Android Programming
Lecture 3

9/9/2011

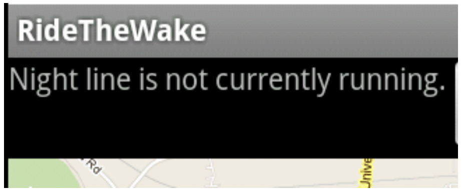
Graphical User Interface Components

- Views:
  - Single widgets or controls
  - How the user interacts with your application

- ViewGroups:
  - One or more views combined together
  - Two uses:
    - Layouts: Invisible, control the flow of other widgets
    - Advanced widgets: Visible, implement complex controls

Simple View Items

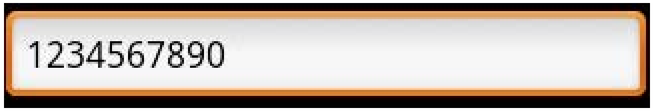
- TextView



RideTheWake

Night line is not currently running.

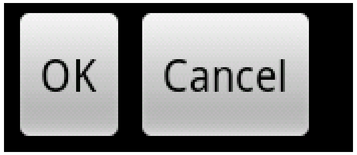
- EditText



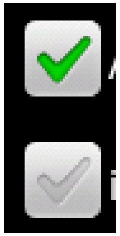
1234567890

Can also be used as a password field

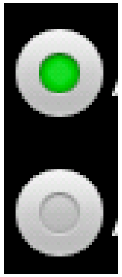
- Button:



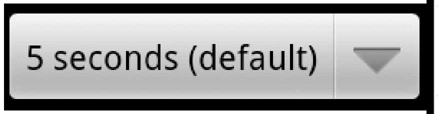
- CheckBox:



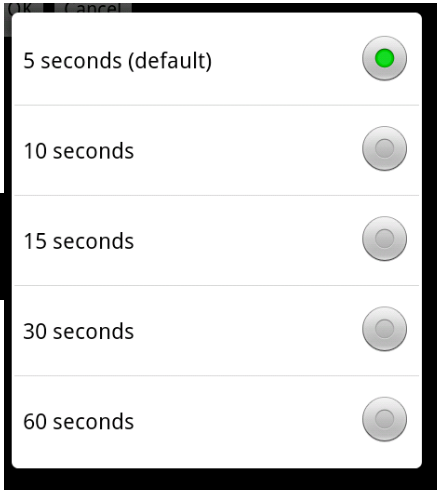
- RadioButton:



- Spinner:

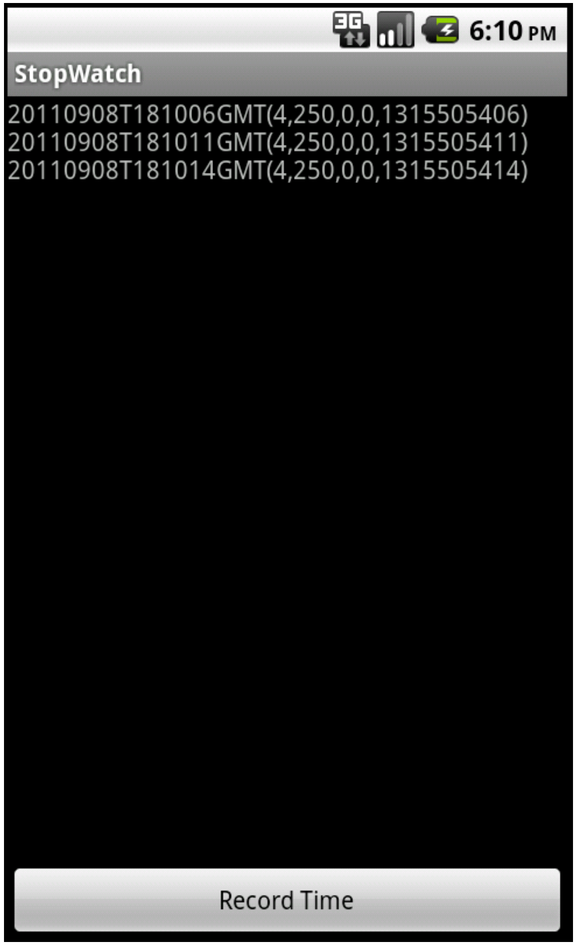


5 seconds (default)



