Mobile Application Development

Produced by

David Drohan (ddrohan@wit.ie)

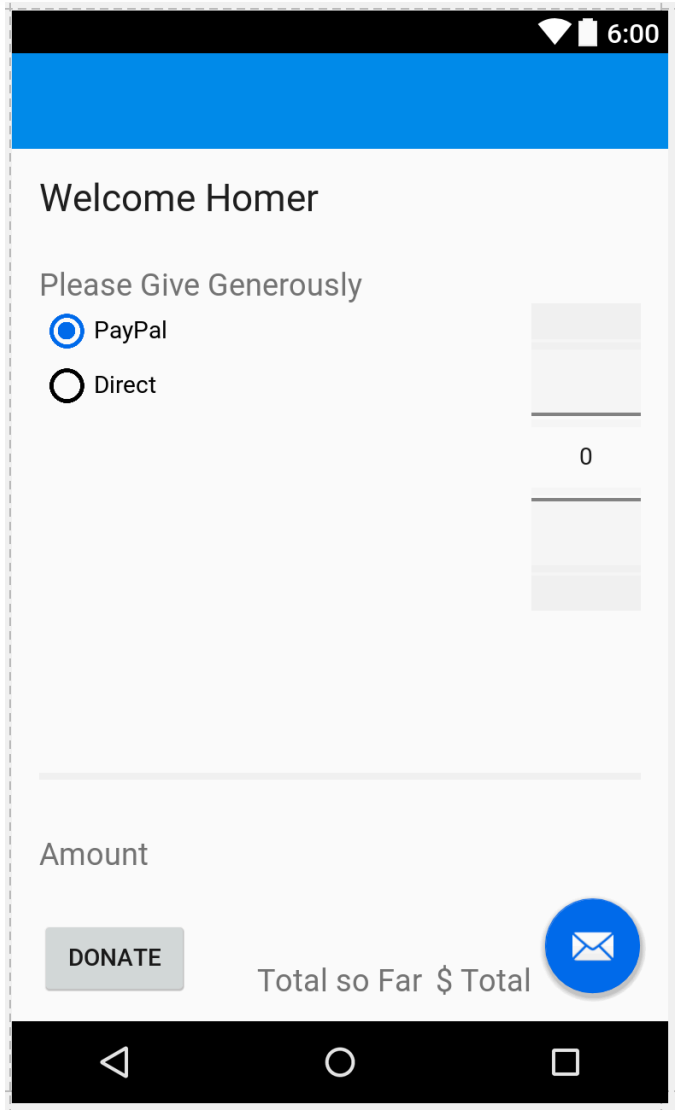
Department of Computing & Mathematics
Waterford Institute of Technology
http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

A First Android Application

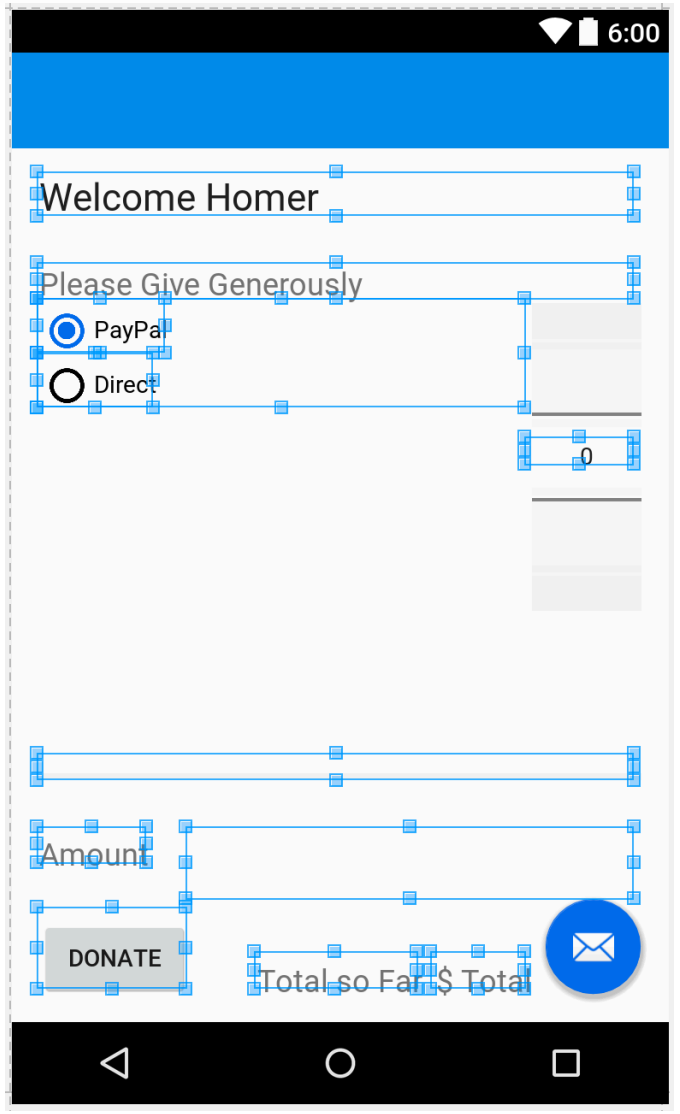
App Basics

- Create a new project (app) called “Donation”
- The *Donation* application will consist of an **activity** and a **layout**:
- An activity is an instance of Activity, a class in the Android SDK.
- An activity is responsible for managing user interaction with a screen of information – the ‘*Controller*’ (in MVC).
- You write subclasses of Activity to implement the functionality that your app requires.
- A simple application may need only one subclass; a complex application can have many.



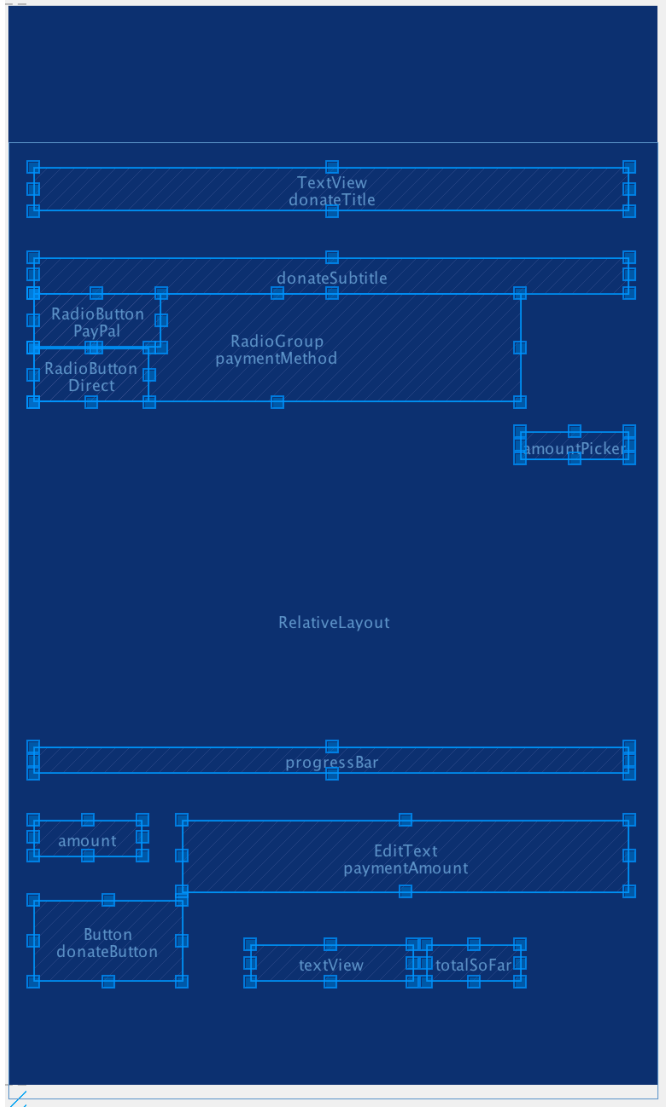
App Basics *

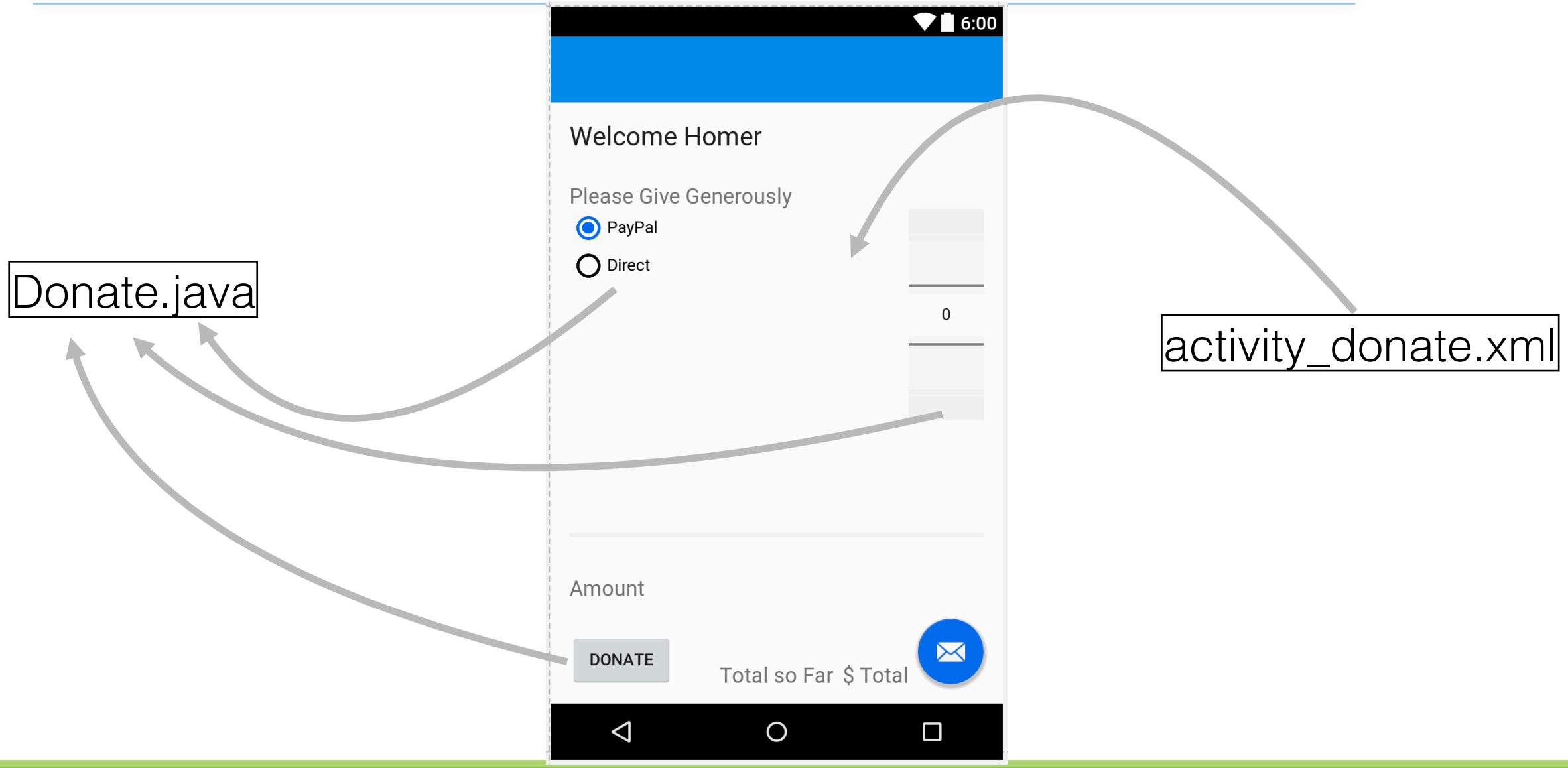
- Donation is a simple app, so it will have a single Activity subclass named Donate.
- Donate will manage (*Control*) the user interface shown
- A **layout** defines a set of user interface objects and their position on the screen – the '*View*' (in MVC)
- A **layout** is made up of definitions written in **XML**. Each definition is used to create an object (a widget) that appears onscreen, like a button, some text or a rating bar etc.
- Donation will include a **layout file** named activity_donate.xml. The XML in this file will define the UI as shown.



