CS 580 Client-Server Programming
Fall Semester, 2012
Doc 5 Android Intro
Sept 10, 2012

Monitoring Client Server Communication

netcat (nc)

network utility

- listens on ports
- connects to ports
- lots more

nc -l portNumber

- listen on given portNumber

Set my Mars client to connect to my local machine then

```
Al pro 34->nc -l 8009
trip;destination:mars;people:1;weight:1.0;mpg:1.0;milesperyear:1.0;;
```

Monitor both Client and Server

Simple Proxy Server that prints out what it sees

Command Line

```
java EchoProxy localPort remoteServer remotePort
```

```
Will listen on localPort
When get connection on localPort
    open connection to remoteServer on remotePort
    Forward all data from localPort to remoteServer
    Send all data from remoteServer to client
```

EchoProxy

```
public class EchoProxy {
    private int localPort;
    private int remotePort;
    private String remoteMachine;

    public static void main(String[] args) throws IOException {
        int localPort = Integer.parseInt(args[0]);
        String host = args[1];
        int remotePort = Integer.parseInt(args[2]);
        new EchoProxy(localPort, host, remotePort).run();
    }

    public EchoProxy(int local, String host, int remote) {
        localPort = local;
        remotePort = remote;
        remoteMachine = host;
    }
```

EchoProxy

```
public void run() throws IOException {
    ServerSocket input = new ServerSocket( localPort );
    System.out.println("Local server "  + input.getLocalPort());
    while (true) {
        Socket client = input.accept();
        System.out.println("New Connection " + client.getInetAddress());
        InputStream fromClient = client.getInputStream();
        OutputStream toClient = client.getOutputStream();
        Socket remote = new Socket(remoteMachine, remotePort);
        OutputStream toServer = remote.getOutputStream();
        InputStream fromServer = remote.getInputStream();
        Thread clientToServer = new Thread(new Forwarder(fromClient,
                toServer));
        clientToServer.start();
        Thread serverToClient = new Thread(new Forwarder(fromServer,
                toClient));
        serverToClient.start();
    }
}
```

EchoProxy - Inner class Forwarder

```
class Forwarder implements Runnable {
    InputStream in;
    OutputStream out;

    public Forwarder(InputStream in, OutputStream out) {
        this.out = out;
        this.in = in;
    }
```

EchoProxy - Inner class Forwarder

```
    @Override
        public void run() {
            int next;
            try {
                while ((next = in.read()) != -1) {
                    out.write(next);
                    System.out.print((char) next);
                    out.flush();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Sample Run - Mars