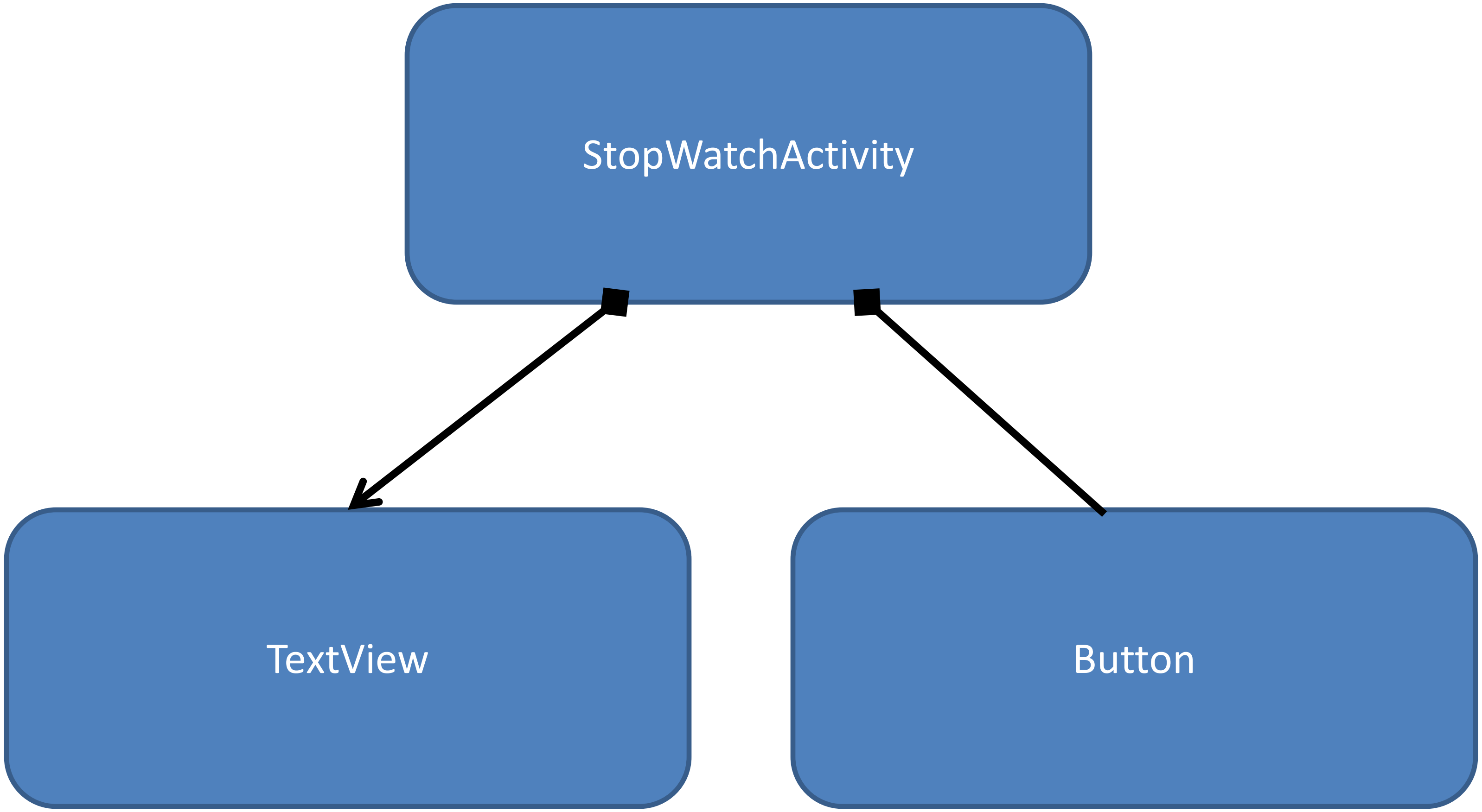
GUIs By Example: StopWatch App

- When the “Record Time” button is clicked, the current time should be printed (along with the result of all previous button presses)



```
20110908T181006GMT(4,250,0,0,1315505406)
20110908T181011GMT(4,250,0,0,1315505411)
20110908T181014GMT(4,250,0,0,1315505414)
```

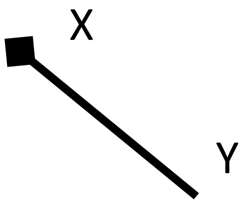
StopWatch App



StopWatchActivity

TextView

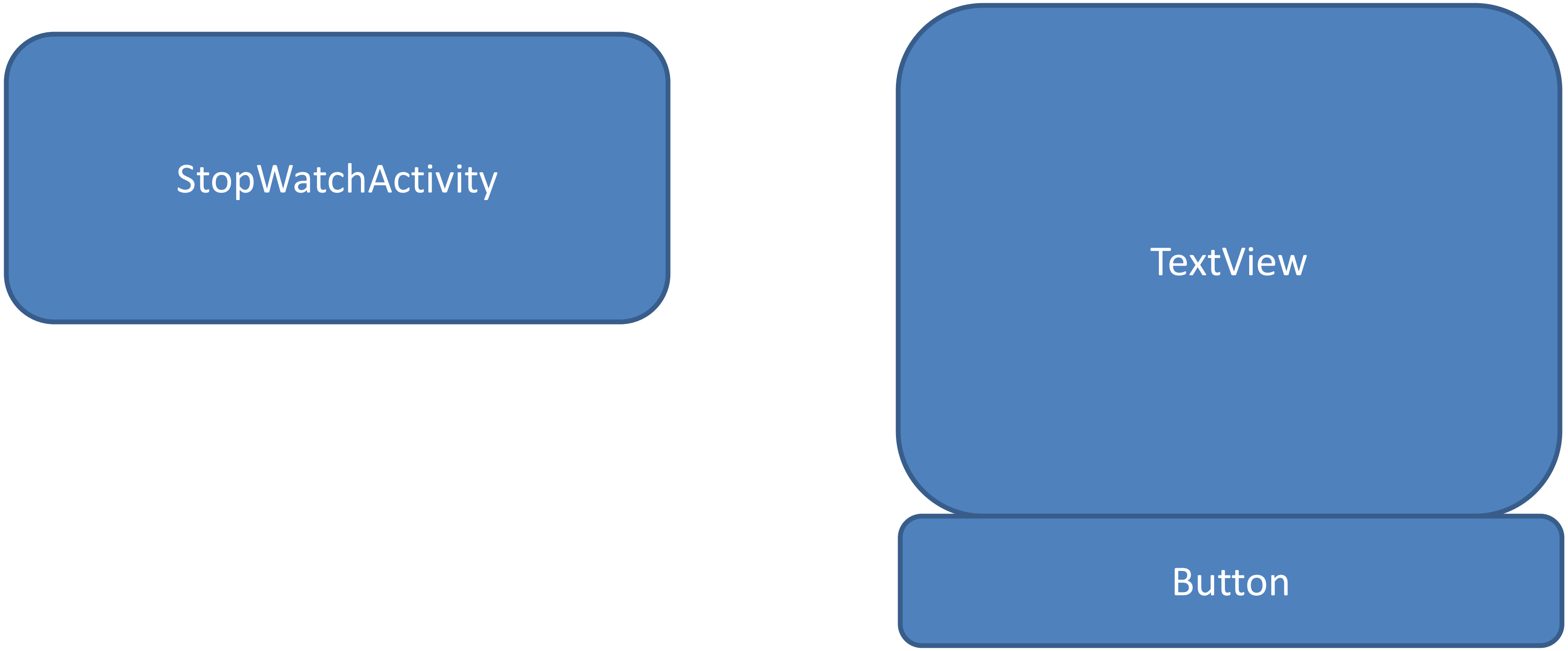
Button



UML Diagram: UML = Unified Modeling Language
http://en.wikipedia.org/wiki/Unified_Modeling_Language

