



# Accelerating Information Technology Innovation

<http://aiti.mit.edu>

India Summer 2012

Lecture 2 – Android User Interface (Controls)



# Overview

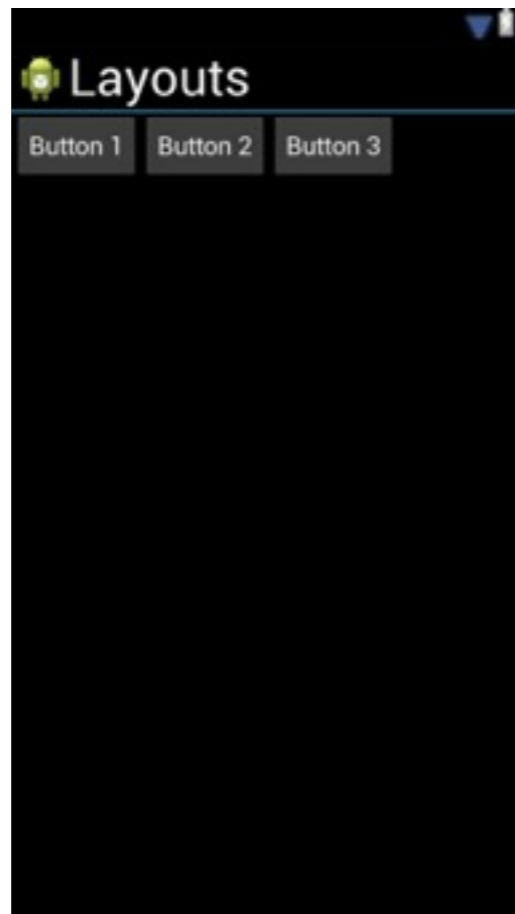
---

- Layouts
- Event “Listeners”
- Types of Android user controls
- Designing user interfaces in Eclipse
- How to use user controls in your code

**What is a layout?**

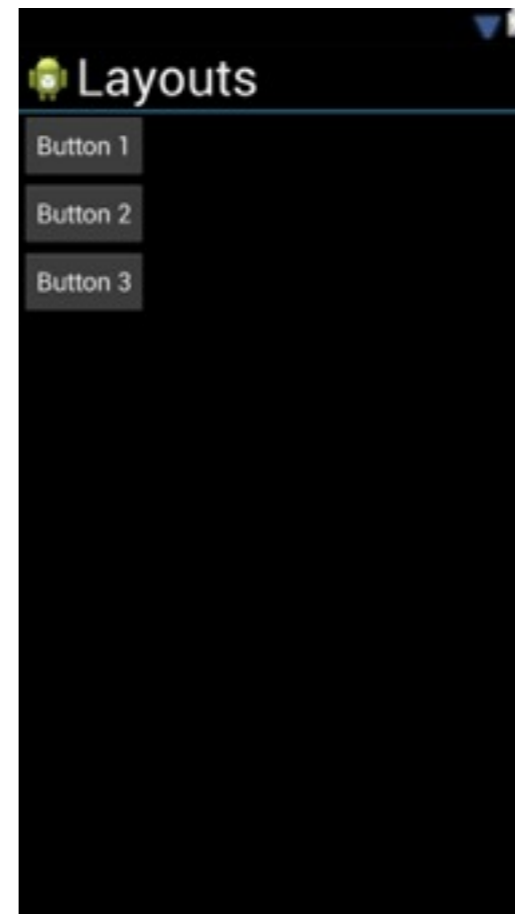
# LinearLayouts

---



Horizontal

*or*

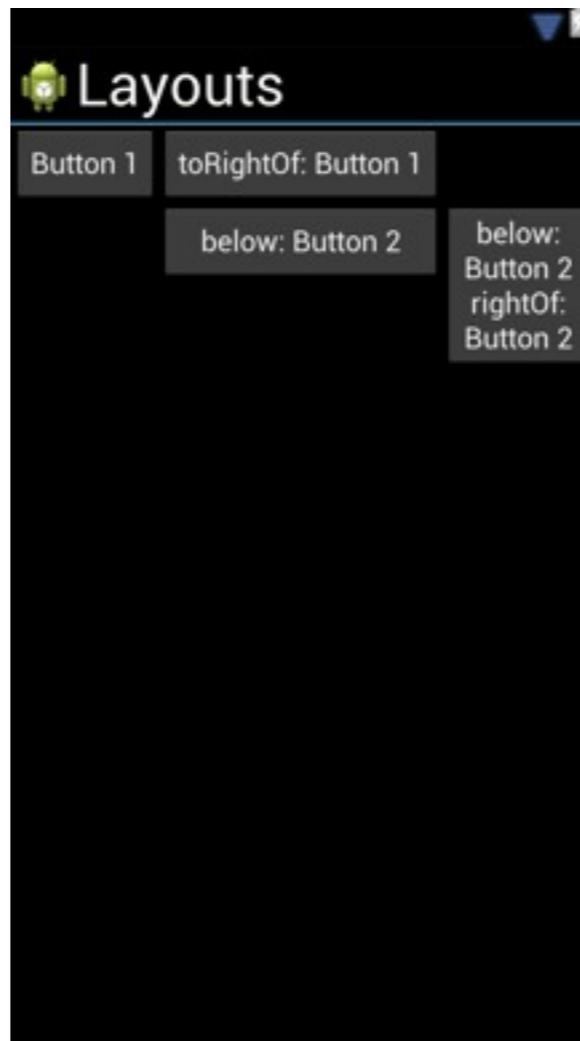


Vertical

Lay out controls in order,  
may display scrollbar

# RelativeLayouts

---



Lay out controls relative to each other

# Event “Listeners”

# Event Listeners

---

- Notify another object/call a method when an object does something
- May be used to communicate changes in state (e.g. when key pressed)
- Related Observer pattern notifies *multiple* objects

