

# Mobile Application Development

---

Produced  
by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics  
Waterford Institute of Technology

<http://www.wit.ie>

<http://elearning.wit.ie>



Waterford Institute of Technology  
INSTITIÚID TEICNEOLAÍOCHTA FHORT LÁIRCE



# A First Android Application

---

Donation Request x

localhost:9000/donation

Home Sign Up Log In Make Donation Report Log Out

## Welcome Homer

Please give generously

Select an amount:

\$1000

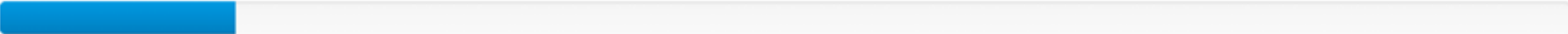
PayPal

Direct

Donate

---

Amount target achieved

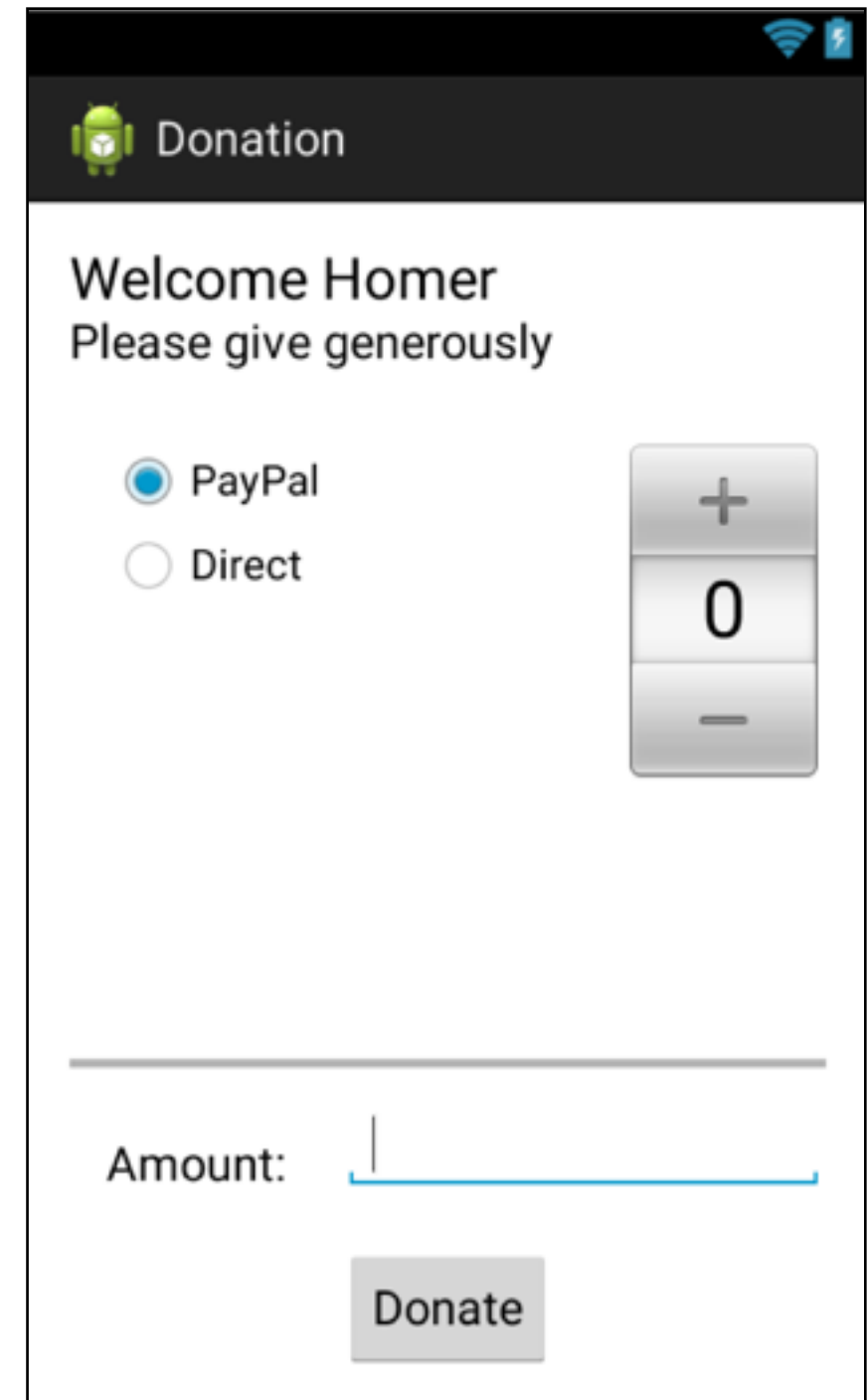


A horizontal progress bar with a blue segment on the left, representing 10% of the target amount.