Composition:
X is composed of Y

StopWatch App: Content View



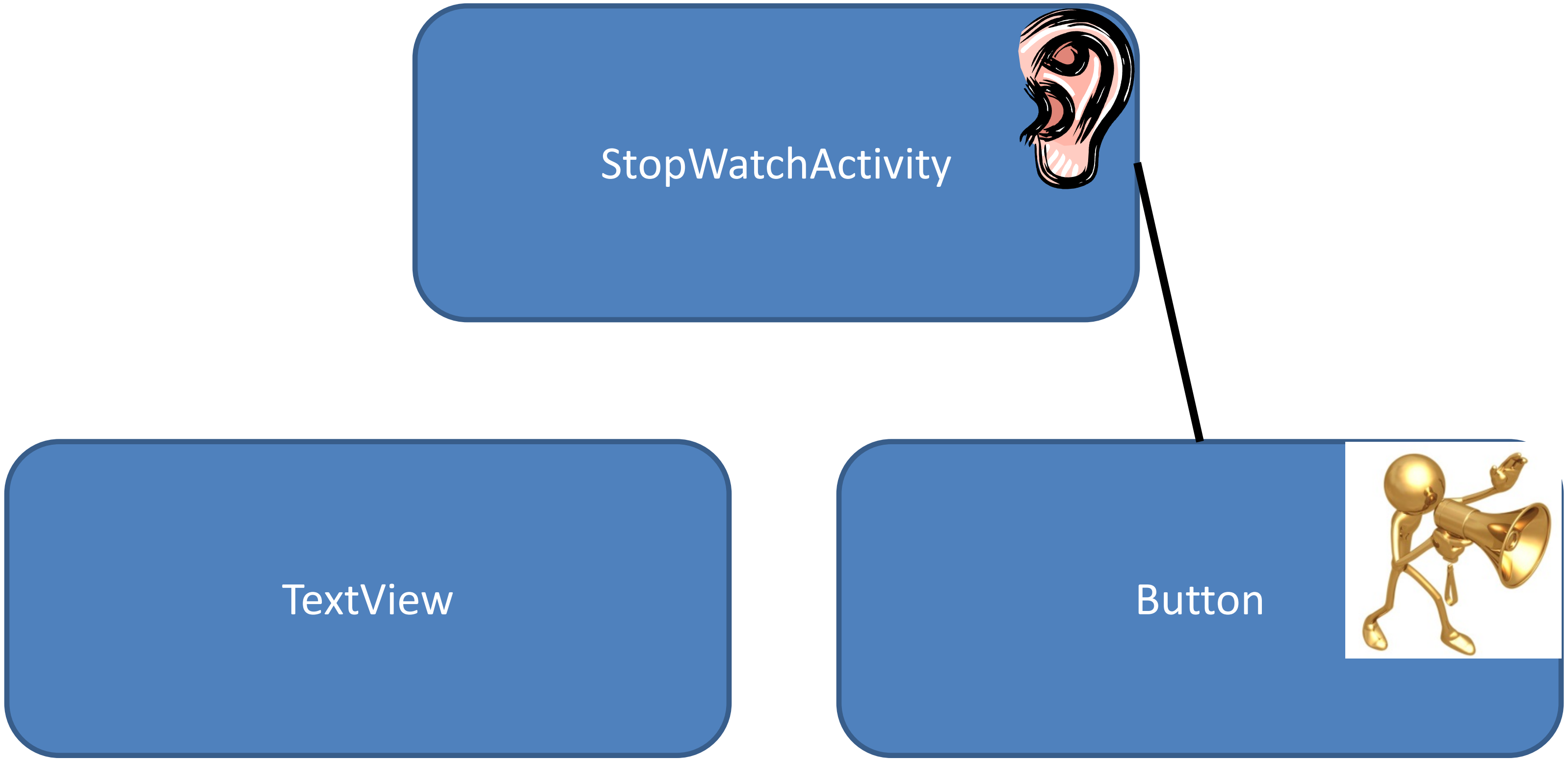
StopWatchActivity

TextView

Button

The ContentView (what is displayed for the Activity [screen]) needs to be set to be an appropriately layed-out positioning of the TextView and Button

StopWatch App



StopWatchActivity

TextView

Button

The Button can be “clicked” by the user. Since the TextView is unaware of the Button, the StopWatchActivity will be set as a listener to listen when the Button announces it was clicked and the StopWatchActivity will then update the TextView.

Layouts

- Layouts are ViewGroups which are used to hold other Views
- Invisible
- Allow *positioning* of different elements
- Layouts can be nested inside of each other

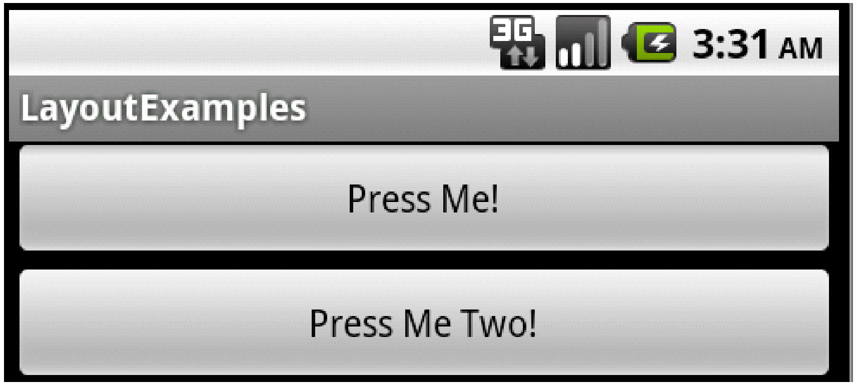
- Common layouts:
  - FrameLayout
  - LinearLayout
  - TableLayout
  - RelativeLayout
  - Gallery

```
import android.widget.NAME;
```

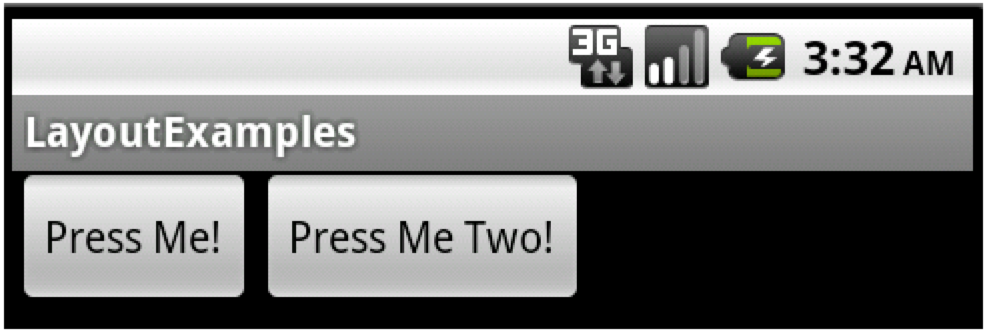
http://developer.android.com/guide/topics/ui/layout-objects.html

