

MMI 2: Mobile Human- Computer Interaction Android (2)

Prof. Dr. Michael Rohs

michael.rohs@ifi.lmu.de

Mobile Interaction Lab, LMU München

Review

- How can UIs be defined in Android?
- What is “R.java”?
- What is “/res”?
- What is “AndroidManifest.xml”?
- What is localization?

ACTIVITIES AND ACTIVITY LIFECYCLES

Applications

- Default: Application \Leftrightarrow Linux process \Leftrightarrow 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

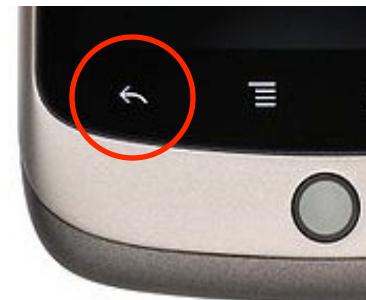
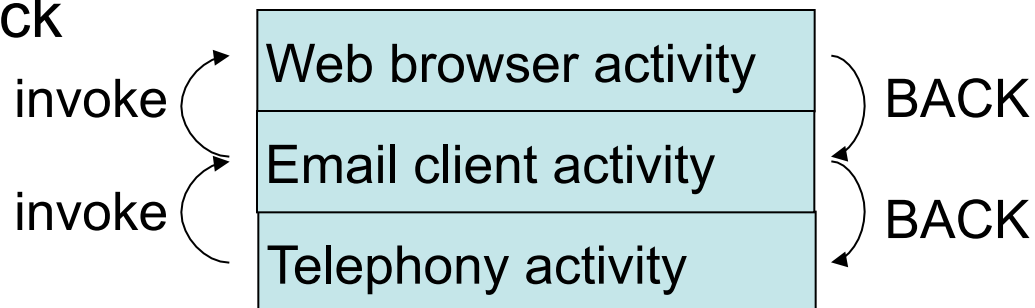
- Common interface for querying an application's data
 - Images, contact information, notes, emails, etc.
 - Content provider defines public URI
 - Expose data as rows and columns of a table
- Data sources (not exposed)
 - File system
 - SQLite database
 - Network
- Content resolvers
 - Dynamic lookup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

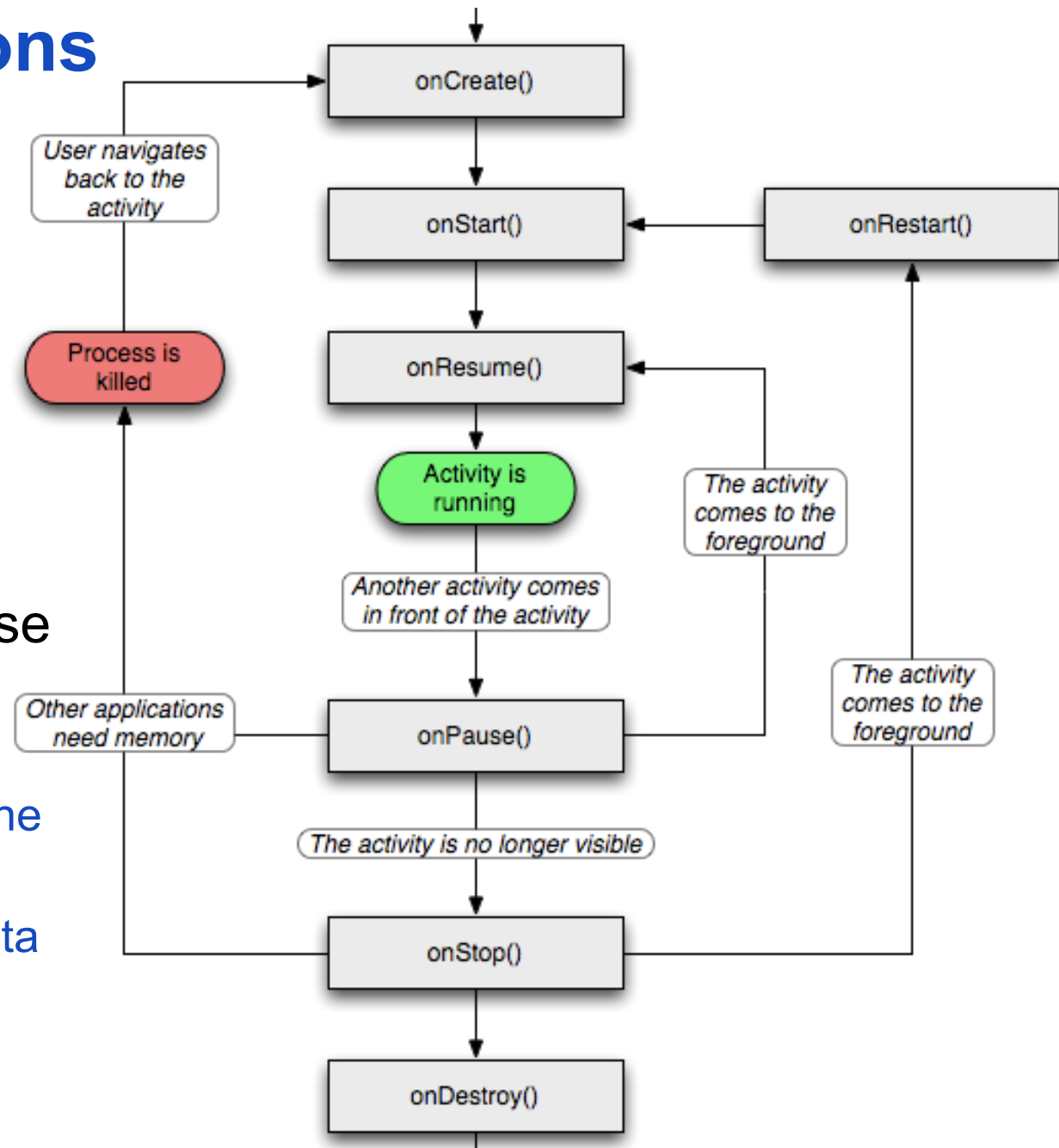


Activity Lifecycle

- 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
 - Represent actions to be performed in the future
 - Register for events (→ publish-and-subscribe)

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="de.lmu.intent.action.ShowBasicView" />  
        <category android:name="android.intent.category.DEFAULT" />  
    </intent-filter>  
</activity>
```

- From another activity

```
Intent intent = new Intent("de.lmu.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

- 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);
```

Intent Categories

- Classifying activities into categories
- Example: CATEGORY_LAUNCHER

```
<intent-filter>
```

```
  <action android:name="android.intent.action.MAIN" />
```

```
  <category android:name="android.intent.category.LAUNCHER" />
```

```
</intent-filter>
```

- Android places icons and names of these activities on the home screen to launch
- Categories
 - CATEGORY_DEFAULT, CATEGORY_TAB, etc.

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.lmu.mobilehci.myapp"
  android:versionCode="1"
  android:versionName="1.0">
```

Uniquely identifies the application!

```
<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" />
```

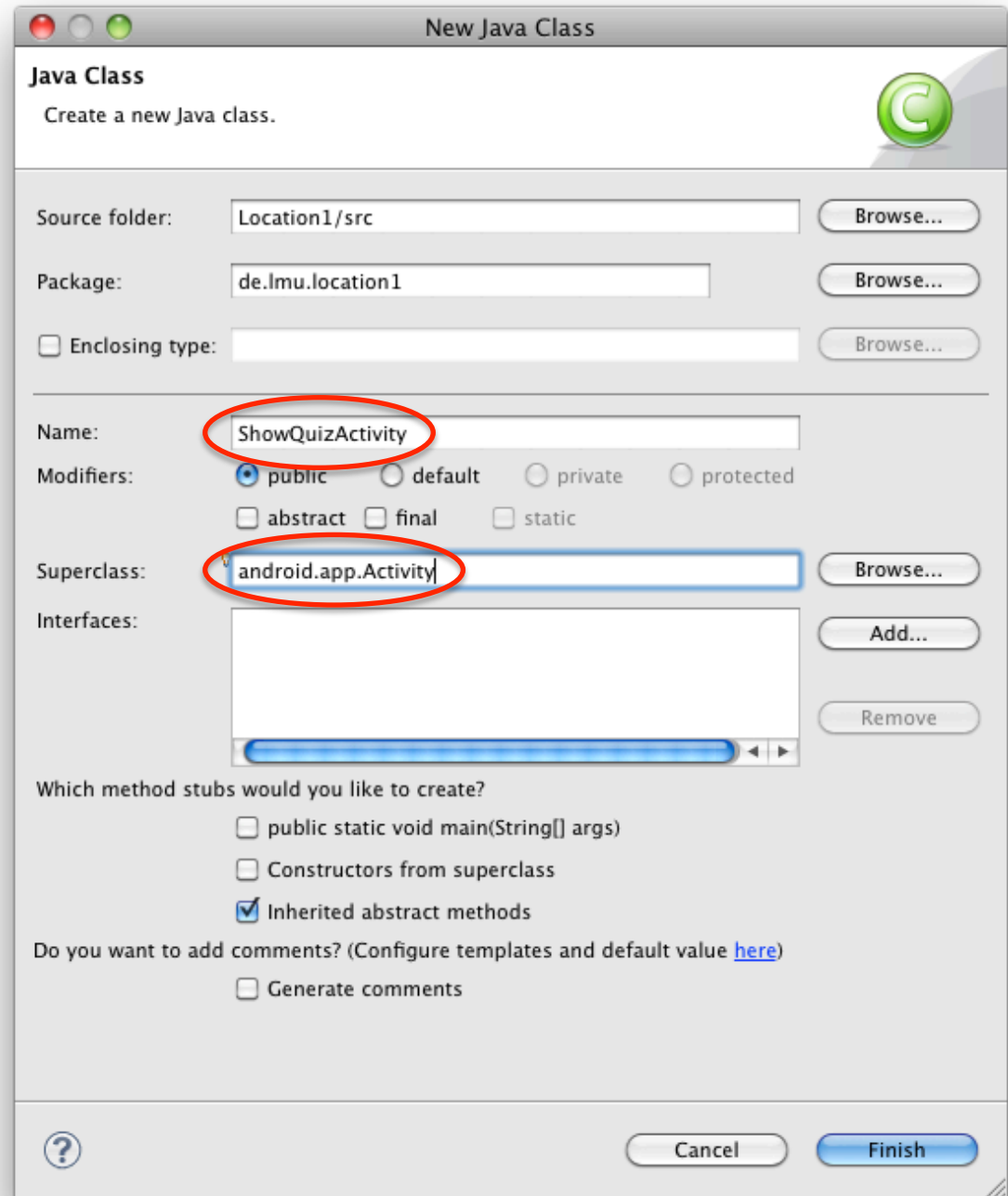
Add for `android:debuggable="true"`
on-device debugging!

