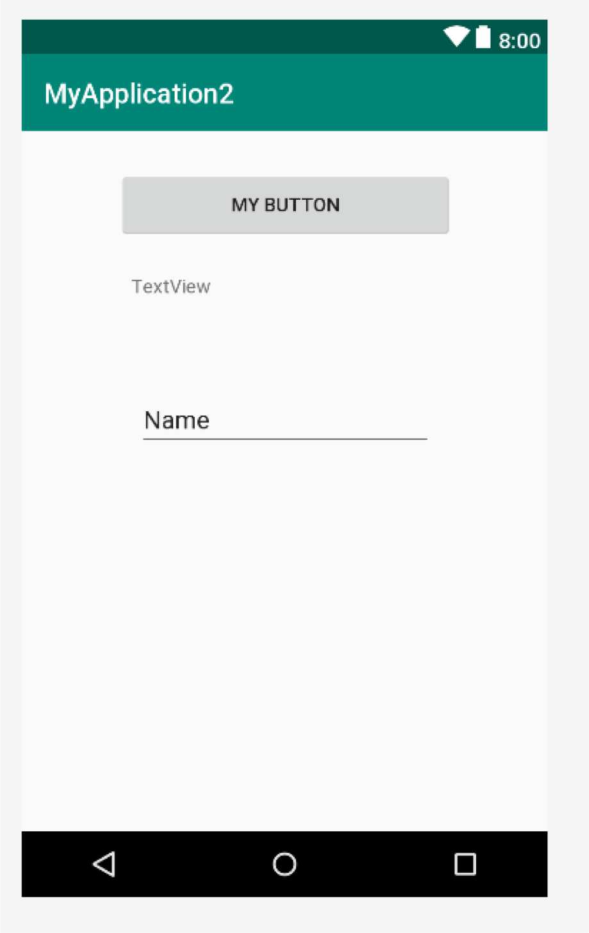
4. The First App

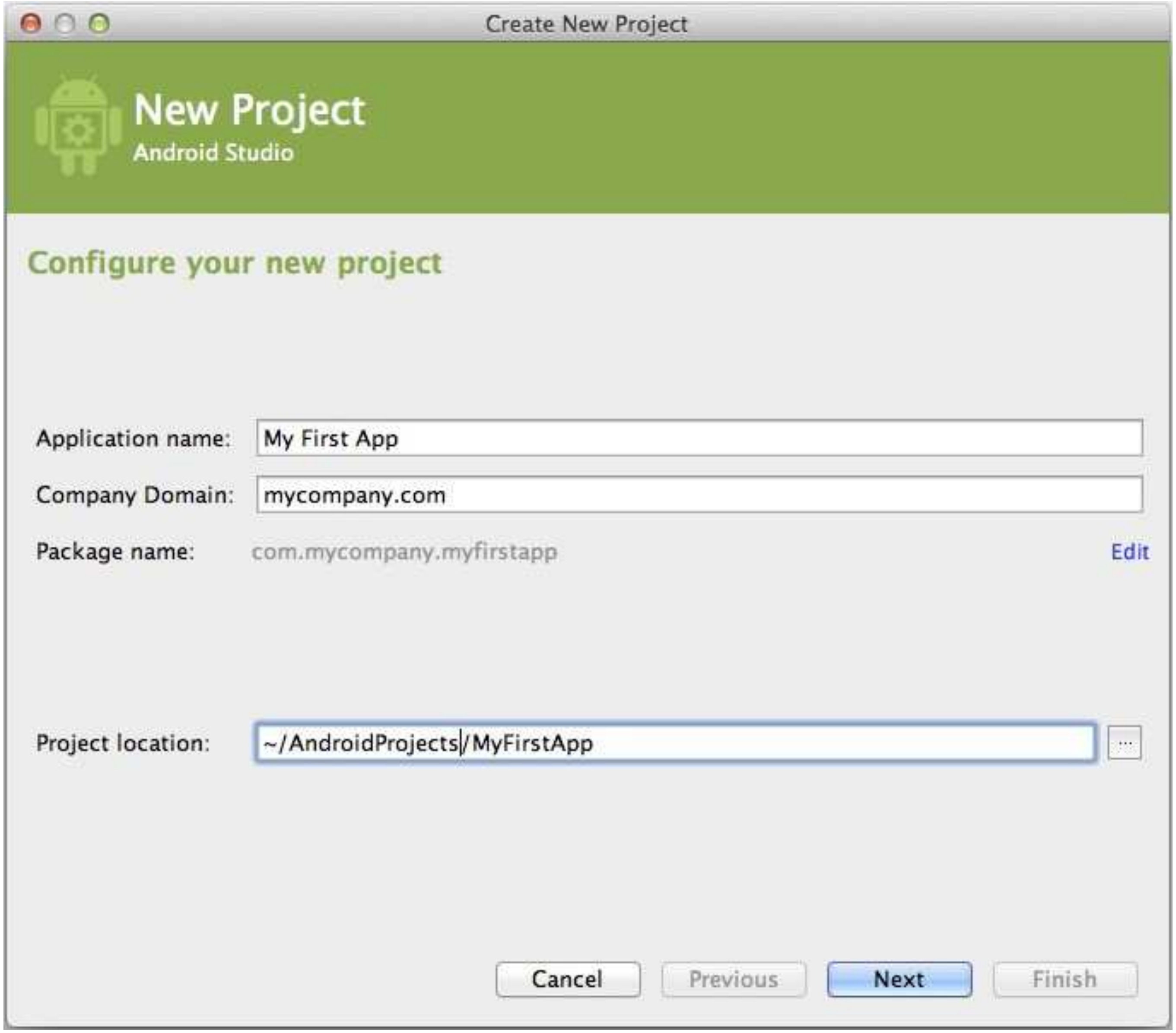
Top-down design

- Let's start from a design of an app that we want to create and then learn the necessary skills to build that app.

- First application
  - user is shown TextView, EditText and Button
  - Enters text in EditText and clicks Button
  - Text appears in TextView



Creating a new project



Configure your new project

Application name: My First App

Company Domain: mycompany.com

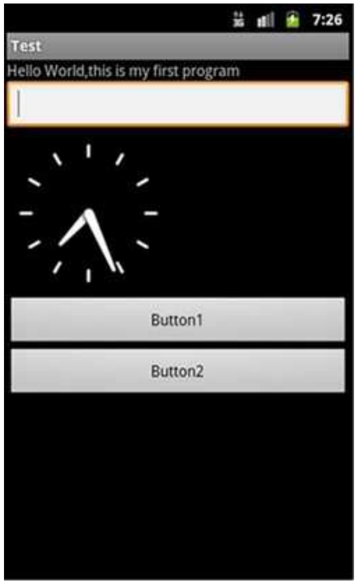
Package name: com.mycompany.myfirstapp

Project location: ~/AndroidProjects/MyFirstApp

Android terminology

- **activity**: a single screen of UI that appears in your app
  - the fundamental units of GUI in an Android app

- **view**: items that appear onscreen in an activity
  - **widget**: GUI control such as a button or text field
  - **layout**: invisible container that manages positions/sizes of widgets



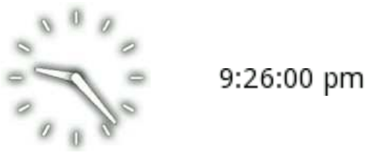
Test

Hello World,this is my first program

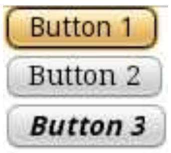
- **event**: action that occurs when user interacts with widgets
  - e.g. clicks, typing, scrolling

- **action bar**: a menu of common actions at top of app
- **notification area**: topmost system menu and icons