LinearLayout

- Vertical: Makes one column of views



- Horizontal: Makes one row of views



Additional Layout Parameters

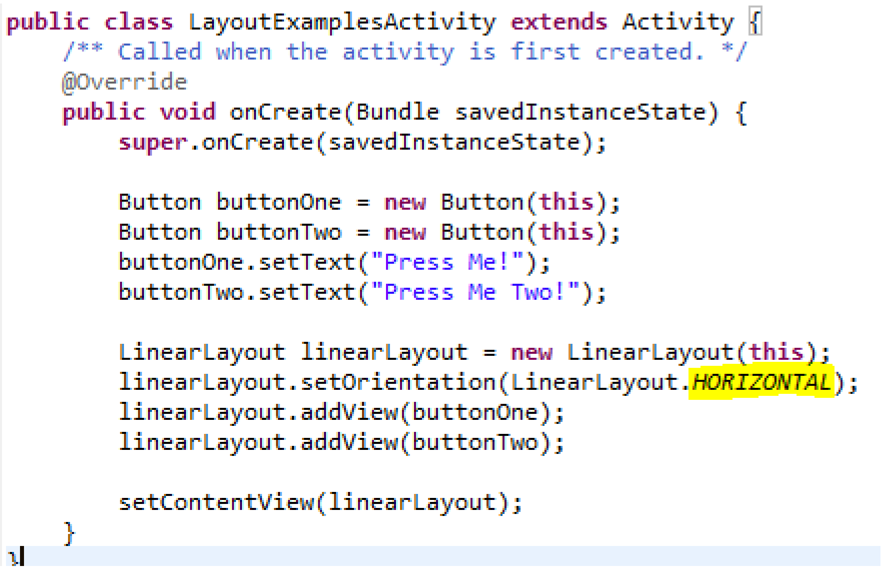
- When a View (such as a Button) is added to a Layout, parameters can be set on how that View is considered within the Layout

- FILL_PARENT vs. WRAP_CONTENT:
  - FILL_PARENT: Expand the View as much as possible to fill the space of the parent container (Layout)
  - WRAP_CONTENT: Make the view be just large enough to hold its contents
  - Can apply to both width and height of View

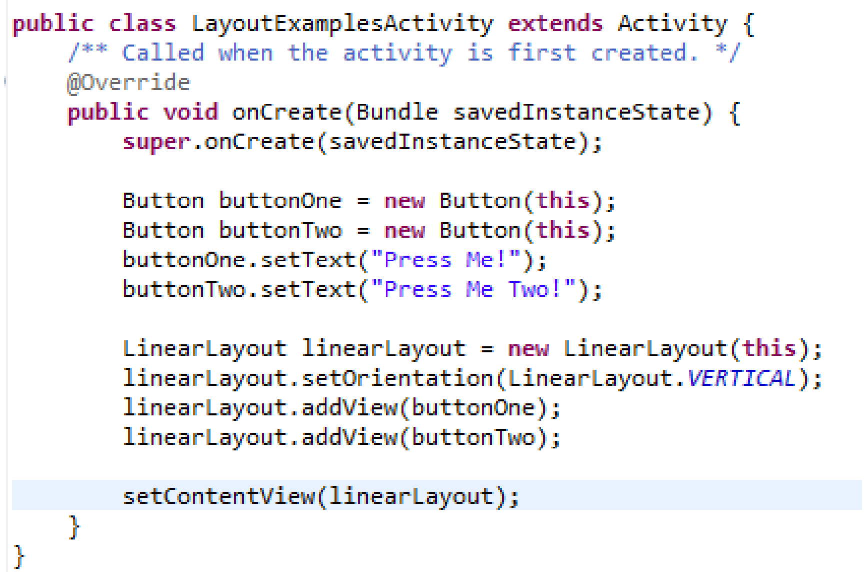
- LAYOUT_WEIGHT: A weighting indicating relative sizes of multiple Views when sharing a layout

- Need to reference API to see defaults

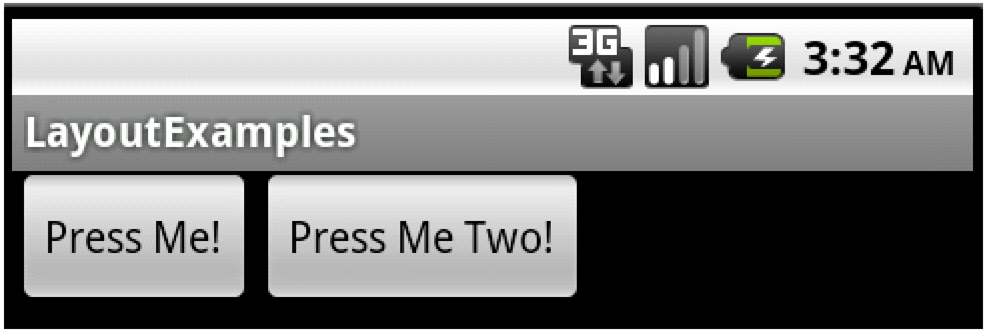
LinearLayout: Simple Examples

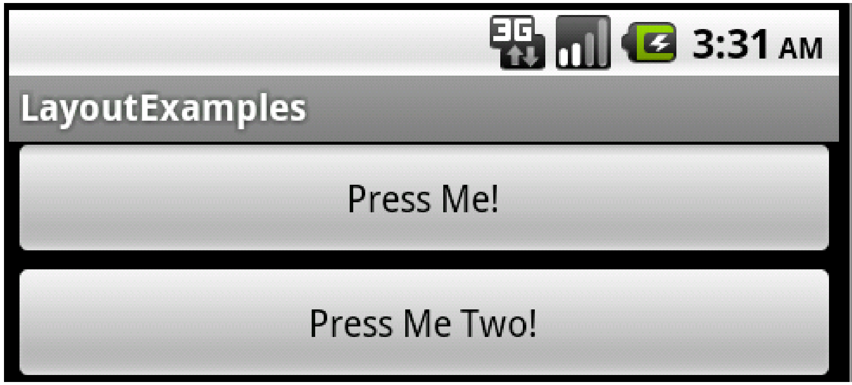


```java
public class LayoutExamplesActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Button buttonOne = new Button(this);
        Button buttonTwo = new Button(this);
        buttonOne.setText("Press Me!");
        buttonTwo.setText("Press Me Two!");

        LinearLayout linearLayout = new LinearLayout(this);
        linearLayout.setOrientation(LinearLayout.HORIZONTAL);
        linearLayout.addView(buttonOne);
        linearLayout.addView(buttonTwo);

        setContentView(linearLayout);
    }
}
```