- Initial activity of application
- Listed in application launcher

USING ACTIVITIES

Activities

- Create new class ShowQuizActivity
- Superclass: android.app.Activity



ShowQuizActivity → AndroidManifest.xml

- Activity class:

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

- AndroidManifest.xml (inside application element)

```
<activity android:name="de.lmu.quiz.ShowQuizActivity"  
        android:label="showquiz"  
        android:screenOrientation="portrait">  
</activity>
```


How to start the new activity?

- Starting an activity:

```
Intent intent = new Intent(this, ShowQuizActivity.class);
startActivityForResult(intent, requestCode);
```

- Processing the result when the activity returns:

```
void onActivityResult(int requestCode, int resultCode, Intent data) {
    // do something with the result...
}
```

How to return to the previous activity?

- Set result and finish the activity
 `setResult(points);`
 `finish();`

How to copy data from one activity to another?

- Add “extras” to Intent objects

```
Intent intent = new Intent(this, ShowQuizActivity.class);  
intent.putExtra("title", "Target 1");  
intent.putExtra("image", R.drawable.location1);  
startActivityForResult(intent, resultCode);
```

- Can put primitive types and Serializable types into extras
 - `java.io.Serializable` is just a “tagging” interface (no methods)

How to share complex data between activities? (Possibility 1)

- In the calling activity, create a public static member (class variable) that references the shared object
`public static PointOfInterest sharedPoi = null;`
- Before starting the new activity, set the shared object
`Intent intent = new Intent(this, ShowQuizActivity.class);`
`sharedPoi = closestPoi;`
`startActivity(intent);`
- Use original shared object in called activity
`TextView titleView = (TextView) findViewById(R.id.showQuestionTitle);`
`titleView.setText(MainActivity.sharedPoi.title);`

How to share complex data between activities? (Possibility 2)

- Subclass `android.app.Application`, put shared data there

```
public class LocationQuiz extends Application {  
    int points = 0;  
    PointOfInterest currentPoi = null;  
}
```

- Change `AndroidManifest.xml`

```
<application android:name="de.lmu.location.LocationQuiz" ...>  
    ...  
</application>
```

- Access shared data in activities

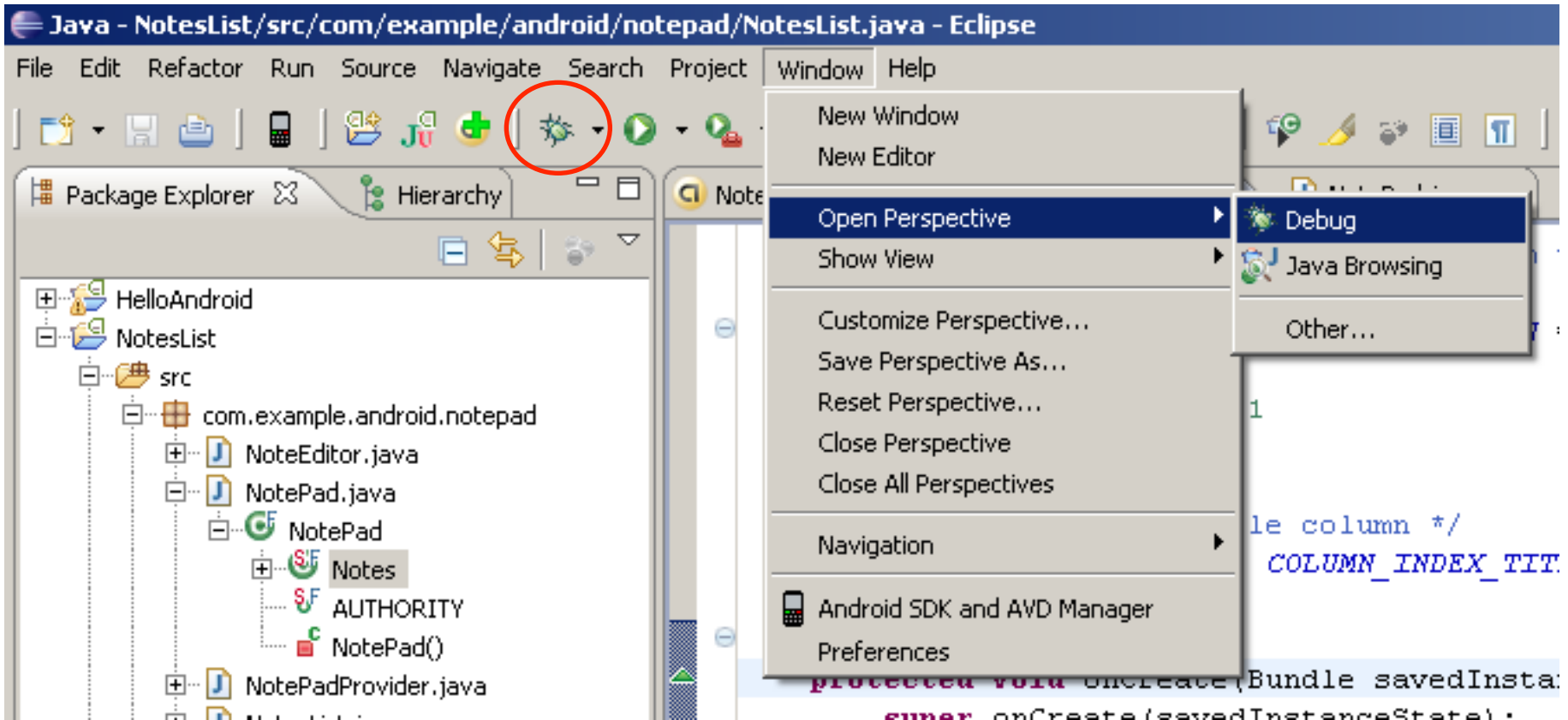
```
LocationQuiz app = (LocationQuiz) getApplication();  
app.currentPoi = ...;  
app.points = 0;
```

Eclipse



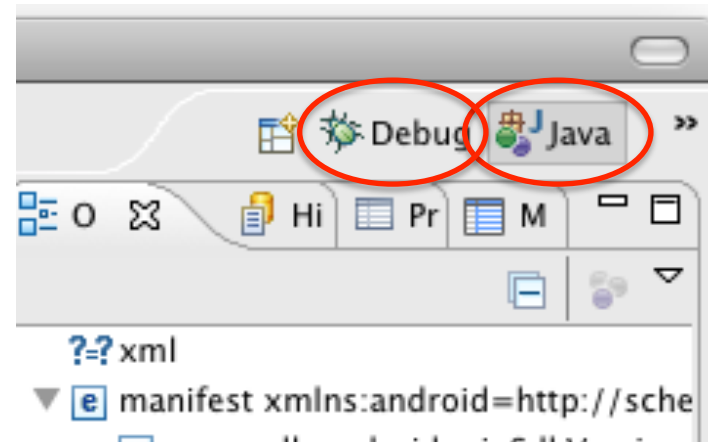
Integrated Development Environment (IDE)

Eclipse Perspectives



Eclipse Perspectives

- Java Perspective
 - Writing source code
 - Adding resources
- Debug Perspective
 - Setting breakpoints
 - Inspecting variables



Eclipse tips:

Ctrl + Shift + O: organize imports

Ctrl + Space: show completions

F3: go to definition (e.g. of a class or method)

Debugging in the Emulator

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

The screenshot shows the Eclipse IDE in a debugging state. The main editor displays the source code of `NotesList.java` with a breakpoint set at line 63. The Debug console shows the thread state, and the Variables view shows the current state of the object. The Outline view shows the class structure.