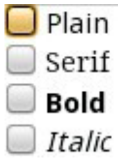
Android widgets



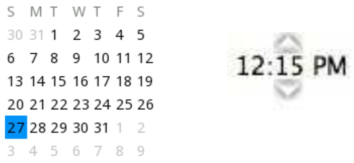
Analog/DigitalClock



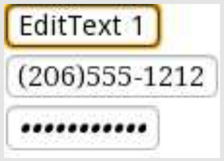
Button



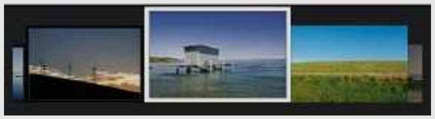
Checkbox



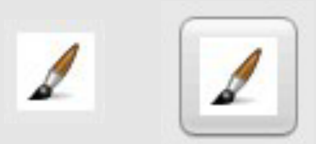
Date/TimePicker



EditText

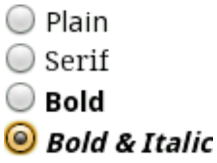


Gallery

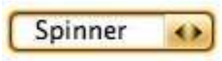


ImageView/Button

ProgressBar

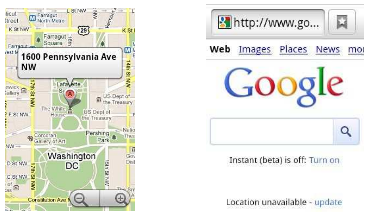


RadioButton



Spinner

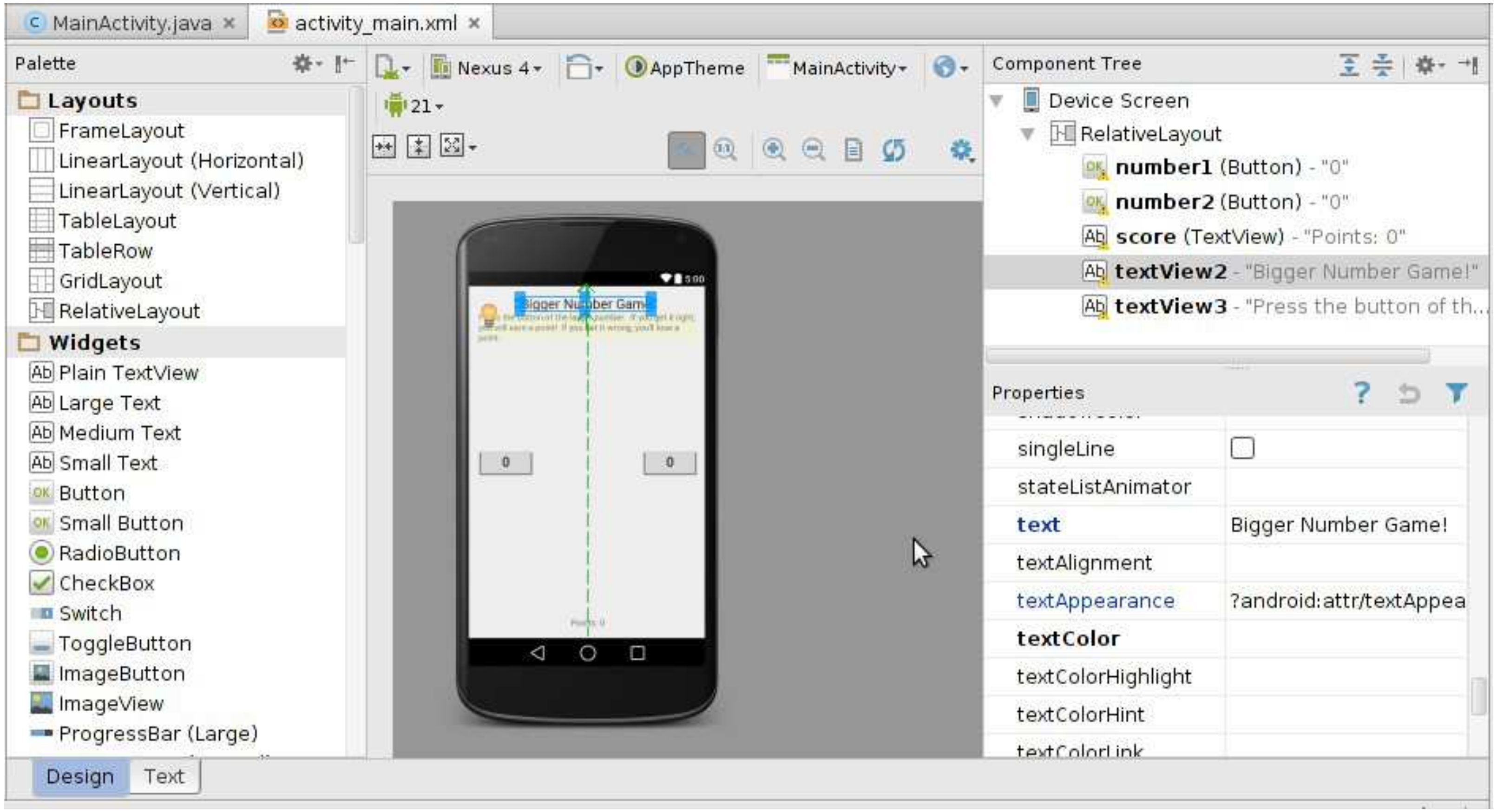
TextView



MapView, WebView

Designing a user interface

- open XML file for your layout (e.g. `activity_main.xml`)
- drag widgets from left **Palette** to the preview image