```
Local server 8009
New Connection /10.0.1.78
trip;destination:mars;people:1;weight:1.0;mpg:1.0;milesperyear:1.0;;food:
53764.3;weight:61.1;;quit;;quit;;
```

Android

Android

Googles mobile phone OS and SDK

Java only

- Special VM
- Nonstandard byte code

Eclipse is development IDE

Linux

Application framework
2D & 3D graphics
Audio, video and still image support
SQLite database
Embeddable web browser

Hardware dependent

GSM, CDMA
Bluetooth, EDGE, 3G, WIFI
Camera, GPS, compass
accelerometer, NFC

Android

IDE - Eclipse

- http://www.eclipse.org/downloads/
- Eclipse Classic recommended by Google
- But any of three types of Eclipse for Java works

Android SDK

- http://developer.android.com/sdk/installing.html
- Follow instruction at that site

Android Development - Emulator and Device

Emulator

- Can be slow to start at times
- Good for basic testing of code

But

- Emulator does not have sensors - accelerometer etc
- Emulator has different set of bugs than devices
- Emulator performance & constraints different than device
- Emulator does not give you feel of how app runs on device

Qualcomm Donation

Qualcomm generously donated to SDSU money to purchase Android phones

We will be using them during the course

Details to be worked out

Android Versions

Version	Name	API Level	% of Devices	Release Date	Notes
1.5	Cupcake	3	0.2%	Apr 2009	
1.6	Donut	4	0.4%	Sept 2009	
2.0/2.1	Eclair	7	3.7%	Oct 2009/ Jan 2010	
2.2.x	Froyo	8	14.0%	May 2010	
2.3.x	Gingerbread	9-10	57.5%	Dec 2010	Kindle Fire
3.x	Honeycomb	11-13	2.1%	Feb 2011	Tablets only
4.0	Ice Cream Sandwich	14-15	20.9%	Oct 2011	October 2011 Phones + Tablets
4.1	Jelly Bean	16	1.2%	Summer 2012	

% devices that accessed Android Market for 14 day period ending Sept 6, 2012