Debug Console:

```
Thread [<3> main] (Suspended (breakpoint at line 63 in NotesList))
  NotesList.onCreate(Bundle) line: 63
  Instrumentation.callActivityOnCreate(Activity, Bundle) line: 1123
  ActivityThread.performLaunchActivity(ActivityThread$ActivityRecord, I
  ActivityThread.handleLaunchActivity(ActivityThread$ActivityRecord, In
  ActivityThread.access$2100(ActivityThread, ActivityThread$ActivityRe
```

Variables View:

Name	Value
this	NotesLis
savedInstanceState	null

Source Code:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setDefaultKeyMode(DEFAULT_KEYS_SHORTCUT);

    // If no data was given in the intent (because w
    // as a MAIN activity), then use our default cor
```

Outline View:

```
com.example.android.notepad
├── import declarations
├── NotesList
│   ├── TAG : String
│   ├── MENU_ITEM_DELETE : int
│   ├── MENU_ITEM_INSERT : int
│   ├── PROJECTION : String[]
│   ├── COLUMN_INDEX_TITLE : int
│   └── onCreate(Bundle) : void
```

Android Debug Bridge

- Android Debug Bridge (adb)
 - Command line tool (tools\adb.exe)
- Start cmd, start emulator, type “adb devices”
 - Output should be:
List of devices attached
emulator-5554 device
- Shell (limited Unix **ash**) on connected device / emulator
 - Type “adb shell”
 - List of commands: ls /system/bin
 - List of databases: ls /data/data
- More information on adb
 - Type “adb help”, output should be... (quite long)
 - <http://developer.android.com/guide/developing/tools/adb.html>

Debugging on a Device

- Declare application as “debuggable”
 - `<application ... android:debuggable="true">`
- Turn on “USB Debugging” on your device
 - Home screen, MENU, Settings, Applications, Development
- Connect via USB, check whether detected

```
C:\>adb devices
List of devices attached
emulator-5554    device
HT91HKV00188    device
```
- If not listed, setup system to detect device
 - <http://developer.android.com/guide/developing/device.html>
- Start in Eclipse, device chooser appears

Inspecting Variables

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);
```

```
    setDefaultKeyMode(DEFAULT_KEYS_SHORTCUT);
```

```
    // If no data was given in the intent (because we were started  
    // as a MAIN activity), then use our default content provider.
```

```
    Intent intent = getIntent();
```

```
    if (intent
```

```
        intent
```

```
    }
```

```
    // Info
```

```
    getListIntent { flg=0x10000000 cmp=com.example.android.notepad/.Note
```

```
    // Perf
```

```
    // when
```

```
    Cursor cursor = managedQuery(getIntent().getData(), PROJECTION, null, null,  
        Notes.DEFAULT_SORT_ORDER);
```

The screenshot shows a variable inspection window for an `Intent` object. The window title is `intent= Intent (id=830060490448)`. It displays the following fields:

- `mAction= null`
- `mCategories= null`
- `mComponent= ComponentName (id=830060490608)`
- `mData= null`

Below the field list, the `toString()` method is expanded, showing the string representation: `Intent { flg=0x10000000 cmp=com.example.android.notepad/.Note`. The window has a scroll bar on the right and a close button in the top right corner.