App Basics

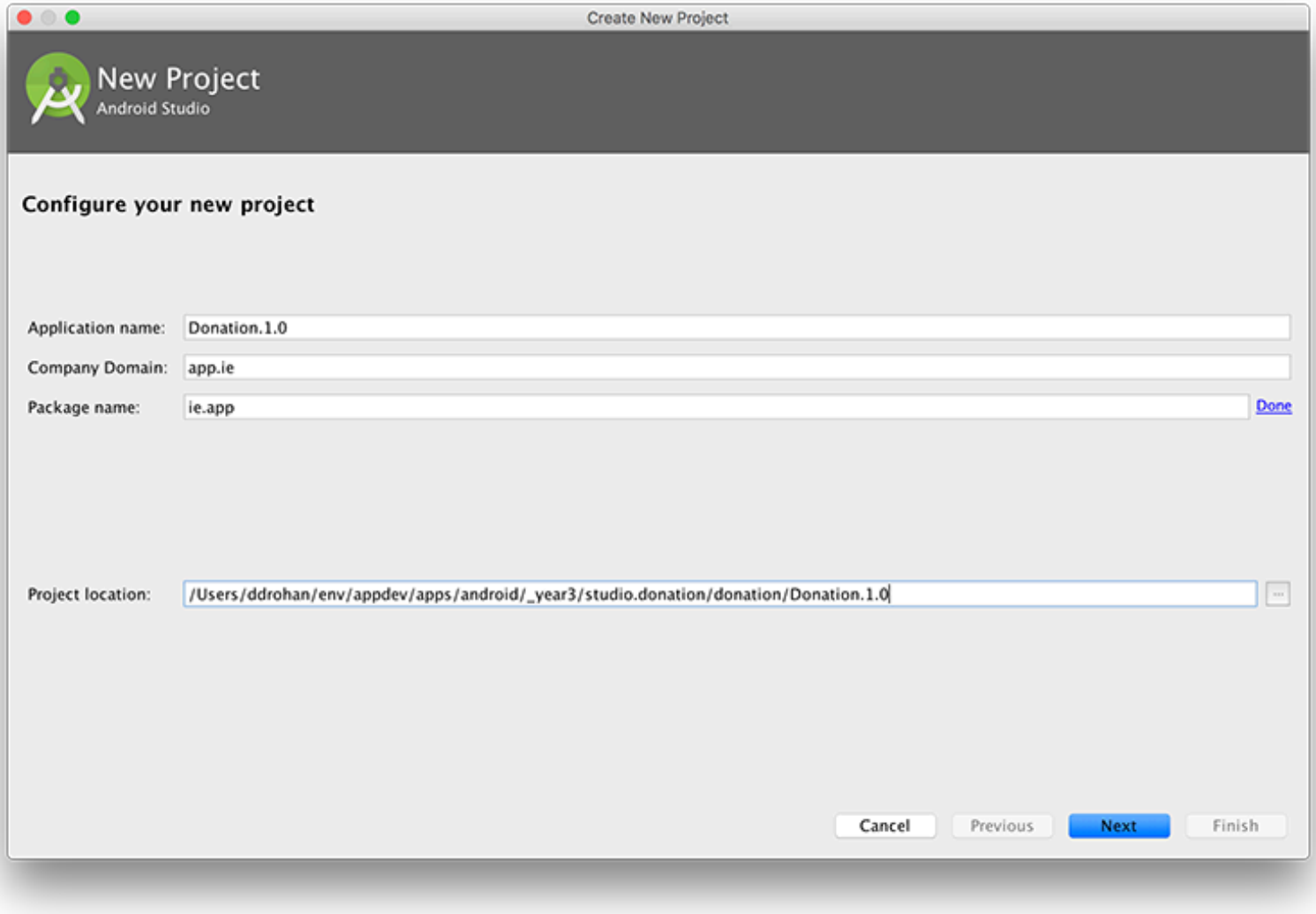
- Donation is a simple app, so it will have a single Activity subclass named Donate.
- Donate will manage (*Control*) the user interface shown
- A **layout** defines a set of user interface objects and their position on the screen – the '*View*' (in MVC)
- A **layout** is made up of definitions written in **XML**. Each definition is used to create an object (a widget) that appears onscreen, like a button, some text or a rating bar etc.
- Donation will include a **layout file** named activity_donate.xml. The XML in this file will define the UI as shown.




6:00
Welcome Homer
Please Give Generously
PayPal
Direct
0
Amount
DONATE
Total so Far $ Total
Donate.java
activity_donate.xml

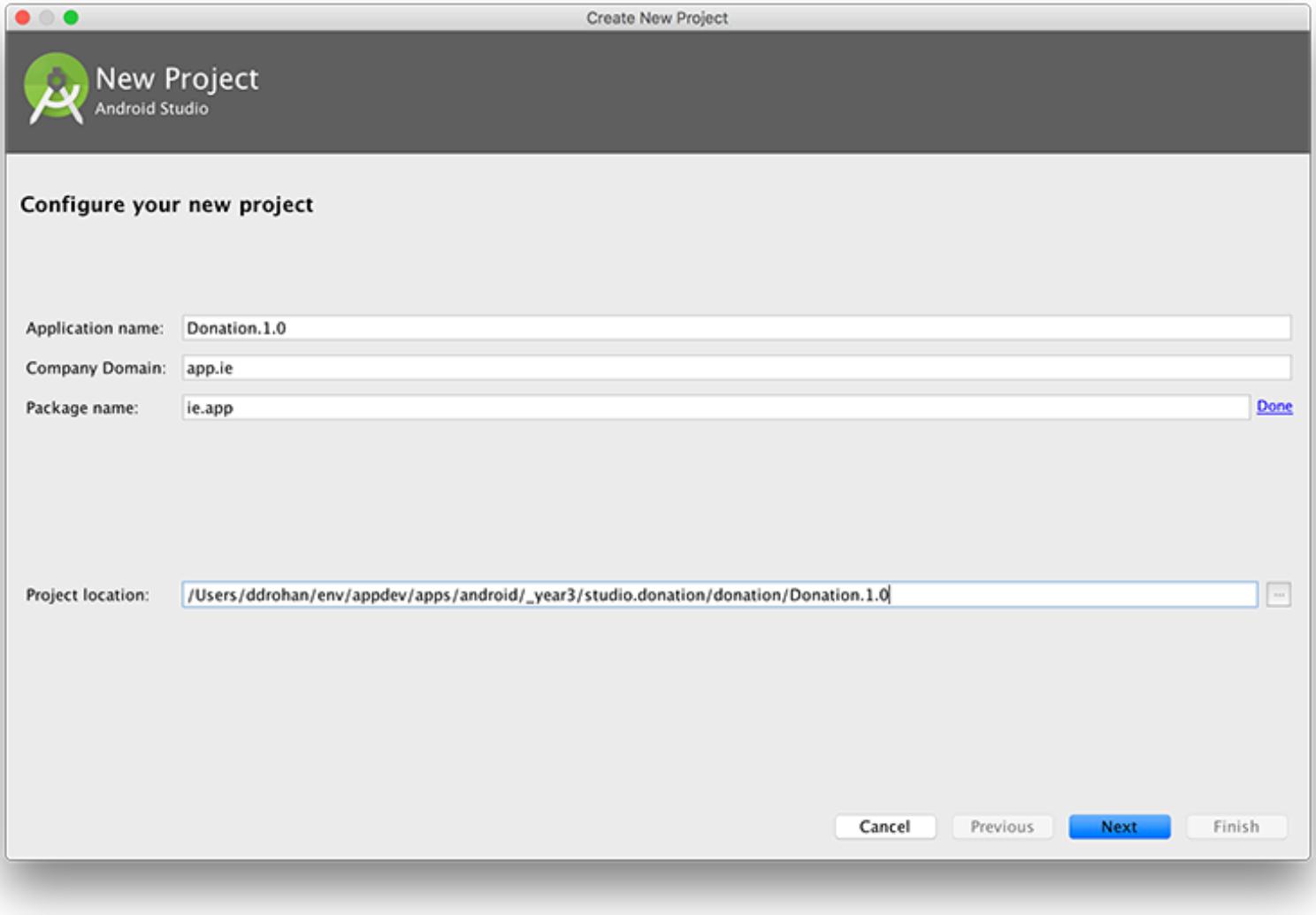
Creating an Android Project

- The first step is to create an Android project. An Android project contains the files that make up an application.
- To create a new project, open Android Studio and choose File -> New —>‘ New Project’