- set their properties in lower-right **Properties** panel

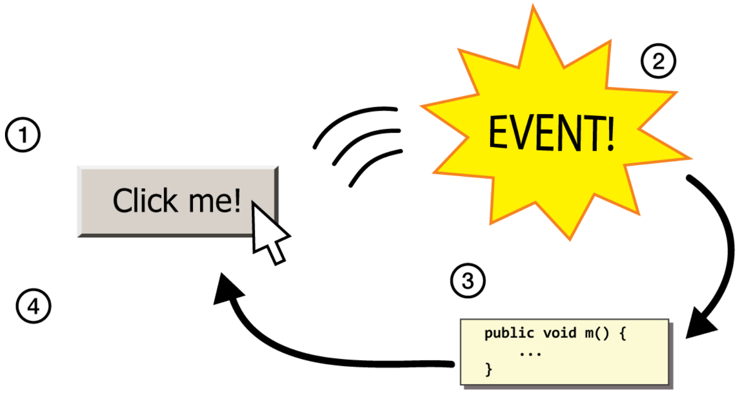


Device Screen
RelativeLayout
number1 (Button) - "0"
number2 (Button) - "0"
score (TextView) - "Points: 0"
textView2 - "Bigger Number Game!"
textView3 - "Press the button of th...

singleLine	☐
stateListAnimator	
text	Bigger Number Game!
textAlignment	
textAppearance	?android:attr/textAppea
textColor	
textColorHighlight	
textColorHint	
textColorLink	

Events

- **event**: An external stimulus your program can respond to.
- Common kinds of events include:
  - Mouse motion / tapping, Keys pressed,
  - Timers expiring, Network data available



