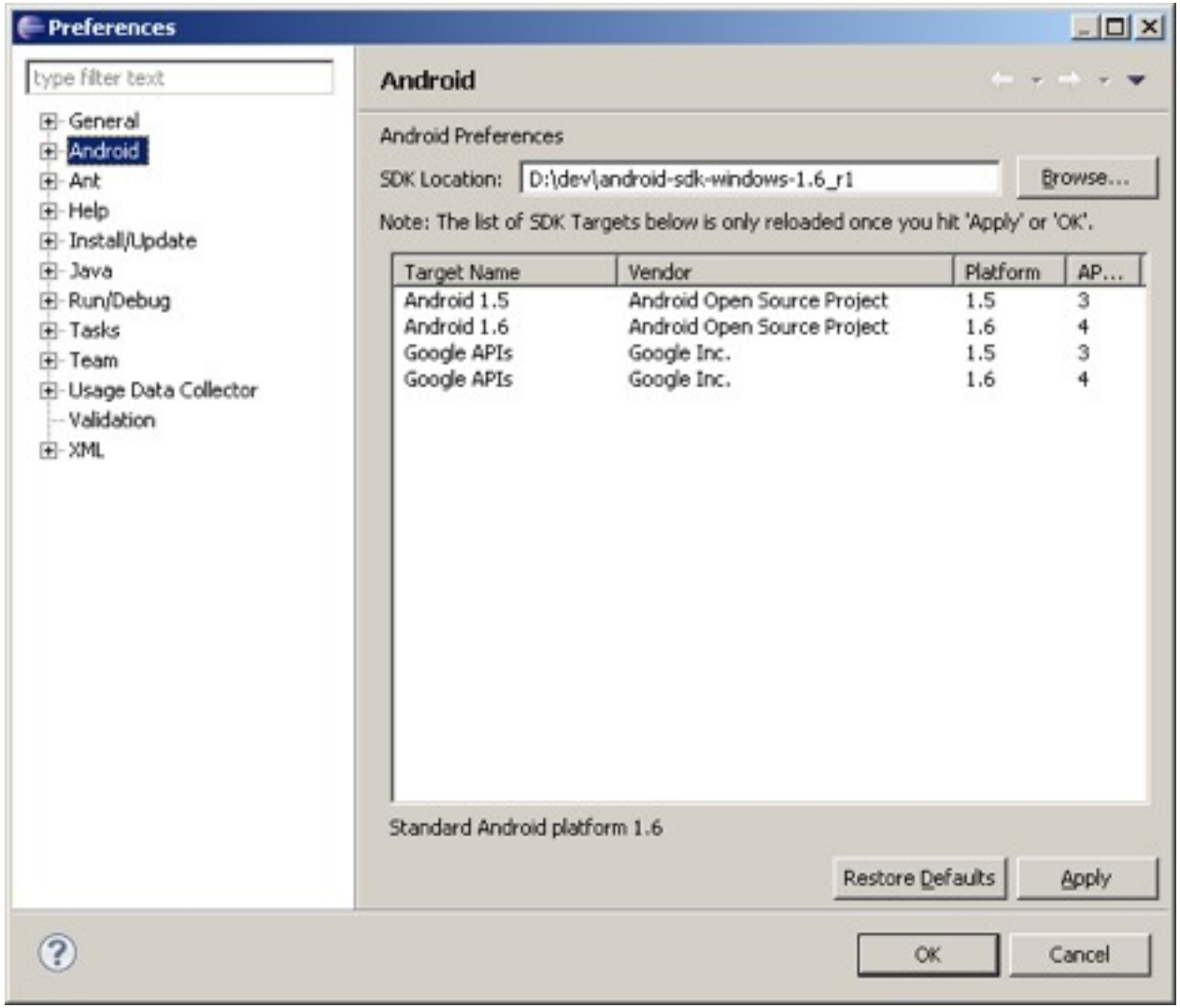
- In Eclipse: Install New Software...
  - https://dl-ssl.google.com/android/eclipse/
- Point Eclipse to the Android SDK starter package
  - Menu: Preferences, Android, SDK Location
- In Eclipse: Android SDK and AVD Manager
  - Install packages, update all...
  - Add new virtual device for newest platform version.

In Eclipse: Install New Software...

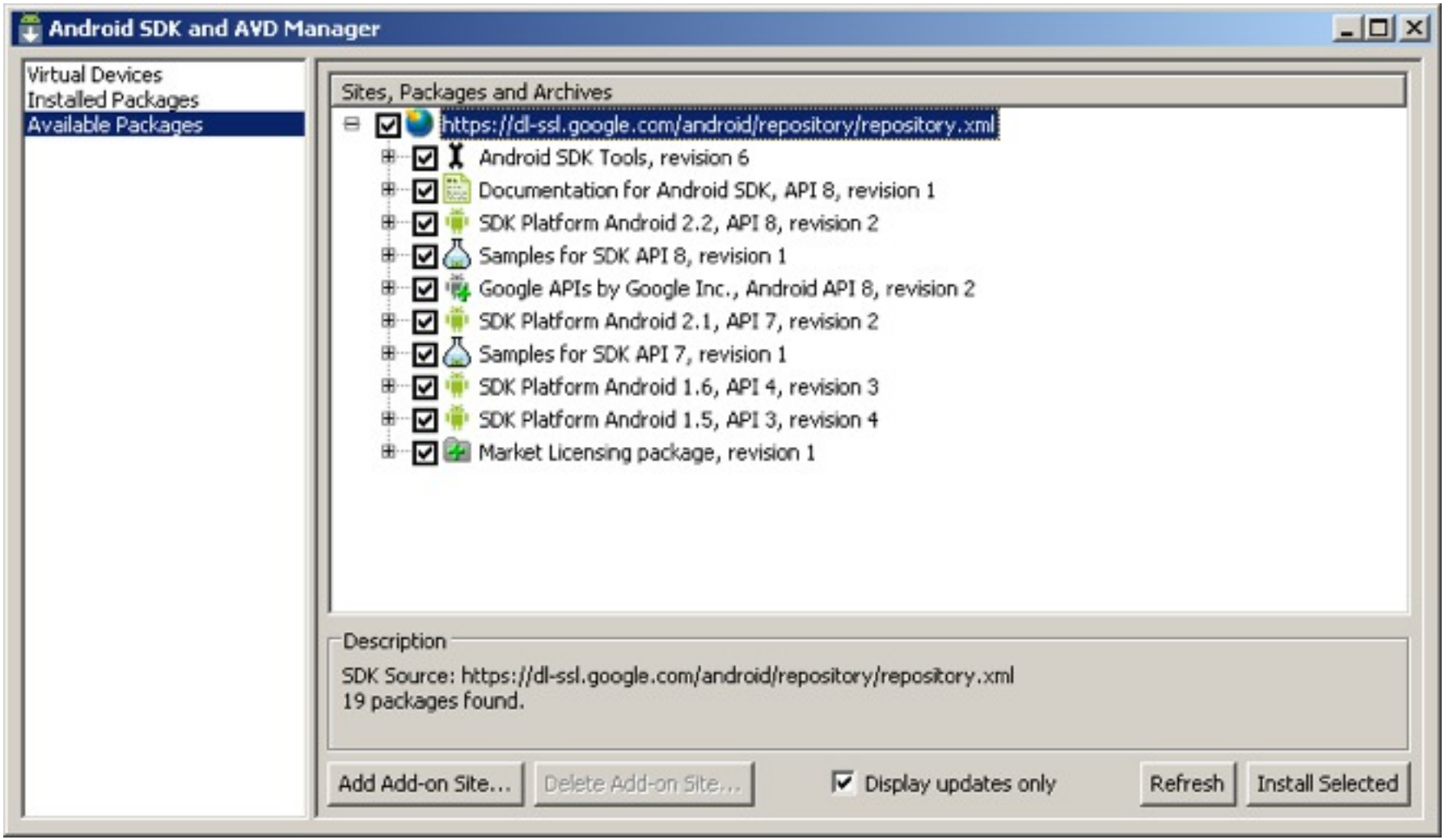
Android Plugin – https://dl-ssl.google.com/android/eclipse/