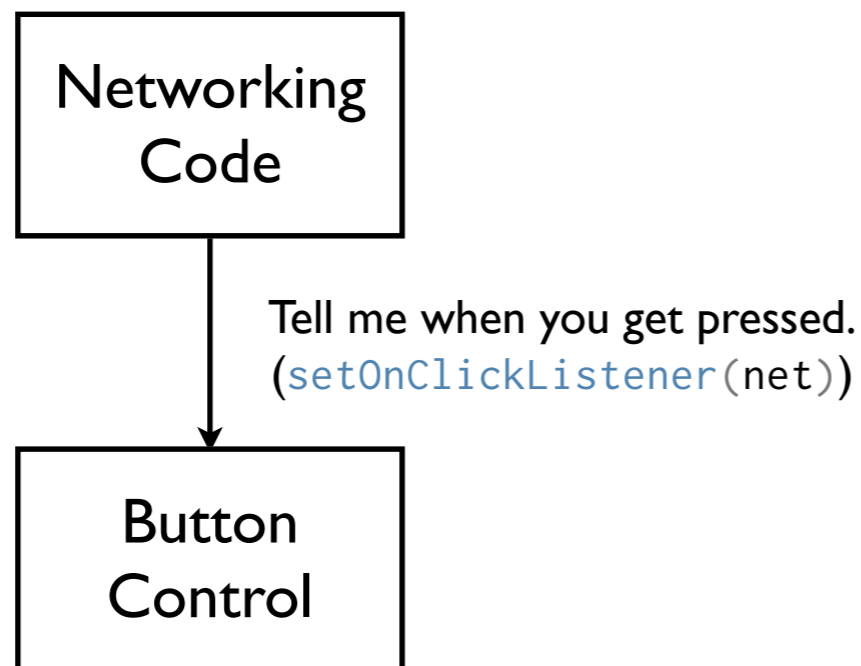
Networking  
Code

Button  
Control

# Event Listeners

---

- Notify another object/call a method when an object does something
- May be used to communicate changes in state (e.g. when key pressed)
- Related Observer pattern notifies *multiple* objects