rying

null,

Logging and Tracing

- android.util.Log

- informational, warning, error methods

- Example:

```
Log.d(TAG, "getAddress: " + s);
```

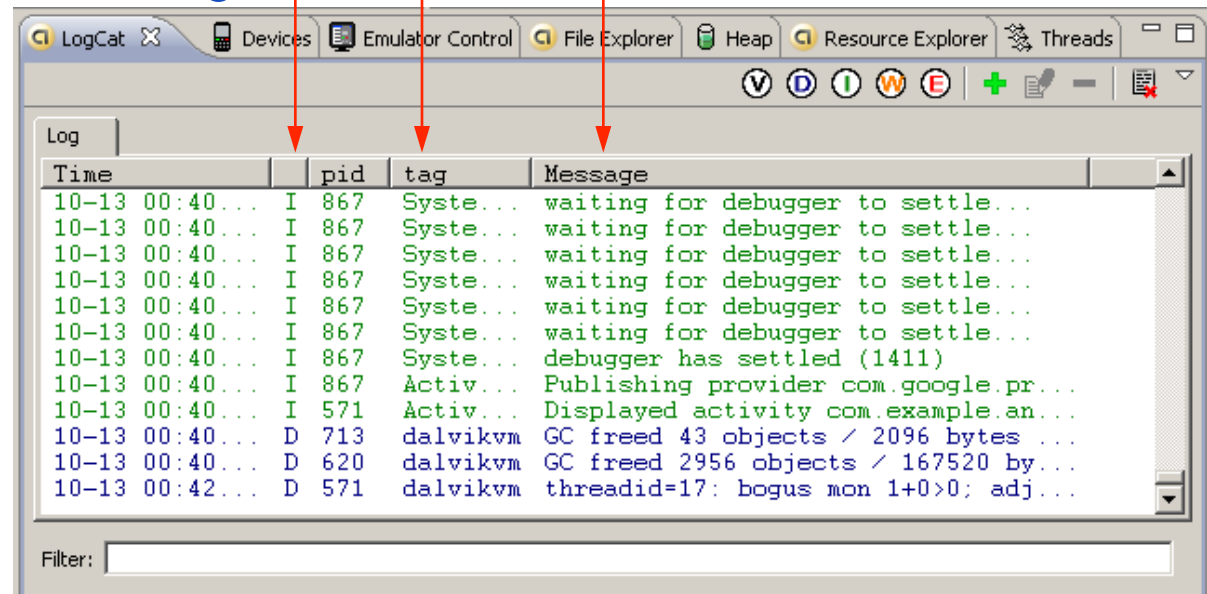
- android.os.Debug

- Debug.startMethodTracing

- Debug.stopMethodTracing

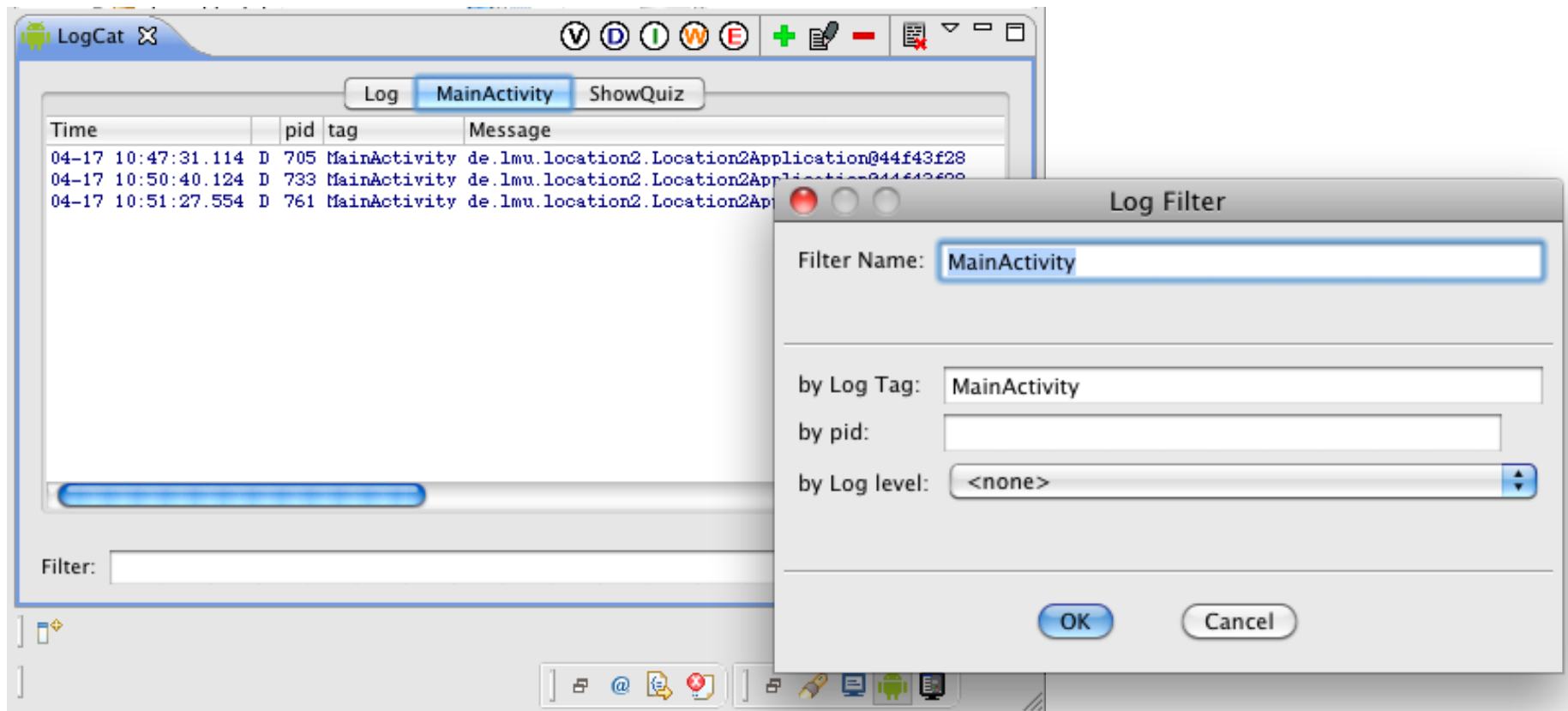
- trace viewer tool

- File explorer tool to view files on the device



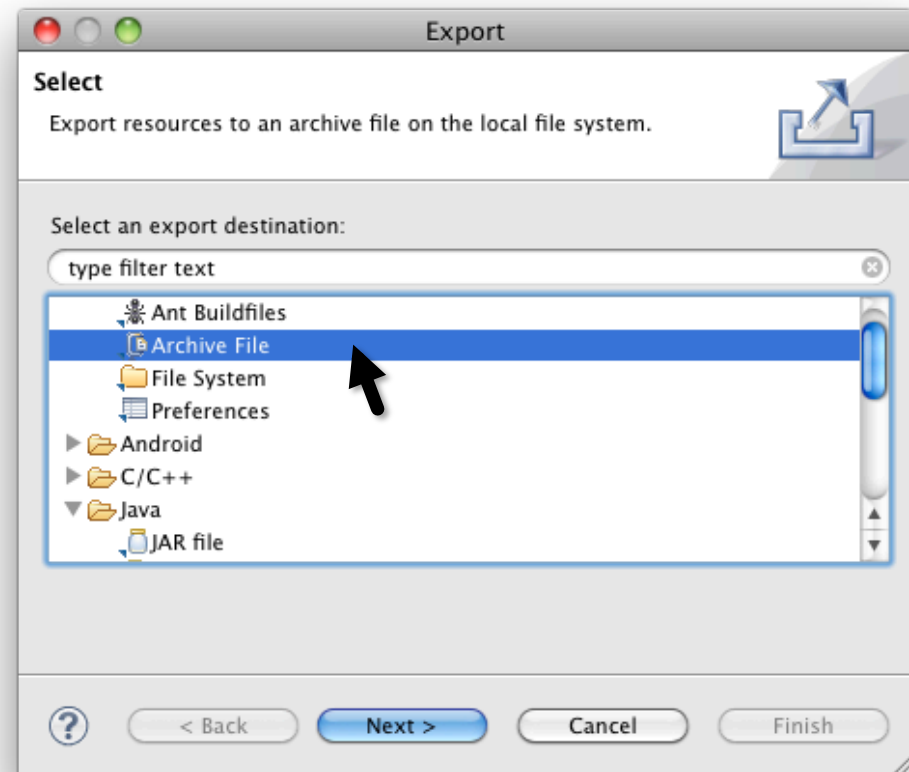
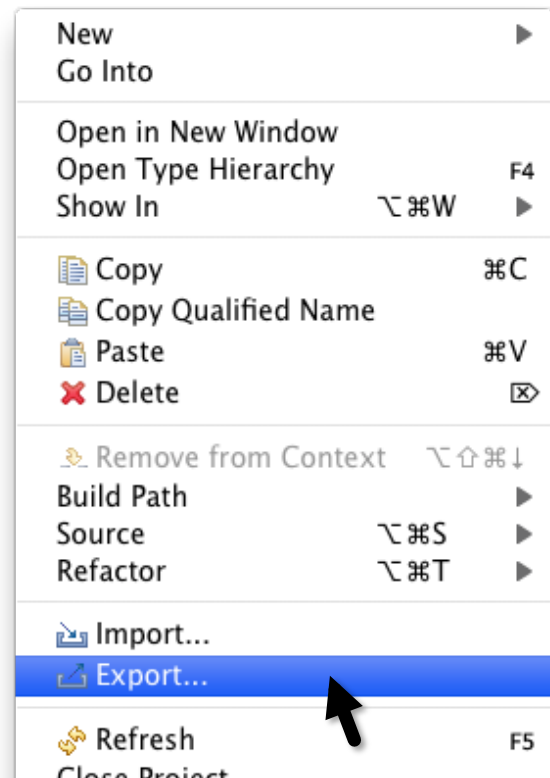
Filtering Eclipse Debug Output

```
Log.d("MainActivity", "onCreate");
```



Exportieren / Importieren von Projekten

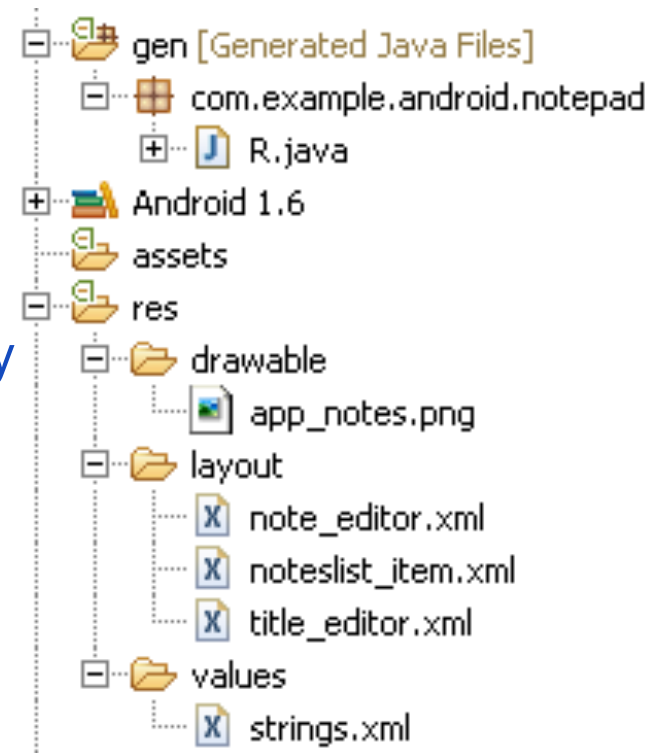
- Android-Projekte exportieren
 - Eclipse → File → Export → General → Archive File (zip)



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

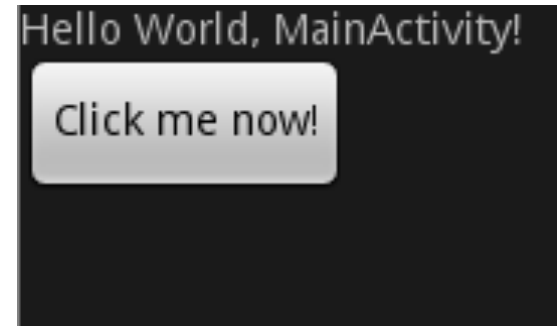
- In /res/values/strings.xml

```
<?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



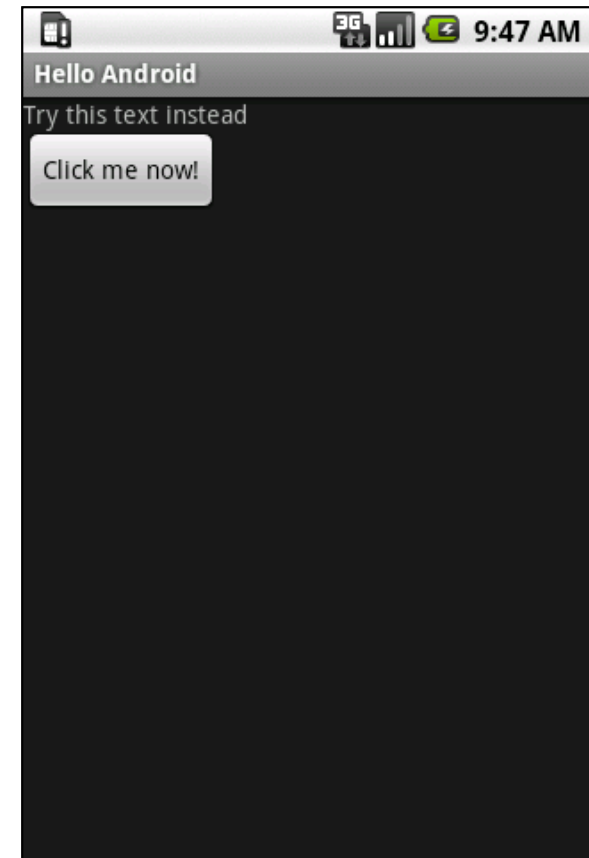
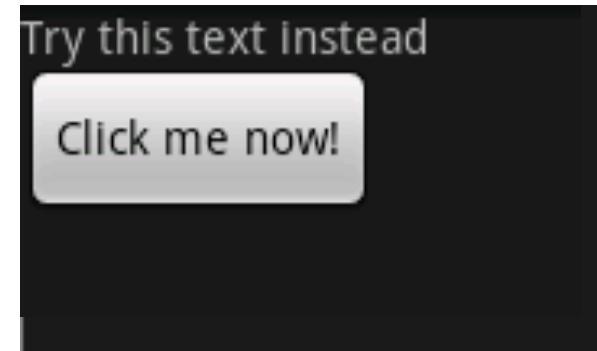
- View for a screen defined in an XML file
- In /res/layout/main.xml

```
<?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

```
 <ERROR Error: No resource found that matches the given name (at 'id' with value '@id/text1').  
    android:text="@string/hello"  
    android:id="@id/text1"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
/>
```

- @+id/text1

```
<TextView  
    android:text="@string/hello"  
    android:id="@+id/text1"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
/>  
<Button
```

Compiled and Noncompiled Resources

- Two types of XML resources
 - Compiled: string resources, layout resources, files in /res/xml/
 - Noncompiled: files in /res/raw, /res/assets/<subfolders>
- Android Asset Packaging Tool (AAPT)
 - Compiles resources (except raw) and places them into .apk file
 - apk = Android package
- Subdirectories of /res/...
 - anim: compiled animation files
 - drawable: bitmaps
 - layout: UI / view definitions
 - values: arrays, colors, dimensions, strings, and styles
 - xml: arbitrary XML files, compiled
 - raw: arbitrary XML files, noncompiled

Android Resource Types

- Color /res/values/<file> R.color.*
- String /res/values/<file> R.string.*
- Dimension /res/values/<file> R.dimen.*
- Image /res/drawable/<files> R.drawable.*
- XML files /res/xml/*.xml R.xml.*
- Raw resources /res/raw/*. * R.raw.*
- Raw assets /assets/*. */*. * arbitrary directory structure, no IDs, access by relative path name