Set Path to Android SDK Starter Package

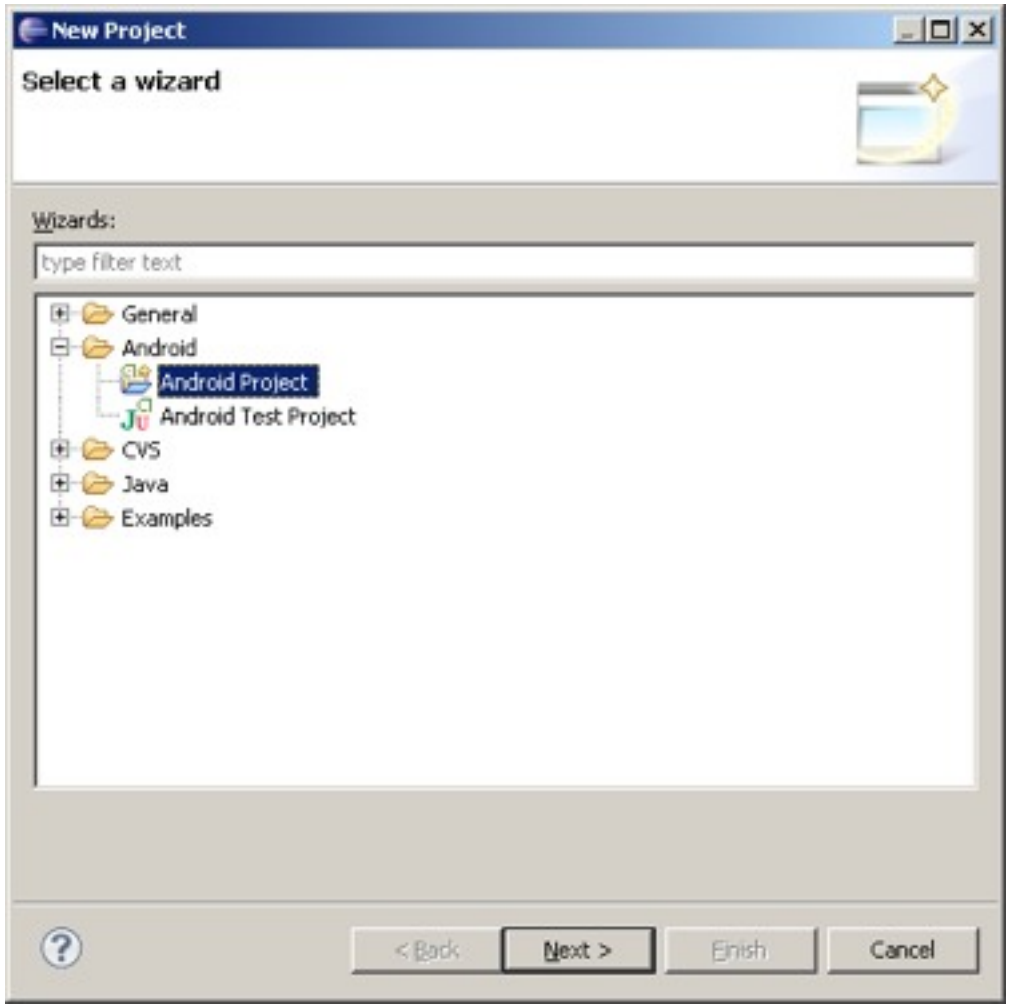


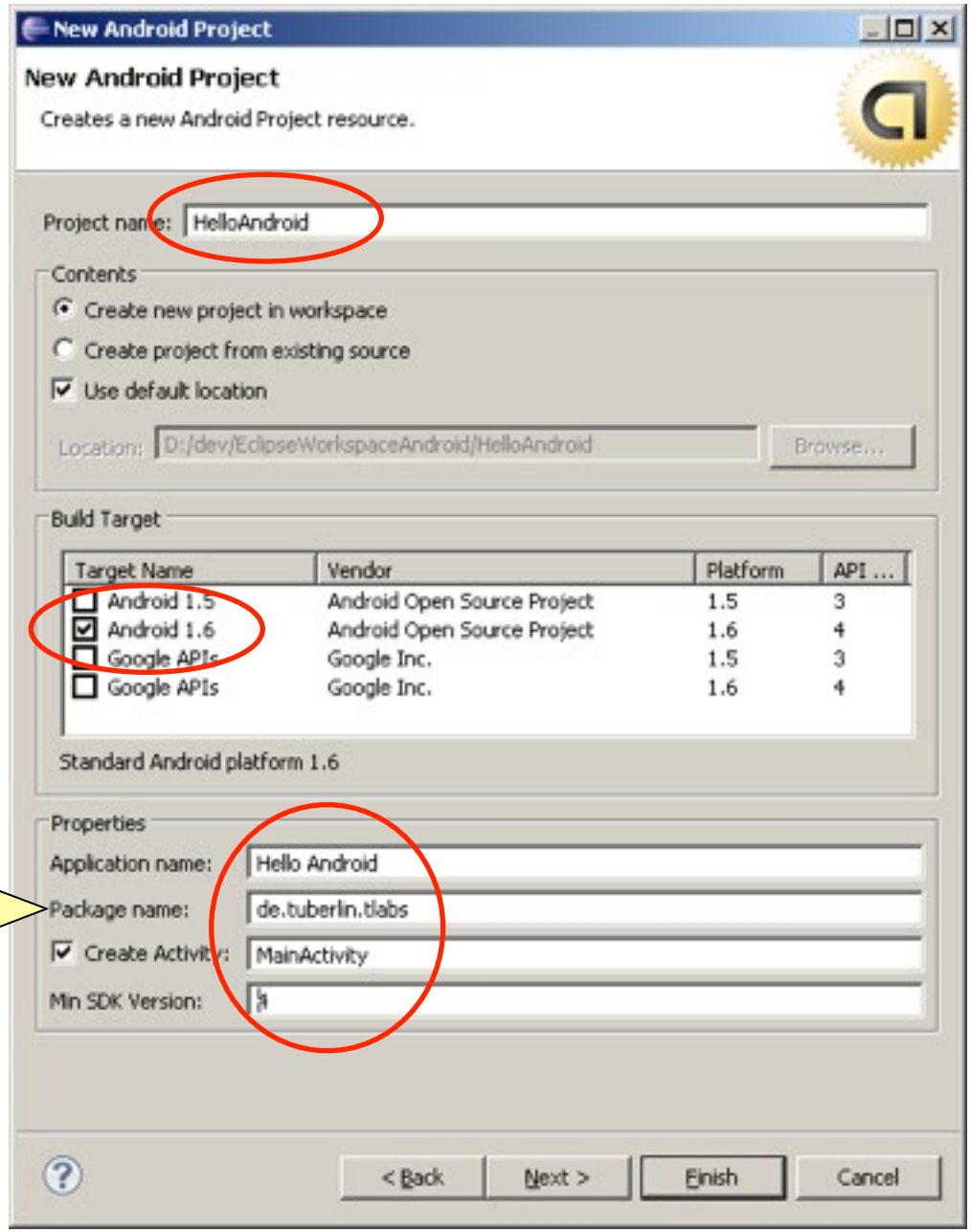
Update Packages



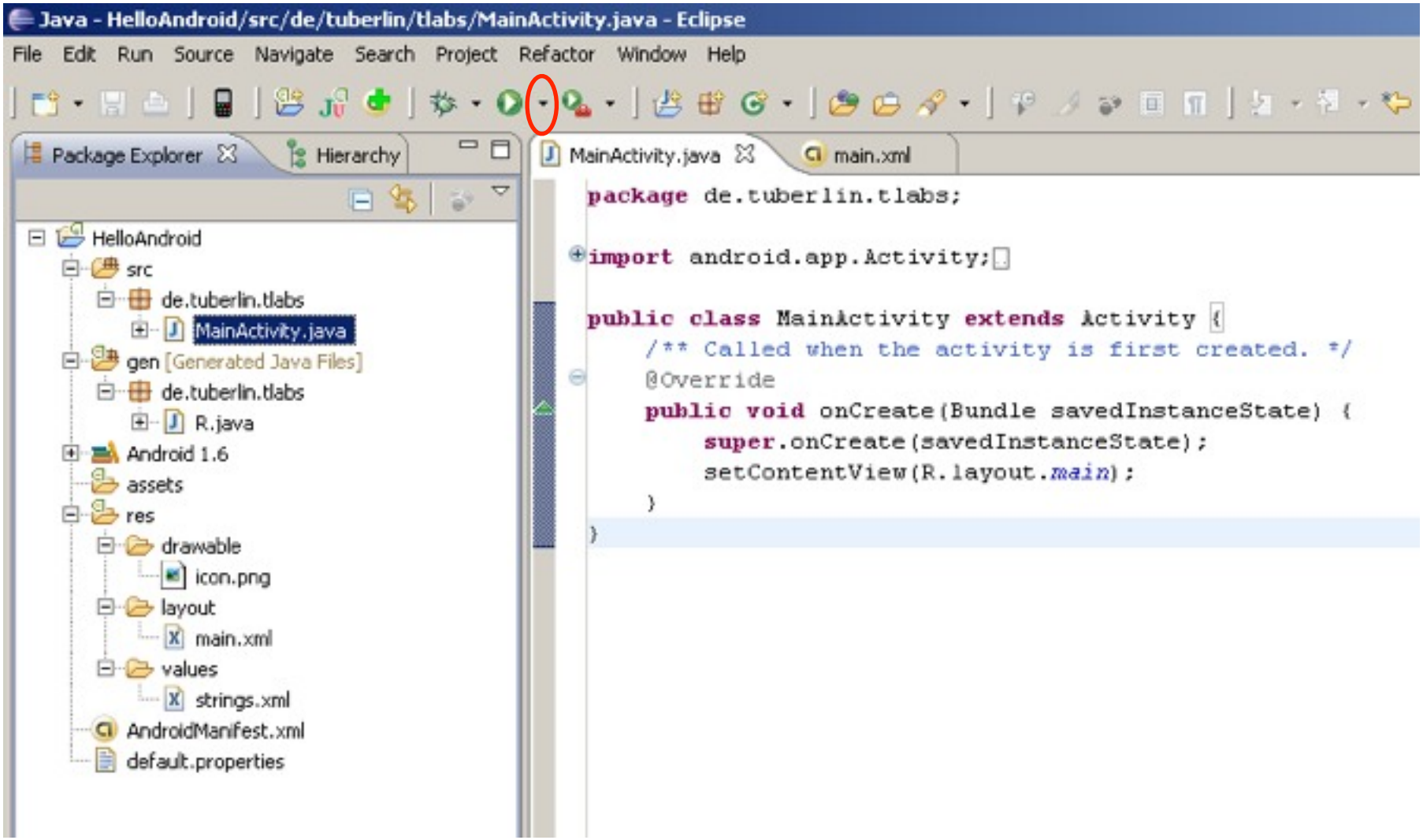
Creating Your First Android Project

File → New Project → Android → Android Project

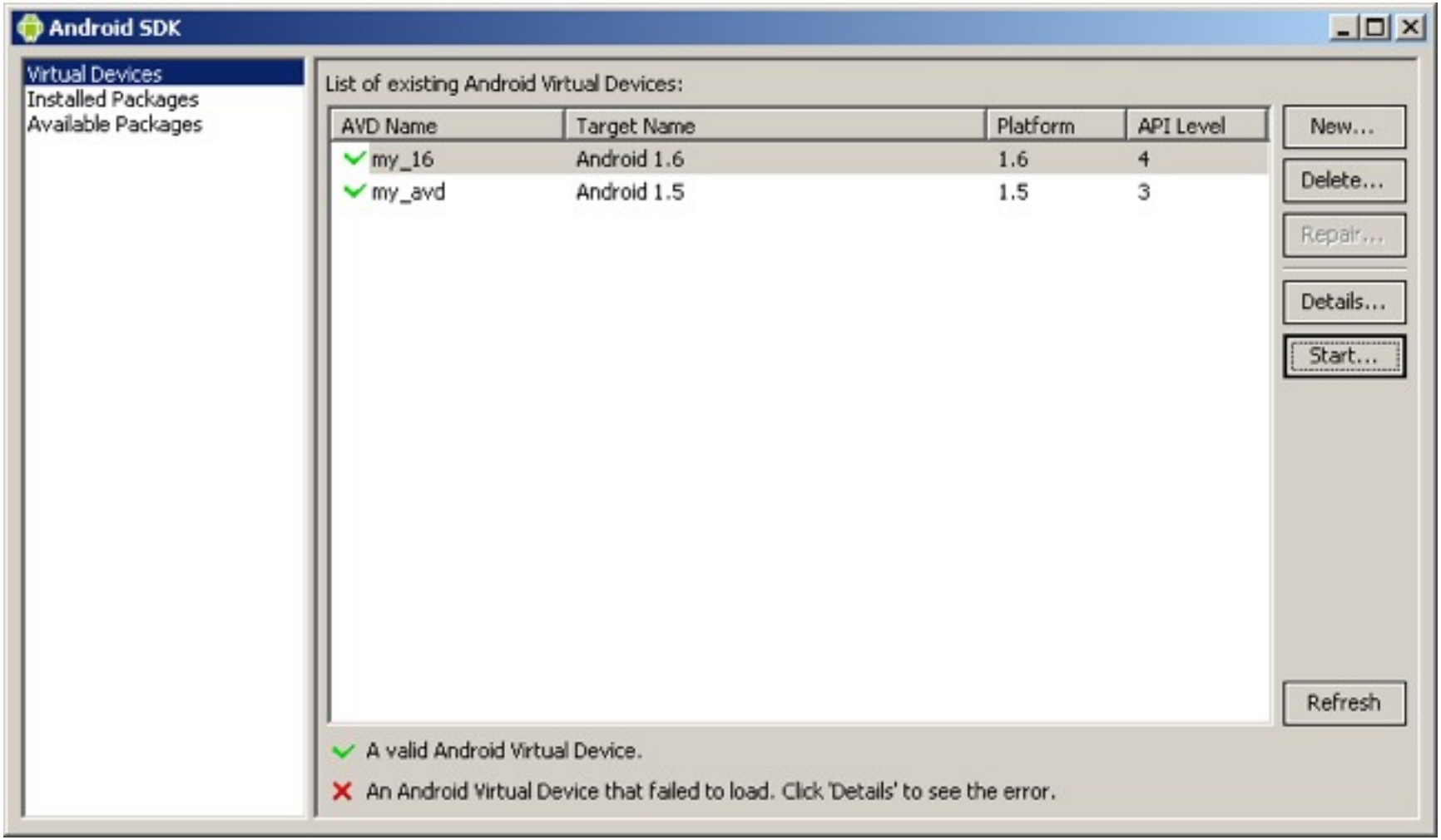


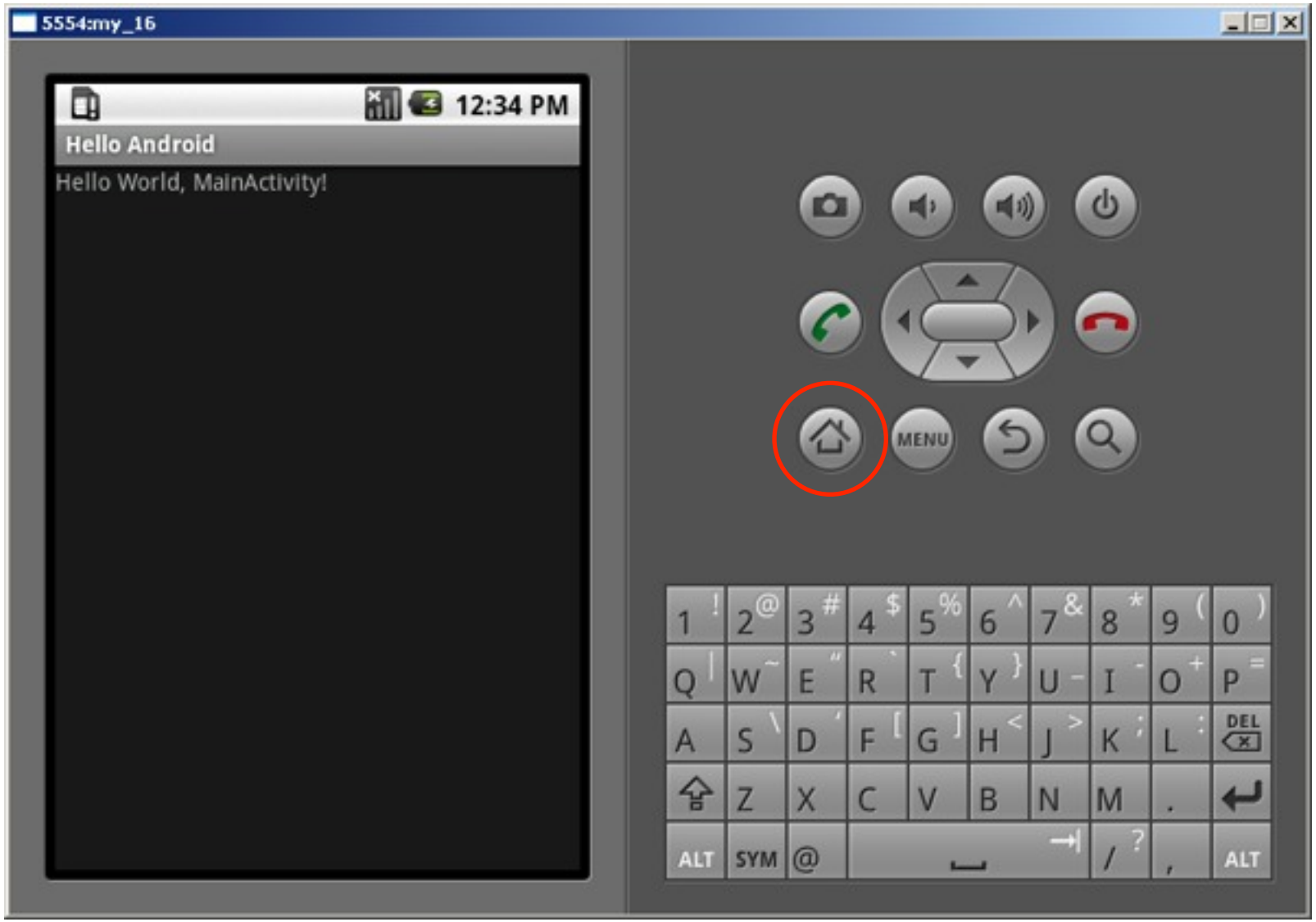


Uniquely identifies the application!