Reference: http://en.wikipedia.org/wiki/Android_(operating_system), referenced 1/19/12. http://developer.android.com/resources/dashboard/platform-versions.html, referenced 1/19/12. http://en.wikipedia.org/wiki/Android_version_history referenced 1/19/12

Android Fragmentation

Fragmentation on different axes

User Interface	Motoblur, HTC Sense UI Kindle Fire etc
Device	Over 70 devices in US Different shapes Different hardware
Operating System	1.6 2.2 2.3 3.x 4.x
Marketplace	50+ Android App stores
Service	Manufacters & venders provide services to increase revenue

Reference:s http://ceklog.kindel.com/2012/01/14/fragmentation-is-not-the-end-of-android/, http://www.android-markets-info.com/android-markets-list, http://www.quora.com/How-many-Android-Phones-models-are-currently-available-in-the-US-Market

Android Fragmentation

Good for some people, not for others

Causes more effort for developers

Installing Android - Step 1

http://developer.android.com/index.html

- Get SDK link on bottom of page
- Follow instructions

  - Download
  - Installing the SDK
    - Adding Platforms & Packages
    - Intalling the Eclipse Plugin

Installing Android - Step 2

Setting Up Virtual Devices

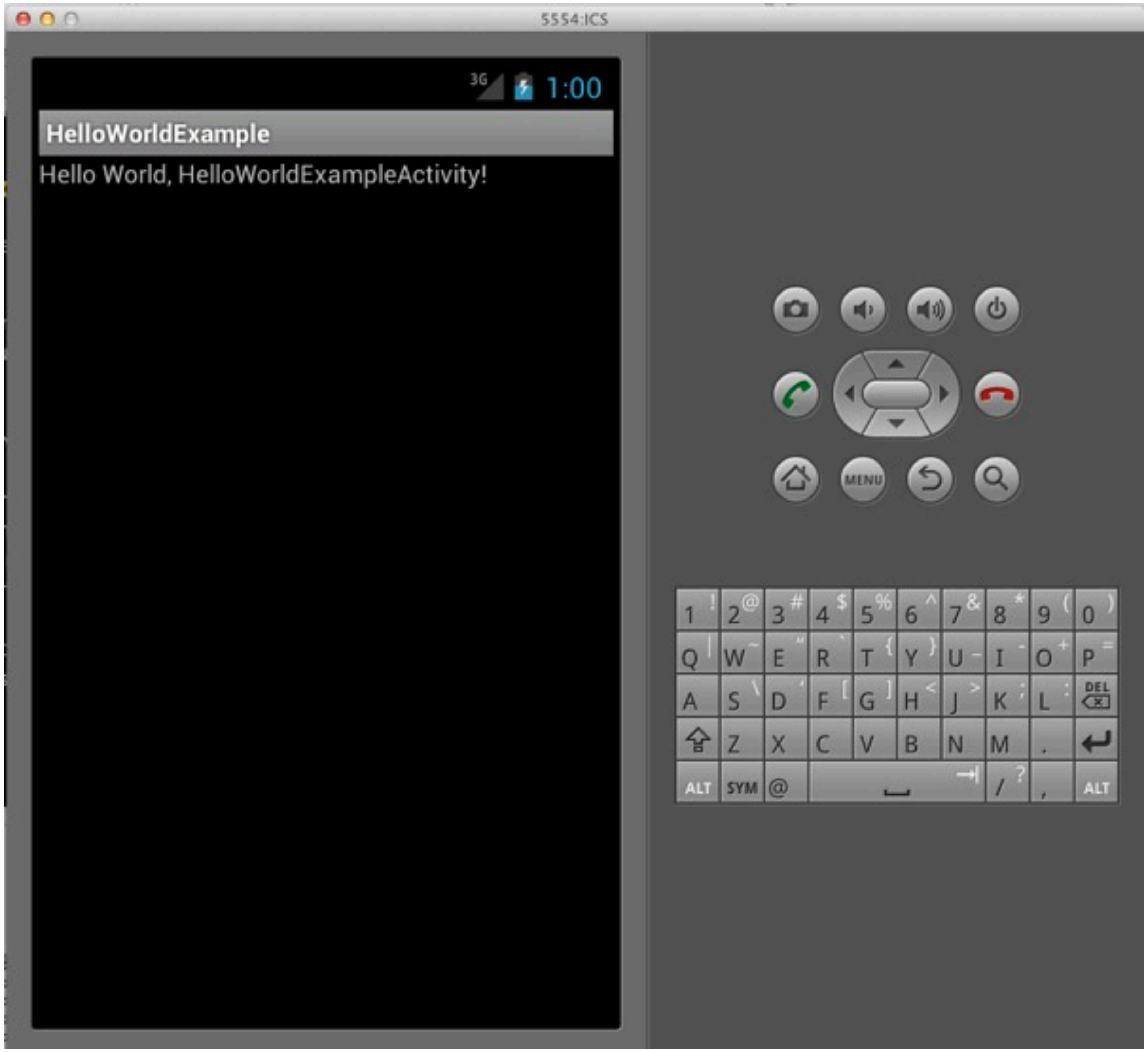
http://developer.android.com/tools/devices/index.html

Set up Virtual Devices with AVD Manager

Run Hello World

http://developer.android.com/training/basics/firstapp/index.html

Hello World - Demo



5554:ICS

HelloWorldExample

Hello World, HelloWorldExampleActivity!

Lots of parts

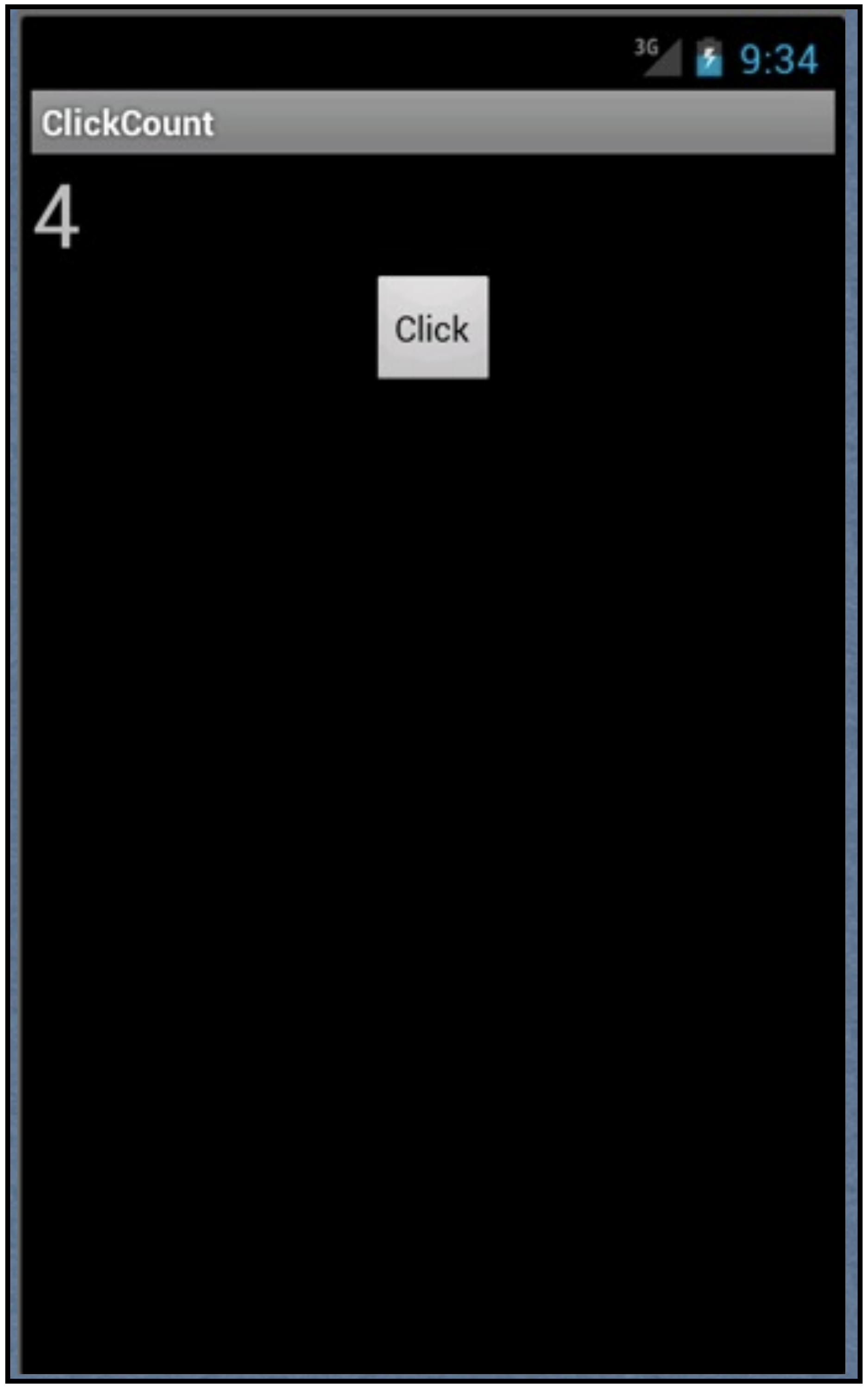
Android apps have lots of parts

Bit intimidating at first, but each part not hard

Difficult to show entire example

Click Count Example

Click Count



Clicking on button increases count

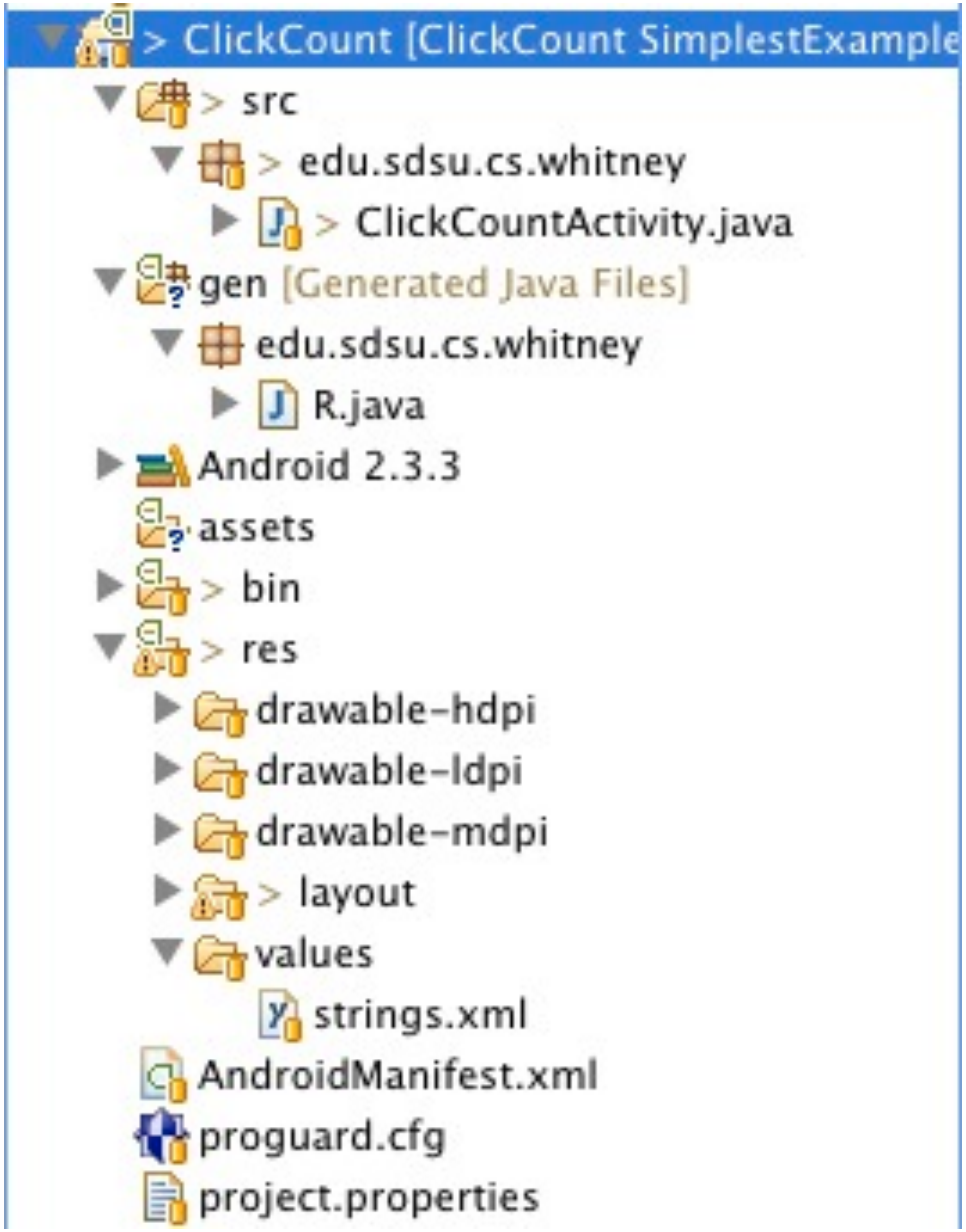
Issues

Adding GUI widgets
Code accessing GUI widgets
Code responding to widget events
Debugging

Issues for Future

Better layout

Lots of Parts