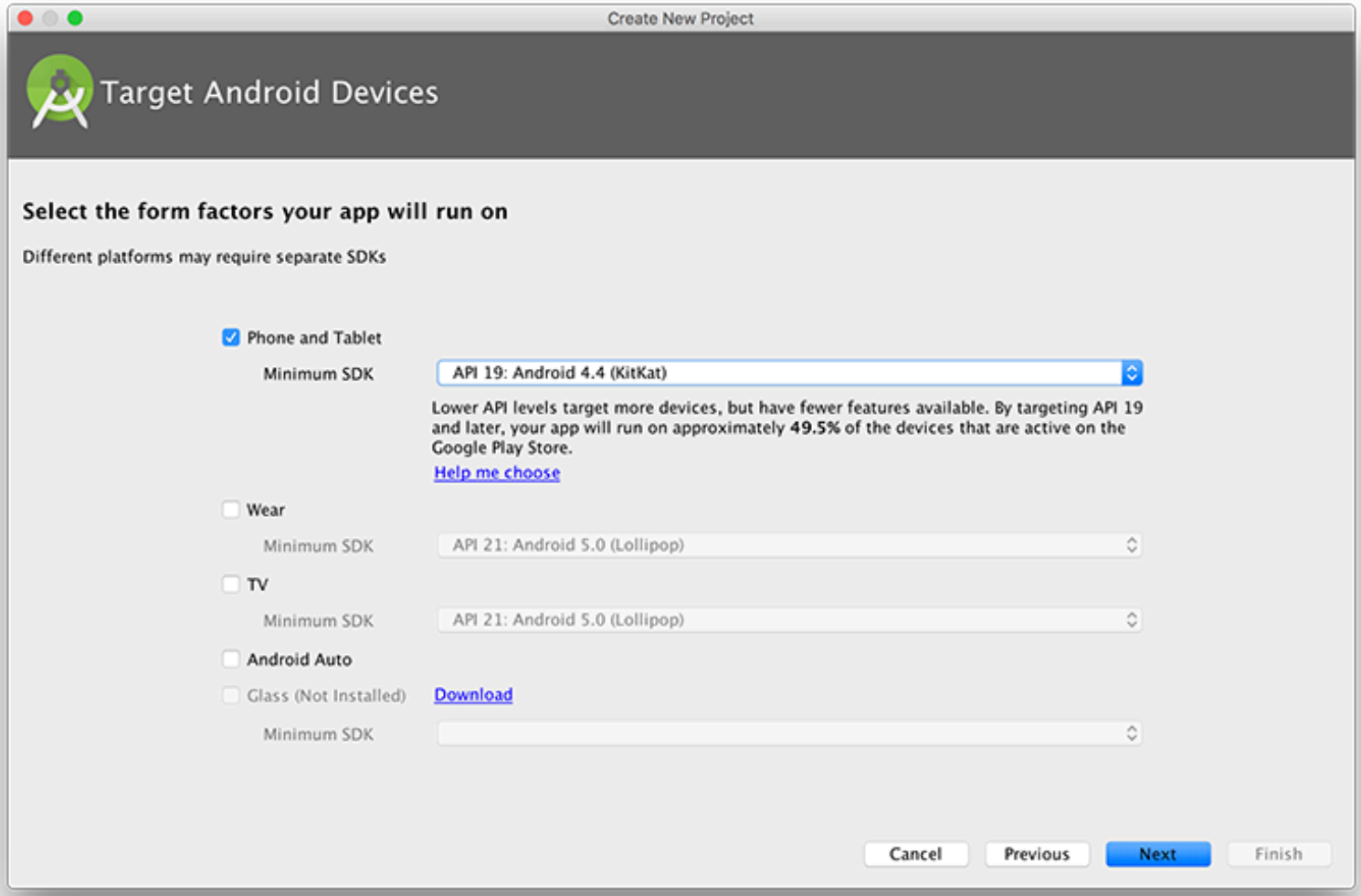
Creating an Android Project

- In the first dialog, enter the application name (Donation.1.0 here). The project name will automatically update to match the application's.
- For the package name, automaticaly entered is *com.example.XYZ,* but that's only a placeholder and it's recommended you change it. Notice that the package name entered uses a "reverse DNS" convention in which the domain name of your organization is reversed and suffixed with further identifiers.
- **This convention keeps package names unique and distinguishes applications from each other on a device and on the Google Play Store.**
- The last field is the project location, which you can change if you wish



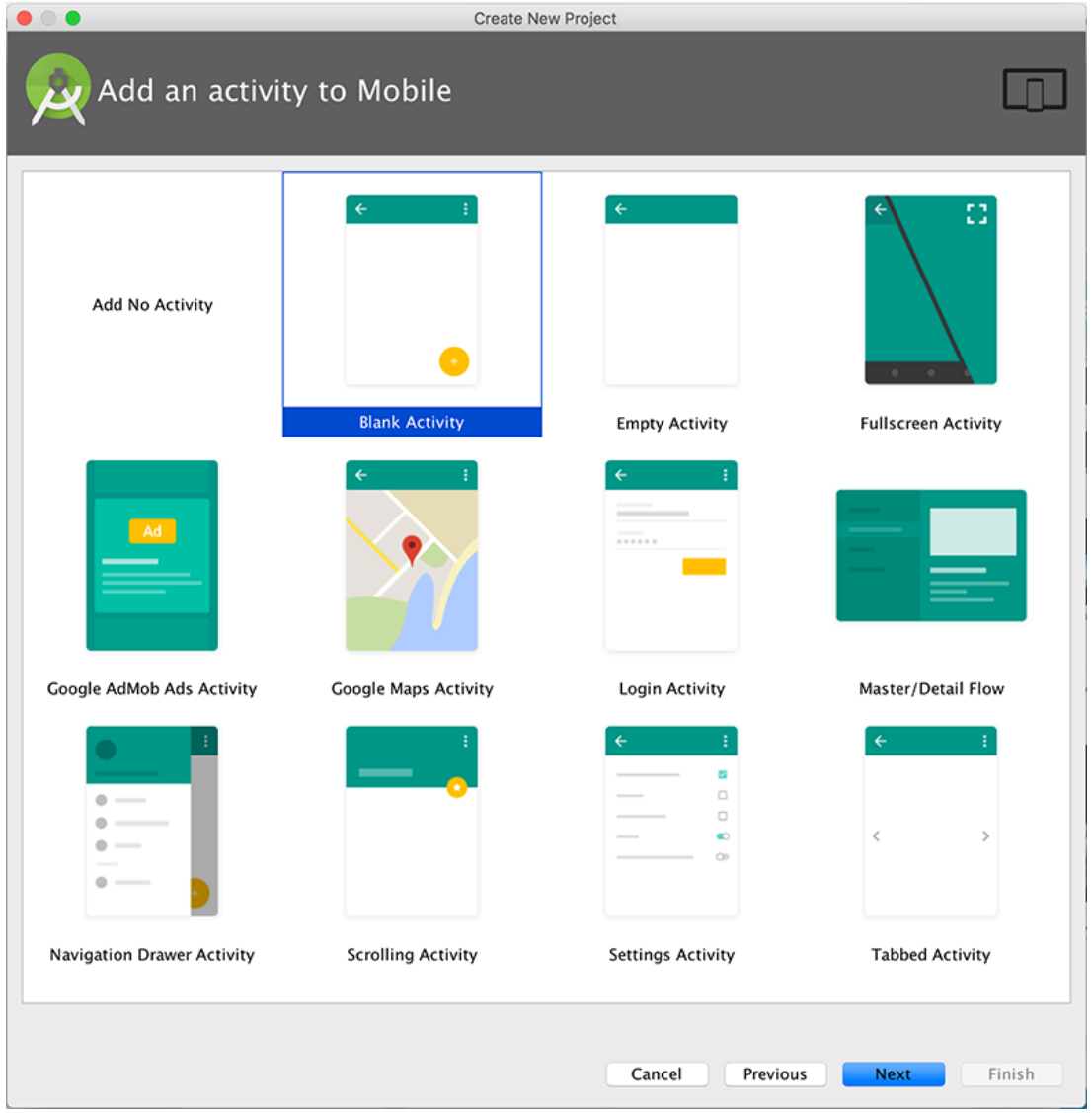
Creating an Android Project

- This step allows you to configure your application to work with different versions of Android and different platforms.
- Take note of the ‘Minimum SDK’ level as the higher it is, the smaller number of devices your app will target.
- For our purposes, ‘Phone and Tablet’ is more than enough, but you can develop for Wearable devices, TV and Auto.



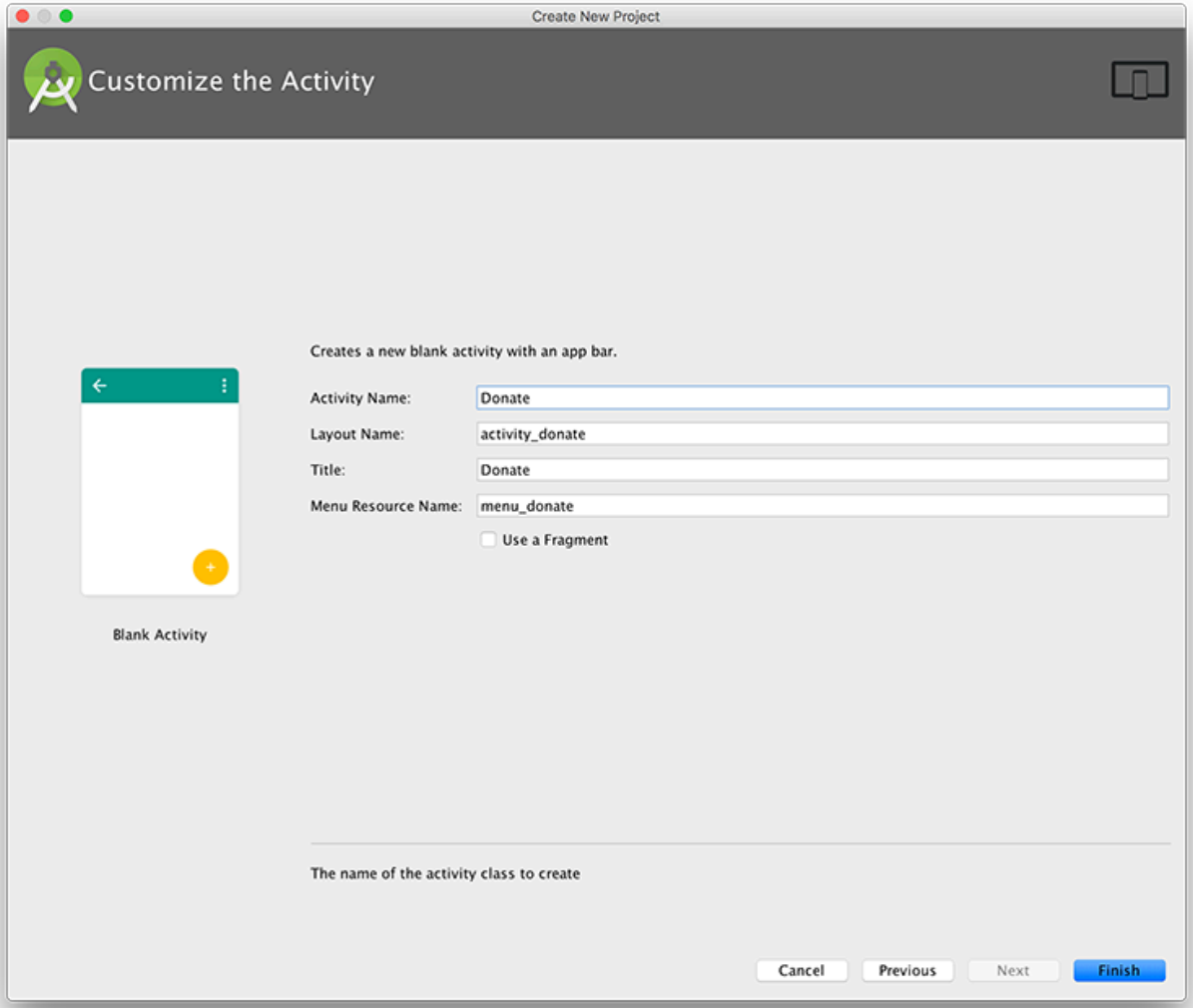
Creating an Android Project

- Here you choose the type of layout you want for your app.
- There are numerous layout ‘templates’ to choose from, including Maps, Tabs, Navigation Drawers etc.
- Be aware that the more complex layout you choose, the more ‘boilerplate’ code is supplied in the ‘startup’ app – this can be quite confusing for a novice Android developer.
- We’ll choose a ‘Blank Activity’ here, to keeps things relatively simple ☺