Java - HelloAndroid/src/de/tuberlin/tlabs/MainActivity.java - Eclipse
File Edit Run Source Navigate Search Project Refactor Window Help
Package Explorer
Hierarchy
HelloAndroid
src
de.tuberlin.tlabs
MainActivity.java
gen [Generated Java Files]
de.tuberlin.tlabs
R.java
Android 1.6
assets
res
drawable
icon.png
layout
main.xml
values
strings.xml
AndroidManifest.xml
default.properties
MainActivity.java
main.xml
package de.tuberlin.tlabs;
import android.app.Activity;
public class MainActivity extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
}
}

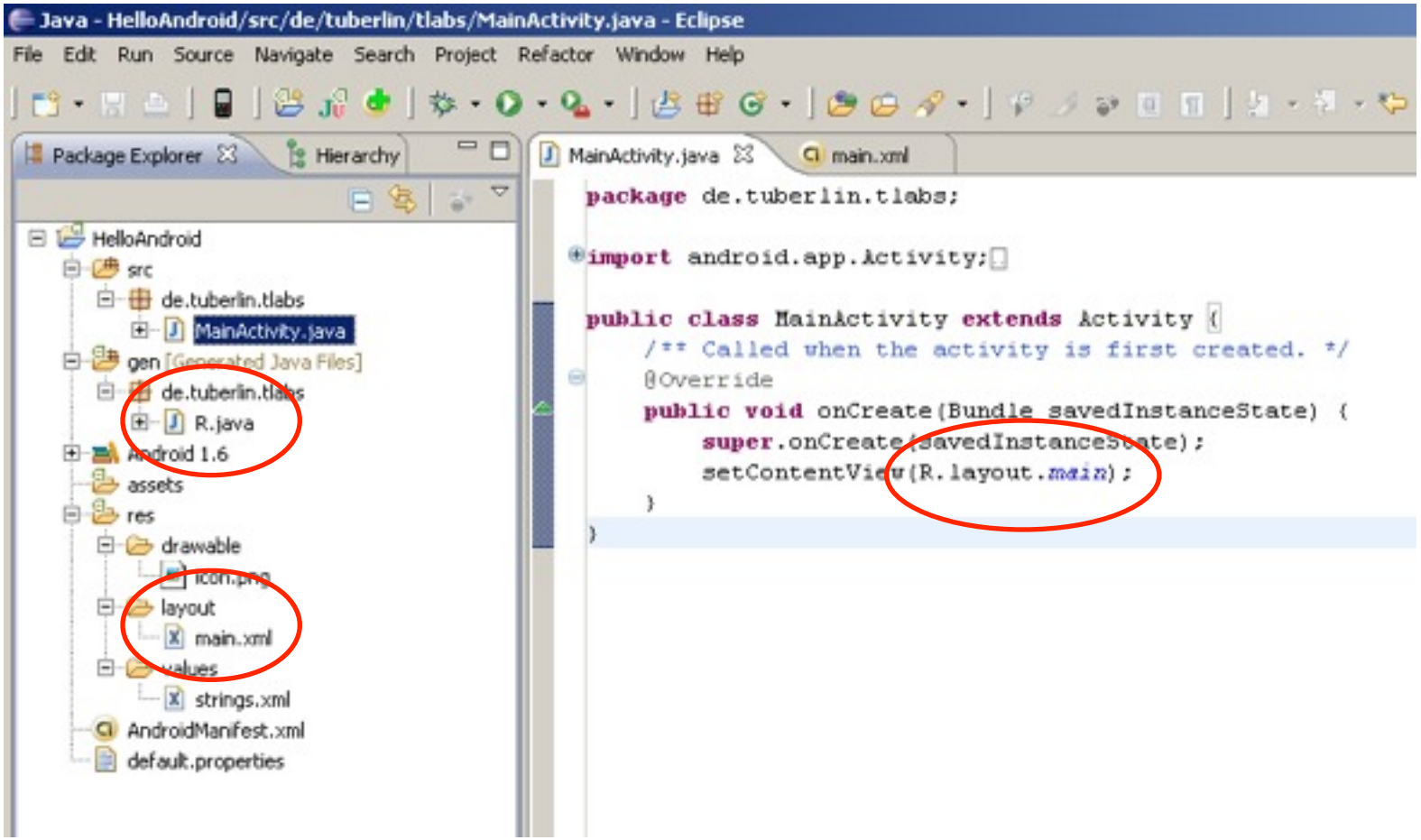
Define Missing Android Virtual Device




5554:my_16
12:34 PM
Hello Android
Hello World, MainActivity!
MENU

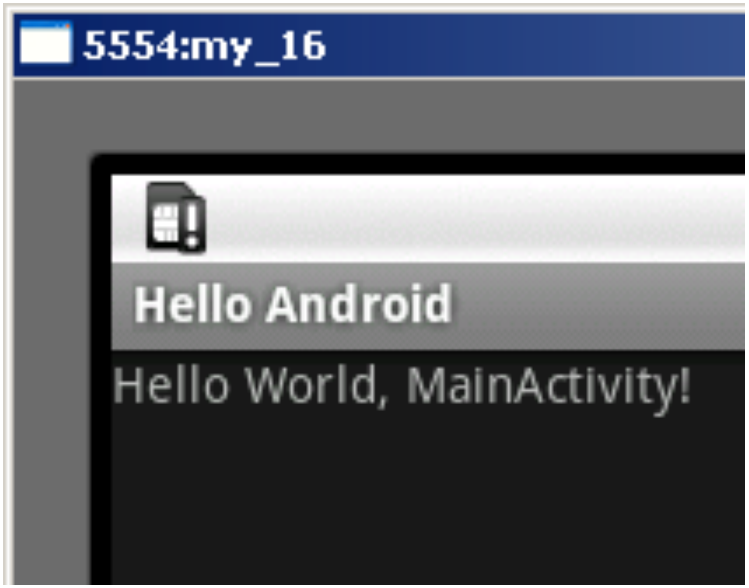

5554:my_16
12:35 PM
Google
Messaging
Dialer
Contacts
Browser
MENU
ALT
SYM
ALT
DEL


5554:my_16
12:35 PM
Google
Alarm Clock
Browser
Calculator
Camcorder
Camera
Contacts
Custom Locale
Dev Tools
Dialer
Email
Gallery
Gestures Builder
Hello Android
Messaging
Music
Settings
MENU
ALT
SYM
DEL


Java - HelloAndroid/src/de/tuberlin/tlabs/MainActivity.java - Eclipse
File Edit Run Source Navigate Search Project Refactor Window Help
Package Explorer
Hierarchy
HelloAndroid
src
de.tuberlin.tlabs
MainActivity.java
gen [Generated Java Files]
de.tuberlin.tlabs
R.java
Android 1.6
assets
res
drawable
icon.png
layout
main.xml
values
strings.xml
AndroidManifest.xml
default.properties
MainActivity.java
main.xml
package de.tuberlin.tlabs;
import android.app.Activity;
public class MainActivity extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
}
}

Declarative definition of UIs
main.xml

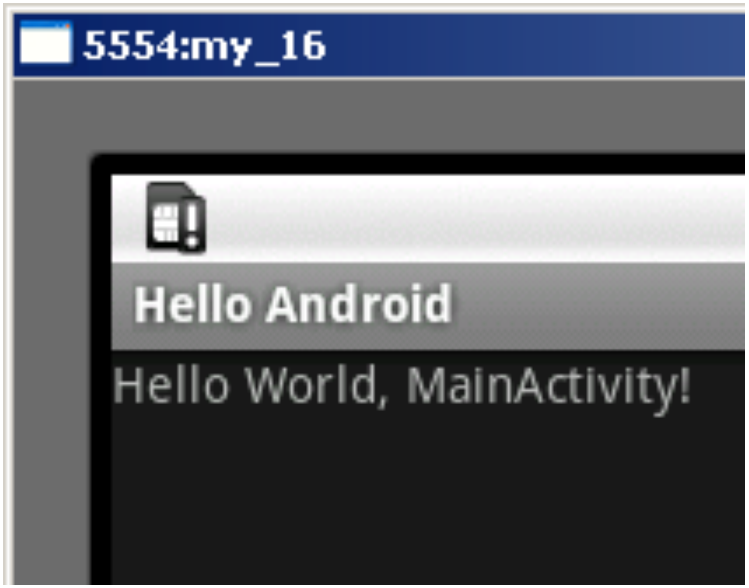
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"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