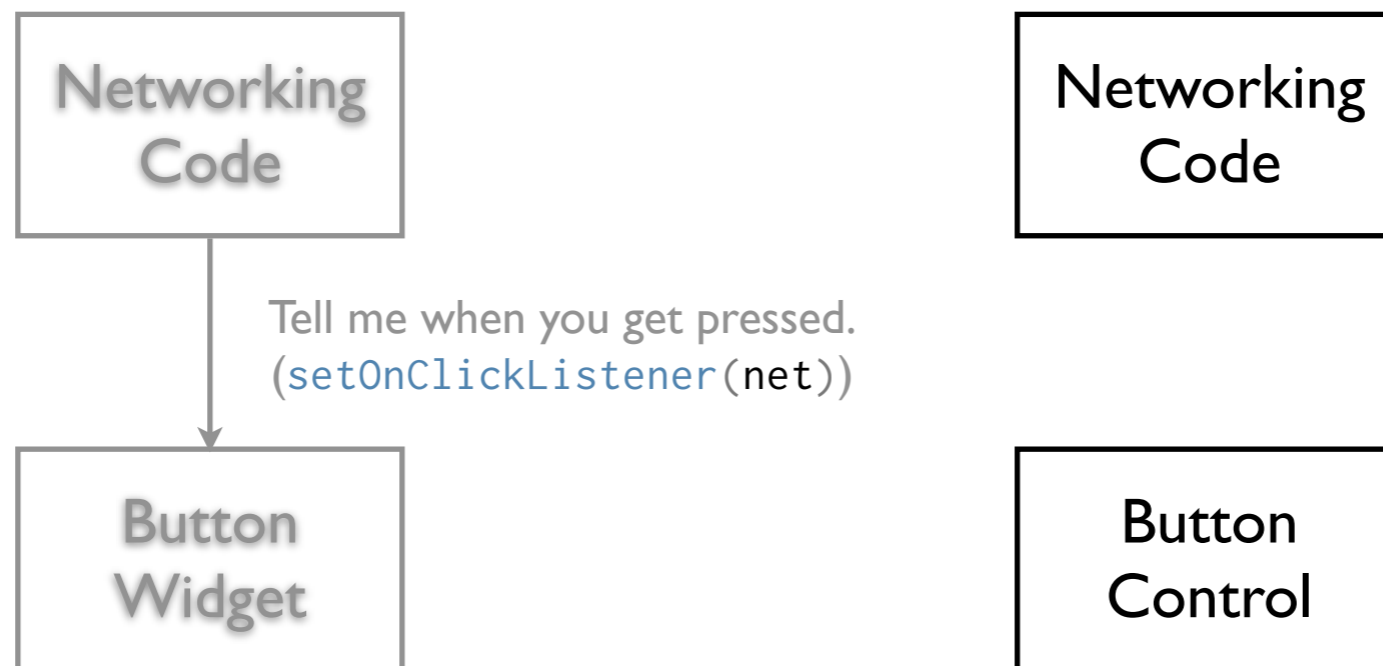




# Event Listeners

---

- Notify another object/call a method when an object does something
- May be used to communicate changes in state (e.g. when key pressed)
- Related Observer pattern notifies *multiple* objects



# Event Listeners

---

- Notify another object/call a method when an object does something
- May be used to communicate changes in state (e.g. when key pressed)
- Related Observer pattern notifies *multiple* objects



# Event Listeners

---

```
public interface OnClickListener {
    abstract void onClick(Position position);
}

public class Button {
    void setListener(OnClickListener x) { this.listener = x; }

    void click() {
        this.mousePosition = Mouse.readPosition();
        if (this.listener != null) {
            this.listener.onClick(this.mousePosition);
        }
    }
}
```

# Event Listeners

---

```
Button myButton = new Button();

myButton.setOnClickListener(new OnClickListener() { // Anonymous!
    public void onClick(Position position) {
        System.out.println("Received click at "
            + position.toString());
    }
});

// The following will print out "Received click at [position]"
myButton.click();
```

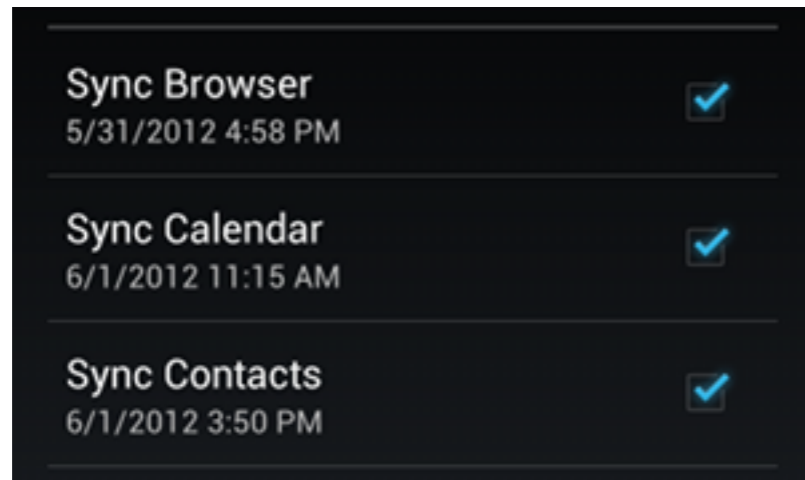
*Note the anonymous class!*

*It could also be an instance of a concrete class implementing  
OnClickListener*

# Types of user controls



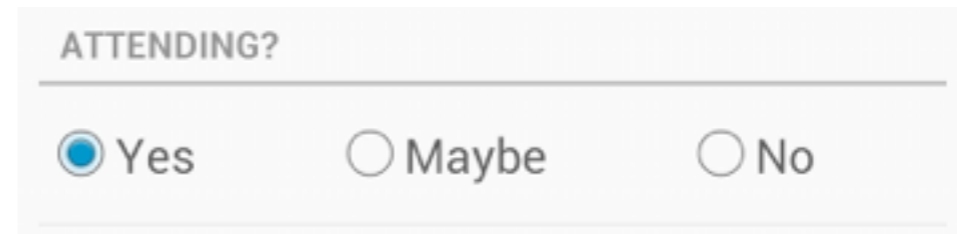
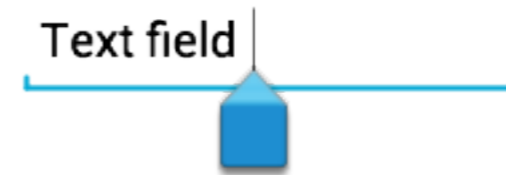
## Buttons