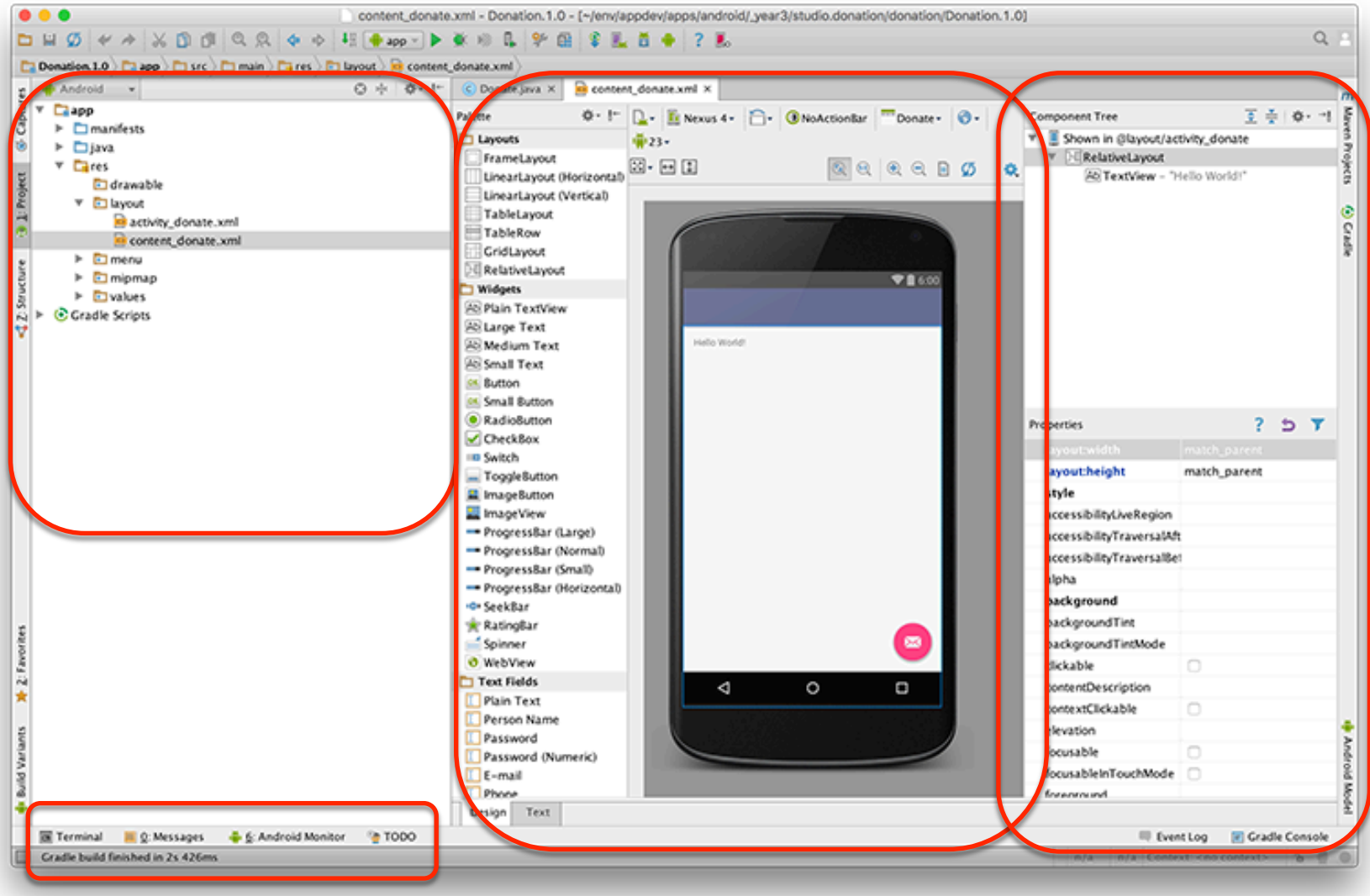
Creating an Android Project

- In the final dialog of this wizard, name the activity subclass **Donate**
- The layout name will automatically update to **activity_donate** to reflect the activity's new name.
- The layout name reverses the order of the activity name, is all lowercase, and has underscores between words.
- This naming style is recommended for layouts as well as other resources that you will learn.



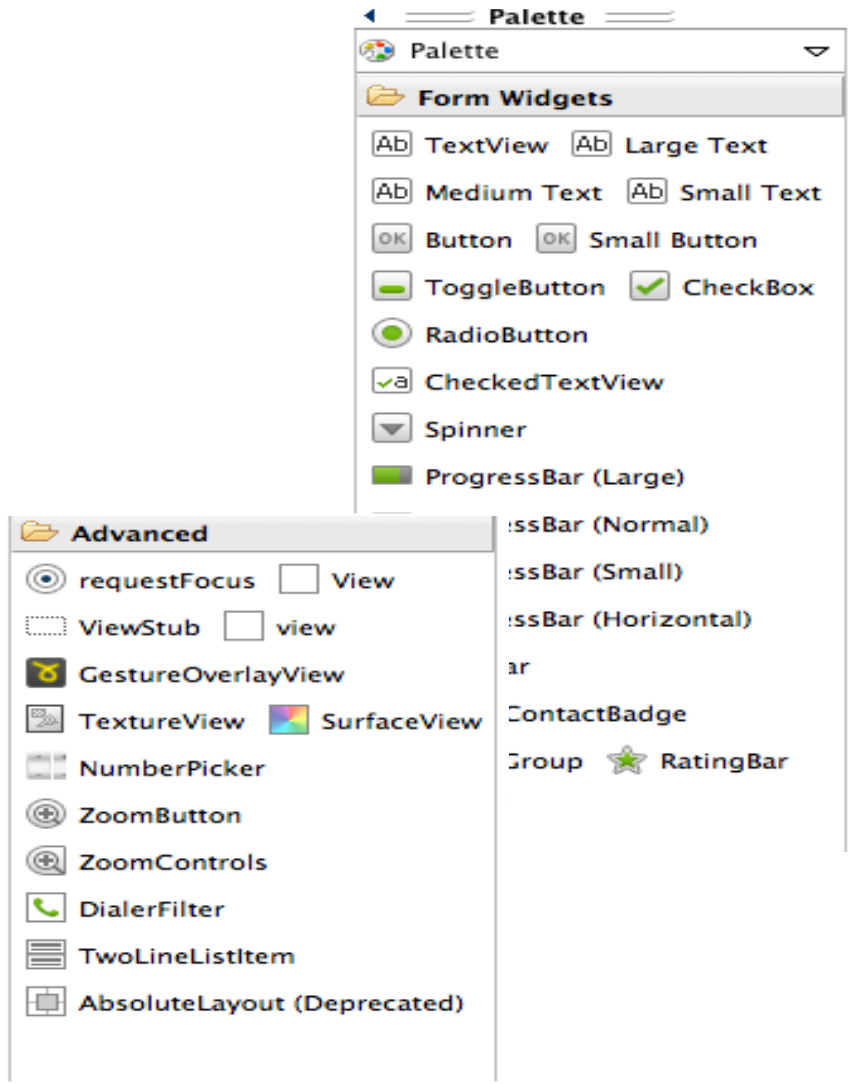
Navigating Android Studio *

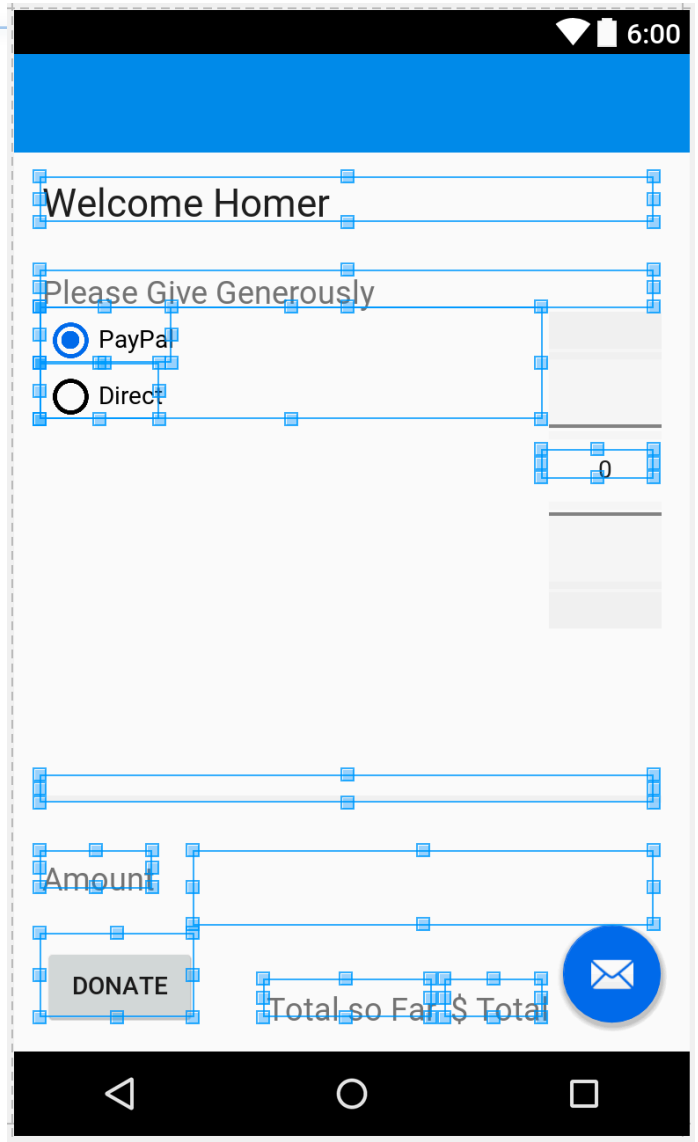
- Android Studio opens your project in the workbench window
- The different panes of the workbench window are called views.
- The lefthand view is the project explorer. From the project explorer, you can manage the files associated with your project.
- The middle view is the editor. Here, Android Studio has open content_donate.xml in the editor.
- There are also views on the righthand side and the bottom of the workbench. Close any views on the righthand side by clicking the x next to the view's name



Laying Out the User Interface *

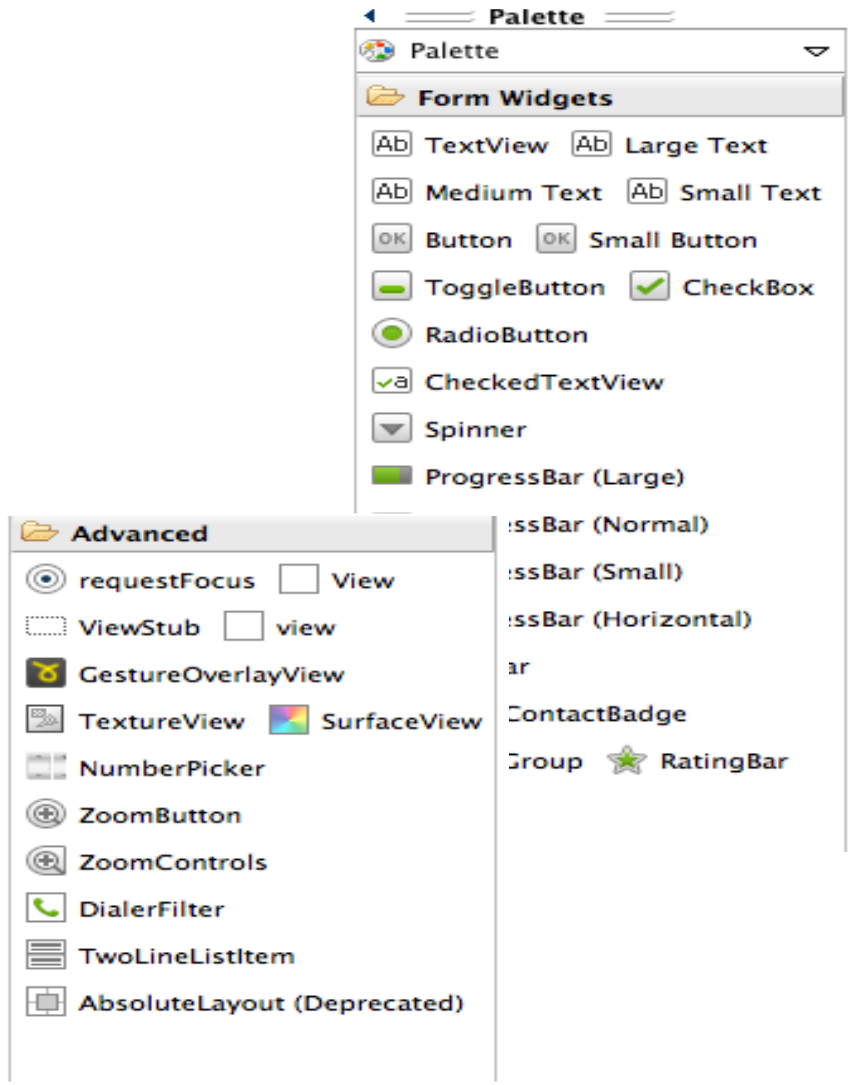
- TextView
- Button
- RadioGroup
- ProgressBar
- NumberPicker
- EditText