Separating text strings from code strings.xml

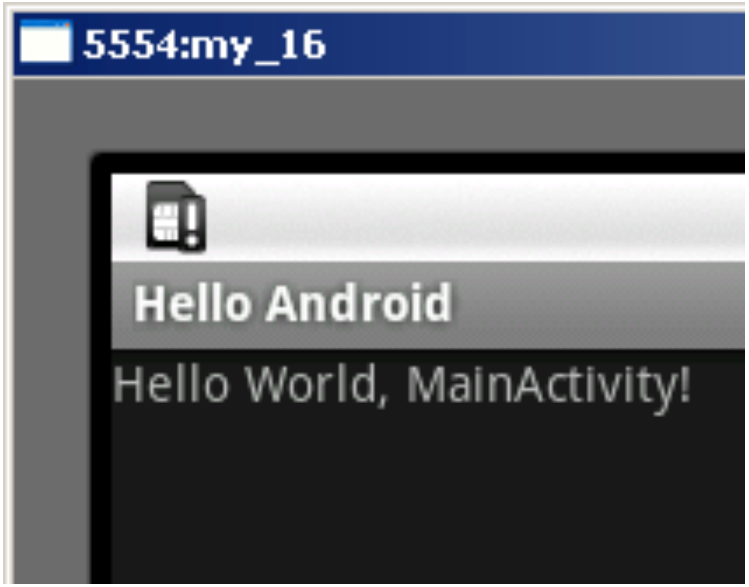
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, MainActivity!</string>
    <string name="app_name">Hello Android</string>
</resources>
```

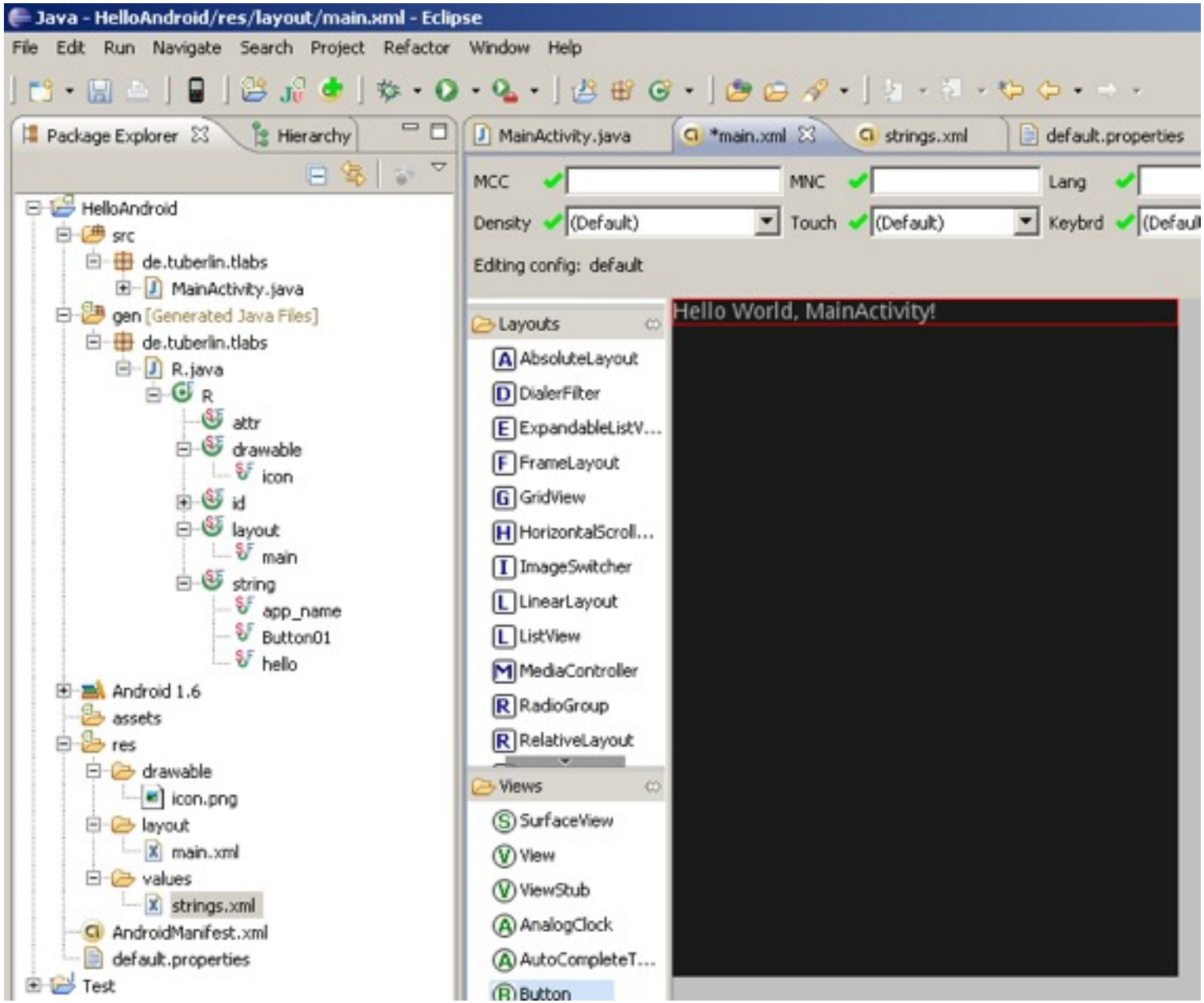
- Default language in res/values/strings.xml
- Localized languages in res/values-xx (language qualifier)
  - French in res/values-fr/strings.xml
  - Hindi in res/values-hi/strings.xml
  - Etc.

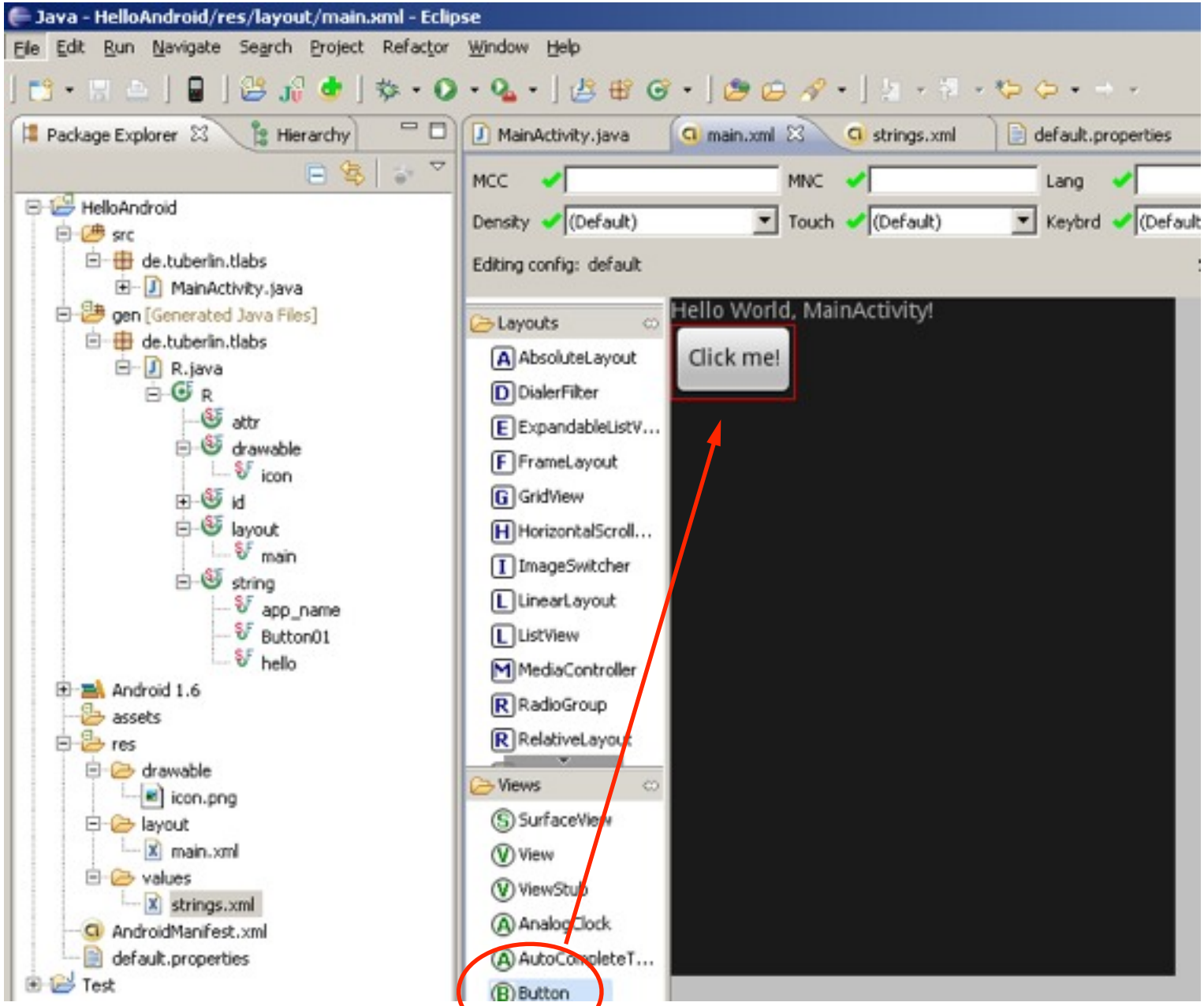


R.java

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package de.tuberlin.tlabs;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int Button01=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int Button01=0x7f040002;
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```




Java - HelloAndroid/res/layout/main.xml - Eclipse
File Edit Run Navigate Search Project Refactor Window Help
Package Explorer
Hierarchy
HelloAndroid
src
de.tuberlin.tlabs
MainActivity.java
gen [Generated Java Files]
de.tuberlin.tlabs
R.java
R
attr
drawable
icon
id
layout
main
string
app_name
Button01
hello
Android 1.6
assets
res
drawable
icon.png
layout
main.xml
values
strings.xml
AndroidManifest.xml
default.properties
Test
MainActivity.java
*main.xml
strings.xml
default.properties
MCC
MNC
Lang
Density
(Default)
Touch
(Default)
Keybrd
Editing config: default
Layouts
AbsoluteLayout
DialerFilter
ExpandableListV...
FrameLayout
GridView
HorizontalScroll...
ImageSwitcher
LinearLayout
ListView
MediaController
RadioGroup
RelativeLayout
Views
SurfaceView
View
ViewStub
AnalogClock
AutoCompleteT...
Button
Hello World, MainActivity!

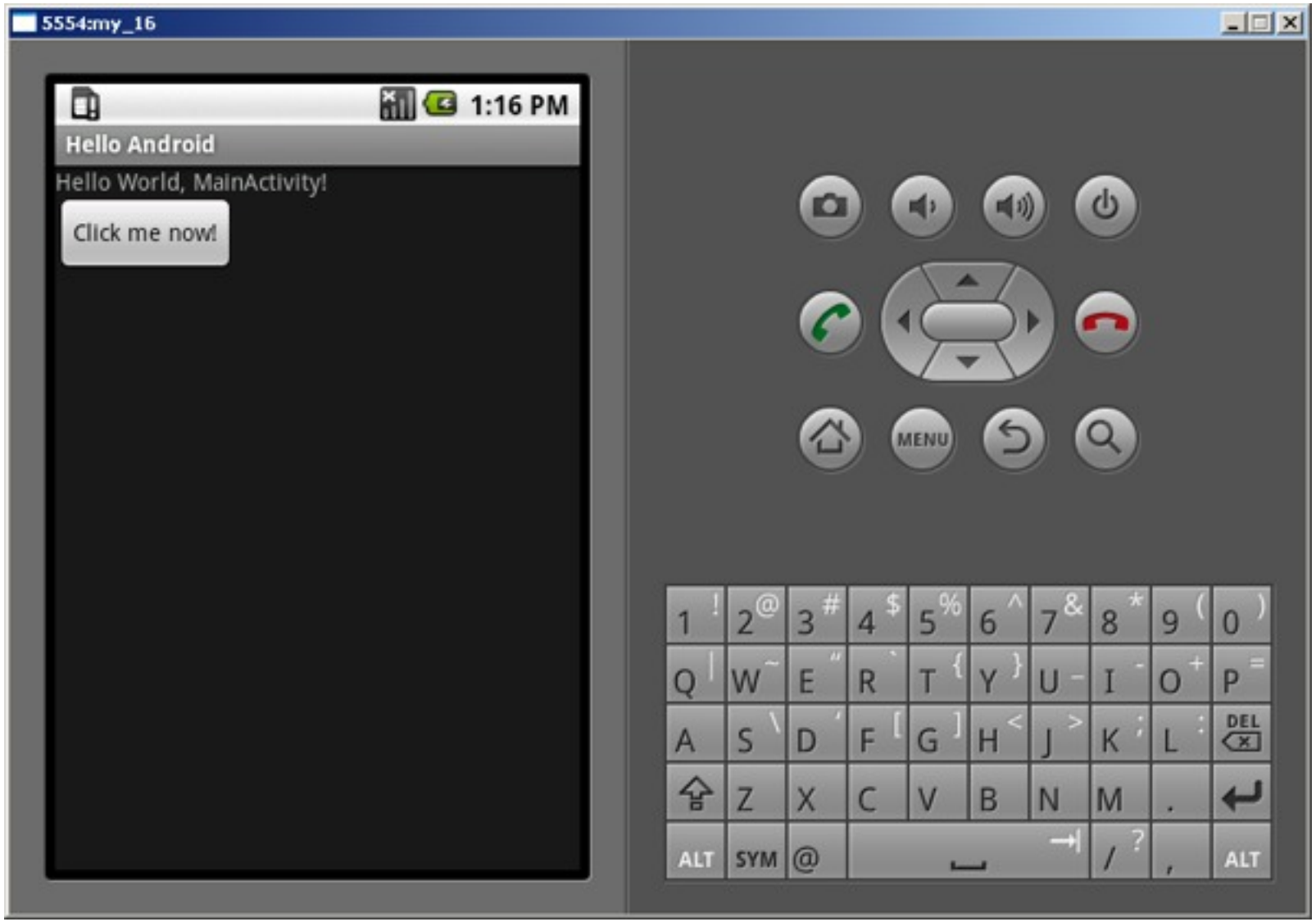

Java - HelloAndroid/res/layout/main.xml - Eclipse
File Edit Run Navigate Search Project Refactor Window Help
Package Explorer
Hierarchy
HelloAndroid
src
de.tuberlin.tlabs
MainActivity.java
gen [Generated Java Files]
de.tuberlin.tlabs
R.java
R
attr
drawable
icon
id
layout
main
string
app_name
Button01
hello
Android 1.6
assets
res
drawable
icon.png
layout
main.xml
values
strings.xml
AndroidManifest.xml
default.properties
Test
MainActivity.java
main.xml
strings.xml
default.properties
MCC
MNC
Lang
Density
(Default)
Touch
(Default)
Keybrd
Editing config: default
Layouts
AbsoluteLayout
DialerFilter
ExpandableListV...
FrameLayout
GridView
HorizontalScroll...
ImageSwitcher
LinearLayout
ListView
MediaController
RadioGroup
RelativeLayout
Views
SurfaceView
View
ViewStub
AnalogClock
AutoCompleteT...
Button
Hello World, MainActivity!
Click me!

Declarative Definition of UIs
main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<Button
    android:text="@string/Button01"
    android:id="@+id/Button01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
</LinearLayout>
```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, MainActivity!</string>
    <string name="app_name">Hello Android</string>
    <string name="Button01">Click me now!</string>
</resources>
```


5554:my_16
1:16 PM
Hello Android
Hello World, MainActivity!
Click me now!
MENU

Define the contents of the application AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.tuberlin.tlabs"
    android:versionCode="1"
    android:versionName="1.0">
  <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".MainActivity“android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
  <uses-sdk android:minSdkVersion="4" />
</manifest>
```

Uniquely identifies the application!

Add for android:debuggable="true" on-device debugging!

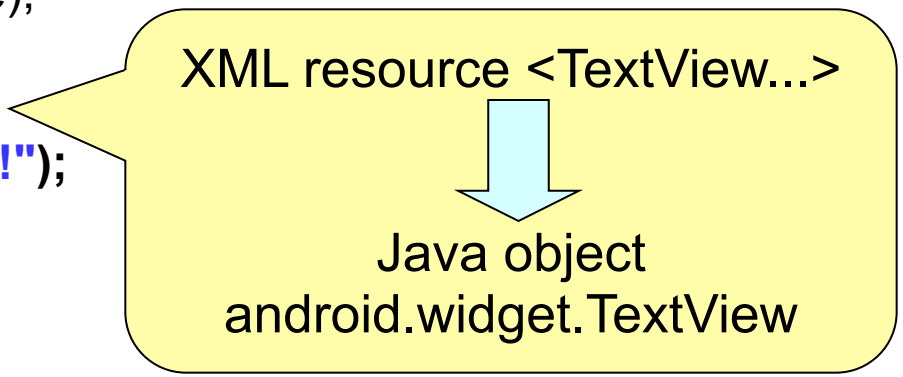
- Initial activity of application
- Listed in application launcher

UI from XML resources
MainActivity.java

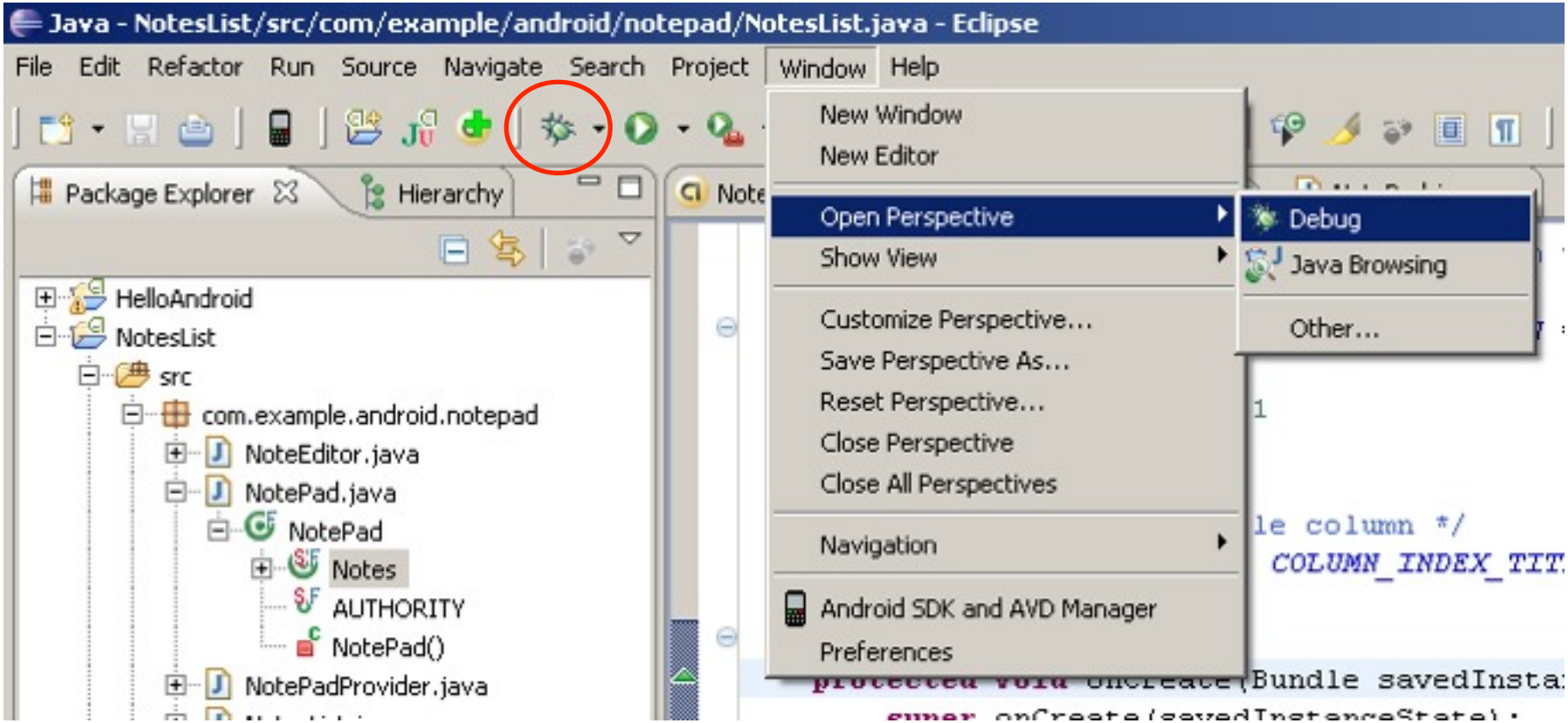
```
package de.tuberlin.tlabs;

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

public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

UI programmatically defined MainActivity.java