```
▼ > ClickCount [ClickCount SimplestExample
  ▼ > src
    ▼ > edu.sdsu.cs.whitney
      ▶ > ClickCountActivity.java
  ▼ gen [Generated Java Files]
    ▼ edu.sdsu.cs.whitney
      ▶ R.java
  ▶ Android 2.3.3
    assets
  ▶ > bin
  ▼ > res
    ▶ drawable-hdpi
    ▶ drawable-ldpi
    ▶ drawable-mdpi
    ▶ > layout
    ▼ values
        strings.xml
    AndroidManifest.xml
    proguard.cfg
    project.properties
```

Basic parts

- Activity
    - Controller
- R.java
    - Runtime location of view
- layouts (res/layouts/main.xml)
    - View
- strings (res/values/strings.xml)
    - Text display on screen

Separation of Concerns

Activity

- Handle events related to View

layouts

- Generates view from xml
- Separates View organization from code

strings

- Text displayed in view
- Separates text & language used from code and view

Main Class

```
package edu.sdsu.cs.whitney;
import android.app.Activity;        import android.os.Bundle;
import android.util.Log;             import android.view.View;
import android.widget.TextView;

public class ClickCountActivity extends Activity {
    TextView countOutput;
    int count = 0;

  @Override
   public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     countOutput = (TextView) this.findViewById(R.id.countOutput);
  }

  public void increase(View button) {
    Log.i("rew", "increase");
    count++;
    countOutput.setText(String.valueOf(count));
  }
}
```

R.java

R = Resource

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY. */
package edu.sdsu.cs.whitney;
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int clickButton=0x7f050001;
        public static final int countOutput=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040000;
        public static final int clickButtonLabel=0x7f040001;
        public static final int initialCount=0x7f040002;
    }
}
```