Normal, Quoted, and HTML Strings

```
<resources>
```

```
<string name="simple_string">simple string</string>
```

```
<string name="quoted_string">"quoted'string"</string>
```

```
quoted'string
```

```
<string name="double_quoted_string">\"double quotes\"</string>
```

```
<string name="java_format_string">
```

```
"double quotes"
```

```
    hello %2$s java format string. %1$s again
```

```
</string>
```

```
String format = getString(R.string.java_format_string);
```

```
String s = String.format(format, "Hello", "Android");
```

```
hello Android java format string. Hello again
```

```
<string name="tagged_string">
```

```
    Hello <b><i>Slanted Android</i></b>, You are bold.
```

```
</string>
```

```
Hello Slanted Android, You are bold.
```

```
</resources>
```


Dimension Resources

- Example
 - `<resources>`
 - `<dimen name="mysize_in_pixels">1px</dimen>`
 - `<dimen name="mysize_in_dp">5dp</dimen>`
 - `<dimen name="medium_size">100sp</dimen>`
 - `</resources>`
- Units
 - px: pixels
 - in: inches (1 inch = 25.4 mm)
 - mm: millimeters
 - pt: points (1/72 inch)
 - dp: density-independent pixel (for 160 dpi screen)
 - sp: scale-independent pixel
- Use in Java
 - `float dimen = getResources().getDimension(r.dimen.mysize);`

Image Resources

- Automatic id generation for images in /res/drawable

- Example: /res/drawable/sample_image.jpg
→ R.drawable.sample_image



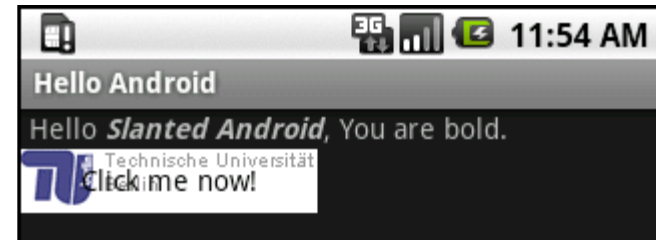
- Supported types: .gif, .jpg, .png

- 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);
```

Arbitrary XML files as Resources

- Stored in /res/xml
- Advantages
 - Referencing via generated resource ID
 - Localization
 - Efficient compilation and storage
- Definition /res/xml/test.xml

```
<?xml version="1.0" encoding="utf-8"?>
<rootelement>
  <subelement1>Hello world</subelement1>
</rootelement>
```
- Usage in Java
 - `XmlResourceParser parser = getResources().getXml(R.xml.test);`

Parsing the XML File

```
XmlResourceParser parser = getResources().getXml(R.xml.test);
StringBuffer sb = new StringBuffer();
parser.next();
int eventType = parser.getEventType();
while (eventType != XmlPullParser.END_DOCUMENT) {
    switch (eventType) {
        case XmlPullParser.START_DOCUMENT:
            sb.append("\nStart document"); break;
        case XmlPullParser.START_TAG:
            sb.append("\nStart tag "+parser.getName()); break;
        case XmlPullParser.END_TAG:
            sb.append("\nEnd tag "+parser.getName()); break;
        case XmlPullParser.TEXT:
            sb.append("\nText "+parser.getText()); break;
    }
    eventType = parser.next();
}
sb.append("\n*****End document");
```

Raw Resources

- Stored in `/res/raw`
- Not compiled
- Identifier generated for each file in `/res/raw`
- Example: Using `/res/raw/test.txt`

```
InputStream is = r.openRawResource(R.raw.test);  
// use input stream...  
is.close();
```

Assets

- Stored in /assets
- Not compiled
- No ID
- Arbitrary directory hierarchy
- AssetManager to access assets
- Example: Using /assets/test.txt

```
AssetManager am = getAssets();  
InputStream is = am.open("test.txt");  
// use input stream...  
is.close();
```

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
```

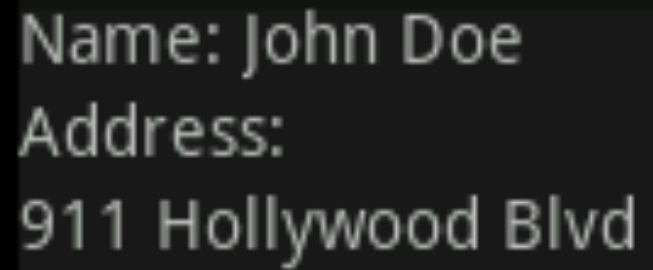
- `android.view.View`
 - Rectangular area on the screen
 - Responsible for drawing and event handling
 - Base class for widgets (buttons, text fields, etc.)
- `android.view.ViewGroup`
 - Is a view and contains other views (“container”)
 - Base class for layouts
- `Layouts`
 - Invisible containers that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

Creating a UI in Java

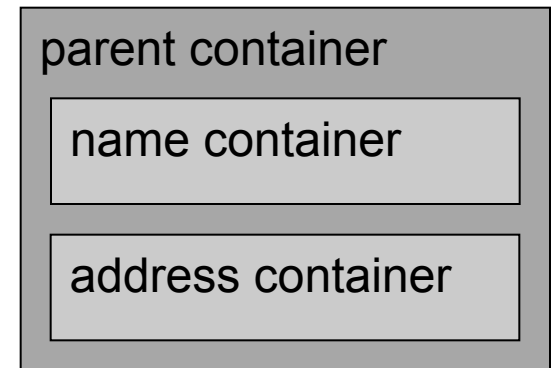
```
package com.androidbook.ch04;
```

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

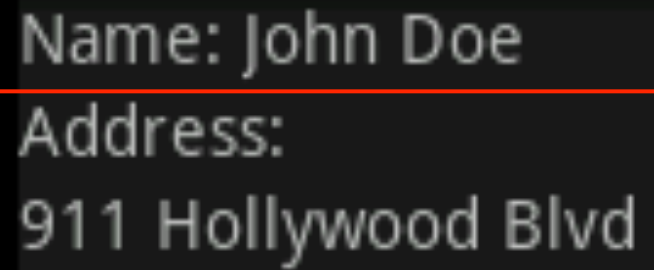
```
public class MainActivity extends Activity {  
    private LinearLayout nameContainer;  
    private LinearLayout addressContainer;  
    private LinearLayout parentContainer;  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        createNameContainer();  
        createAddressContainer();  
        createParentContainer();  
        setContentView(parentContainer);  
    }  
}
```



Name: John Doe
Address:
911 Hollywood Blvd

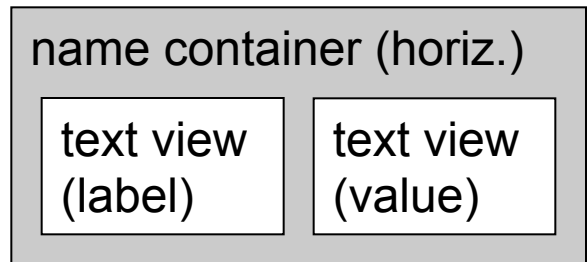


Creating a UI in Java

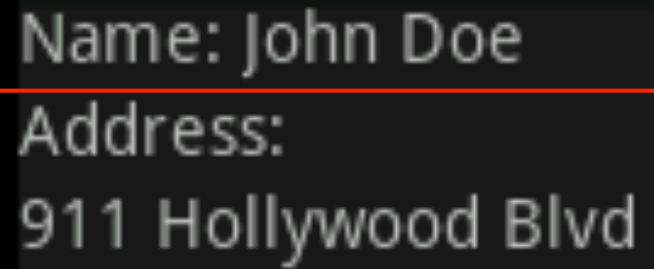


Name: John Doe
Address:
911 Hollywood Blvd

```
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);  
}
```

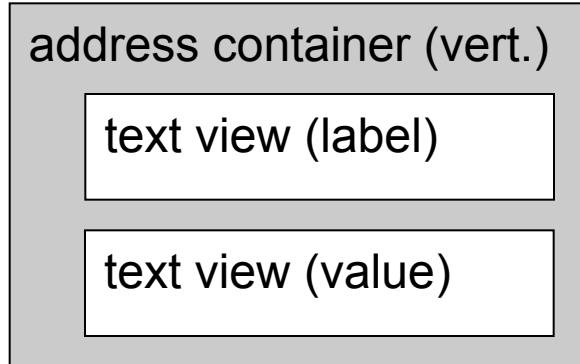


Creating a UI in Java

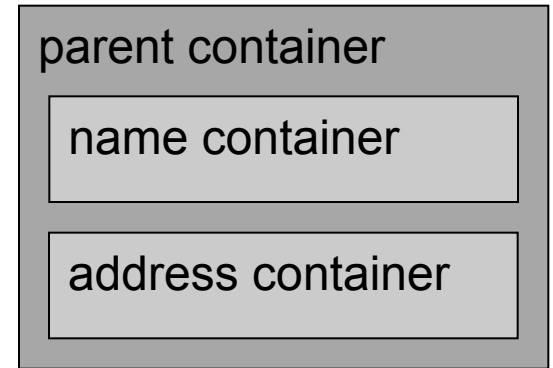


Name: John Doe
Address:
911 Hollywood Blvd

```
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);  
}
```

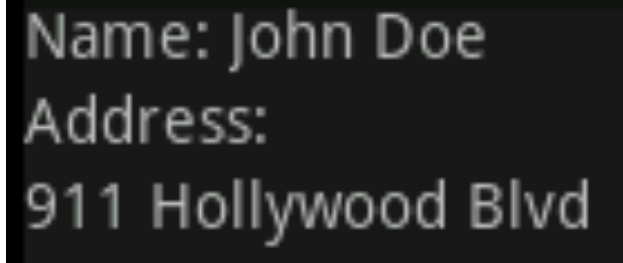


Creating a UI in Java



```
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

TextView Automatic Link Creation

- XML

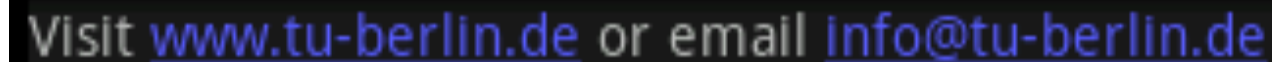
```
<TextView android:id="@+id/nameValue" ... android:autoLink="all" />
```

- Java