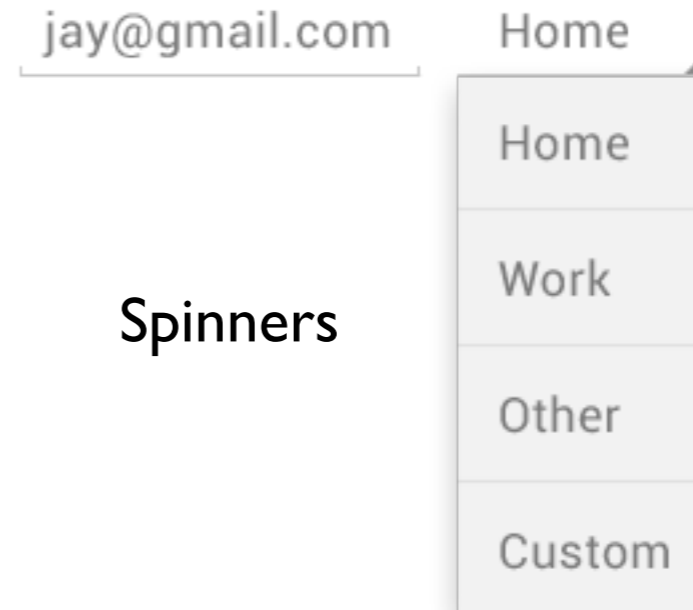
## Checkboxes



## Toggle Buttons



## Radio Buttons



## Spinners

# Other User Interface Components

---

- *TextView* – Display static text
- *WebView* – Display HTML/Websites
- *ImageView* – Display an image
- Composite views to display multiple items:
  - *ListView* – In a vertical list
  - *GridView* – In a (vertical) grid
  - *Gallery* – A horizontal sequential gallery (of images)

# Making interfaces in Eclipse



# Graphical Layout

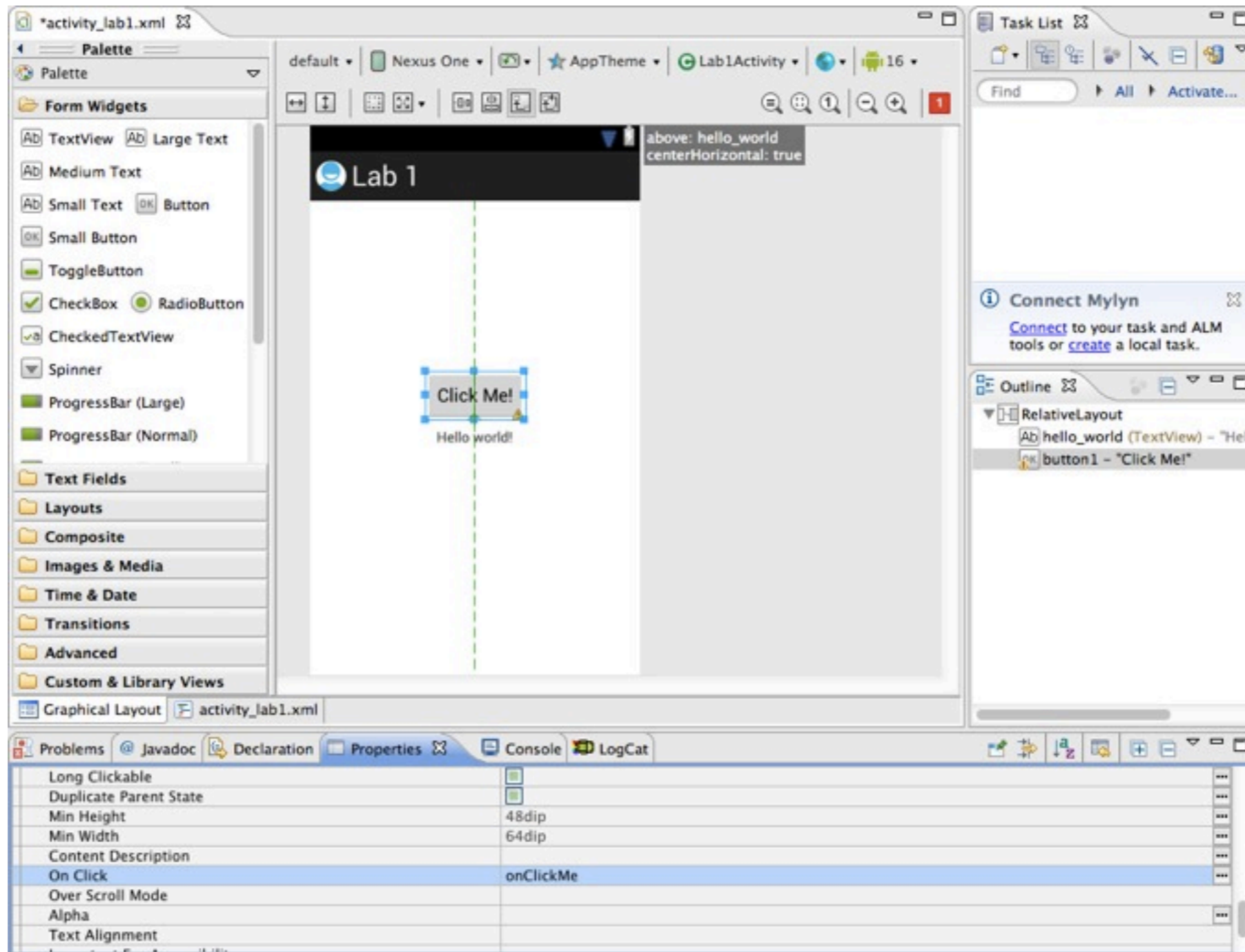
The screenshot displays the graphical layout editor for an Android activity. The interface is divided into several key sections:

- Palette (Left):** A list of UI controls categorized into Form Widgets, Text Fields, Layouts, Composite, Images & Media, Time & Date, Transitions, Advanced, and Custom & Library Views. A red arrow labeled "Controls" points to this area.
- Design View (Center):** A visual representation of the activity layout. It shows a black header bar with a blue back arrow and the text "Lab 1". Below the header, the text "Hello world!" is displayed on a white background.
- Outline (Right):** A tree view showing the layout structure. It indicates a `RelativeLayout` containing a `TextView` with the text "Hello world!". A red arrow labeled "Layout Structure" points to this area.
- Properties (Bottom):** A table showing the properties of the selected `TextView`. A red arrow labeled "Properties" points to this area. A tooltip is visible over the `layout_width` property.