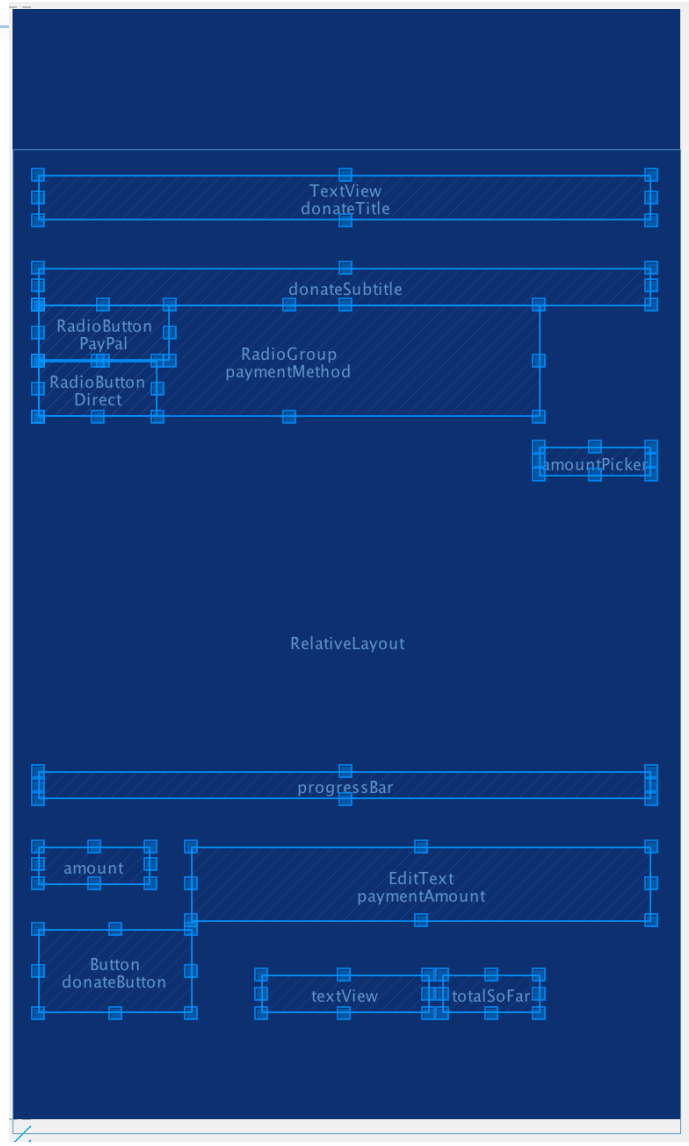




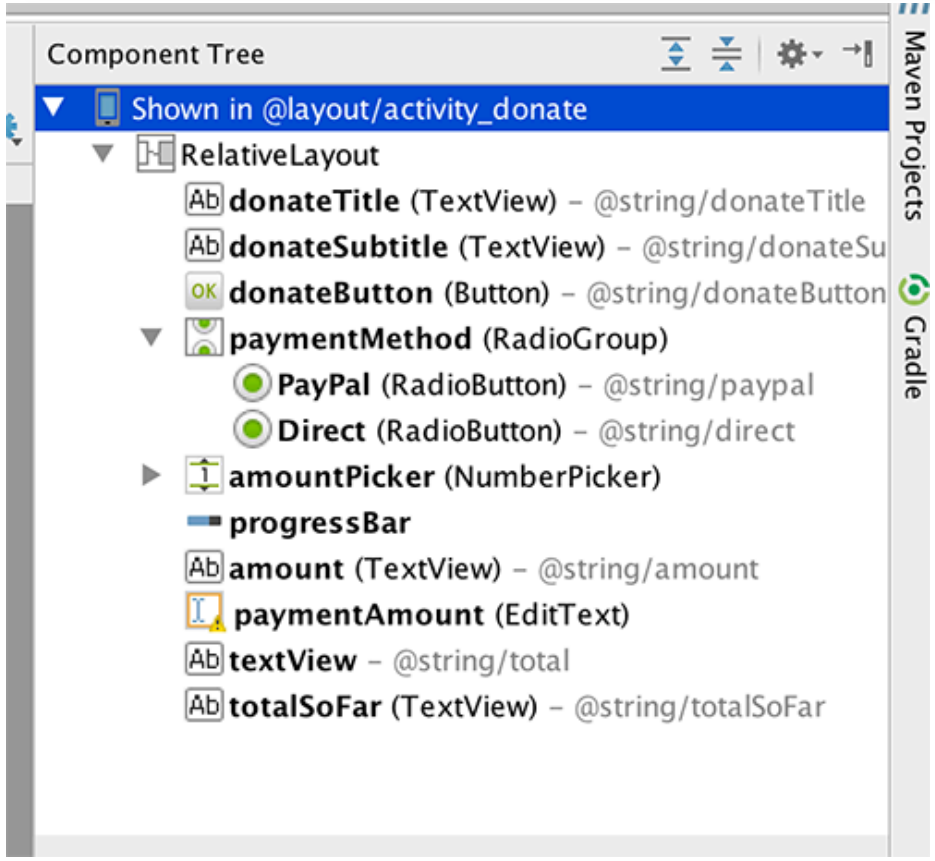
Laying Out the User Interface

- TextView
- Button
- RadioGroup
- ProgressBar
- NumberPicker
- EditText

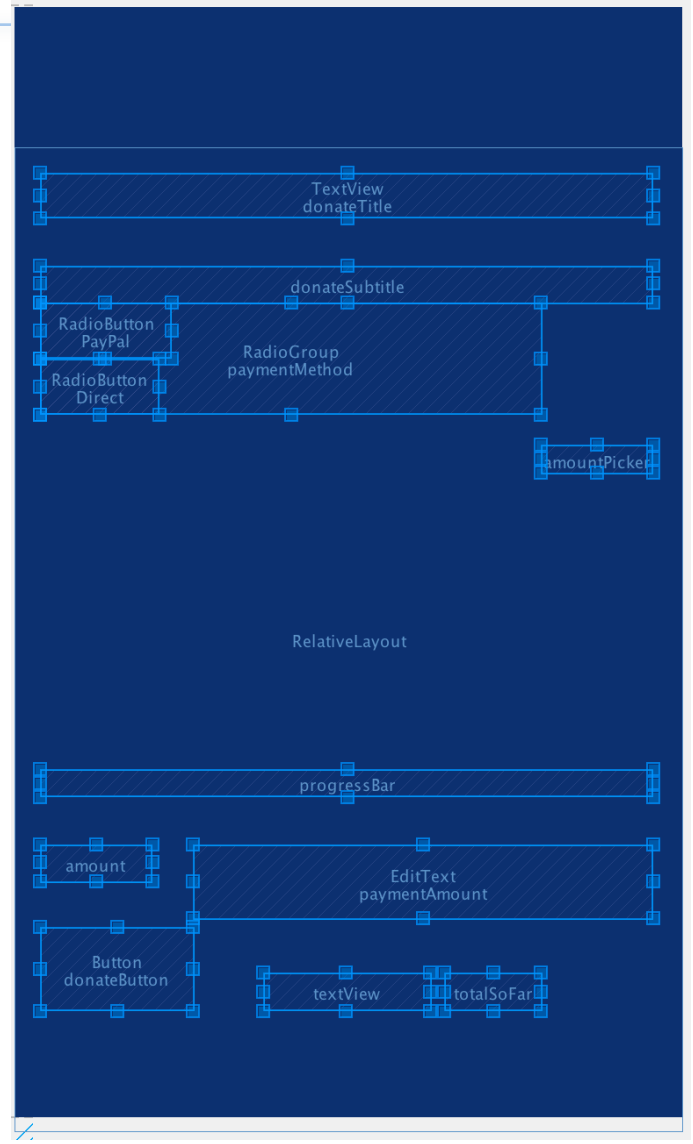




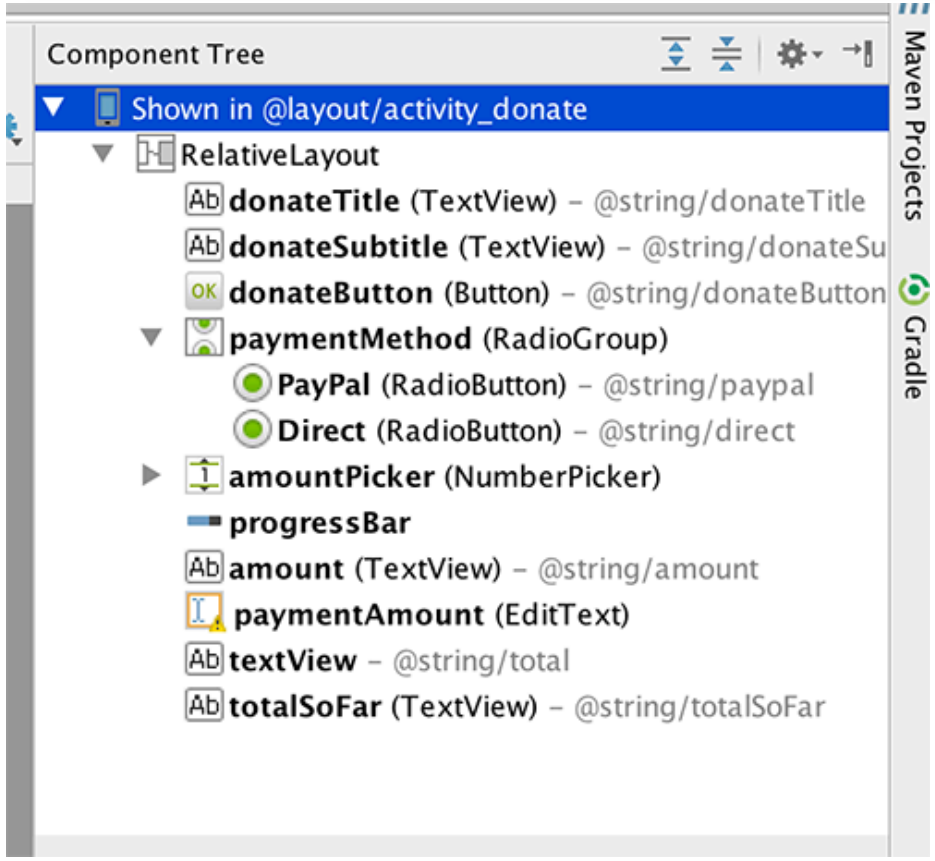
Laying Out the User Interface – Outline View



- Keep track of Outline view
- Name controls appropriately



The Outline View hierarchy



- RelativeLayout is the root
- It has 10 child nodes
  - 5 TextViews
  - 1 Push Button
  - 1 Number Picker
  - 1 Radio Group
    - which has 2 child node RadioButtons
  - 1 EditText
  - 1 ProgressBar