```
setContentView(R.layout.test2);
```

```
TextView nameValue = (TextView)findViewById(R.id.nameValue);
```

```
nameValue.setText("Visit www.tu-berlin.de or email info@tu-berlin.de");
```



Visit www.tu-berlin.de or email info@tu-berlin.de

- Using class Linkify

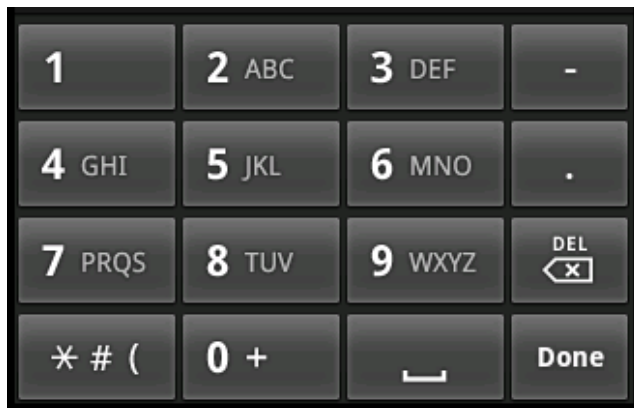
```
Linkify.addLinks(nameValue, Linkify.ALL);
```

EditView Input Type

- `android:inputType="textEmailAddress"`



- `android:inputType="phone"`



AutoCompleteTextView

- XML

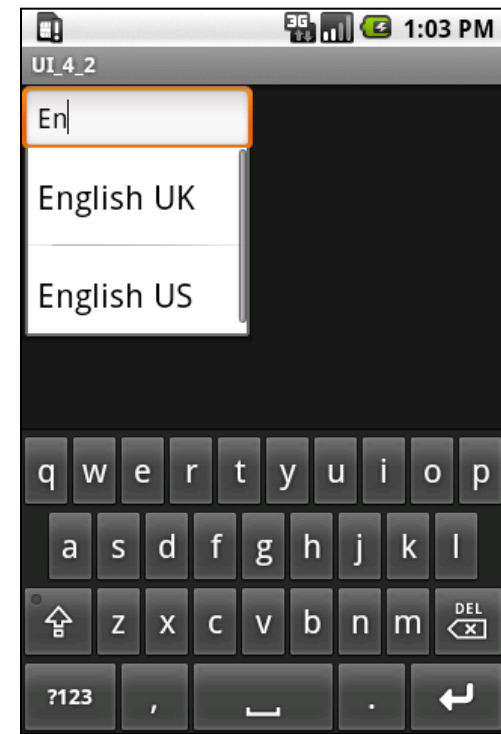
```
<AutoCompleteTextView  
    android:id="@+id/auto" ... />
```

- 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

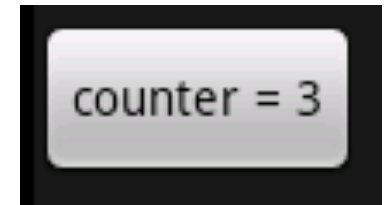
```
<Button android:id="@+id/button1" android:text="Basic Button"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
```

- 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));
    }
}
```



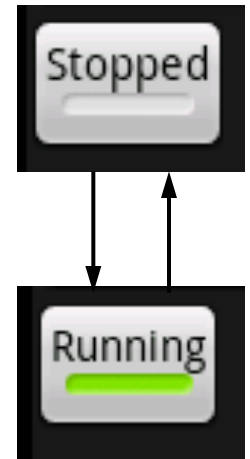
ToggleButton: Two States

- XML

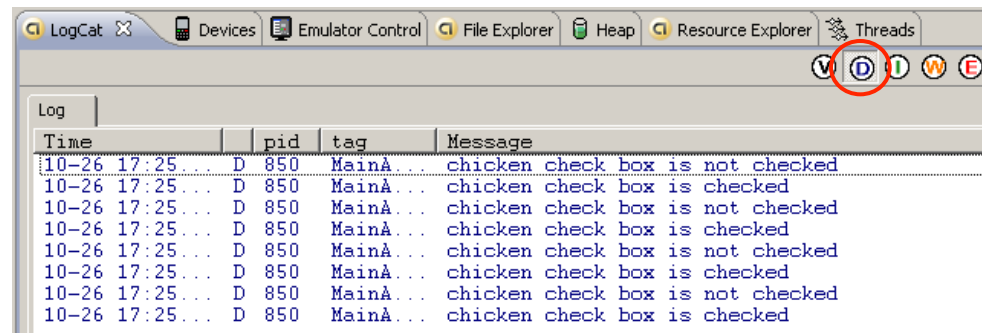
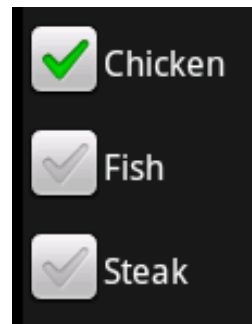
```
<ToggleButton android:id="@+id/cctglBtn"  
  android:layout_width="wrap_content"  
  android:layout_height="wrap_content"  
  android:textOn="Running" android:textOff="Stopped" />
```

- Default text

- “On” for state on
- “Off” for state off



CheckBox



- XML

```
<LinearLayout android:orientation="vertical" ... >  
  <CheckBox android:id="@+id/chicken" android:text="Chicken" ... />  
  <CheckBox android:id="@+id/fish" android:text="Fish" ... />  
  <CheckBox android:id="@+id/steak" android:text="Steak" ... />  
</LinearLayout>
```

- Java

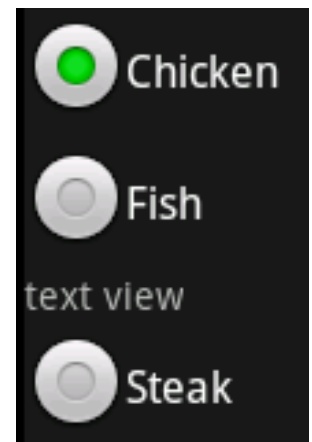
```
CheckBox cb = (CheckBox) findViewById(R.id.chicken);  
cb.setChecked(true);  
cb.setOnCheckedChangeListener(new OnCheckedChangeListener() {  
  public void onCheckedChanged(CompoundButton b, boolean isChecked) {  
    Log.d("MainActivity", "chicken check box is " +  
      (isChecked ? "" : "not ") + "checked");  
  }  
});
```


Radio Button

- XML

```
<LinearLayout android:orientation="vertical"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">
  <RadioGroup android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <RadioButton android:text="Chicken"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
    <RadioButton android:text="Fish"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
    ...
  </RadioGroup>
</LinearLayout>
```

- Radio groups can contain arbitrary views



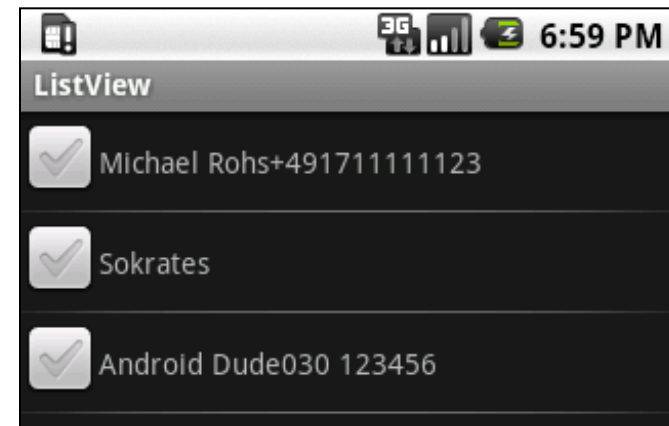
List Controls

- 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>
```

List Controls

- Showing names and numbers from contacts database