# App Basics

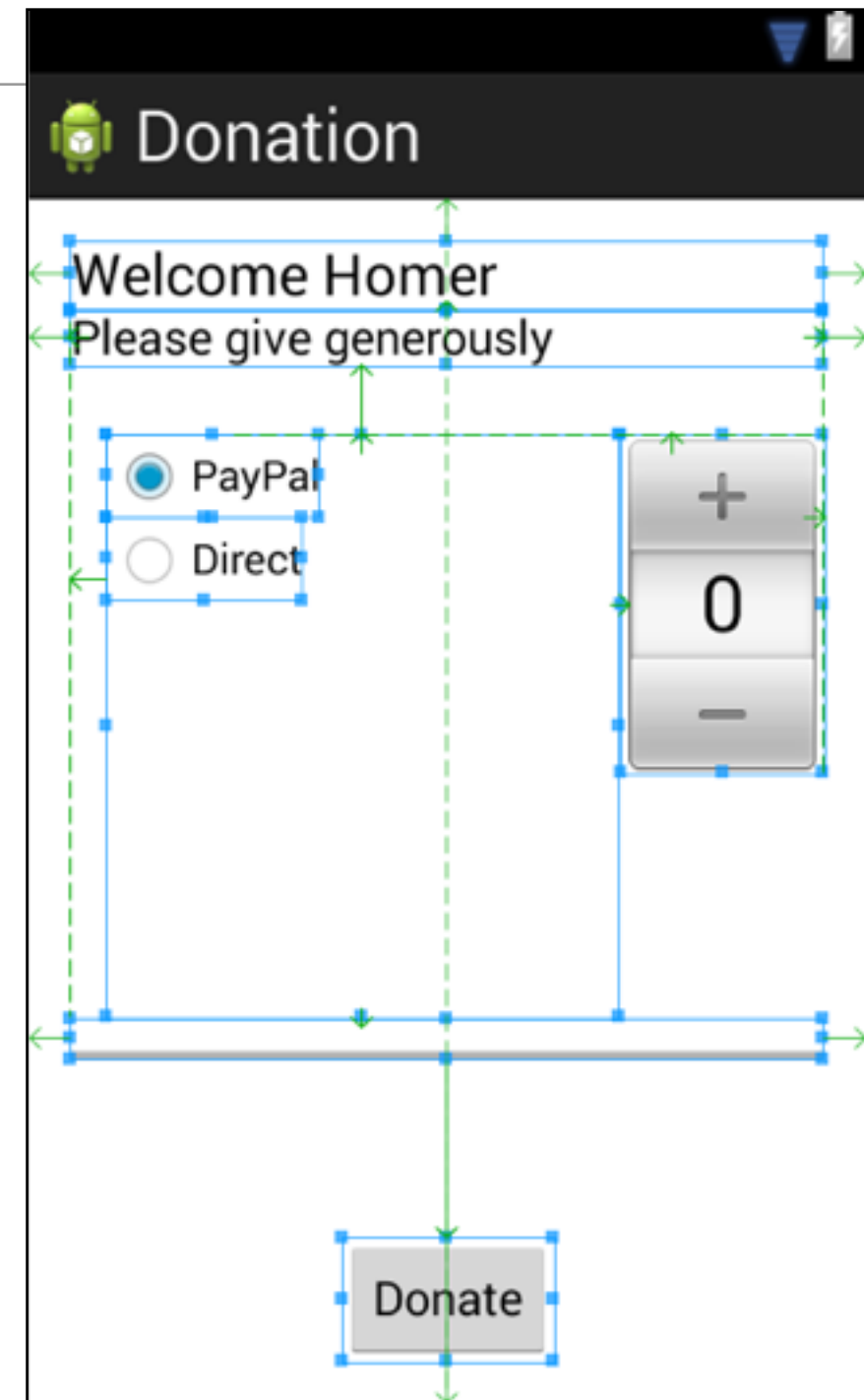
---

- Create a new 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.
- 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 DonationActivity.
- DonationActivity will manage the user interface shown
- A layout defines a set of user interface objects and their position on the screen.
- A layout is made up of definitions written in XML. Each definition is used to create an object that appears onscreen, like a button or some text.
- Donation will include a layout file named activity\_quiz . xml. The XML in this file will define the UI as shown