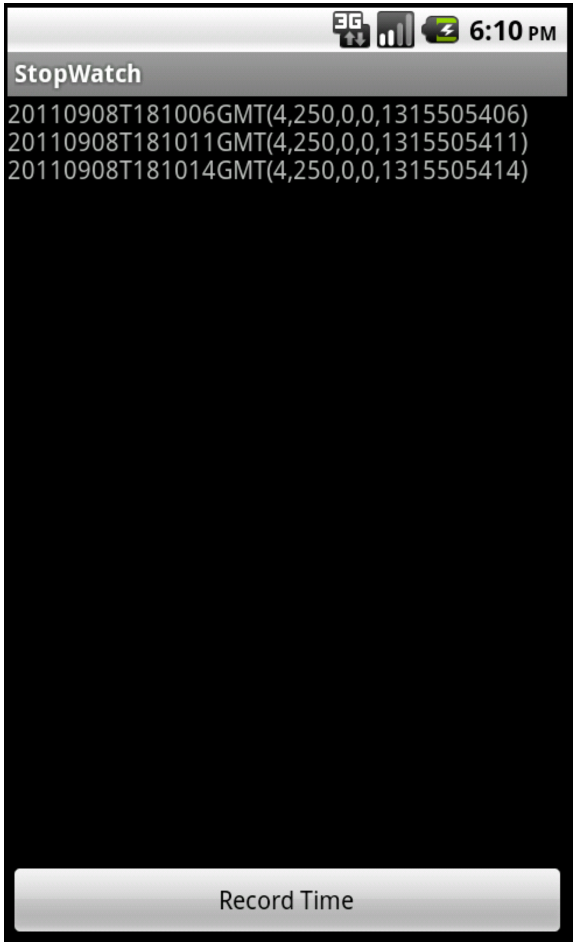
```java
public class LayoutExamplesActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Button buttonOne = new Button(this);
        Button buttonTwo = new Button(this);
        buttonOne.setText("Press Me!");
        buttonTwo.setText("Press Me Two!");

        LinearLayout linearLayout = new LinearLayout(this);
        linearLayout.setOrientation(LinearLayout.VERTICAL);
        linearLayout.addView(buttonOne);
        linearLayout.addView(buttonTwo);

        setContentView(linearLayout);
    }
}
```





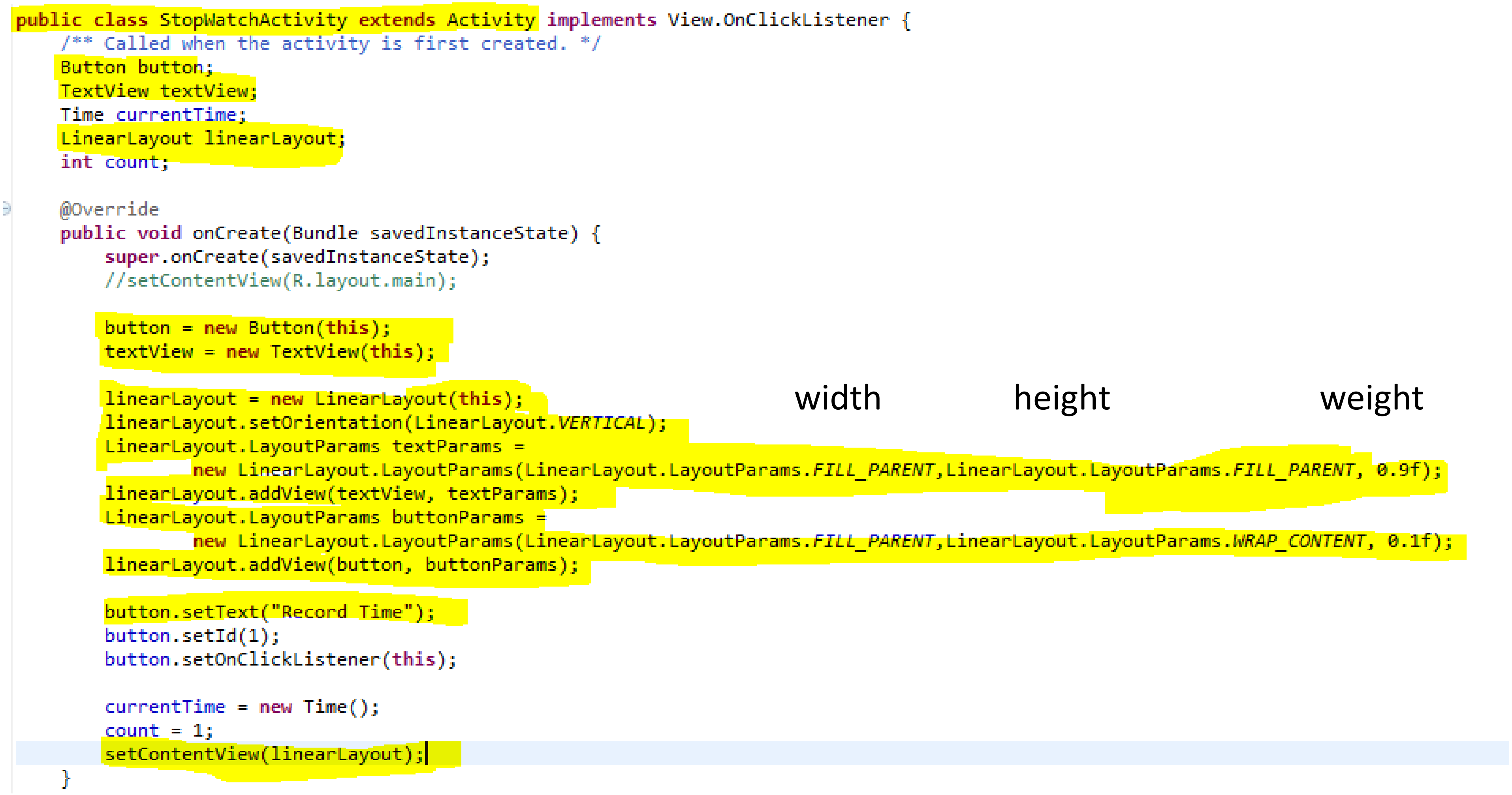
GUIs By Example: StopWatch App

- When the “Record Time” button is clicked, the current time should be printed (along with the result of all previous button presses)



```
20110908T181006GMT(4,250,0,0,1315505406)
20110908T181011GMT(4,250,0,0,1315505411)
20110908T181014GMT(4,250,0,0,1315505414)
```