```
package de.tuberlin.tlabs;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
//        setContentView(R.layout.main);
        TextView tv = new TextView(this);
        tv.setText("Hello World (TextView)!");
        setContentView(tv);
    }
}
```

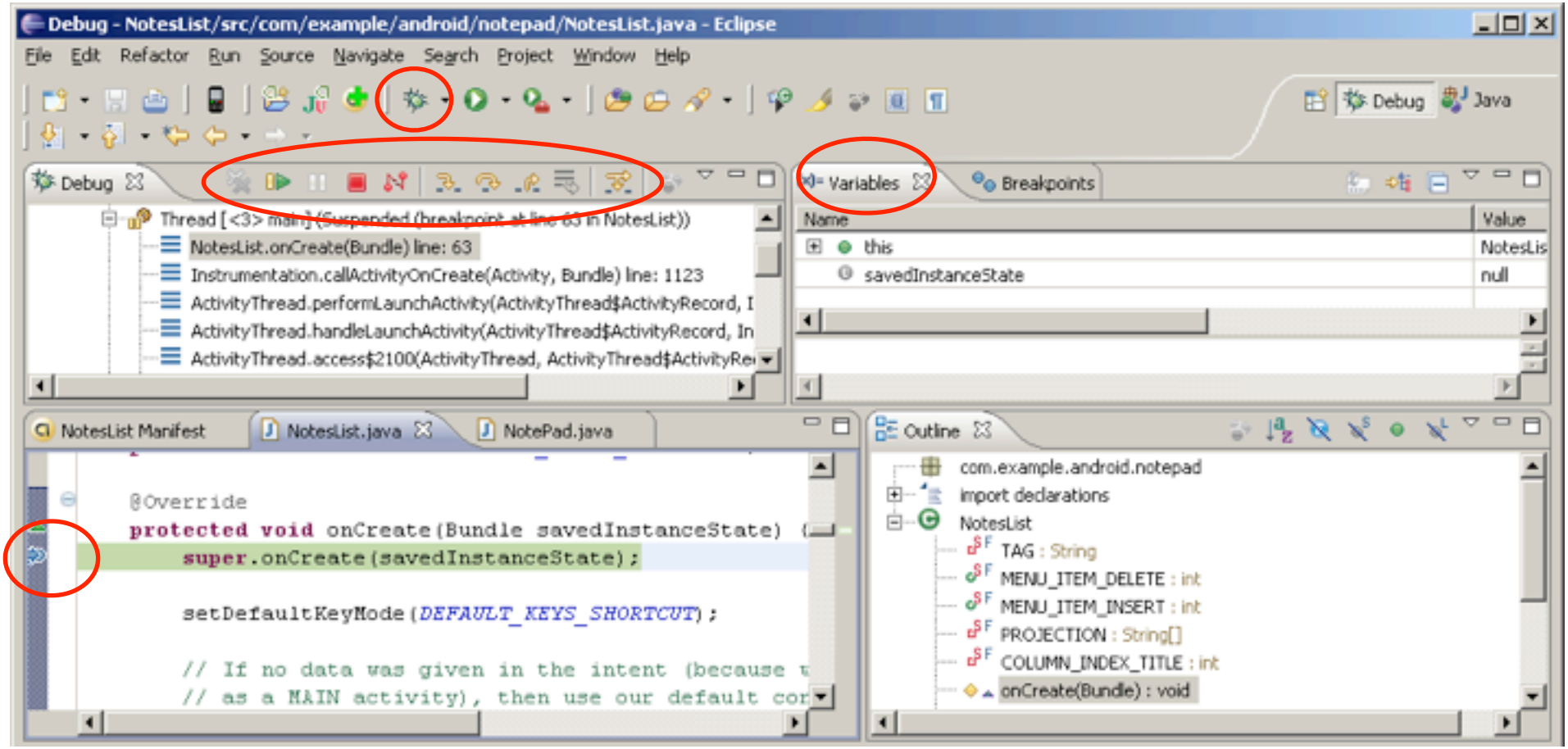


Eclipse Perspectives

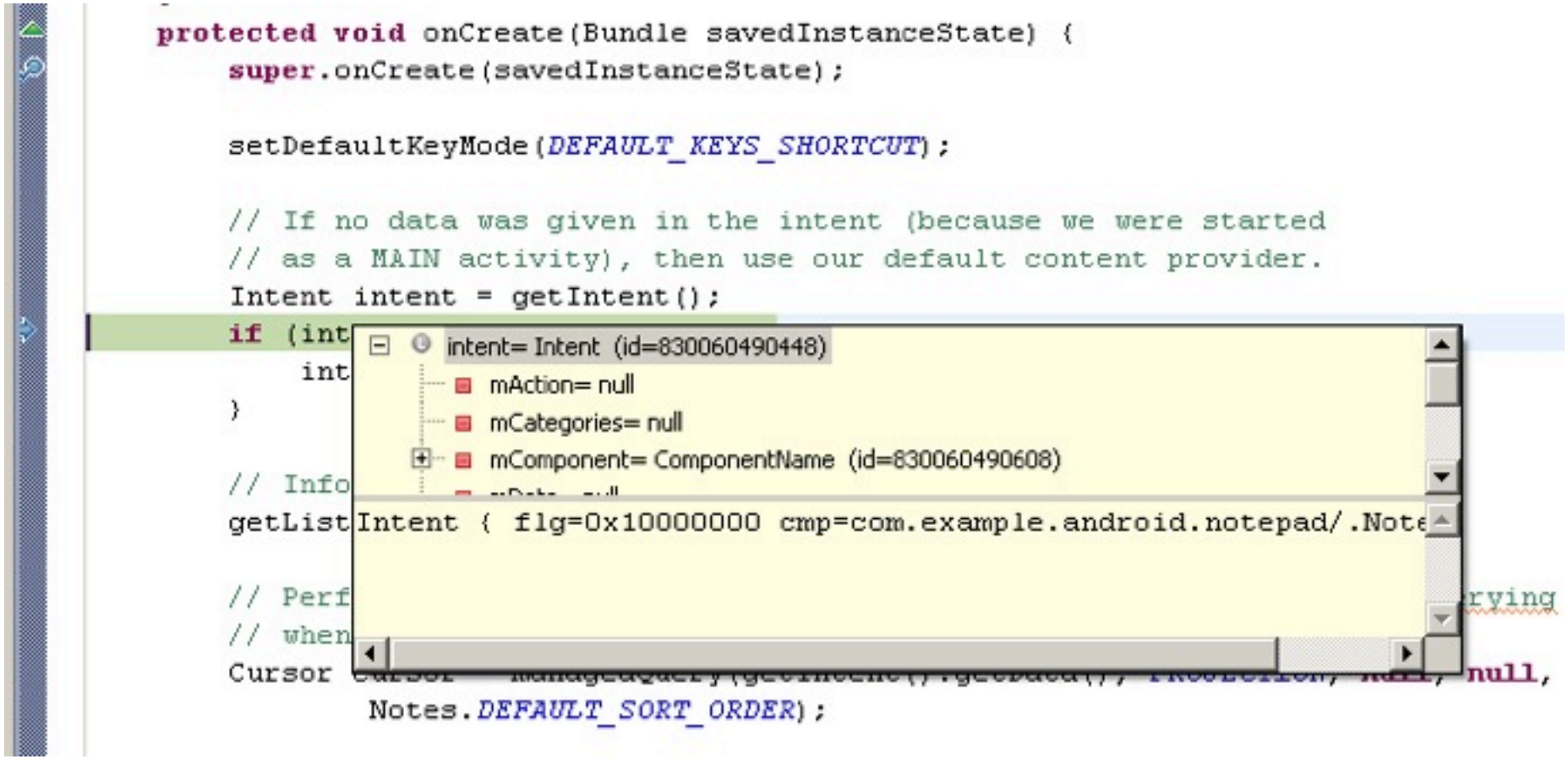


Debugging in the Emulator

- Set Breakpoint with Ctrl+Shift+B (⌘ +Shift+B)
- Step through code with F5, F6, F7 (*fn* + F5, F6, F7)

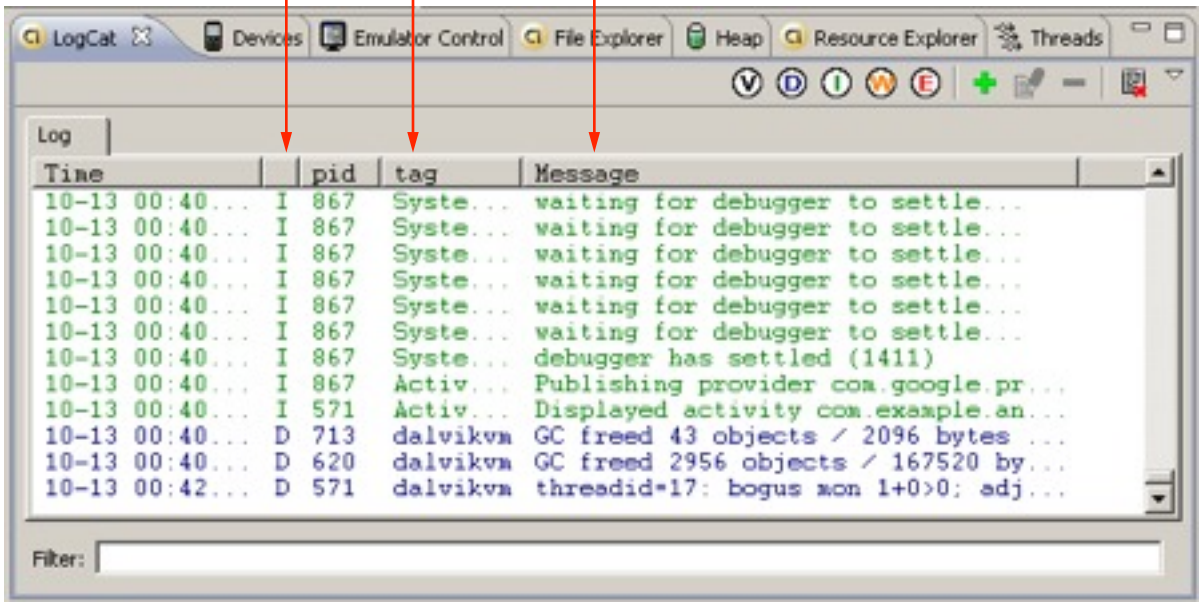


Inspecting Variables



Logging and Tracing

- android.util.Log
  - informational, warning, error methods
  - Example: *Log.d(TAG, "getAddress: " + s);*
- android.os.Debug
  - Debug.startMethodTracing
  - Debug.stopMethodTracing
  - trace viewer tool
- File explorer tool to view files on the device



Concepts so far

- Project directory structure
  - src, gen, assets, res, AndroidManifest.xml
- Resources
  - Declarative view definitions in XML
  - Localization of string resources
  - Selection of icons/images based on resolution
  - Resource identifiers
- Application as collection of Activities
- Manifest file

Activity and Application Lifecycle

Applications

- Default: Application ⇔ Linux process ⇔ Virtual Machine
- Each application has a unique Linux user ID
  - Application files only accessible by this Linux user ID
- Applications can share a user ID
  - Applications with the same ID can share a process/VM
- Application components
  - Activities
  - Services
  - Broadcast receivers
  - Content providers
- Components can register their capabilities with the system
  - Declared in manifest file
  - Example: Barcode recognition service for other application

Activities

- Independent components of the application
  - Components “crash” individually
- Represent data and behavior of one View
  - Roughly: the model and controller of the MVC pattern
- Example: text messaging application
  - Activity 1 shows list of contacts
  - Activity 2 to write a message to a chosen contact
  - Activity 3 to review sent messages
- View of an Activity typically fills the screen
  - Views grouped in hierarchy
  - Parents control layout of children
  - Leaf view react to user actions
  - Associate root view with activity: activity.*setContentView(view id);*

Services

- Application component without a user interface
- Runs in the background and performs some task
- Example: Downloading data from the network
- Local services: invoked from the same process
- Remote services: invoked from other processes
  - But: from same device
  - Android Interface Definition Language (AIDL)
  - Remote Procedure Call (RPC)
  - Exposing service to clients: declaration in manifest file

Broadcast Receivers

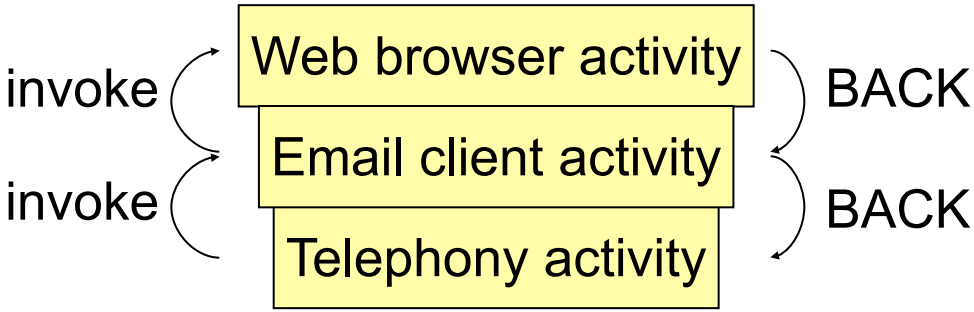
- Application component that receives and reacts to broadcasts
  - No user interface
- System receives and dispatches broadcasts
- Example broadcasts
  - From System: Timezone changed, battery low, language setting changed
  - From an applications: download finished
- Reaction to broadcast
  - Post a notification to the status bar → NotificationManager
  - Start an activity with a user interface
  - Etc.

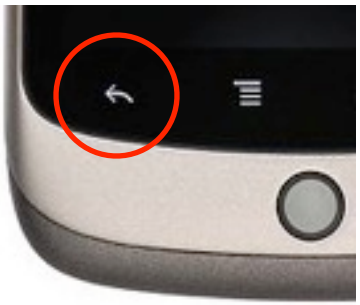
Content Providers

- Common interface for querying an application's data
  - Images, contact information, notes, emails, etc.
  - Content provider defines public URI
  - Expose data as rows and columns of a table
- Data sources (not exposed)
  - File system
  - SQLite database
  - Network
- Content resolvers
  - Dynamic lookup of content provider based on URI
  - Example: content://com.google.provider.NotePad/notes/3

Tasks

- Task: what the user experiences as an “application”
  - Notion of an “application” blurry in component-based system
  - Tasks can span multiple activities and applications
- Example scenario for a task
  - User talks on the phone, looks up an email to answer a question, follows a link to a Web page with the desired information
  - Talk on phone: telephony application
  - Look up email: email client
  - Reading Web page: web browser
- Activity stack of a task:



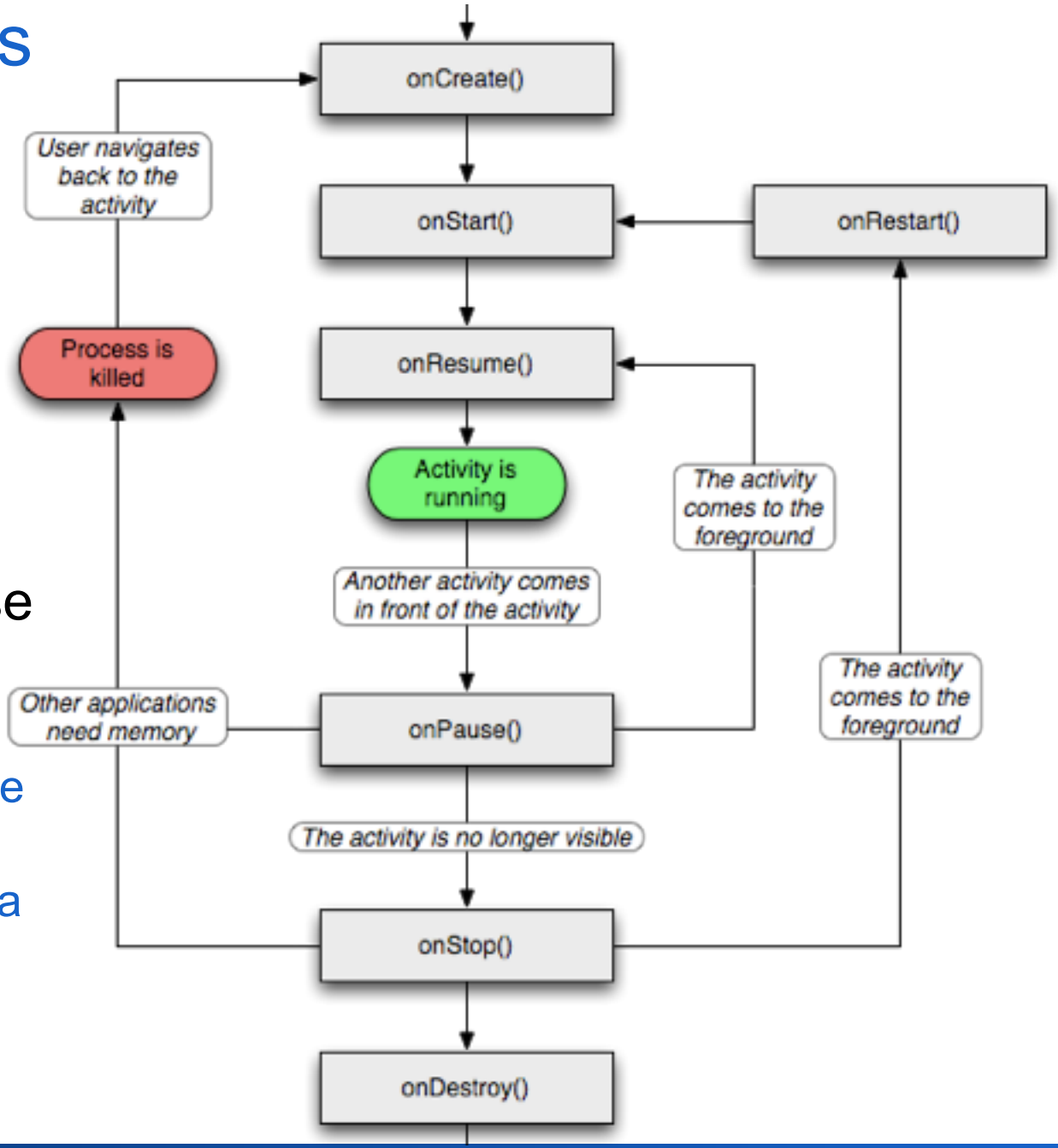


Activity Lifecycle

- Managed by system based on resources and user needs
- States
  - Running: in foreground (at top of activity stack)
  - Paused: partially visible, lost focus (e.g. dialog on top)
  - Stopped: invisible
- Lifecycle callback methods of an Activity
  - **protected void** onCreate(Bundle savedInstanceState);
  - **protected void** onStart();
  - **protected void** onRestart();
  - **protected void** onResume();
  - **protected void** onPause();
  - **protected void** onStop();
  - **protected void** onDestroy();

State Transitions of an Activity

- Use callback methods to manage state and resources of the activity
- Example: onPause
  - Stop OpenGL screen updates
  - Stop game engine updates
  - Stop sending data via the network



Intents

Intents

- Intents are
  - Messages to the system
  - (Passive) representations of an operation to be performed
  - “Glue” between activities
  - Enable late runtime binding across applications
- Primary pieces: action and data
  - Example: action: ACTION_VIEW, data: URI to view
- Intents used to
  - Invoke other applications
  - Raise events (→ publish-and-subscribe)
  - Represent actions to be performed in the future
- Intent registry
  - http://www.openintents.org

Example: Invoking an Activity

- **Activity to be invoked**