```
public class ListDemoActivity extends ListActivity {  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        Cursor c = getContentResolver().query(People.CONTENT_URI,  
                                              null, null, null, null);  
        startManagingCursor(c);  
        String[] cols = new String[] { People.NAME, People.NUMBER };  
        int[] colIds = new int[] { R.id.textview1, R.id.textview2 };  
        SimpleCursorAdapter adapter = new  
            SimpleCursorAdapter(this, R.layout.list_item, c, cols, colIds);  
        setListAdapter(adapter);  
    }  
}
```

AndroidManifest.xml needs:

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

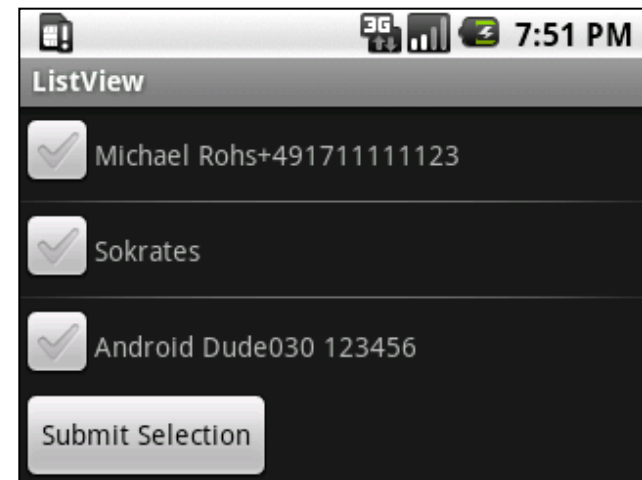
Using a Custom List View

- /res/layout/list.xml

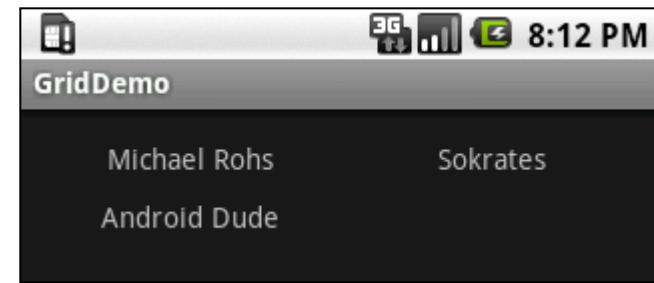
```
<LinearLayout android:orientation="vertical" ...>  
  <LinearLayout android:orientation="vertical" ...>  
    <ListView android:id="@android:id/list"  
      android:layout_width="fill_parent"  
      android:layout_height="0dip" android:layout_weight="1"  
      android:stackFromBottom="true"  
      android:transcriptMode="normal" />  
  </LinearLayout>  
  <Button android:text="Submit Selection"  
    ... />  
</LinearLayout>
```

- Java

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



GridView



- XML

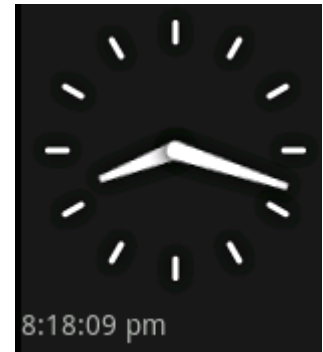
```
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/dataGrid" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:padding="10px"
    android:verticalSpacing="10px" android:horizontalSpacing="10px"
    android:numColumns="auto_fit" android:columnWidth="100px"
    android:stretchMode="columnWidth" android:gravity="center" />
```

- Java

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.gridview);
    GridView gv = (GridView) this.findViewById(R.id.dataGrid);
    Cursor c = getContentResolver().query(People.CONTENT_URI, null, null, null, null);
    startManagingCursor(c);
    String[] cols = new String[] { People.NAME };
    int[] colIDs = new int[] { R.id.textview };
    SimpleCursorAdapter adapter = new SimpleCursorAdapter(
        this, R.layout.grid_item, c, cols, colIDs);
    gv.setAdapter(adapter);
}
```

Android Specific Controls

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



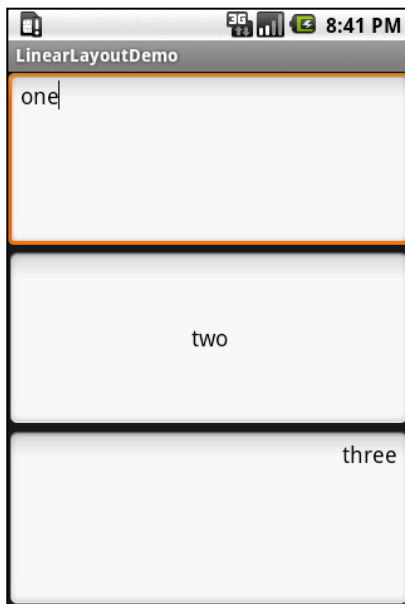
Layout Managers

LayoutManagers

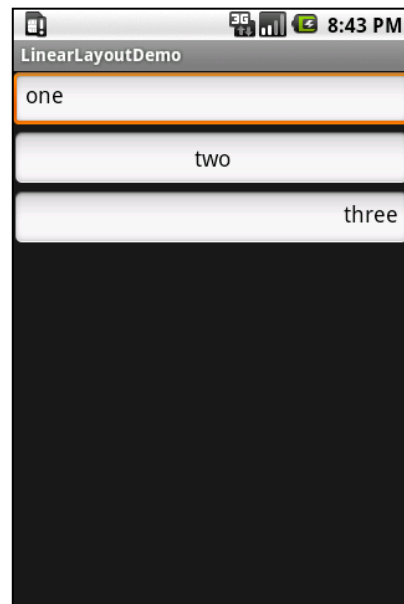
- LayoutManagers
 - Are containers for views (children)
 - Have specific strategy for controlling children's size and position
- Layout Managers in Android
 - LinearLayout: horizontal or vertical arrangement
 - TableLayout: tabular form
 - RelativeLayout: arrange children relative to one another or parent
 - 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

LinearLayout

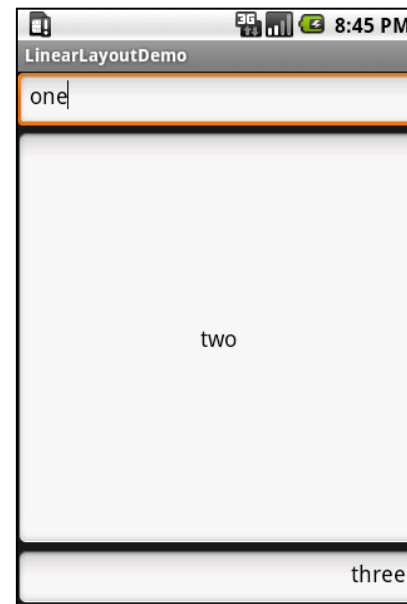
- **Orientation:** horizontal or vertical
- **Gravity:** alignment (left, right, center, top, etc.)
- **Weight:** size importance of one child relative to others



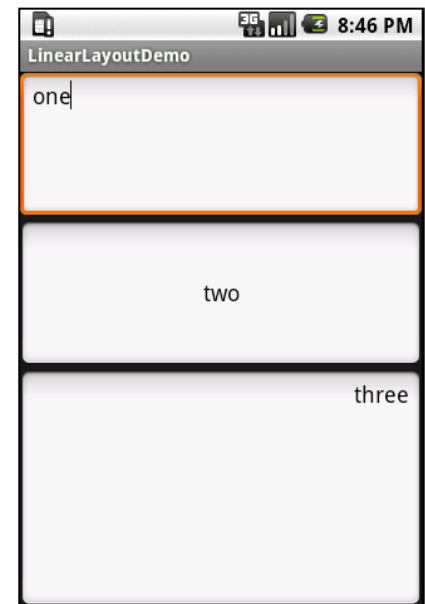
Weights:
1.0, 1.0, 1.0



Weights:
0.0, 0.0, 0.0



Weights:
0.0, 1.0, 0.0



Weights:
0.5, 0.5, 1.0

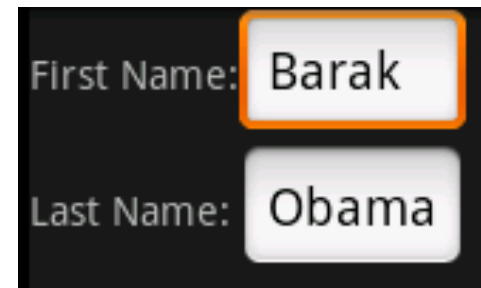
Example LinearLayout with Weights

```
<LinearLayout android:orientation="vertical"
android:layout_width="fill_parent" android:layout_height="fill_parent">
  <EditText android:layout_width="fill_parent"
android:layout_weight="0.5" android:layout_height="wrap_content"
android:text="one" android:gravity="left" />
  <EditText android:layout_width="fill_parent"
android:layout_weight="0.5" android:layout_height="wrap_content"
android:text="two" android:gravity="center" />
  <EditText android:layout_width="fill_parent"
android:layout_weight="1.0" android:layout_height="wrap_content"
android:text="three" android:gravity="right" />
</LinearLayout>
```

TableLayout

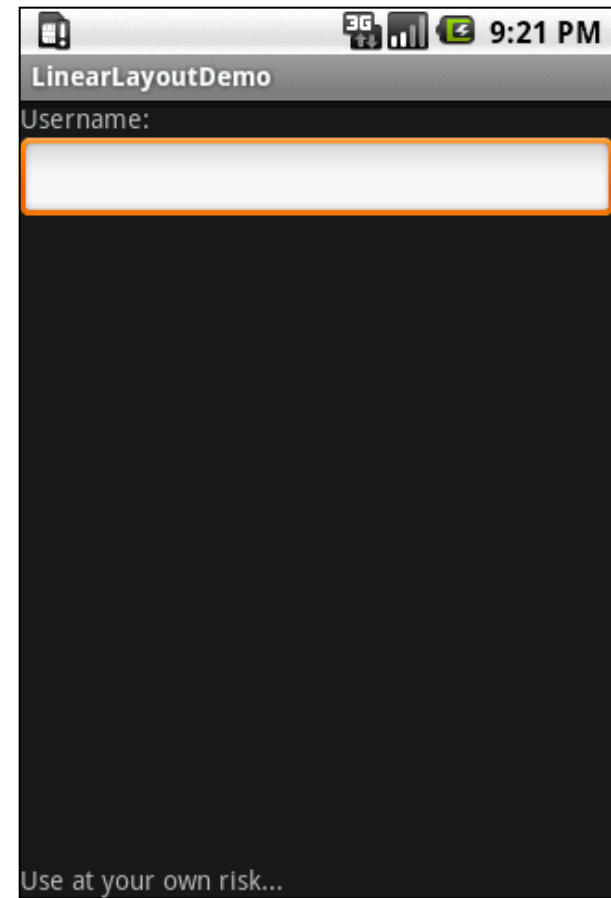
- Extension of LinearLayout
- Example:

```
<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
```