StopWatch App: Layout



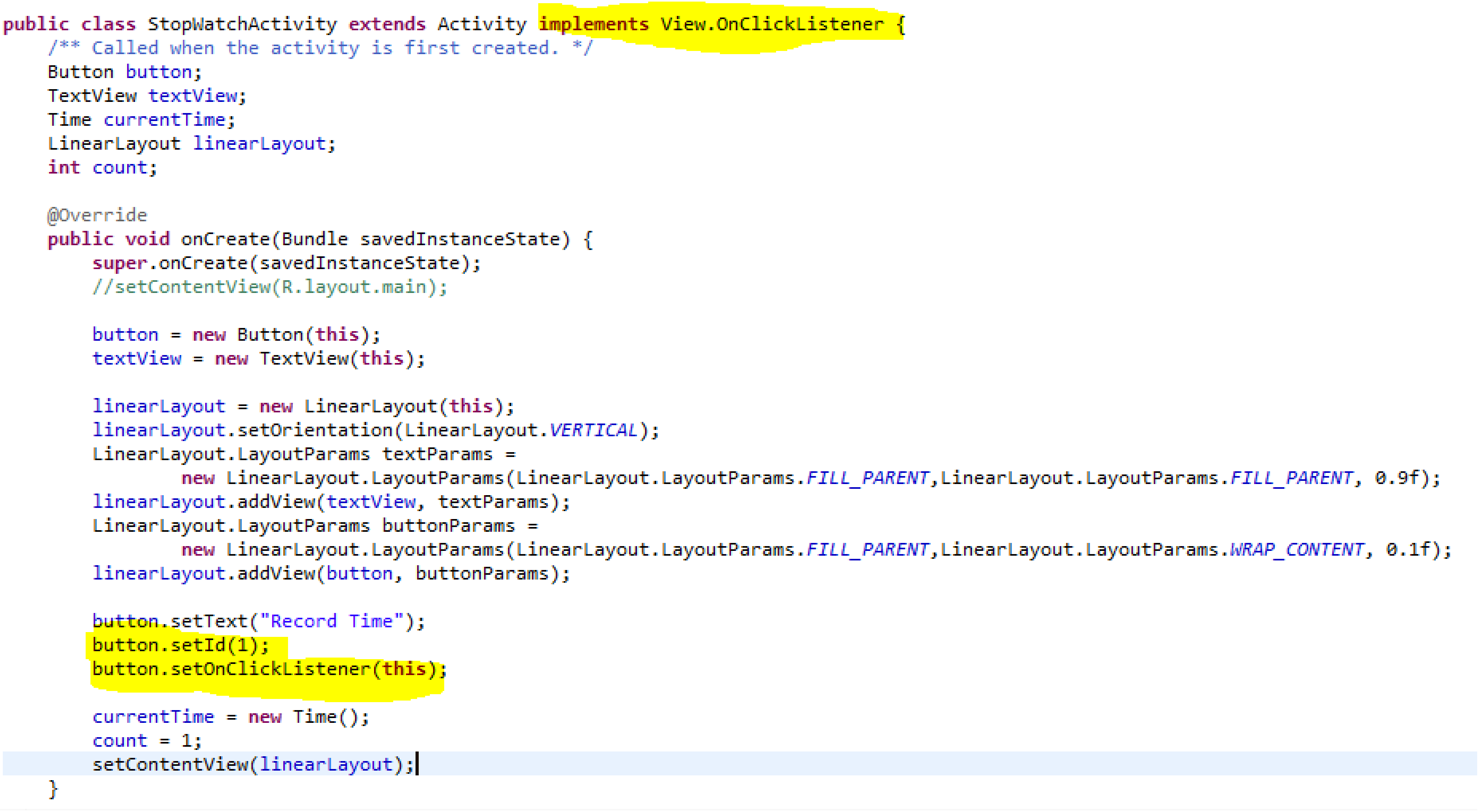
```
public class StopWatchActivity extends Activity implements View.OnClickListener {
    /** Called when the activity is first created. */
    Button button;
    TextView textView;
    Time currentTime;
    LinearLayout linearLayout;
    int count;

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

        button = new Button(this);
        textView = new TextView(this);

        linearLayout = new LinearLayout(this);
        linearLayout.setOrientation(LinearLayout.VERTICAL);
        LinearLayout.LayoutParams textParams =
                new LinearLayout.LayoutParams(LinearLayout.LayoutParams.FILL_PARENT,LinearLayout.LayoutParams.FILL_PARENT, 0.9f);
        linearLayout.addView(textView, textParams);
        LinearLayout.LayoutParams buttonParams =
                new LinearLayout.LayoutParams(LinearLayout.LayoutParams.FILL_PARENT,LinearLayout.LayoutParams.WRAP_CONTENT, 0.1f);
        linearLayout.addView(button, buttonParams);

        button.setText("Record Time");
        button.setId(1);
        button.setOnClickListener(this);

        currentTime = new Time();
        count = 1;
        setContentView(linearLayout);
    }
```