```
public class BasicActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
} }
```

- **In AndroidManifest.xml**

```
<activity android:name="BasicActivity" android:label="My Basic Activity">
    <intent-filter>
        <action android:name="com.andbk.intent.action.ShowBasicView" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

- **From another activity**

```
Intent intent = new Intent("com.andbk.intent.action.ShowBasicView");
startActivity(intent);
```

Available Intents in Android

- Available intents
  - Browser: open a browser window
  - Dialer: calling phone numbers
  - Google Maps: open to the given location
  - Google Streetview: open to the given location
- Intents list
  - http://developer.android.com/guide/appendix/g-app-intents.html
- Examples

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://www.lmu.de"));
startActivity(intent);

Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("geo:52.5127,13.3210?z=17"));
startActivity(intent);
```

Intent Resolution

- Intent resolution maps Intent to component
- If multiple possible receivers, shows selection list

- Matching Intent against all <intent-filter> descriptions in all installed application packages
- Information used for resolution
  - Action
  - Category
  - MIME type / scheme

Matching Intents to Activities

- Generic action ACTION_VIEW

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://www.lmu.de"));
startActivity(intent);
```

- Intent registration names scheme

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <data android:scheme="http" />
        <data android:scheme="https" />
    </intent-filter>
</activity>
```

Matching Intents to Activities

- Other data attributes
  - host, mimeType, port, path, pathPattern, pathPrefix
- Handling a MIME type

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
</intent-filter>
```

- Passing additional information to an intent

```
Bundle b = new Bundle();
// add key/value pairs to bundle
intent.putExtras(b);
```

Explicit Intents

- Invoking an Activity by ComponentName

```
Intent intent = new Intent();
ComponentName cn = new ComponentName
    ("com.android.contacts",
    "com.android.contacts.ContactsEntryActivity");
intent.setComponent(cn);
startActivity(intent);
```

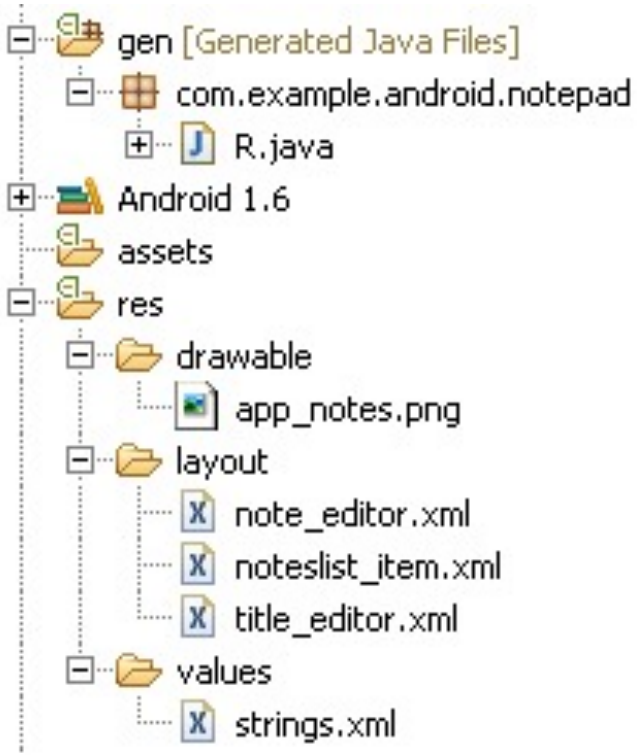
- Invoking an activity by class (is accessible)

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

Resources

Resources

- Declarative definition of UI elements
  - Examples: strings, bitmaps, dialog boxes, audio
- Separate from source code
  - Change resources and code independently
  - Example: localization, look & feel changes
- Resource identifiers → R.java
  - Source code uses resource ID
  - R.java automatically updated



String Resources

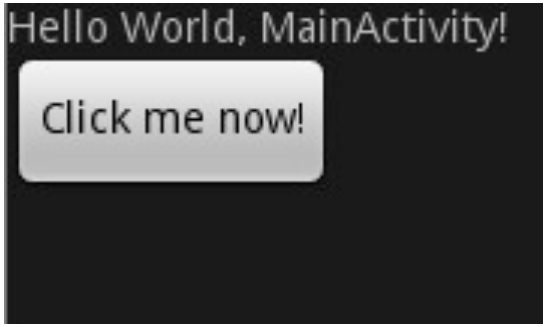
- In /res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Note Pad</string>
    <string name="button_ok">OK</string>
    ...
</resources>
```

- In /gen/<package>/R.java

```
public final class R {
    public static final class string {
        public static final int app_name=0x7f04000b;
        public static final int button_ok=0x7f04000c;
        ...
    }
}
```

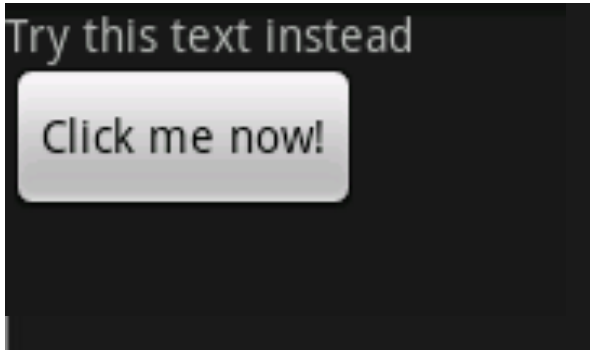
Layout Resources



- View for a screen defined in an XML file
- In /res/layout/main.xml

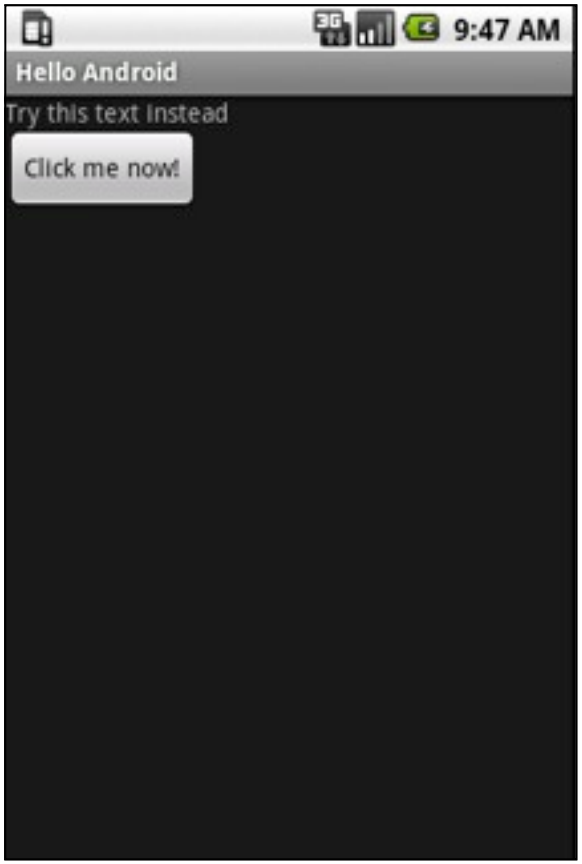
```
<?xml version="1.0" encoding="utf-8"?>
  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/
  android"
   android:orientation="vertical"
   android:layout_width="fill_parent“ android:layout_height="fill_parent“ >
  <TextView
   android:text="@string/hello" /> android:id="@+id/text1"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
  <Button
   android:text="@string/Button01" android:id="@+id/Button01"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content" />
  </LinearLayout>
```

Layout Resources



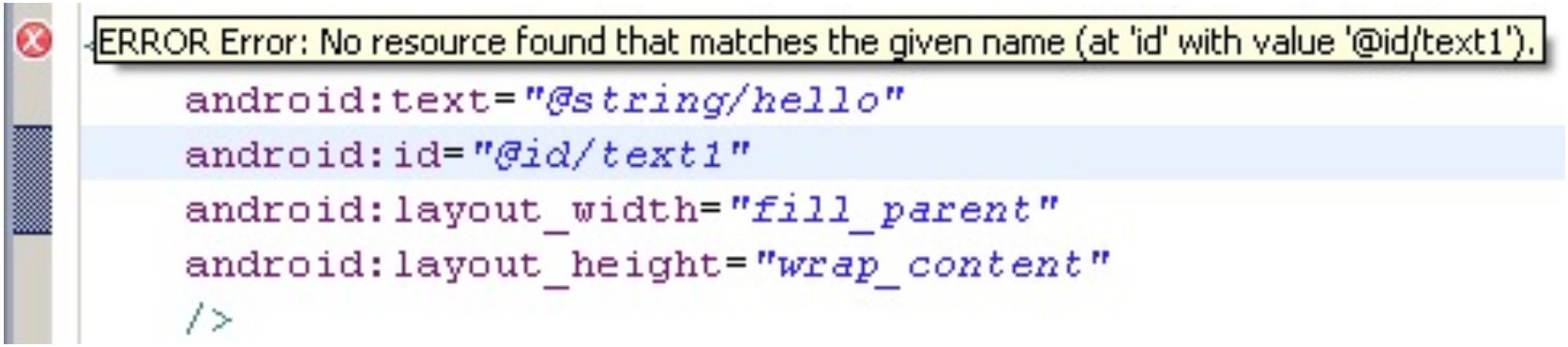
- Instantiated in Java

