# Event handling in Android


DT228/3

Dr. Susan McKeever

# Event-driven programming

- Developing a GUI uses event driven programming..
  - Flow of program is driven by events
    - In this case, user actions
  - Typically :
    - an “event” triggers .. E.g. button click
    - a “callback” method/ event handler
      e.g. onClick()

# Event-driven programming

**Android (and Java Swing) – 3 ways**

1. Declare/implement "listeners" in the activity, with event handler methods – "LONG"!

   OR

2. Use anonymous classes for each widget " MED"!

   OR

3. Embed event handler method name in the XML (ANDROID only) - "EASY!"

**Be aware of tradeoffs**

# First way… Listeners implemented in the class

```
public class MyPass extends Activity
        implements View.OnClickListener {

        @Override
        public void onCreate(Bundle icicle) {

                //  other onCreate code
                use Button class "setOnClickListener"
                 to assign the button listener;

        }


        public void onClick(View view) {
                        // do whatever you want as a result of the
                        // the button click;
        }

}
```

1. Implement the "listener" needed

2. Assigning that listener to the widget that takes the user action (e.g. button)

3. Implementing the "behaviour" we wanted when the user action was taken e.g. button clicked

# First way… Listeners implemented in the class – more than one button?

```
public class MainActivity extends Activity
                    implements View.OnClickListener {

        Button btn;
        @Override
        public void onCreate(Bundle icicle) {

                //  other onCreate code

        btn = (Button)findViewbyId(R.id.whateverbuttoniscalled)
        btn.setOnClickListener(this);

        }


        public void onClick(View view) {
                    // do whatever you want as a result of the
                    // the button click;
        }

}
```

**What code needs to change?**

**The onClick() Method Will be triggered when any button is clicked so first thing in the method is to figure out which button clicked it… The View object getID() method to check which button was clicked..**

# Second way … Use anonymous listener classes for each widget

```
public class MainActivity extends Activity {
    private Button button;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        button = (Button) findViewById(R.id.buttonToast);

        button.setOnClickListener(// anonymous listener
    class goes here://  new OnClickListener()
                {
                  public void onClick(View v)
                   {
                        // whatever you want to happen when
                        // button is clicked
                   }
                });
    }
```

# Third way… Embed event handler method into XML

```
<?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/buttonToast"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content“
        android:onClick=“showToast”
        android:text="Show Toast" />

</LinearLayout>
```

# Third way... Embed callback method into XML

```
public class MainActivity extends Activity {
    private Button button;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        button = (Button) findViewById(R.id.buttonToast);

        public void showToast(View v)
        {
             Toast.makeText(getApplicationContext(),
                                  "Button is clicked",
              Toast.LENGTH_LONG).show();
        }
    }
}
```

# Implementing event Programming –best way?

- **1st way:** Implement listeners as interfaces in the activity, with
Shared event handlers
  - Dynamic code is in one place +
  - Shared listener across widgets = shared callback -
  - Longer to code - OR
- **2nd way:** Anonymous class for each widget
  - *Can't reuse – separate class per widget* -
  - potential performance hit -
  - Easier to follow code? +

# Implementing listeners +s/ -s

- **3rd way :** Put callback method as an "onclick" attribute value in the XML

Simplest to implement +

But

1. Presentation coupled with logic- bad –
2. Changes to method name - > need to remember to refactor the xml –
3. Multiple XML files using a single method can lead to maintenance problems if functionality diverges -

# Examples of other widgets..using <EditText> in an activity

- Implement ?
- **Respond to text changes?**

# How do we know which listener to implement?

e.g.
Could have a **checkbox** OR

A **button** OR

An **input field**.. etc

Is it the same listener for all widgets?

Unfortunately.. NO.. Need to use the right one, for the right Widget, and the right user action (clicking, hovering with the mouse etc)