Maps

Names to runtime locations of resources

Generated from ids in main.xml

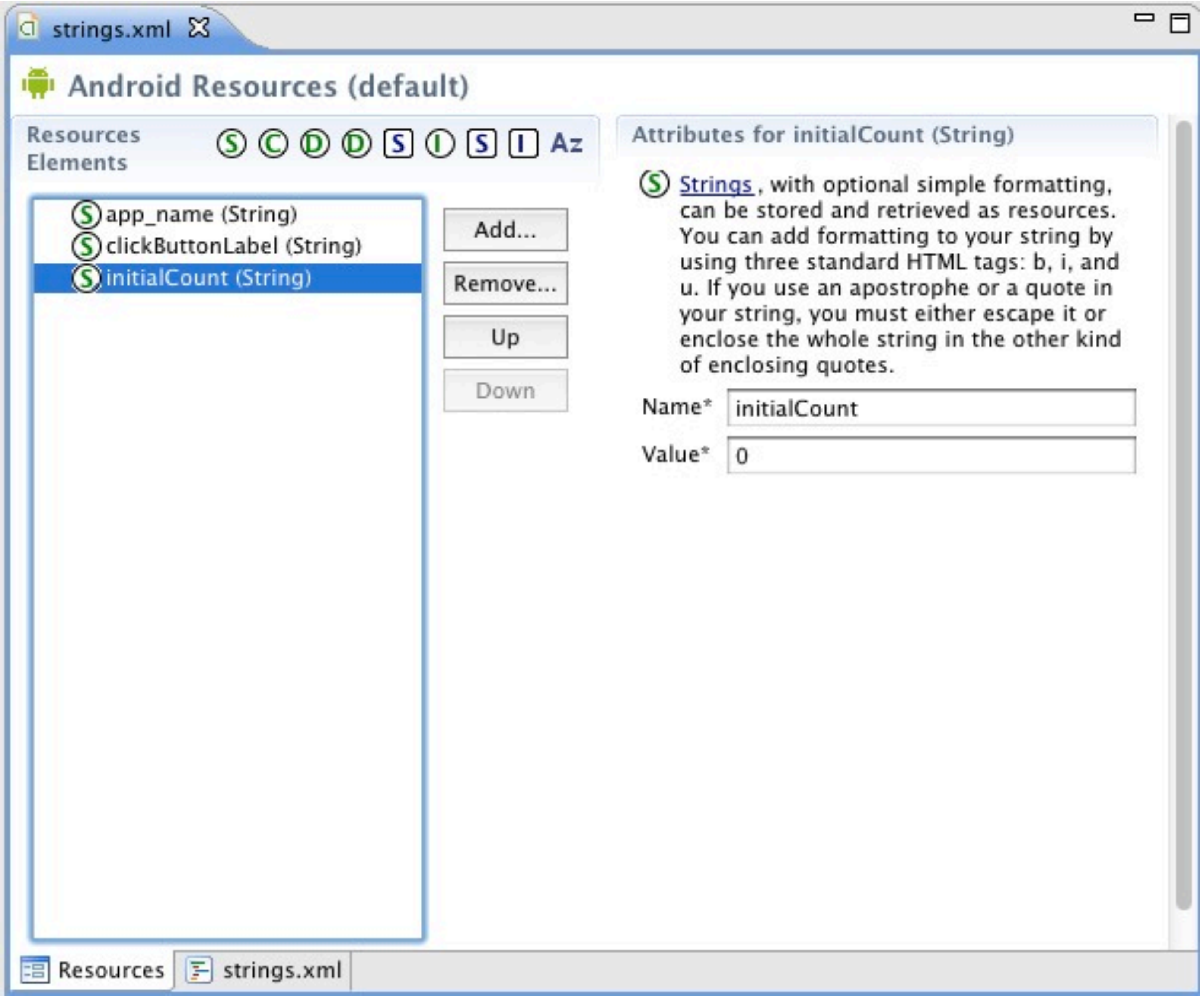
Generated from files in res/layout

Generated from strings.xml

res/values/strings.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">ClickCount</string>
    <string name="clickButtonLabel">Click</string>
    <string name="initialCount">0</string>
</resources>
```

Two Eclipse views of res/values/strings.xml



app_name (String)
clickButtonLabel (String)
initialCount (String)

Attributes for initialCount (String)

Strings, with optional simple formatting, can be stored and retrieved as resources. You can add formatting to your string by using three standard HTML tags: b, i, and u. If you use an apostrophe or a quote in your string, you must either escape it or enclose the whole string in the other kind of enclosing quotes.

Name* initialCount

Value* 0

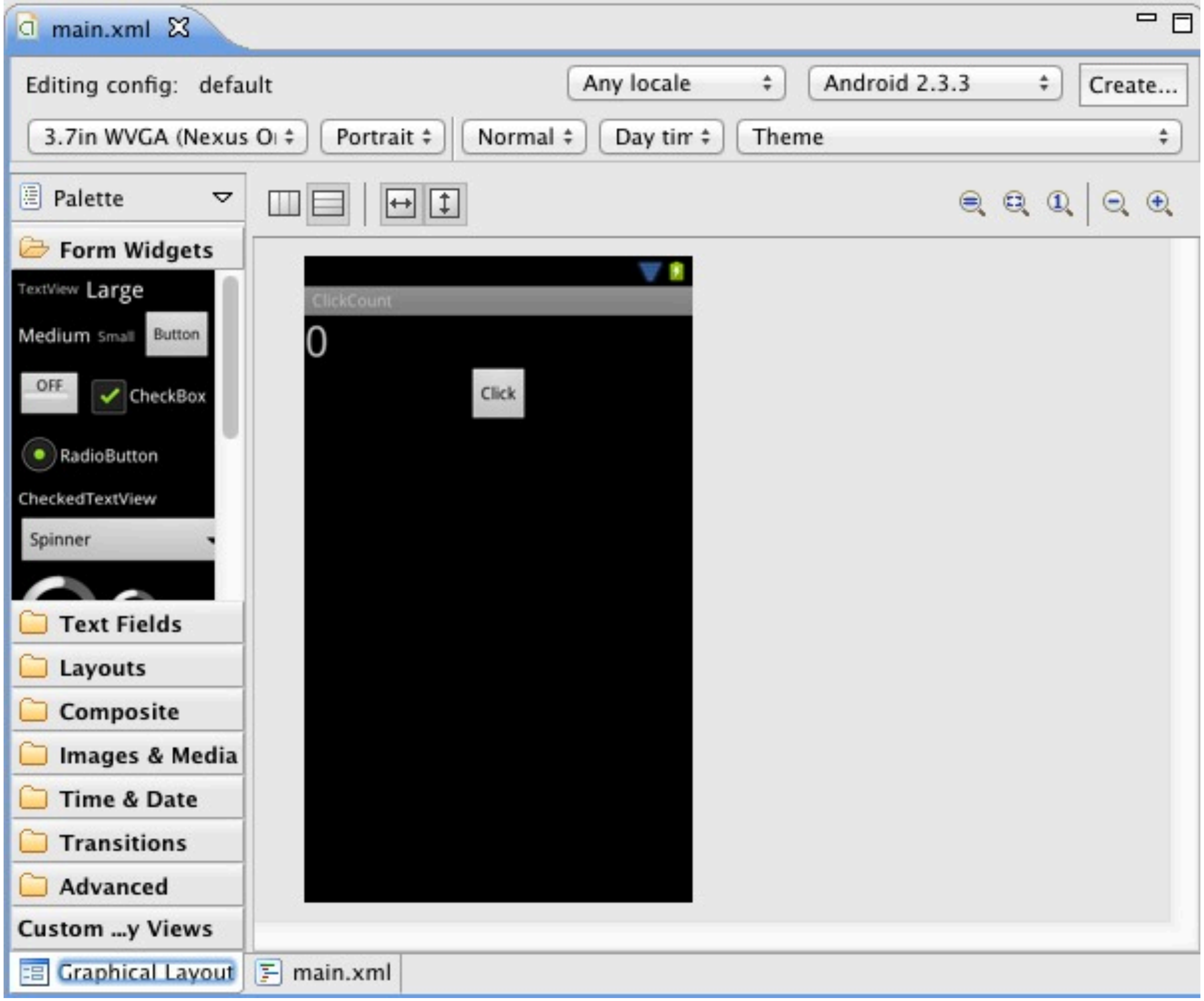
You can edit file directly

Or use this visual editor

res/layout/main.xml - Source View

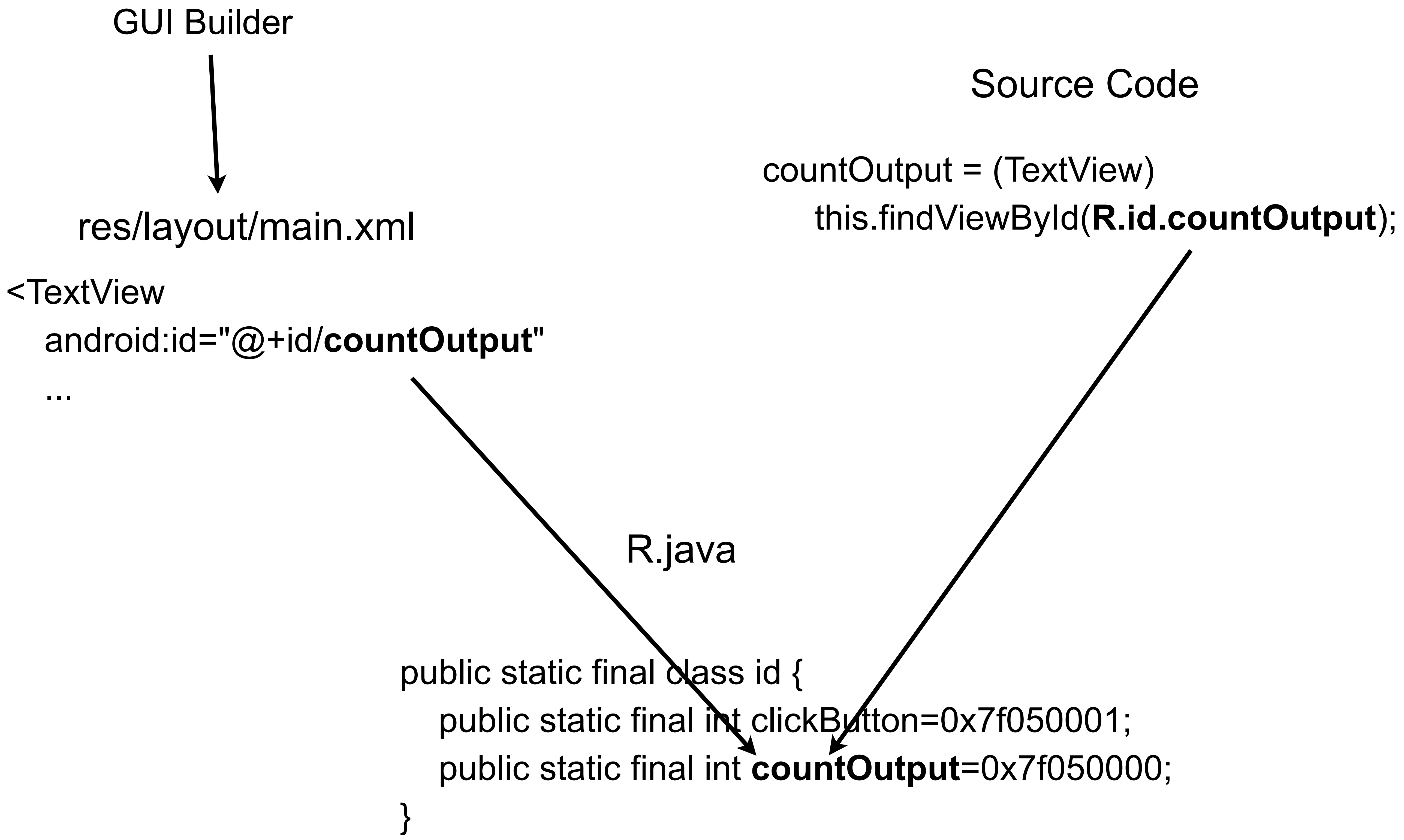
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:gravity="center|top"
   android:orientation="vertical" >

   <TextView
      android:id="@+id/countOutput"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/initialCount" android:textSize="35sp"/>

   <Button
      android:id="@+id/clickButton"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/clickButtonLabel"
      android:onClick="increase" />
</LinearLayout>
```

res/layout/main.xml - Graphical Editor View



How all this works

R - Connection between resources & code



GUI Builder

res/layout/main.xml

```
<TextView
   android:id="@+id/countOutput"
   ...
```

Source Code

```
countOutput = (TextView)
     this.findViewById(R.id.countOutput);
```