```
public class MainActivity extends Activity {

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);

    TextView tv = (TextView)
            this.findViewById(R.id.text1);
    tv.setText("Try this text instead");

}
}
```



Resource-Reference Syntax

- “+” Use id if it already exists, otherwise create new id
- @id/text1



- @+id/text1

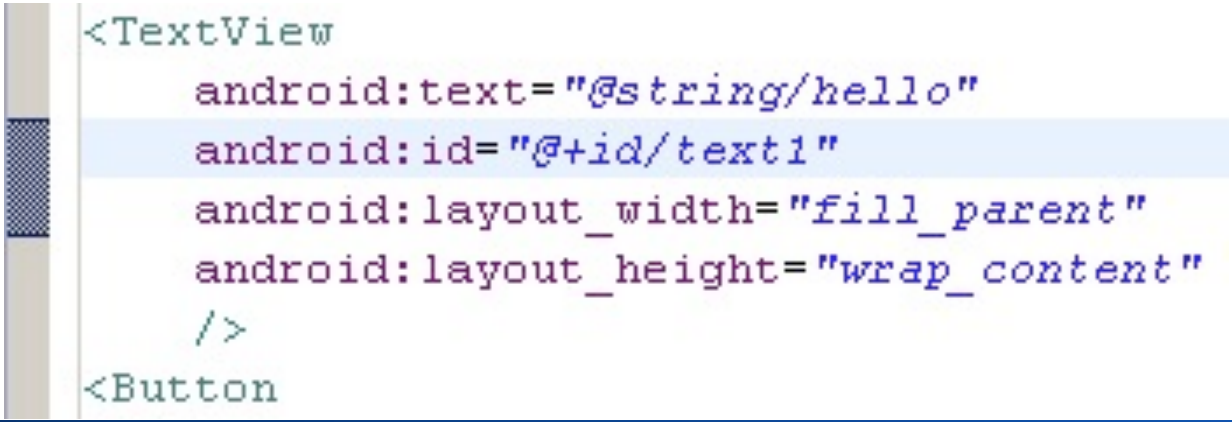
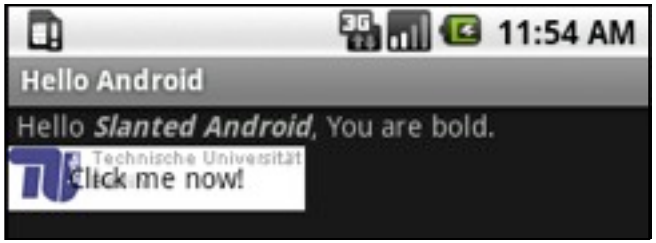


Image Resources

- Automatic id generation for images in /res/drawable
  - Example: /res/drawable/sample_image.jpg
    → R.drawable.sample_image
- Supported types: .gif, .jpg, .png



- Usage in XML

```
<Button android:text="@string/Button01"
...
android:background="@drawable/sample_image" />
```

- Usage in Java

```
Button b = (Button)this.findViewById(R.id.Button01);
b.setBackgroundResource(R.drawable.sample_image);
```

UI Components

- Common Controls
- Layout Managers
- Menus
- Dialogs

Common Controls

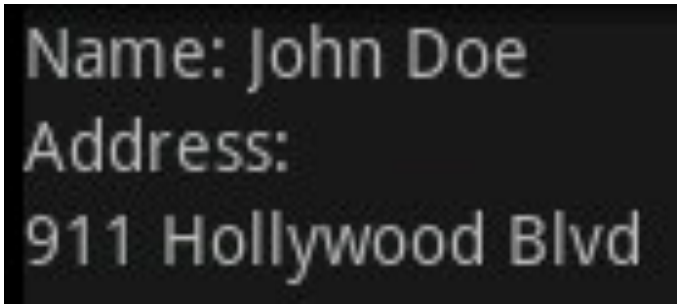
- Predefined user interface elements (“controls”, “widgets”)
  - Define basic interaction patterns
  - Semantics known to users
- Standard widgets
  - Text fields, buttons, lists, grids, date & time controls
- Android-specific controls
  - MapView (display a geographic map)
  - Gallery (display a list of photos)

Core UI Component Classes

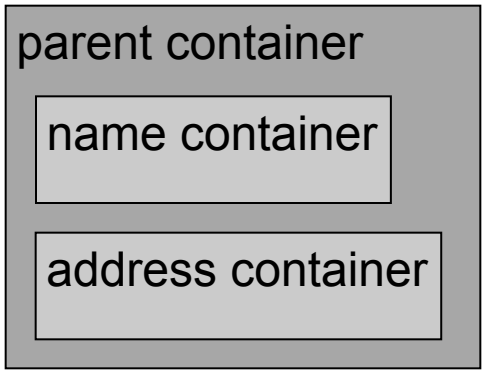
java.lang.Object
↑ android.view.View
↑ android.view.ViewGroup
↑ android.widget.LinearLayout

- android.view.View
  - Rectangular area on the screen
  - Responsible for drawing and event handling
  - Base class for widgets (buttons, text fields, etc.)
- android.view.ViewGroup
  - Is a view and contains other views (“container”)
  - Base class for layouts
- Layouts
  - Invisible containers that hold other Views
  - Define their layout properties (position, padding, size, etc.)
  - Example: LinearLayout (horizontal / vertical list of children)

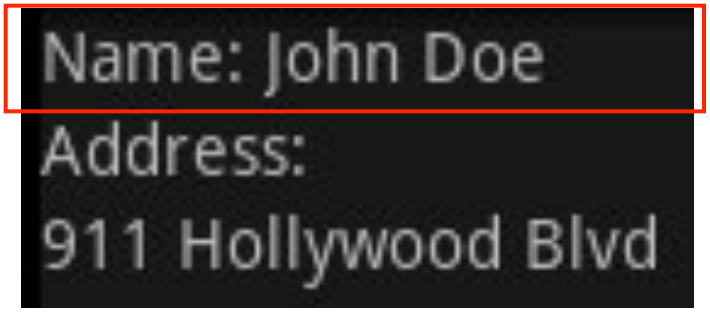
Creating a UI in Java

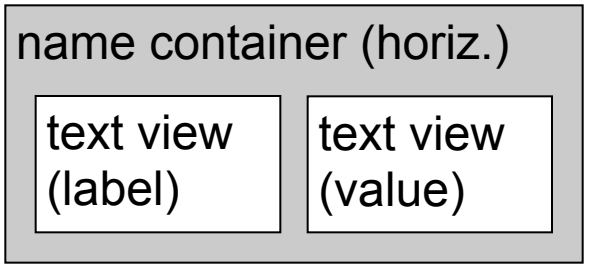



```
package com.androidbook.ch04;

import android.app.Activity;
import android.os.Bundle;
import android.view.ViewGroup.LayoutParams;
import android.widget.LinearLayout;
import android.widget.TextView;

public class MainActivity extends Activity {
    private LinearLayout nameContainer;
    private LinearLayout addressContainer;
    private LinearLayout parentContainer;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        createNameContainer();
        createAddressContainer();
        createParentContainer();
        setContentView(parentContainer);
}
```



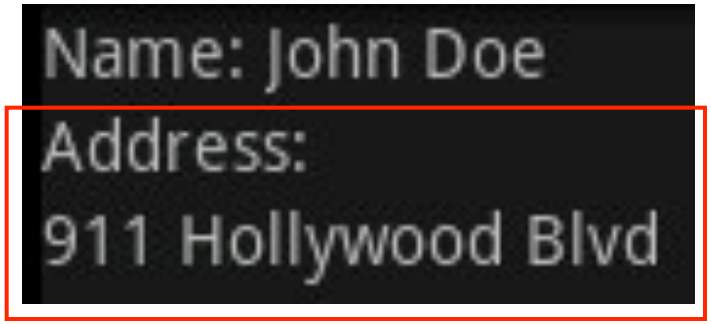
Creating a UI in Java

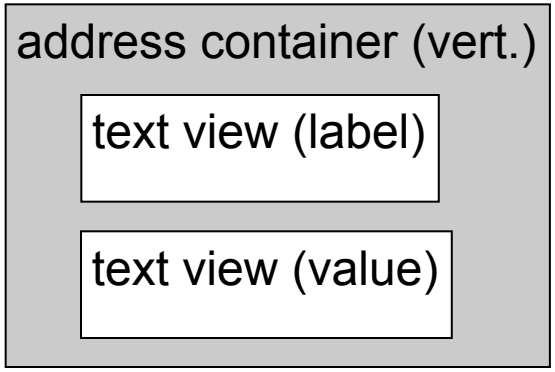





```
private void createNameContainer() {
    nameContainer = new LinearLayout(this);
    nameContainer.setLayoutParams(
        new LayoutParams(
            LayoutParams.FILL_PARENT,
            LayoutParams.WRAP_CONTENT));
    nameContainer.setOrientation(LinearLayout.HORIZONTAL);
    TextView nameLbl = new TextView(this);
    nameLbl.setText("Name: ");
    nameContainer.addView(nameLbl);
    TextView nameValueLbl = new TextView(this);
    nameValueLbl.setText("John Doe");
    nameContainer.addView(nameValueLbl);
}
```

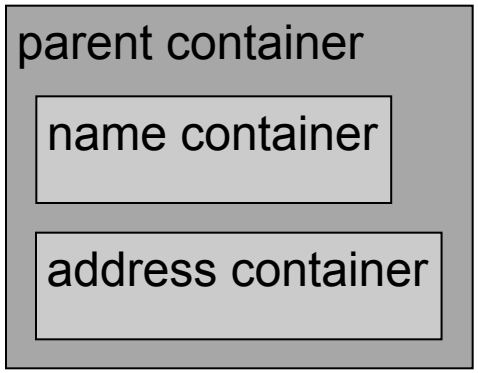
Creating a UI in Java





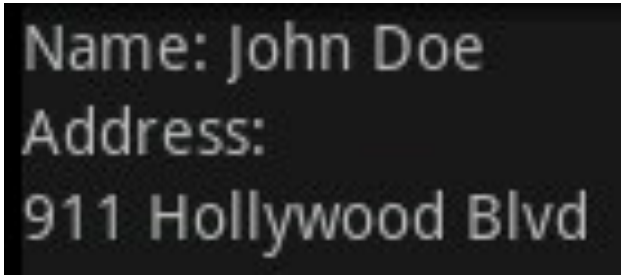

```
private void createAddressContainer() {
    addressContainer = new LinearLayout(this);
    addressContainer.setLayoutParams(
        new LayoutParams(
            LayoutParams.FILL_PARENT,
            LayoutParams.WRAP_CONTENT));
    addressContainer.setOrientation(LinearLayout.VERTICAL);
    TextView addrLbl = new TextView(this);
    addrLbl.setText("Address:");
    TextView addrValueLbl = new TextView(this);
    addrValueLbl.setText("911 Hollywood Blvd");
    addressContainer.addView(addrLbl);
    addressContainer.addView(addrValueLbl);
}
```

Creating a UI in Java




```
private void createParentContainer() {
    parentContainer = new LinearLayout(this);
    parentContainer.setLayoutParams(new LayoutParams(
        LayoutParams.FILL_PARENT,
        LayoutParams.FILL_PARENT));
    parentContainer.setOrientation(LinearLayout.VERTICAL);
    parentContainer.addView(nameContainer);
    parentContainer.addView(addressContainer);
}

}
```

Creating a UI in XML (/res/layout/test.xml)




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="horizontal" android:layout_width="fill_parent"
      android:layout_height="wrap_content">
    <TextView android:layout_width="wrap_content"
      android:layout_height="wrap_content" android:text="Name: " />
    <TextView android:layout_width="wrap_content"
      android:layout_height="wrap_content" android:text="John Doe" />
  </LinearLayout>
  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="wrap_content">
  <TextView android:layout_width="fill_parent"
    android:layout_height="wrap_content" android:text="Address:" />
  <TextView android:layout_width="fill_parent"
    android:layout_height="wrap_content" android:text="911 Hollywood Blvd." />
  </LinearLayout>
</LinearLayout>
```

Setting the XML UI in Java

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

Design UI in XML, Reference in Java

- Assign IDs in XML

```
<TextView android:id="@+id/nameValue" .../>
<TextView android:id="@+id/addrValue" ... />
```

- Refer to controls using IDs

```
TextView nameValue = (TextView) findViewById(R.id.nameValue);
nameValue.setText("John Doe");
TextView addrValue = (TextView)findViewById(R.id.addrValue);
addrValue.setText("911 Hollywood Blvd.");
```

- View must have been loaded before referencing IDs

```
setContentView(R.layout.test);
```

Common Controls

Text Controls

- TextView
  - Display text, no editing
  - Automatic link creation if text contains URLs

    android:autoLink="all“
- EditText
  - Text editing
  - Expands as needed
  - Correct spelling errors