width

height

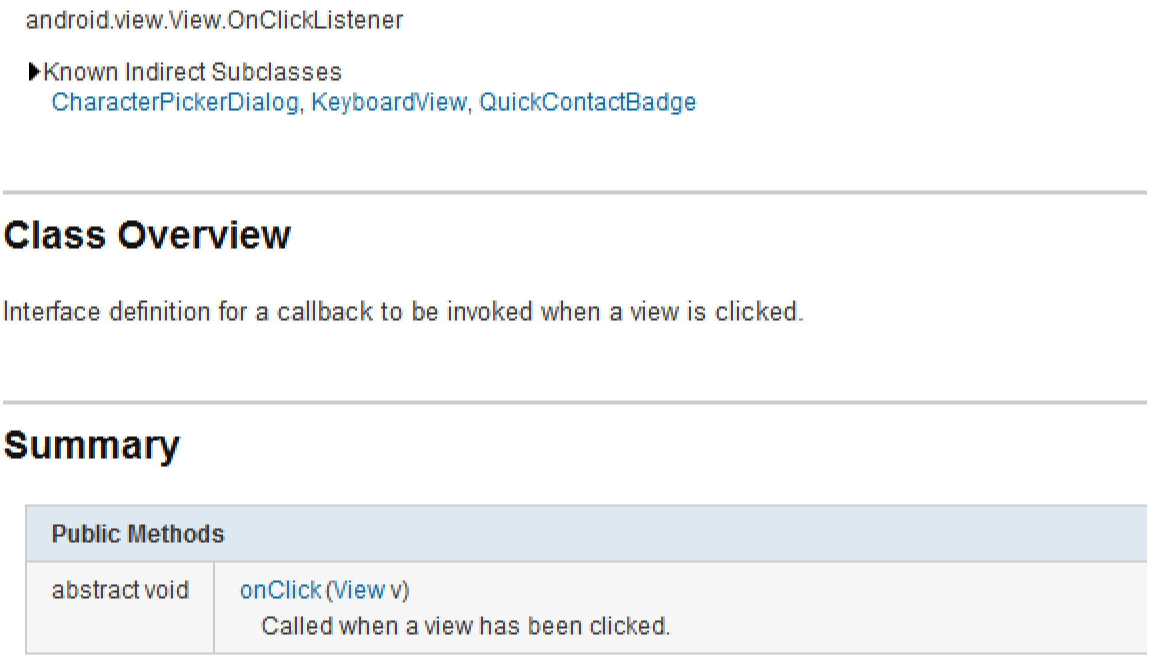
weight

StopWatch App: Button Click Listener (Adding Activity as Listener to Button)



```
public class StopWatchActivity extends Activity implements View.OnClickListener {
    /** Called when the activity is first created. */
    Button button;
    TextView textView;
    Time currentTime;
    LinearLayout linearLayout;
    int count;

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

        button = new Button(this);
        textView = new TextView(this);

        linearLayout = new LinearLayout(this);
        linearLayout.setOrientation(LinearLayout.VERTICAL);
        LinearLayout.LayoutParams textParams =
                new LinearLayout.LayoutParams(LinearLayout.LayoutParams.FILL_PARENT,LinearLayout.LayoutParams.FILL_PARENT, 0.9f);
        linearLayout.addView(textView, textParams);
        LinearLayout.LayoutParams buttonParams =
                new LinearLayout.LayoutParams(LinearLayout.LayoutParams.FILL_PARENT,LinearLayout.LayoutParams.WRAP_CONTENT, 0.1f);
        linearLayout.addView(button, buttonParams);

        button.setText("Record Time");
        button.setId(1);
        button.setOnClickListener(this);

        currentTime = new Time();
        count = 1;
        setContentView(linearLayout);
    }
```

StopWatch App: Button Click Listener (Responding to the onClick Message)



android.view.View.OnClickListener

▸Known Indirect Subclasses
CharacterPickerDialog, KeyboardView, QuickContactBadge

Class Overview

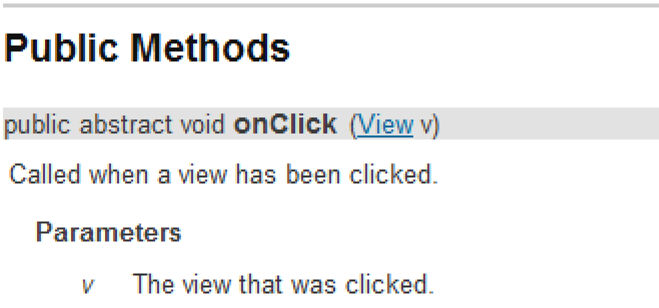
Interface definition for a callback to be invoked when a view is clicked.

Summary

Public Methods	
abstract void	onClick (View v) Called when a view has been clicked.

One Listener can listen to multiple subjects

Check id of listener that generated the event



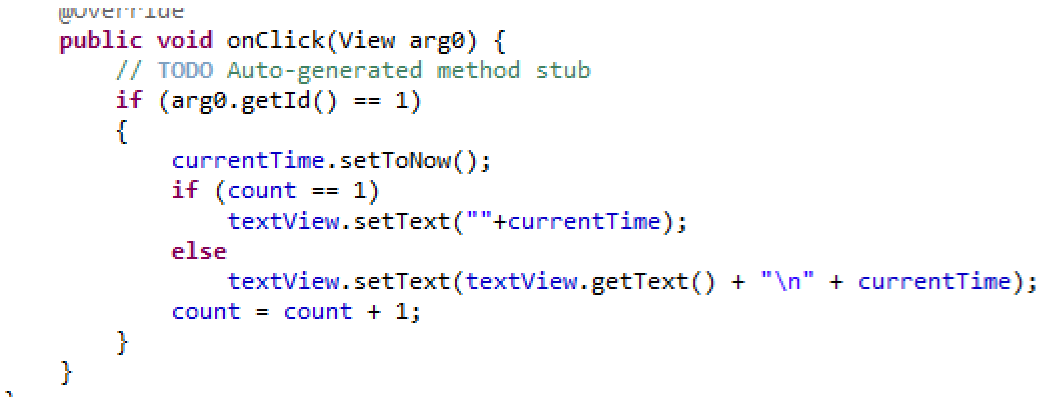
Public Methods

public abstract void **onClick** (View v)

Called when a view has been clicked.

Parameters

v The view that was clicked.