R.java

```
public static final class id {
    public static final int clickButton=0x7f050001;
    public static final int countOutput=0x7f050000;
}
```

onCreate

```
public class ClickCountActivity extends Activity {
       TextView countOutput;
       int count = 0;

    @Override
     public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        countOutput = (TextView) this.findViewById(R.id.countOutput);
    }
```

When app is started ClickCountActivity is created and "onCreate" is called. The file res/layout/main.xml is read and the view described in it is creates. R.layout.main is a reference to that view object. In the file res/layout/main.xml one can give GUI elements ids. R.id.countOutput returns the GUI element with the id "countOutput"

layout magic

```
<Button
   android:id="@+id/clickButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/clickButtonLabel"
   android:onClick="increase" />
```

"@+id/clickButton" is how we give the id a name. The "@+" tells the compiler(?) to add this to the R.java file. "onClick" indicates the method to call when the button is clicked. The method has one argument of type view, which is the GUI element that generated the click.

Responding to the click

```
public class ClickCountActivity extends Activity {
        TextView countOutput;
        int count = 0;

    public void increase(View button) {
        Log.i("rew", "increase");
        count++;
        countOutput.setText(String.valueOf(count));
    }
```

increase is the method that the button will call when clicked (see previous slide). The argument is the button that caused the method to be called. In this case we only have one button so we do not use the argument. Log.i is a log statement. All log statements are sent to "LogCat" not the console. See next slide for more about logging. You may have go to DDMS and select the emulator for the output to show in "LogCat". System.out.print does work, but its output also goes "LogCat" not the console.

Logging

X (or levels)

Log.X(tag, message)
Log.X(tag, message, Exception)

Log file contains a lot of messages

Can filter based on
Tags & Levels

v	Verbose
d	Debug
i	Info
w	Warning
e	Error
wtf	What a Terrible Failure Report condition that should not happen

Documentation states that debug log messages are stripped at runtime
That is false

Android Building Blocks

Basic Android Application Parts

Activities
- UI building block
- Views & Activity subclasses

Content Providers
- Shares data between applications

Intents
- How your code starts a new activity

Services
- Long-running nonGUI code

AndroidManifest.xml

R.java

layouts

Fragments
- Sub-activity UI container
- Android 3.0 & 4

Activity

Code that does some work

Single, focused thing that a user can do

Usually each screen(View) has its own activity

An application may have multiple screens, hence multiple activities

An application runs in its own Linux process

Activities can be viewless

Application

One or more screens (view)

Each screen has an activity

When go to new screen previous activity is stored on back stack

Back button

- Kills current activity
- Makes activity on top of back stack current

Home button

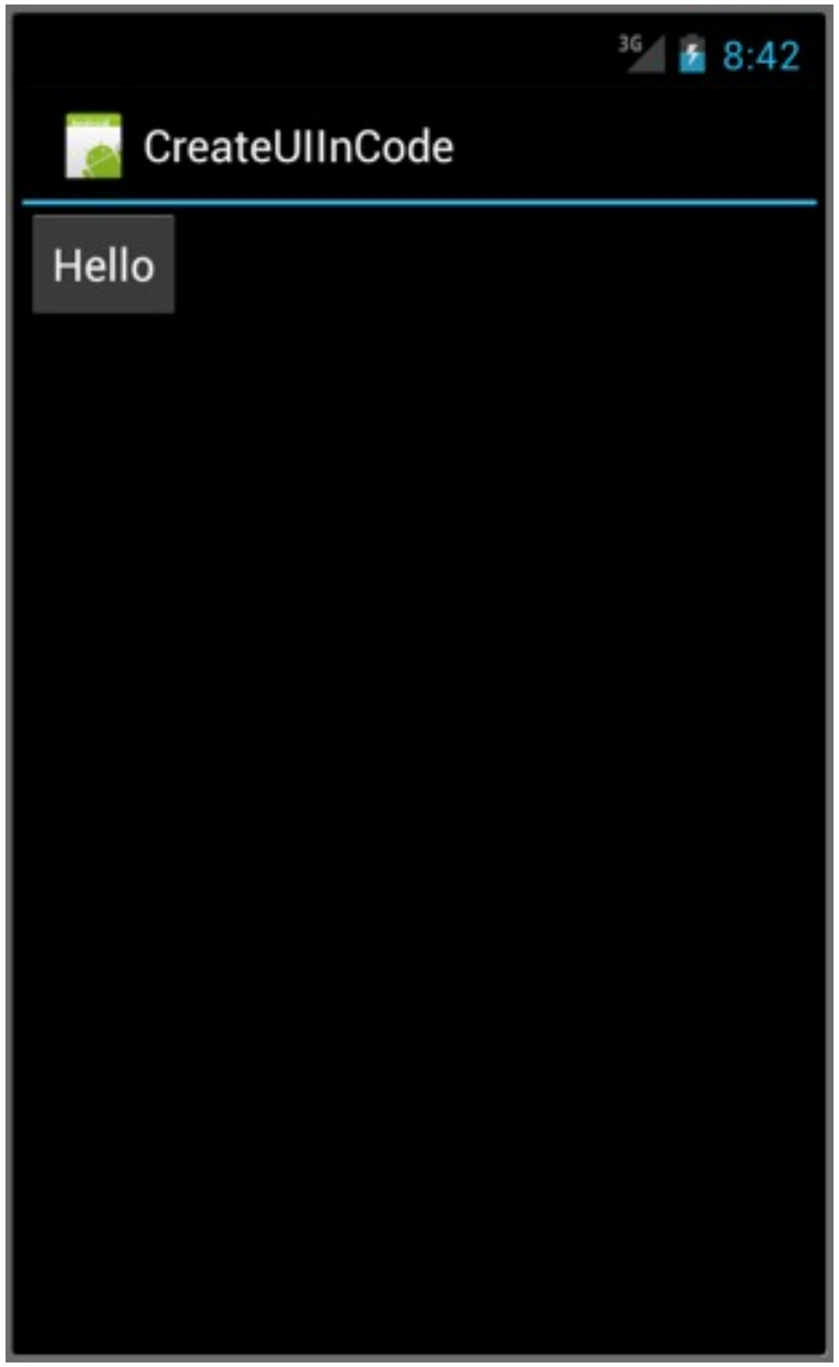
- Suspends current application
- Application and its activities just paused