Id	Value
Layout Parameters	{}
Gravity	
<b>layout_width</b>	match_parent
Specifies the basic width of the view.	match_parent
[dimension, enum]	{}
Background	
Padding Left	
Content Description	

# Using controls in your code

# A Simple Button



# A Simple Button: Code View

---

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/hello_world"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:padding="@dimen/padding_medium"
        android:text="@string/hello_world"
        tools:context=".Lab1Activity" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/hello_world"
        android:layout_centerHorizontal="true"
        android:onClick="onClickMe"
        android:text="Click Me!" />

</RelativeLayout>
```

# A Simple Button: Code View

---

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent" >
```

```
<TextView
```

```
  android:id="@+id/hello_world"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_centerHorizontal="true"
  android:layout_centerVertical="true"
  android:padding="@dimen/padding_medium"
  android:text="@string/hello_world"
  tools:context=".Lab1Activity" />
```

*[name the view]*

```
<Button
```

```
  android:id="@+id/button1"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_above="@+id/hello_world"
  android:layout_centerHorizontal="true"
  android:onClick="onClickMe"
  android:text="Click Me!" />
```

*[name the view]*

```
</RelativeLayout>
```

# A Simple Button: Code View

---

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent" >
```

*[expand to fill parent]*

```
<TextView
  android:id="@+id/hello_world"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_centerHorizontal="true"
  android:layout_centerVertical="true"
  android:padding="@dimen/padding_medium"
  android:text="@string/hello_world"
  tools:context=".Lab1Activity" />
```

*[minimum needed size]*

*[horizontally and vertically center]*

```
<Button
  android:id="@+id/button1"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_above="@+id/hello_world"
  android:layout_centerHorizontal="true"
  android:onClick="onClickMe"
  android:text="Click Me!" />
```

*[minimum needed size]*

*[horizontally center above hello\_world]*

```
</RelativeLayout>
```

# A Simple Button: Code View

---

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
```

```
<TextView
    android:id="@+id/hello_world"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:padding="@dimen/padding_medium"
    android:text="@string/hello_world"
    tools:context=".Lab1Activity" />
```

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/hello_world"
    android:layout_centerHorizontal="true"
    android:onClick="onClickMe"
    android:text="Click Me!" />
```

*[name of the method to call when clicked]*

```
</RelativeLayout>
```

# A Simple Button: Code View

---

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent" >
```

```
<TextView
  android:id="@+id/hello_world"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_centerHorizontal="true"
  android:layout_centerVertical="true"
  android:padding="@dimen/padding_medium"
  android:text="@string/hello_world"
  tools:context=".Lab1Activity" />
```

*[reference a string resource]*

```
<Button
  android:id="@+id/button1"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_above="@+id/hello_world"
  android:layout_centerHorizontal="true"
  android:onClick="onClickMe"
  android:text="Click Me!" />
```

*[literal string of the button]*

```
</RelativeLayout>
```



# String Resources

The screenshot displays an IDE interface with the Package Explorer on the left and the Android Resources editor on the right. The Package Explorer shows a project structure with folders like 'src', 'gen', 'Android 4.1', 'Android Dependencies', 'bin', 'libs', 'res', and 'values'. The 'strings.xml' file is selected in the 'values' folder. The Android Resources editor shows a list of resources elements: 'app\_name (String)', 'hello\_world (String)', 'menu\_settings (String)', and 'title\_activity\_lab1 (String)'. The 'hello\_world (String)' resource is selected, and its attributes are shown in the right-hand pane. The 'Name' field contains 'hello\_world' and the 'Value\*' field contains 'Hello world!'. The 'Attributes for hello\_world (String)' pane also includes a descriptive text about strings and their formatting options.

Package Explorer

- AITI Website [trunk/technical/website]
- ILikeMumbai [ILikeMumbai NO-HEAD]
- Lab 1 [lab1 master]
  - src
  - gen [Generated Java Files]
  - Android 4.1
  - Android Dependencies
  - bin
  - libs
  - res
    - drawable-hdpi
    - drawable-ldpi
    - drawable-mdpi
    - drawable-xhdpi
    - layout
      - activity\_lab1.xml
    - menu
    - values
      - dimens.xml
      - strings.xml
      - styles.xml
    - values-large
    - values-v11
    - values-v14
  - AndroidManifest.xml
  - ic\_launcher-web.png
  - proguard-project.txt
  - project.properties

activity\_lab1.xml strings.xml

Android Resources (default)

Resources Elements S C D D S I S I Az

- app\_name (String)
- hello\_world (String)
- menu\_settings (String)
- title\_activity\_lab1 (String)

Add...  
Remove...  
Up  
Down

Attributes for hello\_world (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 hello\_world

Value\* Hello world!