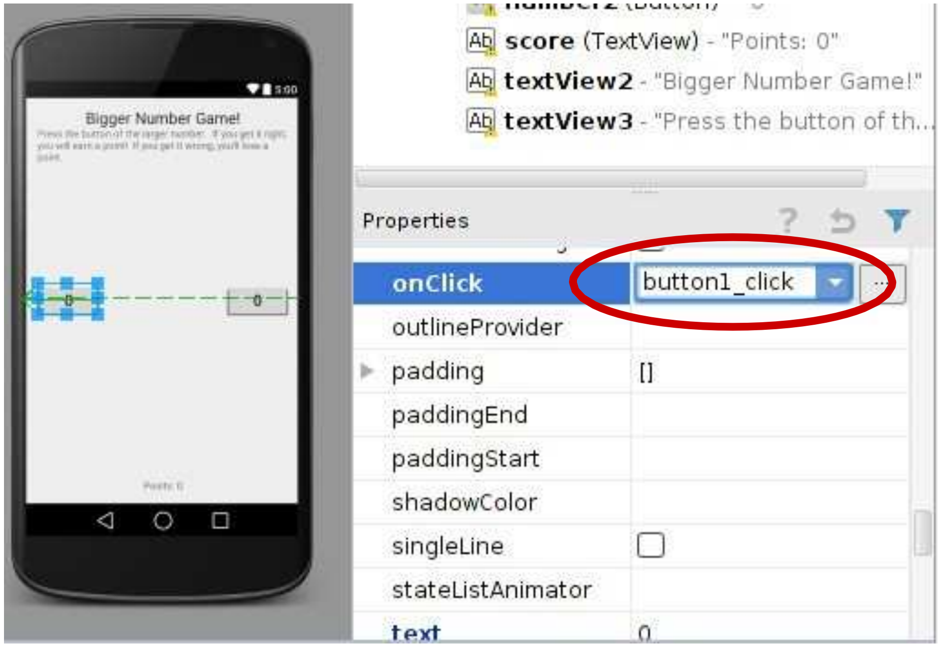
```
public void m() {
    ...
}
```

- **event-driven programming**: Overall execution of your program is largely dictated by user events.
  - Commonly used in graphical programs.

- To respond to events in a program, you must:
  - Write methods to handle each kind of event ("listener" methods).
  - Attach those methods to particular GUI widgets.

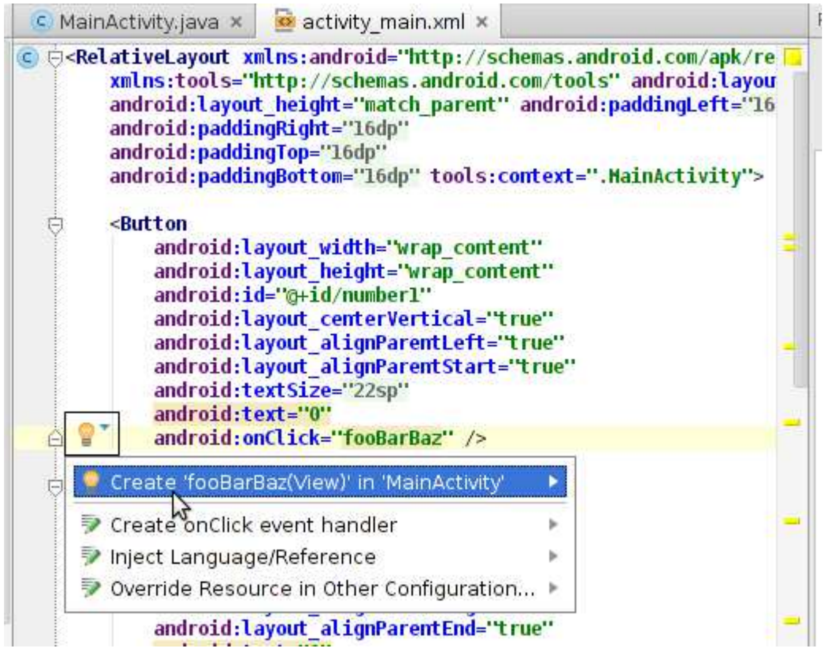
Setting an event listener

- select the widget in the **Design** view
- scroll down its **Properties** until you find `onClick`
- type the name of a method you'll write to handle the click
- switch to the **Text view** and find the XML for that button
- click the "Light Bulb" and choose to "**Create**" the method



score (TextView) - "Points: 0"
textView2 - "Bigger Number Game!"
textView3 - "Press the button of th...

onClick	button1_click
outlineProvider	
padding	[]
paddingEnd	
paddingStart	
shadowColor	
singleLine	☐
stateListAnimator	
text	0