The *View* (activity_donate.xml) 'Source' *

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:fitsSystemWindows="true" tools:context=".Donate">

    <android.support.design.widget.AppBarLayout android:layout_height="wrap_content"
        android:layout_width="match_parent" android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar android:id="@+id/toolbar"
            android:layout_width="match_parent" android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary" app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_donate" />

    <android.support.design.widget.FloatingActionButton android:id="@+id/fab"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_gravity="bottom|end" android:layout_margin="16dp"
        android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

The *View* (content_donate.xml) 'Source' *

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="64dp"
    android:paddingRight="64dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    app:layout_behavior="android.support.design.widget.AppBarLayout$ScrollingVie..."
    tools:showIn="@layout/activity_donate" tools:context=".Donate">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Welcome Homer"
        android:id="@+id/donateTitle"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />

    <TextView...>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Donate"
        android:id="@+id/donateButton"
        android:onClick="donateButtonPressed"
        android:layout_marginBottom="53dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true" />

    <RadioGroup
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/donateSubtitle"
        android:layout_alignParentStart="true"
        android:id="@+id/paymentMethod"
        android:layout_toStartOf="@+id/amountPicker">

        <RadioButton...>

        <RadioButton...>
    </RadioGroup>
```

```
    <NumberPicker
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/amountPicker"
        android:layout_alignTop="@+id/paymentMethod"
        android:layout_alignEnd="@+id/donateSubtitle" />

    <ProgressBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="?android:attr/progressBarStyleHorizontal"
        android:id="@+id/progressBar"
        android:indeterminate="false"
        android:layout_marginBottom="27dp"
        android:layout_above="@+id/amount"
        android:layout_alignParentStart="true"
        android:layout_alignEnd="@+id/donateSubtitle" />

    <TextView...>

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:id="@+id/paymentAmount"
        android:layout_alignTop="@+id/amount"
        android:layout_alignEnd="@+id/progressBar"
        android:layout_toEndOf="@+id/donateButton" />

    <TextView...>

    <TextView...>
</RelativeLayout>
```

Widget attributes

- The android:layout_width and android:layout_height attributes are required for almost every type of widget.
- They are typically set to either match_parent or wrap_content:
  - match_parent view will be as big as its parent
  - wrap_content view will be as big as its contents require

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Welcome Homer"
    android:id="@+id/donateTitle"
    android:layout_alignParentTop="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true" />
```

String resources *

- Notice that the values of strings are not literal strings. They are references to string resources
- A string resource is a string that lives in a separate XML file called a strings file.
- You can give a widget a hard-coded string, like *android:text="True"*, but it is usually not a good idea.
- Placing strings into a separate file and then referencing them is better, making localization easy.

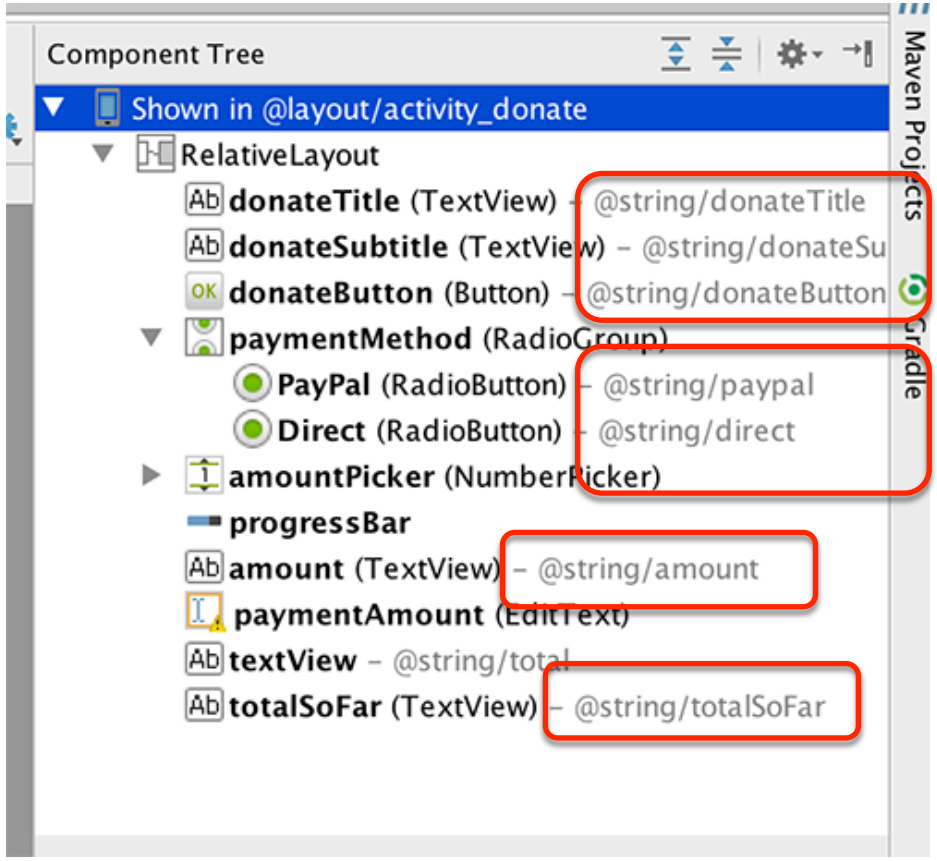
```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/donateSubtitle"
    android:layout_alignParentStart="true"
    android:id="@+id/paymentMethod"
    android:layout_toStartOf="@+id/amountPicker">

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/paypal"
        android:id="@+id/PayPal"
        android:checked="true" />

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/direct"
        android:id="@+id/Direct"
        android:checked="false" />
</RadioGroup>
```

String resources file – strings.xml *

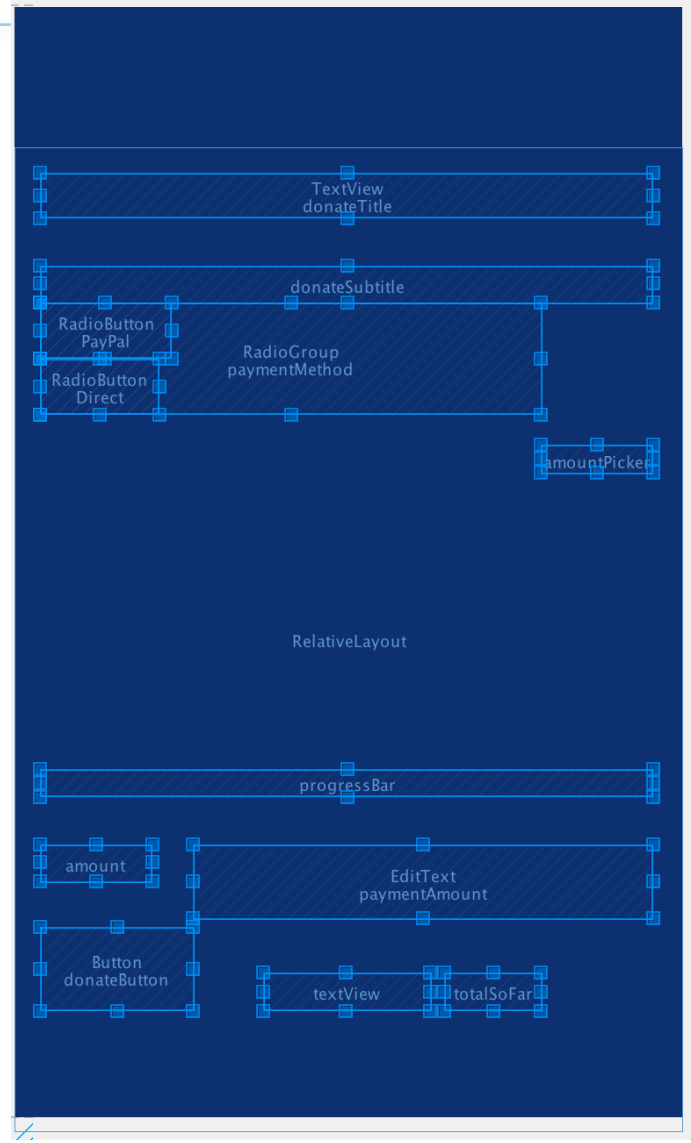
- Every project includes a default strings file named strings.xml.
- Whenever you refer to, for example, '*@string/direct*' in any XML file in the project, you will get the literal string "Direct" at runtime.
- The default strings file is named strings.xml, but you can name a strings file anything you want.
- You can also have multiple strings files in a project. As long as the file is located in **res/values/**, has a resources root element, and contains child string elements, your strings will be found and used appropriately.

```
<resources>
    <string name="app_name">Donation.1.5</string>
    <string name="action_settings">Settings</string>
    <string name="donateTitle">Welcome Homer</string>
    <string name="donateSubtitle">Please Give Generously</string>
    <string name="donateButton">Donate</string>
    <string name="paypal">PayPal</string>
    <string name="direct">Direct</string>
    <string name="amount">Amount</string>
    <string name="total">Total so Far</string>
    <string name="totalSoFar">$ Total</string>
</resources>
```

String resources file & Outline View *



- Keep track of Outline view
- Name controls appropriately



The *Controller* (Donate.java) 'Source' *

The **`onCreate(Bundle)`** method is called when an instance of the activity subclass is created. When an activity is created, it needs a user interface (a View) to manage. To get the activity its user interface, you call the following Activity method *:

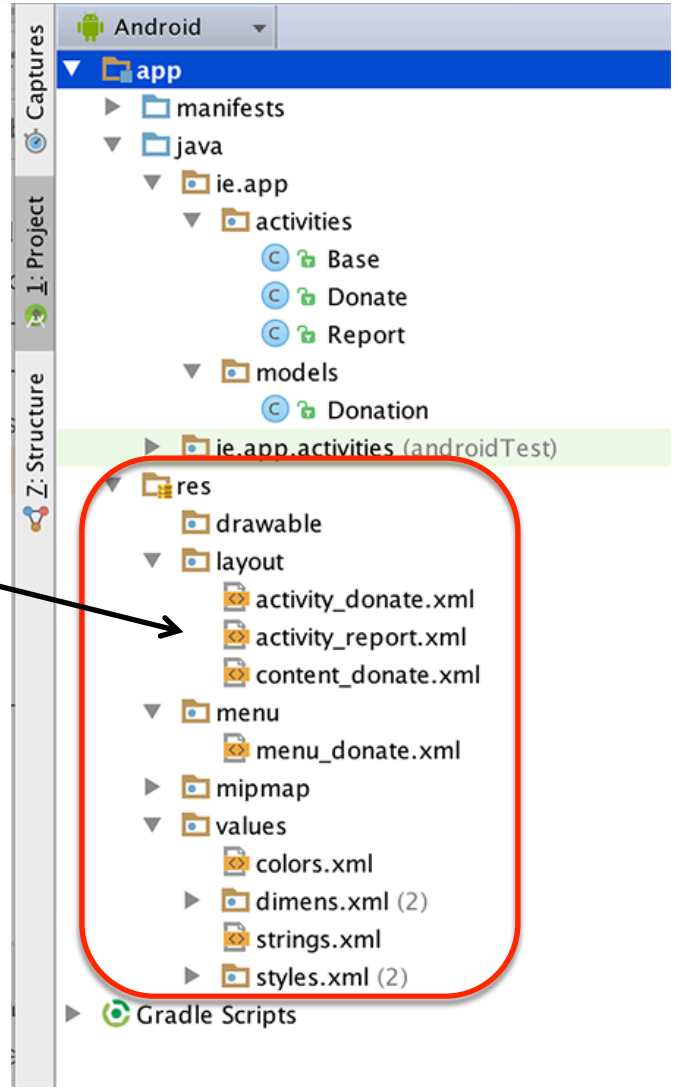
`public void setContentView(int layoutReslD)`