Resources strings.xml

**But what about the code?**

# The Code

The screenshot shows an IDE window with the Package Explorer on the left and the code editor on the right. The Package Explorer shows the project structure for 'Lab 1 [lab1 master]', including the 'src' directory, 'edu.mit.aiti.india.lab1' package, and 'Lab1Activity.java' file. The code editor displays the following Java code:

```
package edu.mit.aiti.india.lab1;

import android.os.Bundle;

public class Lab1Activity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lab1);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_lab1, menu);
        return true;
    }

}
```

# The Code

```
package edu.mit.aiti.india.lab1;

import android.os.Bundle;

public class Lab1Activity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lab1); [instantiate activity_lab1.xml]
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_lab1, menu);
        return true;
    }

}
```

# The Code:

## onClickMe()

---

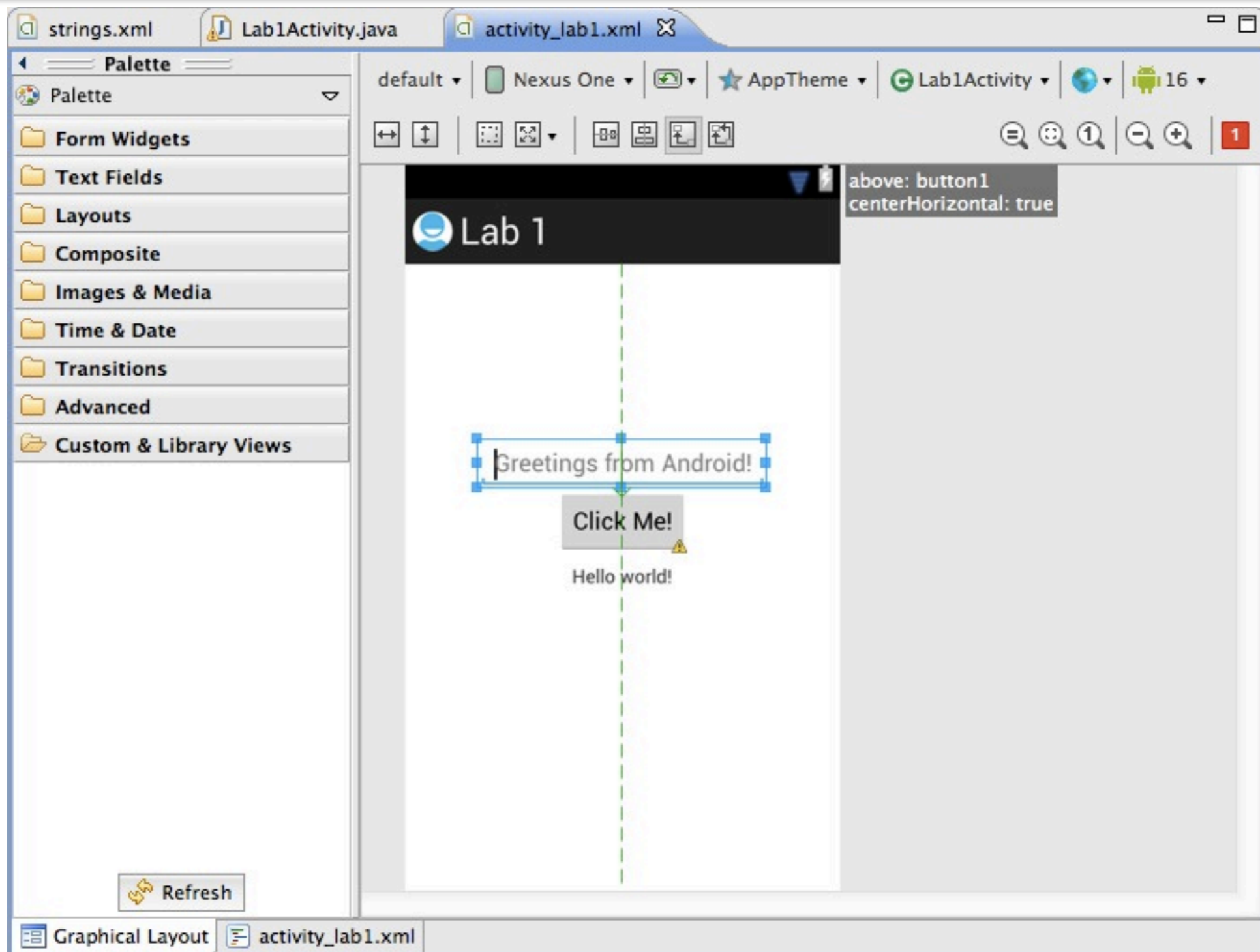
```
public void onClickMe(View button) {
    /* get TextView @+id/hello_world */
    TextView hello_world =
        (TextView) findViewById(R.id.hello_world);

    /* set its text to the string resource @string/greetings */
    hello_world.setText(getString(R.string.greetings));
}
```