Some Details

How to create UI

In source Code

Using XML and Graphical Editor



In Source Code

```
public class CreateUIInCodeActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout layout = new LinearLayout(this);
        Button test = new Button(this);
        test.setText("Hello");
        layout.addView(test,
            new LinearLayout.LayoutParams(
                ViewGroup.LayoutParams.WRAP_CONTENT,
                ViewGroup.LayoutParams.WRAP_CONTENT,
                0));
        setContentView(layout);
    }
}
```

Layouts

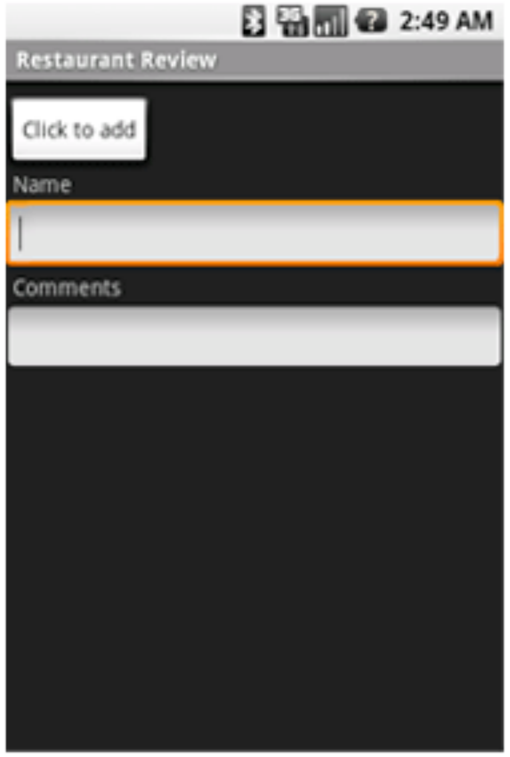
Organize UI elements on screen

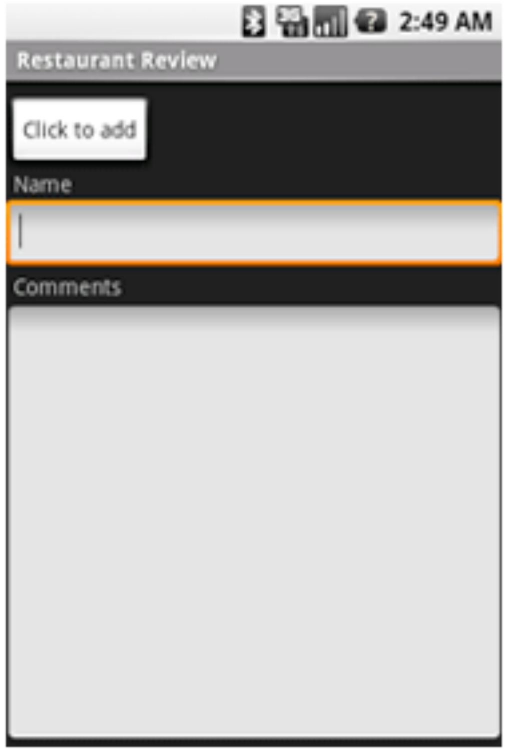
Common Layouts

FrameLayout - displays one child

LinearLayout

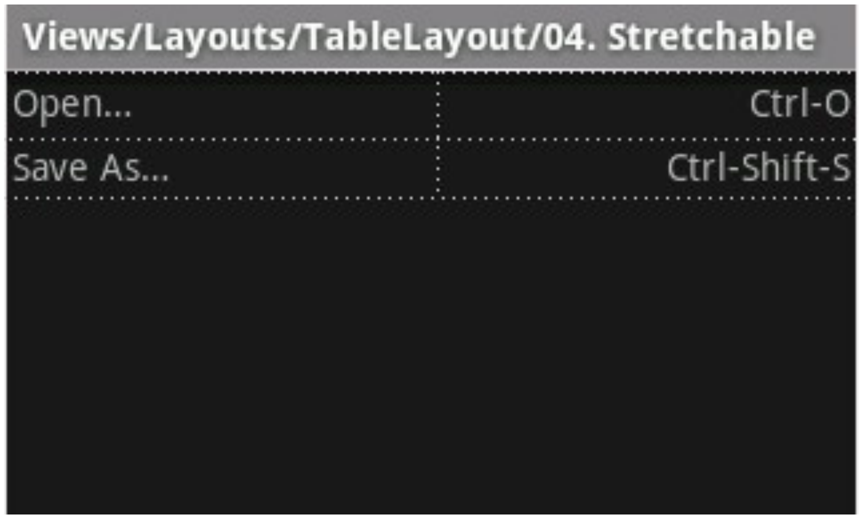
Items stacked vertically or horizontally