This method inflates a layout and puts it on screen. When a layout is inflated, each widget in the layout file is instantiated as defined by its attributes. You specify which layout to inflate by passing in the layouts resource ID (next slide).

```
public class Donate extends AppCompatActivity {

    private Button          donateButton;
    private RadioGroup      paymentMethod;
    private ProgressBar     progressBar;
    private NumberPicker    amountPicker;
    private EditText        amountText;
    private TextView        amountTotal;

    private int             totalDonated = 0;
    private boolean         targetAchieved = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_donate);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        //...

        donateButton = (Button) findViewById(R.id.donateButton);

        if (donateButton != null) {Log.v("Donate", "Really got the donate button");}

        //...
    }
```

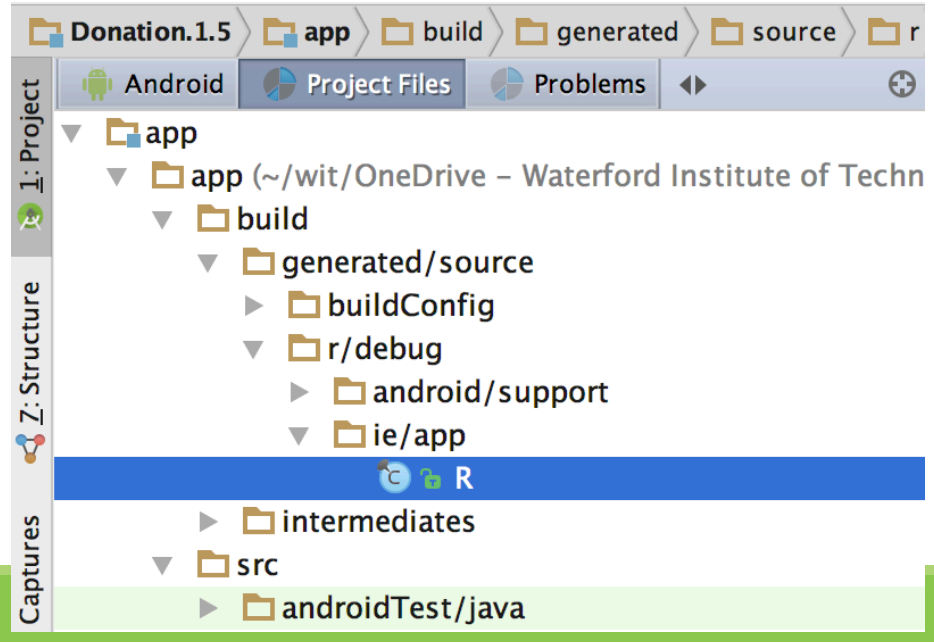
Resources and resource IDs *

- A layout is a resource. A resource is a piece of your application that is not code - things like image files, audio files, and XML files.
- Resources for your project live in a subdirectory of the **res** directory *.
- To access a resource in code, you use its resource ID.
- To see the current resource IDs for your app, go to the package explorer and reveal the contents of the gen directory. Find and open R.java.
- Because this is generated by the Android build process, you should not change it, as you are subtly warned at the top of the file.



R.Java *

- This is a file generated by the android build system.
- It bridges the world of resources and Java, allowing resource IDs to be used in pure java code (next slide *).
- *Never* edit or modify this file, it is automatically updated as new resources are added/edited.

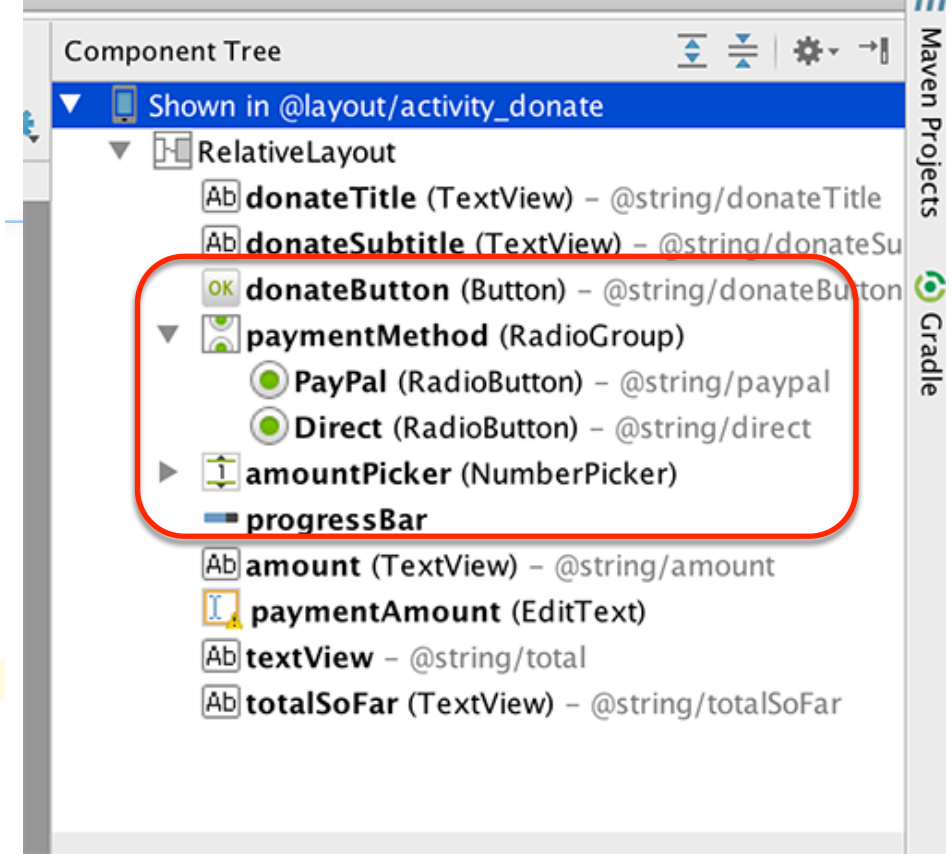


R.java ×

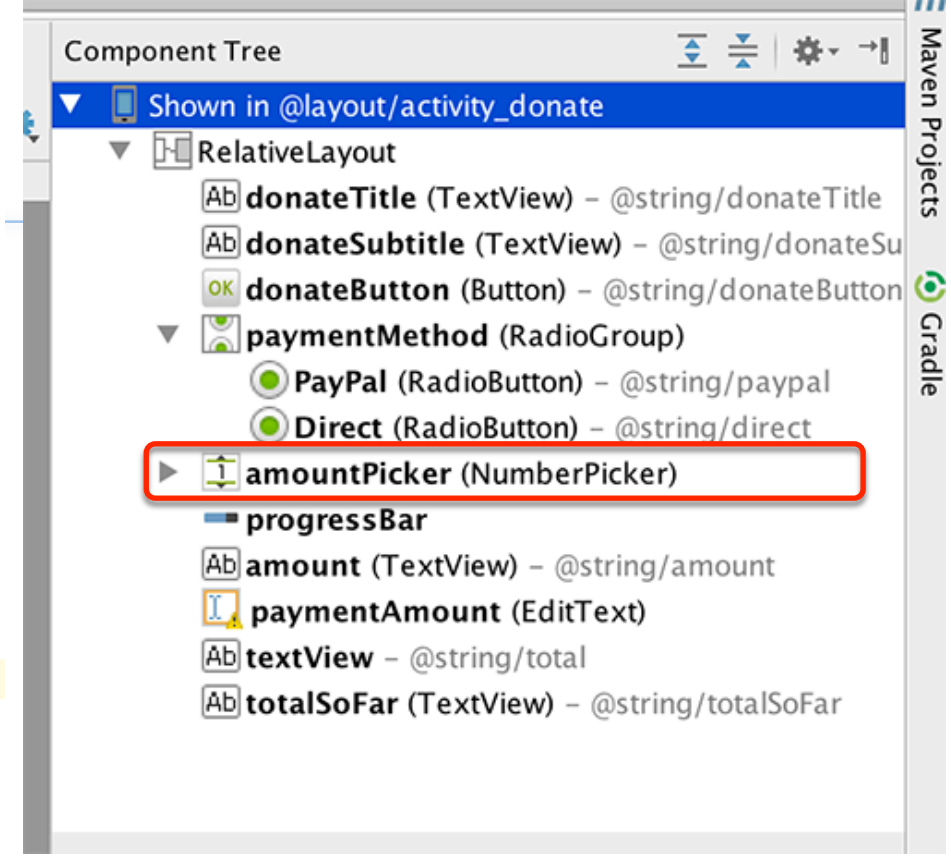
Files under the build folder are generated and should not be edited.

```
1    /.../
7
8    package ie.app;
9
10   public final class R {
11       public static final class anim {
12           public static final int abc_fade_in=0x7f050000;
13           public static final int abc_fade_out=0x7f050001;
14           public static final int abc_grow_fade_in_from_bottom=0x7f050002;
15           public static final int abc_popup_enter=0x7f050003;
16           public static final int abc_popup_exit=0x7f050004;
             public static final int disableHome=0x7f0c000d;
             public static final int donateButton=0x7f0c0068;
             public static final int donateSubtitle=0x7f0c0067;
             public static final int donateTitle=0x7f0c0066;
             public static final int edit_query=0x7f0c0057;
             public static final int end=0x7f0c001e;
             public static final int end_padder=0x7f0c0080;
             public static final int parentPanel=0x7f0c0042;
             public static final int paymentAmount=0x7f0c006f;
             public static final int paymentMethod=0x7f0c0069;
             public static final int pin=0x7f0c0019;
             public static final int progressBar=0x7f0c006d;
             public static final int progress_circular=0x7f0c0005;
             public static final int progress_horizontal=0x7f0c0006;
7998         /**
7999           <p>This symbol is the offset where the {@link android.R.attr#layou
8000           attribute's value can be found in the {@link #ViewStubCompat} arra
8001           @attr name android:layout
8002         */
8003         public static final int ViewStubCompat_android_layout = 1;
8004     };
8005 }
8006
```

```java
public class Donate extends AppCompatActivity {

    private Button          donateButton;
    private RadioGroup      paymentMethod;
    private ProgressBar     progressBar;
    private NumberPicker    amountPicker;
    private EditText        amountText;
    private TextView        amountTotal;

    private int             totalDonated = 0;
    private boolean         targetAchieved = false;

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

        donateButton = (Button) findViewById(R.id.donateButton);

        if (donateButton != null) {Log.v("Donate", "Really got the donate button");}

        paymentMethod = (RadioGroup)   findViewById(R.id.paymentMethod);
        progressBar   = (ProgressBar)  findViewById(R.id.progressBar);
        amountPicker  = (NumberPicker) findViewById(R.id.amountPicker);
        amountText    = (EditText)     findViewById(R.id.paymentAmount);
        amountTotal   = (TextView)     findViewById(R.id.totalSoFar);

        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
        progressBar.setMax(10000);
        amountTotal.setText("$0");
    }
```



```java
public class Donate extends AppCompatActivity {

    private Button          donateButton;
    private RadioGroup      paymentMethod;
    private ProgressBar     progressBar;
    private NumberPicker    amountPicker;
    private EditText        amountText;
    private TextView        amountTotal;

    private int             totalDonated = 0;
    private boolean         targetAchieved = false;

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

        donateButton = (Button) findViewById(R.id.donateButton);

        if (donateButton != null) {Log.v("Donate", "Really got the donate button");}

        paymentMethod = (RadioGroup)   findViewById(R.id.paymentMethod);
        progressBar   = (ProgressBar)  findViewById(R.id.progressBar);
        amountPicker  = (NumberPicker) findViewById(R.id.amountPicker);
        amountText    = (EditText)     findViewById(R.id.paymentAmount);
        amountTotal   = (TextView)     findViewById(R.id.totalSoFar);

        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
        progressBar.setMax(10000);
        amountTotal.setText("$0");
    }
```



Setting listeners

- Android applications are typically event-driven.
- Unlike command-line programs or scripts, event-driven applications start and then wait for an event, such as the user pressing a button.
  - (Events can also be initiated by the OS or another application, but user-initiated events are the most obvious.)
- When your application is waiting for a specific event, we say that it is "*listening for*" that event.
- The object that you create to respond to an event is called a listener. A listener is an object that implements a listener interface for that event.