# A Simple Button: Demo

# Capturing User Input: Adding an EditText

# Adding an EditText





# Adding an EditText: Code View

---

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/button1"
    android:layout_centerHorizontal="true"
    android:ems="10"
    android:hint="@string/greetings_from_android"
    android:inputType="textNoSuggestions" >

    <requestFocus />
</EditText>
```

*[“placeholder” text]*  
*[workaround for bug in SDK]*

# Aside:

# EditText inputTypes

---

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/button1"
    android:layout_centerHorizontal="true"
    android:ems="10"
    android:hint="@string/greetings_from_android"
    android:inputType="textNoSuggestions" >

    <requestFocus />
</EditText>
```



# Aside:

## EditText inputTypes

---

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/button1"
    android:layout_centerHorizontal="true"
    android:ems="10"
    android:hint="@string/greetings_from_android"
    android:inputType="textNoSuggestions|textEmailAddress" >

    <requestFocus />
</EditText>
```



# Aside:

## EditText inputTypes

---

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/button1"
    android:layout_centerHorizontal="true"
    android:ems="10"
    android:hint="@string/greetings_from_android"
    android:inputType="textNoSuggestions|phone" >

    <requestFocus />
</EditText>
```



# Aside:

# EditText inputTypes

---

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/button1"
    android:layout_centerHorizontal="true"
    android:ems="10"
    android:hint="@string/greetings_from_android"
    android:inputType="textNoSuggestions|textPassword" >

    <requestFocus />
</EditText>
```



# The Code:

## onClickMe()

---

```
public void onClickMe(View button) {
    TextView hello_world =
        (TextView) findViewById(R.id.hello_world);
    EditText text_box =
        (EditText) findViewById(R.id.text_box);

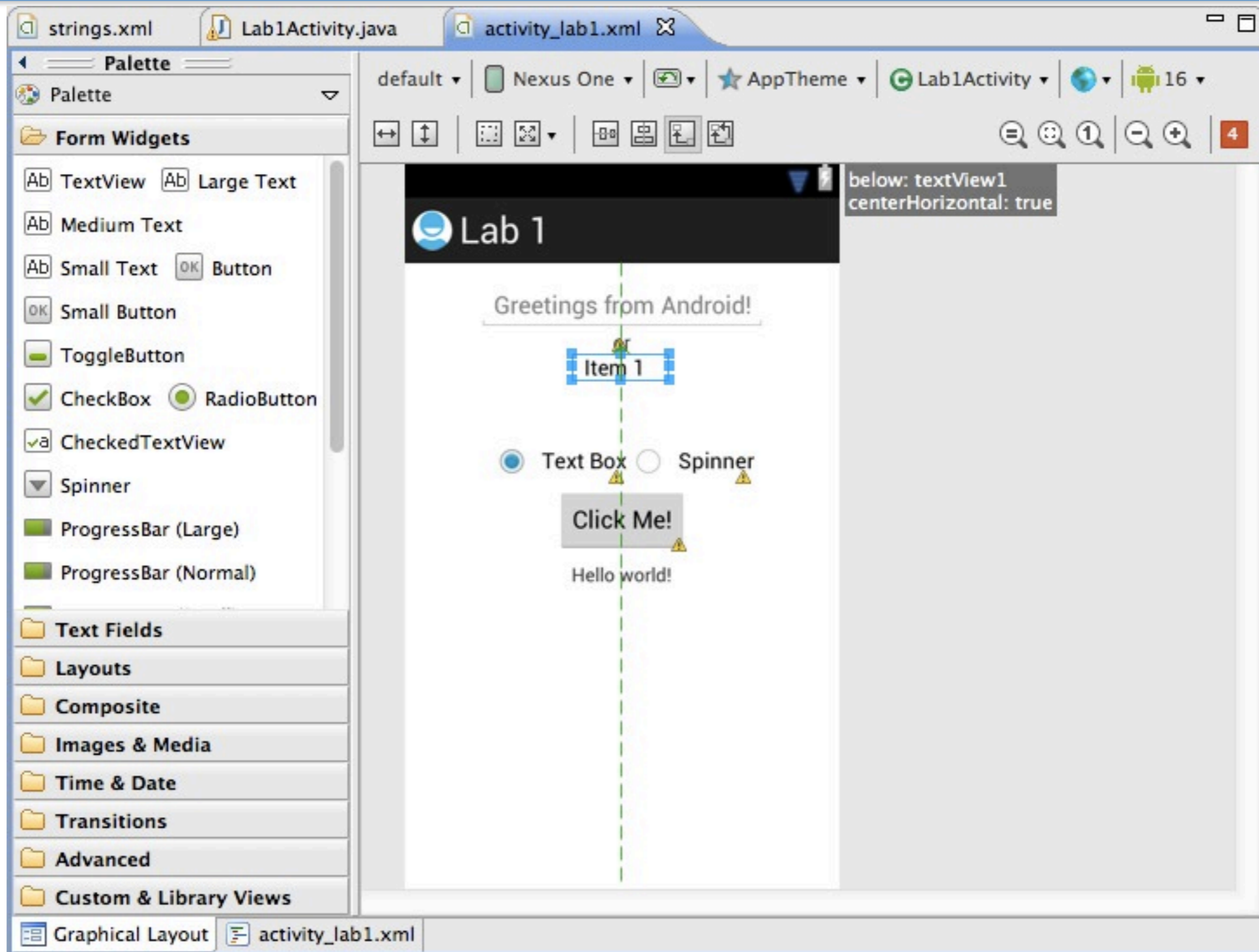
    if (text_box.getText().length() == 0) {
        hello_world.setText(getString(R.string.greetings));
    } else {
        /* set the text to the text in text_box if not empty */
        hello_world.setText(text_box.getText());
    }
}
```

# Adding an EditText: Demo

# Making choices: Radio Buttons and Spinners



# Radio Buttons and Spinners



# Radio Buttons and Spinners

## Code View

---

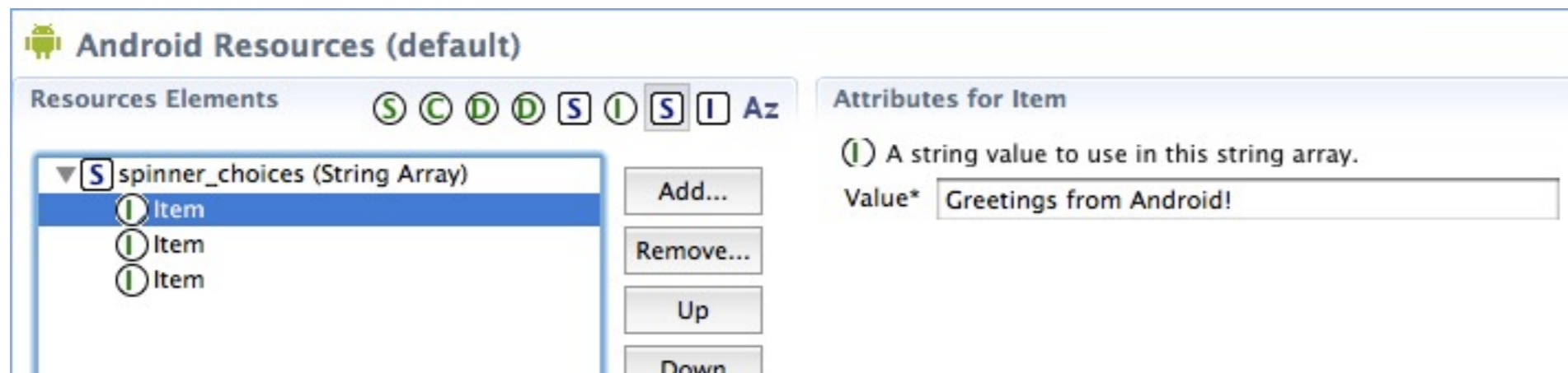
```
<RadioGroup android:id="@+id/chooseSource"  
    android:orientation="horizontal" >  
    <RadioButton android:id="@+id/radio0"  
        android:checked="true"  
        android:text="Text Box" />  
    <RadioButton android:id="@+id/radio1"  
        android:text="Spinner" />  
</RadioGroup>
```

*[This button is checked]*