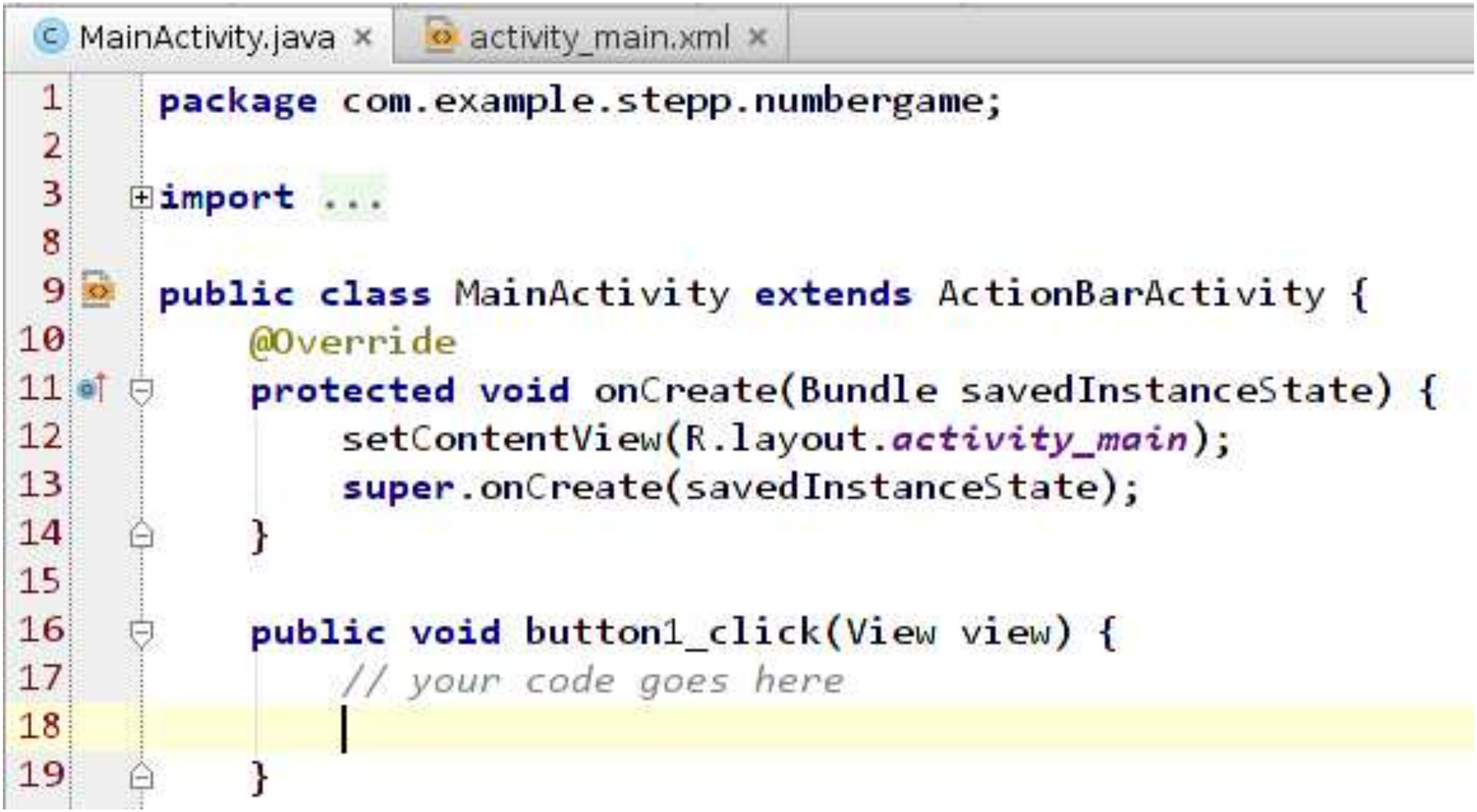
MainActivity.java
activity_main.xml
Create 'fooBarBaz(View)' in 'MainActivity'
Create onClick event handler
Inject Language/Reference
Override Resource in Other Configuration...