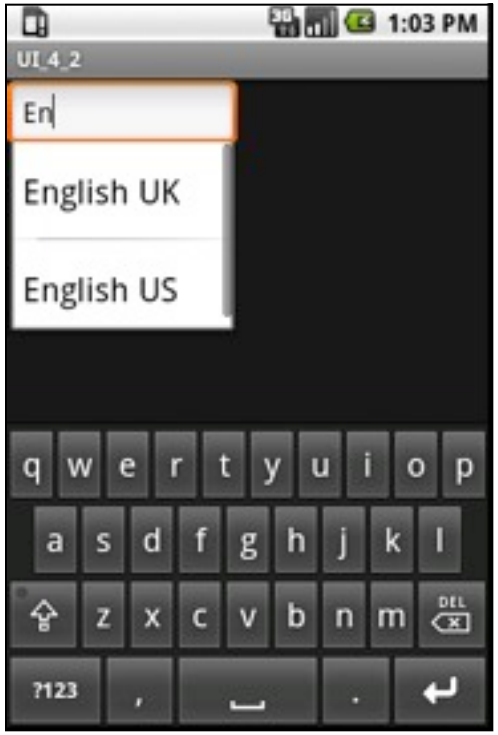
    android:autoText="true"
- AutoCompleteTextView
  - Displays suggestions for word completion
- MultiCompleteTextView
  - Displays suggestions for each word

EditView Input Type

- android:inputType=*"textEmailAddress"*

- android:inputType=*“phone“*

AutoCompleteTextView



- XML

```
<AutoCompleteTextView
    android:id="@+id/auto" ... />
```

- Java

```
AutoCompleteTextView actv =
    (AutoCompleteTextView) findViewById(R.id.auto);
ArrayAdapter<String> aa = new ArrayAdapter<String>(this,
        android.R.layout.simple_dropdown_item_1line,
        new String[] {"English UK", "English US", "Hebrew", "Hindi", ... });
    actv.setAdapter(aa);
```

- Adapter
  - Resource ID for showing a single item
  - The data to use

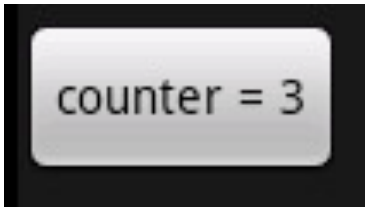
Handling Button Click Events

- XML

```
<Button android:id="@+id/button1" android:text="Basic Button"
android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

- Java

```
public class MainActivity extends Activity implements
    View.OnClickListener {
    public void onCreate(Bundle savedInstanceState) {
        ...
        Button b = (Button) findViewById(R.id.button1);
        b.setOnClickListener(this);
    }

    private int counter = 0;

    public void onClick(View v) {
        Button b = (Button)v;
        b.setText("counter = " + (++counter));
    }
}
```



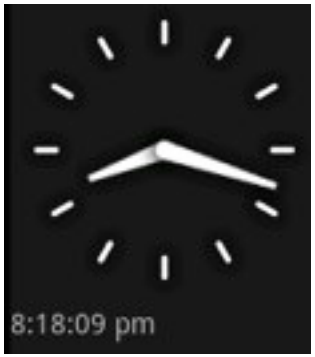
List Controls

- Vertical list of items
- Usage
  - Derive from android.app.ListActivity.ListActivity
  - Set a ListView
  - Setting data for the list view via setListAdapter
- Definition of list item in list_item.xml

```
<LinearLayout ...>
    <CheckBox android:id="@+id/checkbox" ... />
    <TextView android:id="@+id/textview1" ... />
    <TextView android:id="@+id/textview2" ... />
    ...
</LinearLayout>
```

Android Specific Controls

- DatePicker and TimePicker
- AnalogClock and DigitalClock
- MapView
- Gallery

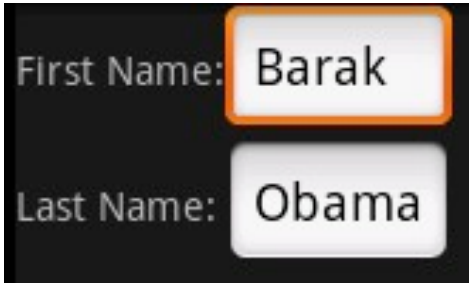


Layout Managers

LayoutManagers

- LayoutManagers
  - Are containers for views (children)
  - Have specific strategy for controlling children's size and position
- Layout Managers in Android
  - LinearLayout: horizontal or vertical arrangement
  - TableLayout: tabular form
  - RelativeLayout: arrange children relative to one another or parent
  - AbsoluteLayout: absolute coordinates
  - FrameLayout: dynamically change controls
- Layout_width and layout_height
  - fill_parent: child wants to fill available space within the parent
  - wrap_content: child wants to be large enough to fit its content

TableLayout



- Extension of LinearLayout
- Example:

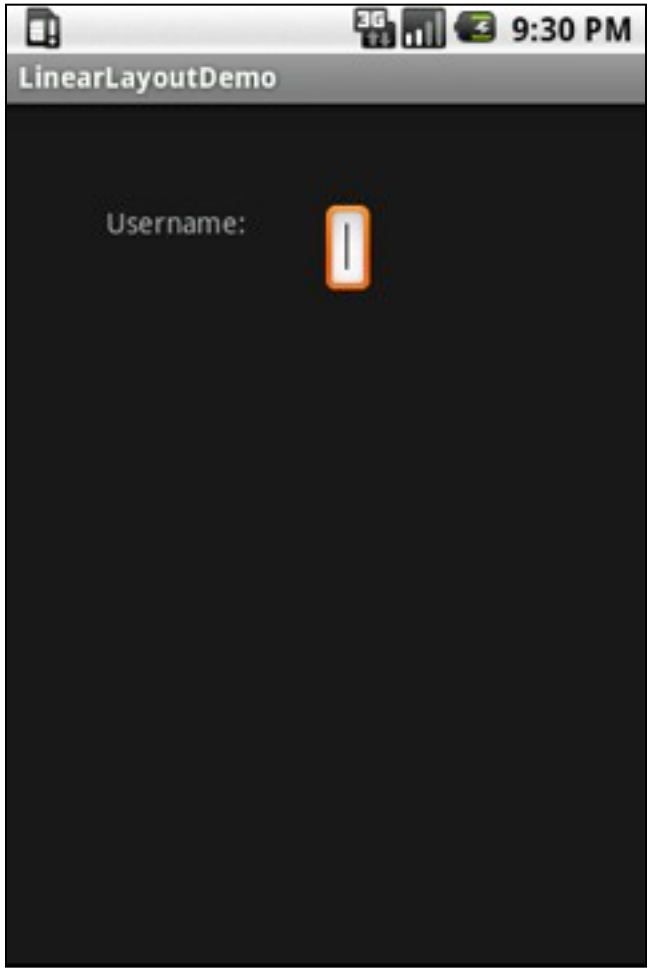
```
<TableLayout android:layout_width="fill_parent"
    android:layout_height="fill_parent">
<TableRow>
    <TextView android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:text="First Name:" />
    <EditText android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:text="Barak" />
</TableRow>
<TableRow>
    <TextView android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:text="Last Name:" />
    <EditText android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:text="Obama" />
</TableRow>
</TableLayout>
```

RelativeLayout


```
<RelativeLayout android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <TextView android:id="@+id/userNameLbl"
        android:text="Username: "
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true" />
    <EditText android:id="@+id/userNameText"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/userNameLbl" />
    <TextView android:id="@+id/disclaimerLbl"
        android:text="Use at your own risk... "
        android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:layout_alignParentBottom="true" />
</RelativeLayout>
```

AbsoluteLayout

```
<AbsoluteLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >
    <TextView android:text="Username:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="50px"
        android:layout_y="50px" />
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="160px"
        android:layout_y="50px" />
</AbsoluteLayout>
```



Screen Configurations

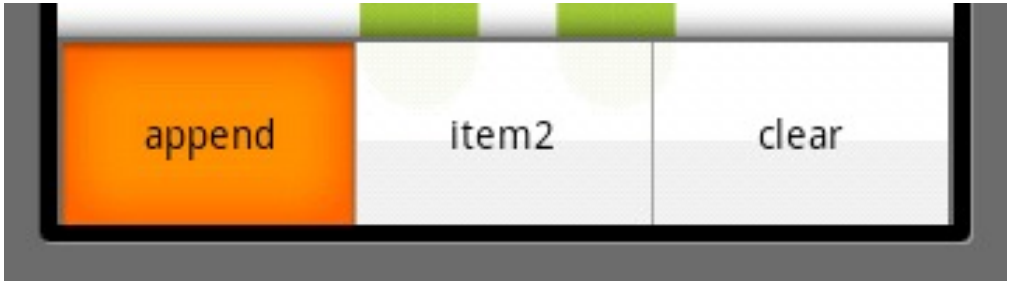
- Configurations
  - Portrait
  - Landscape
  - Square
- Different layouts for different configurations
  - Screen resolutions
- Configuration-specific resource subdirectories
  - /res/layout-port /res/drawable-port
  - /res/layout-land /res/drawable-land
  - /res/layout-square /res/drawable-square
  - /res/layout /res/drawable (default)

Menus

Menus

- An activity is associated with a single menu
- Use onCreateOptionsMenu(Menu m) to populate menu
- Creating an options menu

```
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    menu.add(0, 1, 0, "append"); // group, id, order, title
    menu.add(0, 2, 1, "item2");
    menu.add(0, 3, 2, "clear");
    return true; // return true to enable menu
}
```



Responding to Menu Selection

- Overriding onOptionsItemSelected

```
public boolean onOptionsItemSelected(MenuItem item) {
    Log.d("MainActivity", "menu id = " + item.getItemId() +
            ", title = " + item.getTitle().toString());
    switch (item.getItemId()) {
        case X: // id of handeled item
        // handle item X
        return true;
        ...
    }
}
```

Defining Menus in XML

- XML files in /res/menu
- /res/menu/menu1.xml

```
<menu ...>
<group android:id="@+id/menuGroup_Main">
    <item android:id="@+id/menu_clear"
      android:orderInCategory="10"
      android:title="clear" />
    <item android:id="@+id/menu_dial"
      android:orderInCategory="7"
      android:title="dial" />
  ...
</group>
</menu>
```

- Java

```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu1, menu);
    return true;
}
```

Dialogs

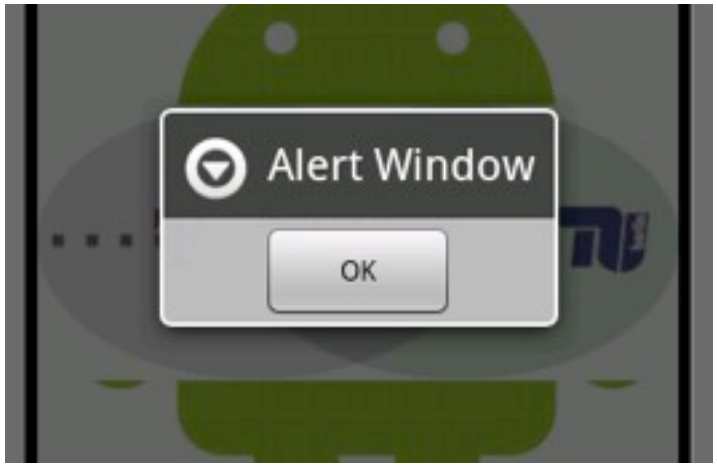
Alert Dialogs

- Alert dialog examples
  - Respond to prompt
  - Pick item or option from list
  - View progress
- Steps
  - Construct android.app.AlertDialog.Builder object
  - Set data (list of items) and parameters (e.g. number of buttons)
  - Set callback methods for buttons
  - Build and show the dialog

Example Alert Dialog

- Java

```
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.menu_testPick) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Alert Window");
        builder.setPositiveButton("OK", listener);
        AlertDialog ad = builder.create();
        ad.show();
        return true;
    }
    ...
}
```



Basic Graphics and Touch Input

Basic Graphics: Drawing

- Screen drawing in View.onDraw
- Canvas class for “draw” calls ← Graphics in Java SE
  - drawRect, drawLines, drawCircle, drawText, etc.
  - Transformation matrix
  - Clipping
- Paint class
  - Describes colors and drawing styles
  - Examples: anti-aliasing, stroke width, text size, etc.
- Bitmap class for offscreen drawing
  - Explicit creation of canvas and bitmap
  - Canvas draws into the bitmap

Method View.onDraw

```
public class MyView extends View {
    private final Rect rect = new Rect();
    private final Paint paint = new Paint();

    public MyView(Context c) {
        super(c);
        paint.setAntiAlias(false);
        paint.setARGB(255, 255, 255, 255);
    }

    protected void onDraw(Canvas c) {
        rect.set(10, 10, 310, 20);
        c.drawRect(rect, paint);
    }
    ...
```

Touch Input: MotionEvent

- Method View.onTouchEvent(MotionEvent e)
- Motion event data
  - x, y, time, action, source, pressure, size
- Sources depend on hardware
  - Mouse, pen, finger, trackball
- Actions
  - ACTION_DOWN
  - ACTION_MOVE
  - ACTION_UP
  - ACTION_CANCEL
- Motion history
  - Sequence of coordinates between events

Touch Input Painting

```
public class TouchPaint extends Activity {

    private MyView myView;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        myView = new MyView(this);
        setContentView(myView);
    }
}
```

Touch Input Painting

```
public class MyView extends View {
    private final Paint paint = new Paint();
    private int x = 0, y = 0;

    public MyView(Context c) {
        super(c);
        paint.setARGB(255, 255, 255, 255);
    }

    protected void onDraw(Canvas c) {
        c.drawCircle(x, y, 3, paint);
    }

    public boolean onTouchEvent(MotionEvent e) {
        x = (int)e.getX(); y = (int)e.getY();
        invalidate();
        return true;
    }
}
```

Touch Input Painting (Off-Screen Image)

```
public class MyView extends View {
...
private Bitmap bitmap;
private Canvas canvas;
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
    bitmap = Bitmap.createBitmap(w, h, Bitmap.Config.RGB_565);
    canvas = new Canvas(bitmap);
}
protected void onDraw(Canvas c) {
    if (bitmap != null) c.drawBitmap(bitmap, 0, 0, null);
}
public boolean onTouchEvent(MotionEvent e) {
    if (canvas != null) {
        int x = (int)e.getX(); int y = (int)e.getY();
        canvas.drawCircle(x, y, 3, paint);
        invalidate();
    }
    return true;
}   }
```