```
<Spinner android:id="@+id/spinner"  
    android:entries="@array/spinner_choices" /> [The choices in the spinner]
```

# Radio Buttons and Spinners

## Resources



[or]

```
<resources>
```

```
<string name="app_name">Lab 1</string>
<string name="hello_world">Hello world!</string>
<string name="menu_settings">Settings</string>
<string name="title_activity_lab1">Lab 1</string>
<string name="greetings_from_android">Greetings from Android!</string>
<string-array name="spinner_choices">
    <item >Greetings from Android!</item>
    <item >Les salutations des Android!</item>
    <item >iSaludos desde Android!</item>
</string-array>
```

```
</resources>
```

# The Code:

## onClickMe()

---

```
public void onClickMe(View button) {
    RadioButton radio_text_box =
        (RadioButton) findViewById(R.id.radio0);

    if (radio_text_box.isChecked()) {
        /* as described earlier ... */
    } else {
        Spinner spinner = (Spinner) findViewById(R.id.spinner);
        /* set the text to the text in the spinner */
        hello_world.setText(
            spinner.getItemAtPosition(
                spinner.getSelectedItemPosition()
            ).toString());
    }
}
```

# Radio Buttons and Spinners Demo

# Lab I

[https://bitbucket.org/aiti\\_india/labI](https://bitbucket.org/aiti_india/labI)

# Resources

---

- Android Developer Website “User Interface”  
<<http://developer.android.com/guide/topics/ui/index.html>>, especially:
  - “Layouts”  
<<http://developer.android.com/guide/topics/ui/declaring-layout.html>>
  - “Input Controls”  
<<http://developer.android.com/guide/topics/ui/controls.html>>
  - “Input Events”  
<<http://developer.android.com/guide/topics/ui/ui-events.html>>
- “Android User Interface Design” on MobileTuts  
<<http://mobile.tutsplus.com/series/android-user-interface-design/>>