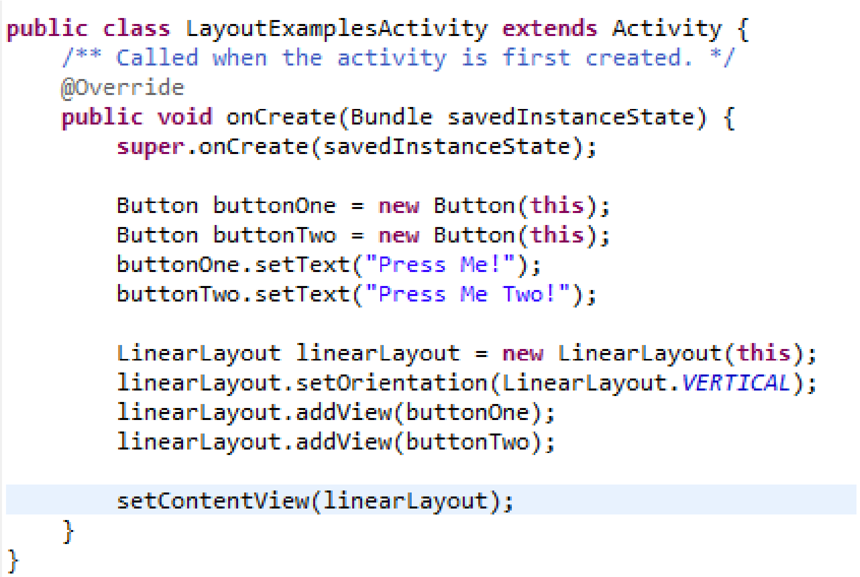


```
@Override
public void onClick(View arg0) {
    // TODO Auto-generated method stub
    if (arg0.getId() == 1)
    {
        currentTime.setToNow();
        if (count == 1)
            textView.setText(""+currentTime);
        else
            textView.setText(textView.getText() + "\n" + currentTime);
        count = count + 1;
    }
}
```

Specifying Layouts in XML

- It is very common to specify layouts in a text instead of code format
- For main activity, layout specified in *res/layout/main.xml*
- **XML: Extensible Markup Language**
  - Similar to HTML
  - Markup tags (< >, opening, /closing), Attributes=Values (x=y), Content (text [rare actually])
  - Nesting

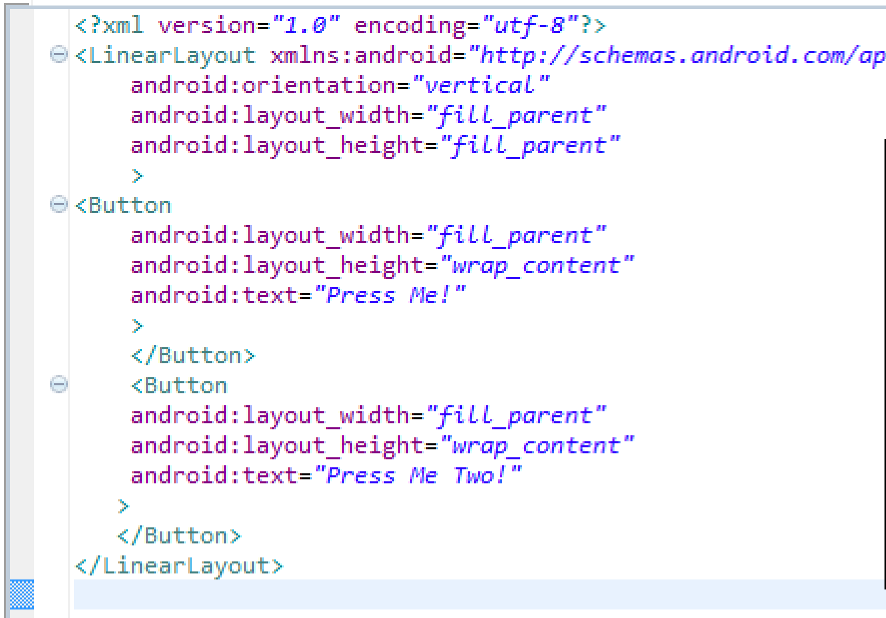
Specifying Layouts in XML



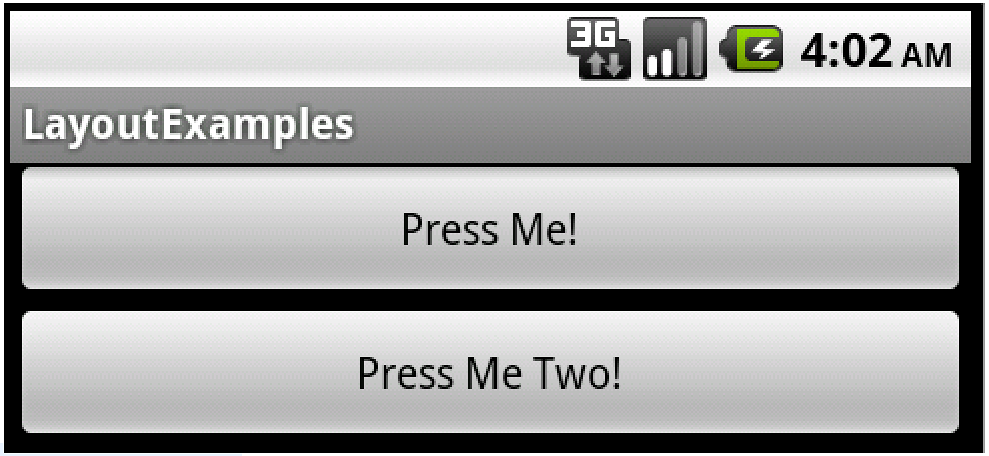
```java
public class LayoutExamplesActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Button buttonOne = new Button(this);
        Button buttonTwo = new Button(this);
        buttonOne.setText("Press Me!");
        buttonTwo.setText("Press Me Two!");

        LinearLayout linearLayout = new LinearLayout(this);
        linearLayout.setOrientation(LinearLayout.VERTICAL);
        linearLayout.addView(buttonOne);
        linearLayout.addView(buttonTwo);

        setContentView(linearLayout);
    }
}
```

Code and XML approaches
that generate the same interface

In XML version, the two Buttons are
Nested inside the LinearLayout
(between <LinearLayout></LinearLayout>)



```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"
    >
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Press Me!"
    >
    </Button>
    <Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Press Me Two!"
   >
   </Button>
</LinearLayout>
```


3G
4:02 AM
LayoutExamples
Press Me!
Press Me Two!