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/re
    xmlns:tools="http://schemas.android.com/tools" android:layou
    android:layout_height="match_parent" android:paddingLeft="16
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/number1"
        android:layout_centerVertical="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:textSize="22sp"
        android:text="0"
        android:onClick="fooBarBaz" />

        android:layout_alignParentEnd="true"
```

Event listener Java code



```
package com.example.stepp.numbergame;

import ...

public class MainActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setContentView(R.layout.activity_main);
        super.onCreate(savedInstanceState);
    }

    public void button1_click(View view) {
        // your code goes here

    }
```

View objects

- each widget has an associated Java object you can access
- they are subclasses of parent class **`View`**
  - examples: `Button`, `TextView`, `EditText`, ...

- `View` objects have many `get` and `set` methods that correspond to the properties in the Design view:
  - background, bottom, ID, left, margin, padding, right, text, textAlignment, textSize, top, typeface, visibility, x, y, z, ...
  - example: for a Button's **text** property, there will be methods:

```
public String getText()
public void setText(String text)
```

  - Find list of properties in Design view, or typing ".get" on a button in Java code, or at: https://developer.android.com/reference/

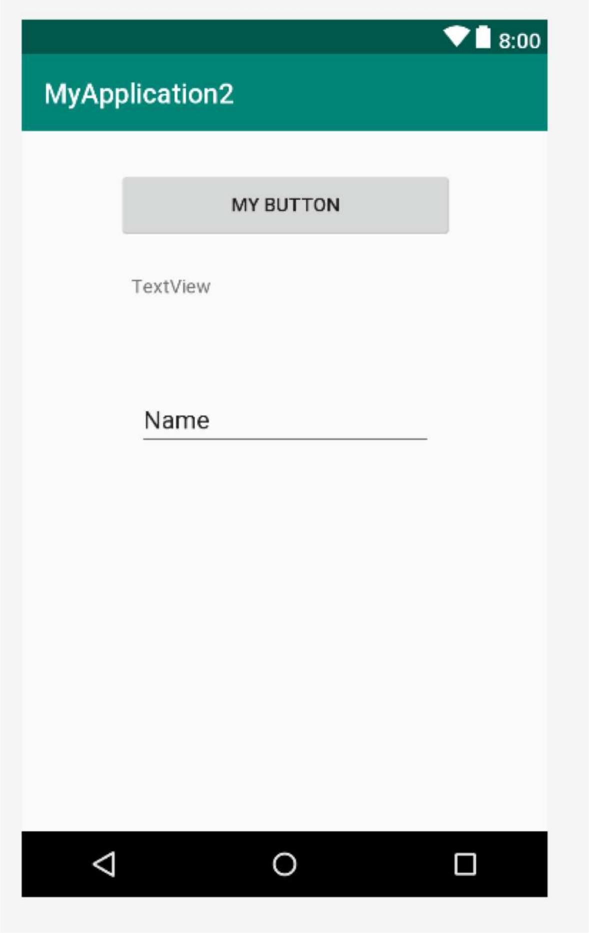
Interacting with widgets

- accessing a widget in the Java code:
  1. in Design view, give that view a unique **ID** property value
  2. in Java code, call `findViewById` to access its View object
     - pass it a parameter of `R.id.`***`your_unique_ID`***
     - cast the returned value to the appropriate type (`Button`, `TextView`, etc.)

```
public void button1_onclick(View view) {
    TextView tv = (TextView) findViewById(R.id.mytextview);
    tv.setText("You clicked it!");
}
```

Exercise: Number game

- New let's build that "First Application”, Recall:
  - user is shown TextView, EditText and Button
  - Enters text in EditText and clicks
  - Button
  - Text appears in TextView

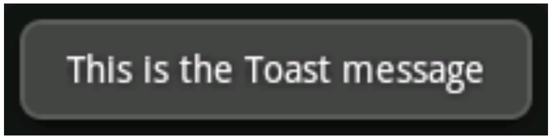


Displaying Toasts

```
Toast.makeText(this,
                "message",
                duration).show();
```

- where ***duration*** is `Toast.LENGTH_SHORT` or `LENGTH_LONG`

- A "Toast" is a pop-up message that appears for a short time.
- Useful for displaying short updates in response to events.
- Should not be relied upon extensively for important info.



This is the Toast message