Common Layouts

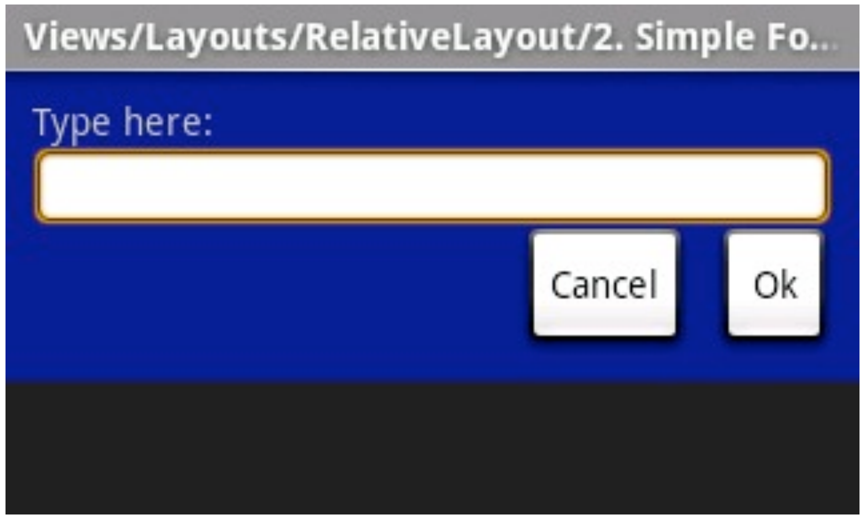
Tablelayout



Views/Layouts/TableLayout/04. Stretchable

Open...	Ctrl-O
Save As...	Ctrl-Shift-S

RelativeLayout



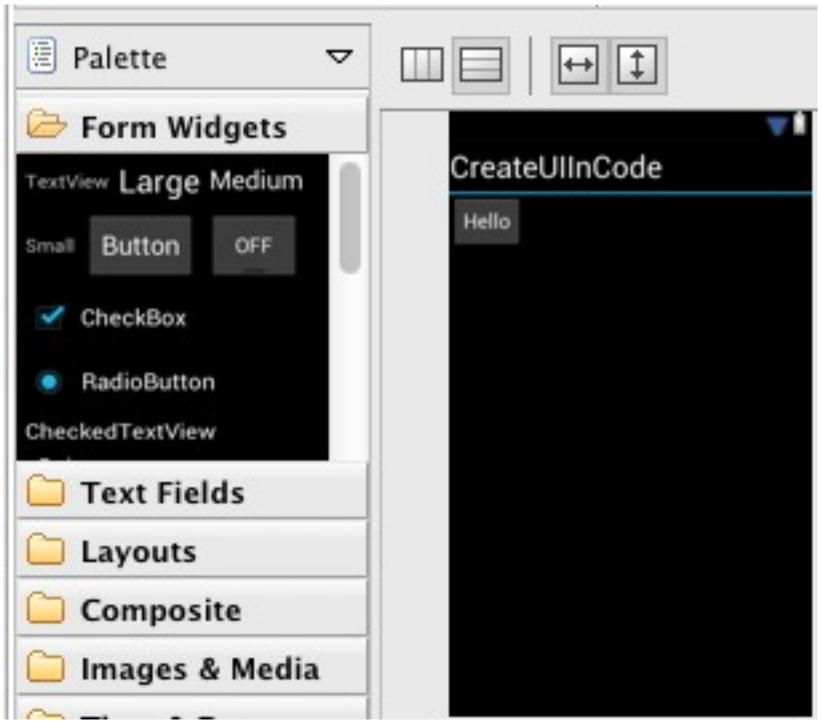
Views/Layouts/RelativeLayout/2. Simple Fo...

Type here:

Layout Documentation with Examples

http://developer.android.com/guide/topics/ui/layout-objects.html

In XML and Graphical Editor



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

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

Graphical Editor Demo

How to connect UI widgets to code

Two Directions

How does code get reference to UI elements

How do UI elements call code

How does code get reference to UI elements

If create UI elements in code - have reference

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    LinearLayout layout = new LinearLayout(this);
    Button test = new Button(this);
    test.setText("Hello");
    layout.addView(test,
        new LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.WRAP_CONTENT,
            ViewGroup.LayoutParams.WRAP_CONTENT,
            0));
    setContentView(layout);
 }
```

How does code get reference to UI elements

If create UI in XML layout - id

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button test = (Button) this.findViewById(R.id.test);
}
```

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

How do UI elements call code

In xml layout - onXXX

```
<Button
    android:id="@+id/test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    android:onClick="makeToast"/>
```

```
public class CreateUIInCodeActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void makeToast(View source) {
       Toast.makeText(this, "Hello World", Toast.LENGTH_SHORT).show();
    }
}
```

How do UI elements call code

In code - Listeners

GestureDetector.OnGestureListener	Notify when gestures occur
MenuItem.OnMenuItemClickListener	a menu item is clicked.
View.OnClickListener	a view is clicked.
View.OnCreateContextMenuListener	the context menu for this view is being built.
View.OnFocusChangeListener	the focus state of a view changed.
View.OnKeyListener	a key event is dispatched to this view.
View.OnLongClickListener	a view has been clicked and held.
View.OnTouchListener	a touch event is dispatched to this view.
ViewGroup.OnHierarchyChangeListener	the hierarchy within this view changed.
ViewStub.OnInflateListener	ViewStub has successfully inflated its layout resource.
ViewTreeObserver.OnGlobalFocusChangeListener	the focus state within the view tree changes.
ViewTreeObserver.OnGlobalLayoutListener	the global layout state or the visibility of views within the view tree changes.
ViewTreeObserver.OnPreDrawListener	the view tree is about to be drawn.
ViewTreeObserver.OnTouchModeChangeListener	the touch mode changes.

See View overview at: http://code.google.com/android/reference/android/view/package-summary.html for more information and links to each listener

What is a Listener?

Java Interface

View.OnClickListener

abstract void onClick(View v)

Called when a view has been clicked.

UI elements call methods on concrete Listener object in response to user action.

Have to add the concrete Listener to UI element

Using the Listener

```
public class CreateUIInCodeActivity extends Activity implements View.OnClickListener{
      Button test;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        test = (Button) this.findViewById(R.id.test);
        test.setOnClickListener(this);
    }

    public void onClick(View source) {
      Toast.makeText(this, "Hello World", Toast.LENGTH_SHORT).show();
    }
}
```

What if we have Multiple Things to click

```
public class CreateUIInCodeActivity extends Activity implements View.OnClickListener{
    Button test;

  @Override
  public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     this.findViewById(R.id.test);
     test.setOnClickListener(this);
     this.findViewById(R.id.OtherButton).setOnClickListener(this);
  }

  public void onClick(View source) {
    if (source == test )
         handle button test click
   else
       handle other other case
  }
}
```

It does not take too many cases for the onClick method to become ugly. The xml onclick is cleaner, but it does hide the connection in the layout file. Or one can use anonymous classes

Using an Anonymous Class

```
public class CreateUIInCodeActivity extends Activity  {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      Button test = (Button) this.findViewById(R.id.test);
      test.setOnClickListener(new View.OnClickListener() {
                  @Override
                  public void onClick(View source) {
                       makeToast();
                  }
      });
   }

   public void makeToast() {
     Toast.makeText(this, "Hello World", Toast.LENGTH_SHORT).show();
   }
```

The anonymous class allows us to specify which method to call instead of onClick, but syntax is verbose and ugly.