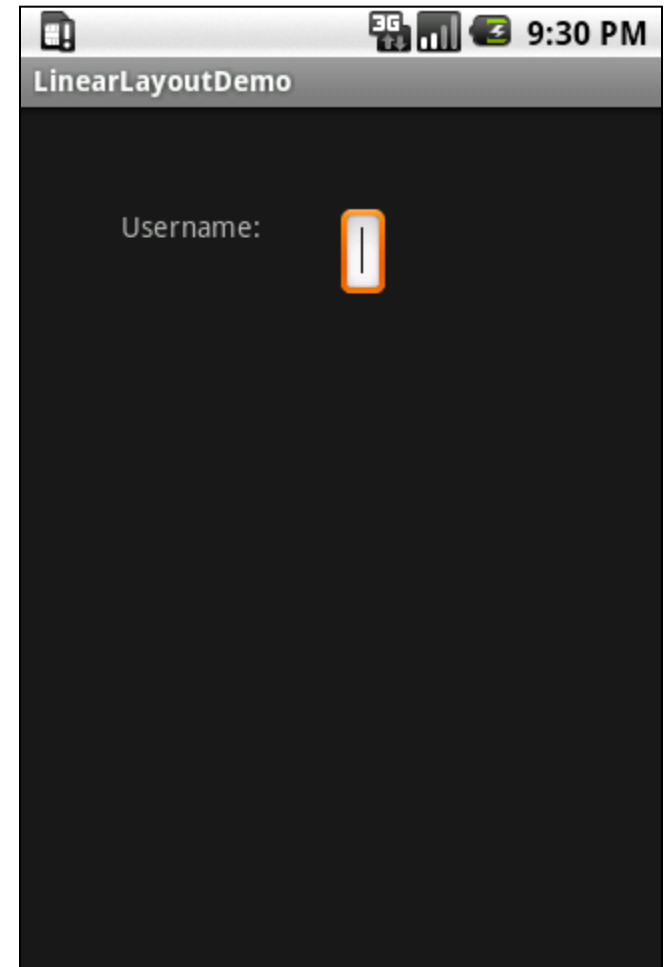
```
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent" >
```

```
<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>
```



FrameLayout

- Displays one item at a time
- Stacks items if multiple visible
- XML

```
<FrameLayout... >
```

```
<ImageView
```

```
    android:id="@+id/imgView1"
```

```
    android:src="@drawable/one"
```

```
    android:scaleType="fitCenter"
```

```
    android:layout_width="fill_parent" android:layout_height="fill_parent" />
```

```
<ImageView android:id="@+id/imgView2"
```

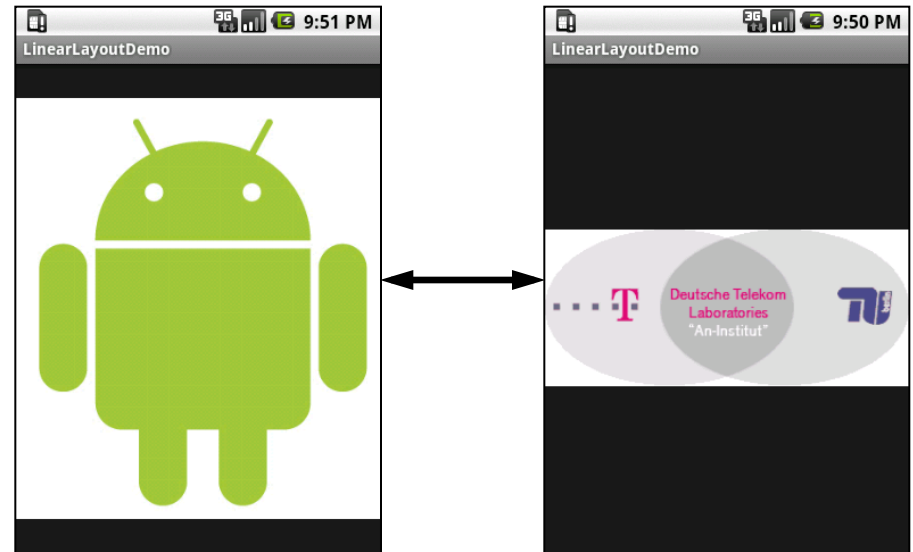
```
    android:src="@drawable/two"
```

```
    android:scaleType="fitCenter"
```

```
    android:layout_width="fill_parent" android:layout_height="fill_parent"
```

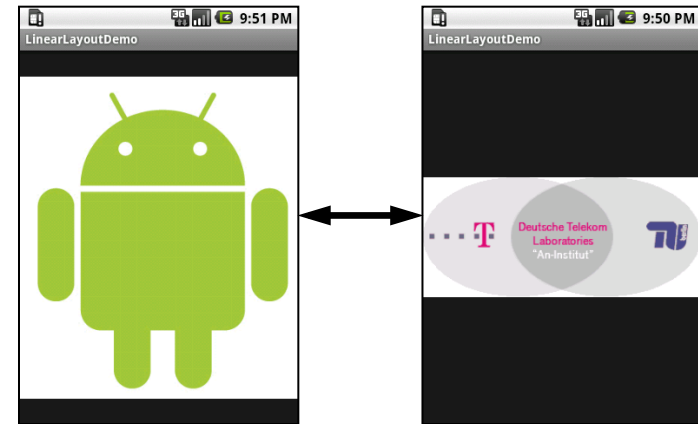
```
    android:visibility="gone" />
```

```
</FrameLayout>
```



FrameLayout

```
public class FrameActivity extends Activity {  
    protected void onCreate(Bundle state) {  
        super.onCreate(state);  
        setContentView(R.layout.frame);  
        ImageView one = (ImageView) findViewById(R.id.oneImgView);  
        ImageView two = (ImageView) findViewById(R.id.twoImgView);  
        one.setOnClickListener(new OnClickListener() {  
            public void onClick(View view) {  
                ImageView two = (ImageView) findViewById(R.id.twoImgView);  
                two.setVisibility(View.VISIBLE);  
                view.setVisibility(View.GONE);  
            });  
        });  
        two.setOnClickListener(new OnClickListener() {  
            public void onClick(View view) {  
                ImageView one = (ImageView) findViewById(R.id.oneImgView);  
                one.setVisibility(View.VISIBLE);  
                view.setVisibility(View.GONE);  
            });  
        });  
    }  
}
```



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)

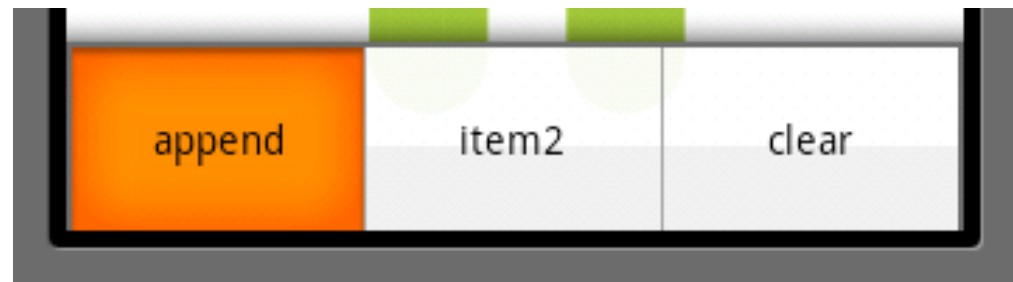
Menus



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  
}
```



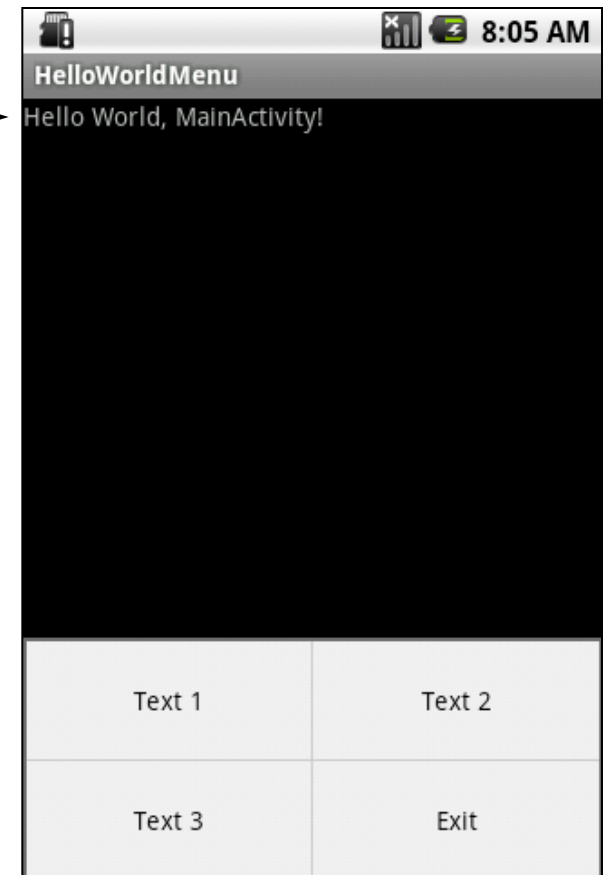
Responding to Menu Selection

- 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 handled item  
            // handle item X  
            return true;  
        ...  
    }  
}
```

Exercise: A Menu for Hello World

- Add a menu with four items to “Hello World”
- Menu items 1-3 changes text shown in the top of the display
 - `setText(...)` →
- Menu item 1 → MMI2
- Menu item 2 → LMU
- Menu item 3 → Android
- Menu item 4: Exit the application
 - `finish()`



The End



Prof. Dr. Michael Rohs

michael.rohs@ifi.lmu.de

Mobile Interaction Lab, LMU München