Setting Listeners - 3 Different Styles

- The three styles are:
  1. Explicitly set in Resource File
  2. Using a dedicated Listener Interface
  3. Using an Anonymous Inner Class
- Ultimately, we need to master all three.

1. Explicitly set in Resource File *

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Donate"
    android:id="@+id/donateButton"
    android:onClick="donateButtonPressed"
    android:layout_marginBottom="53dp"
    android:layout_alignParentBottom="true"
    android:layout_alignParentStart="true" />
```

- Add onClick property to xml element
- Implement corresponding method in associated class (note the View parameter)

```
public void donateButtonPressed (View view)
{
    String method = paymentMethod.getCheckedRadioButtonId() == R.id.PayPal ? "PayPal" : "Direct";

    int donatedAmount =  amountPicker.getValue();
    if (donatedAmount == 0)
    {
        String text = amountText.getText().toString();
        if (!text.equals(""))
            donatedAmount = Integer.parseInt(text);
    }

 //...
}
```

Making Toast

- A toast is a short message that informs the user of something but does not require any input or action
- To create a toast, you call the following method from the Toast class:
  - public static Toast makeText(Context context, int resId, int duration)
  - The Context parameter is typically an instance of Activity (Activity is a subclass of Context).
  - The second parameter is the resource ID of the string that the toast should display. The Context is needed by the Toast class to be able to find and use the string's resource ID.
  - The third parameter is usually one of two Toast constants that specify how long the toast should be visible.

Displaying Toasts *

- After you have created a toast, you call Toast.show() on it to get it on screen

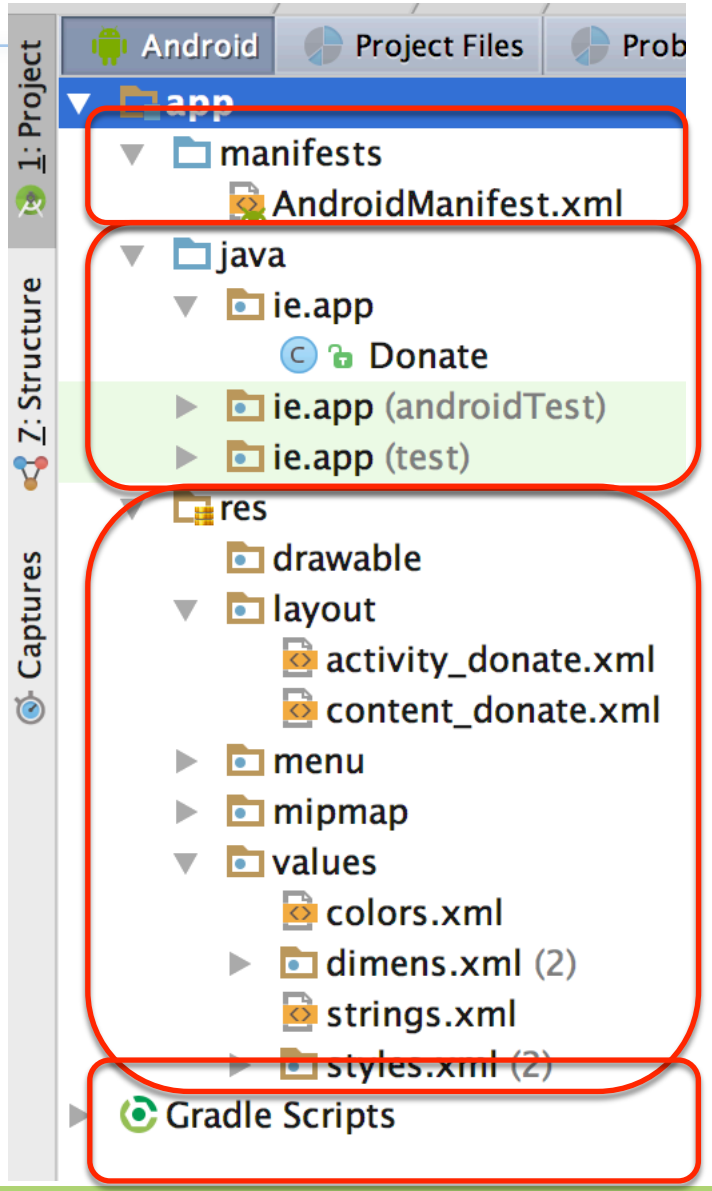
```
if (!targetAchieved)
{
    totalDonated  = totalDonated + donatedAmount;
    targetAchieved = totalDonated >= 10000;
    progressBar.setProgress(totalDonated);
    String totalDonatedStr = "$" + totalDonated;
    amountTotal.setText(totalDonatedStr);
}
else
{
    Toast toast = Toast.makeText(this, "Target Exceeded!", Toast.LENGTH_SHORT);
    toast.show();
}
```

- Alternatively you can 'chain' the method calls, like so, (removing the need for a Toast instance)

```
Toast.makeText(this, "Target Exceeded!", Toast.LENGTH_SHORT).show();
```

Project Structure – Detail *

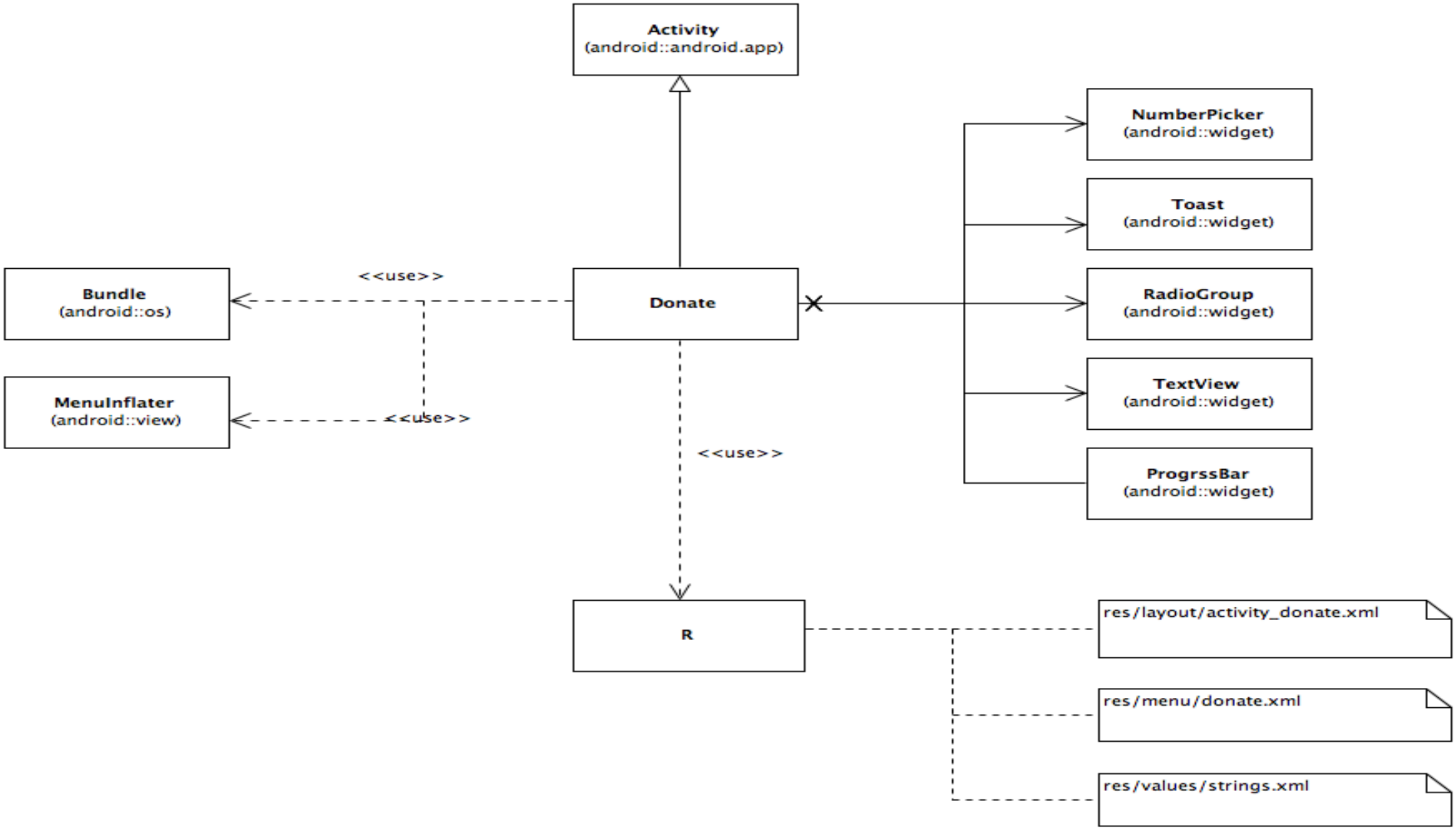
- All Java source files, unit testing : *java*
- All resources - *res*
- Overall project config: *AndroidManifest.xml*
- *Gradle Build for dependencies, libraries etc.*



```
public class Donate extends AppCompatActivity {

    private Button          donateButton;
    private RadioGroup      paymentMethod;
    private ProgressBar     progressBar;
    private NumberPicker    amountPicker;
    private EditText        amountText;
    private TextView        amountTotal;

    private int             totalDonated = 0;
    private boolean         targetAchieved = false;

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

        donateButton = (Button) findViewById(R.id.donateButton);

        if (donateButton != null) {Log.v("Donate", "Really got the donate button");}

        paymentMethod = (RadioGroup)   findViewById(R.id.paymentMethod);
        progressBar   = (ProgressBar)  findViewById(R.id.progressBar);
        amountPicker  = (NumberPicker) findViewById(R.id.amountPicker);
        amountText    = (EditText)     findViewById(R.id.paymentAmount);
        amountTotal   = (TextView)     findViewById(R.id.totalSoFar);

        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
        progressBar.setMax(10000);
        amountTotal.setText("$0");
    }
```

A Model? (MVC)*

- Only a single class, so model not particularly useful
- However, the Donate class interacts with at least 8 android framework classes *


Activity
(android::android.app)
NumberPicker
(android::widget)
Toast
(android::widget)
Bundle
(android::os)
<<use>>
Donate
RadioGroup
(android::widget)
TextView
(android::widget)
MenuInflater
(android::view)
<<use>>
ProgrssBar
(android::widget)
<<use>>
R
res/layout/activity_donate.xml
res/menu/donate.xml
res/values/strings.xml

Questions?


Thanks!