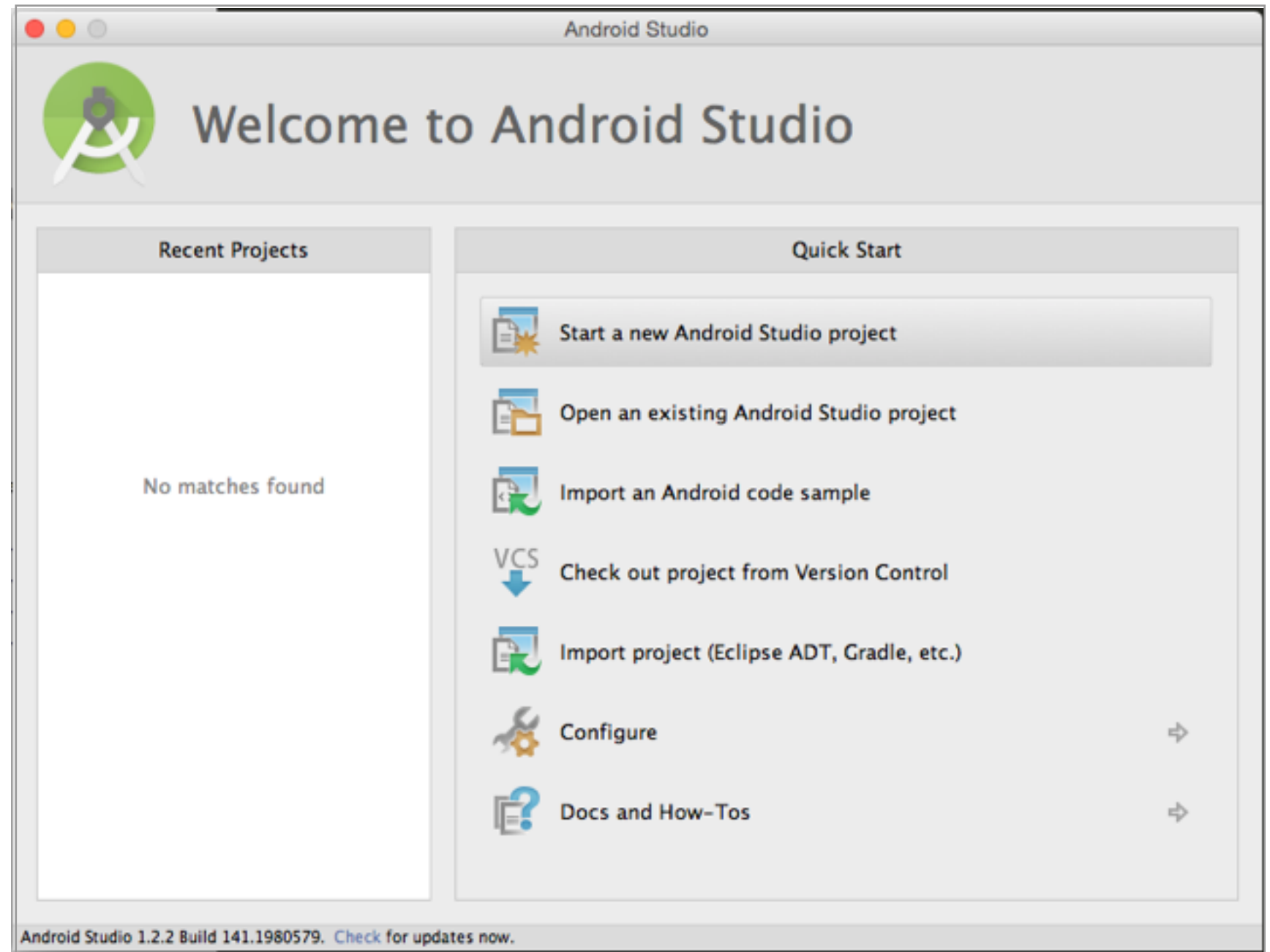
Donate.java



donate\_activity.xml

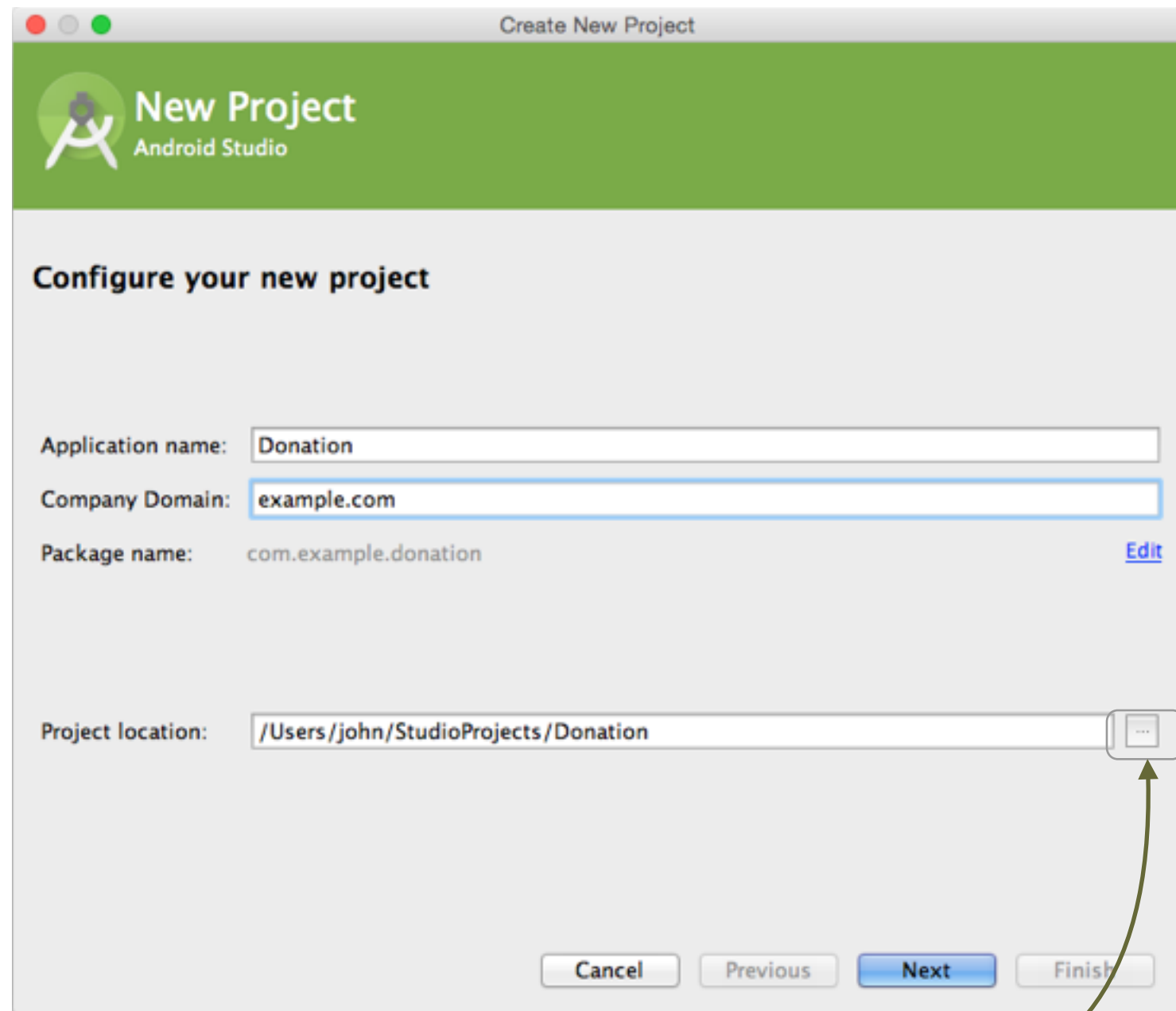
# Creating an Android Project

- 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: *Start a new Android Studio project*



# Creating an Android Project

- In the first dialog, enter Donation as the application name. The project name will automatically update to match the application's.
- For the Company Domain, enter *example.com*. Notice that the package name is automatically generated using a “reverse DNS” convention in which the domain name of your organization is reversed and suffixed with further identifiers: *com.example.donation*
- This convention keeps package names unique and distinguishes applications from each other on a device and on Google Play.
- A default Project location is presented. You may change this if you wish.

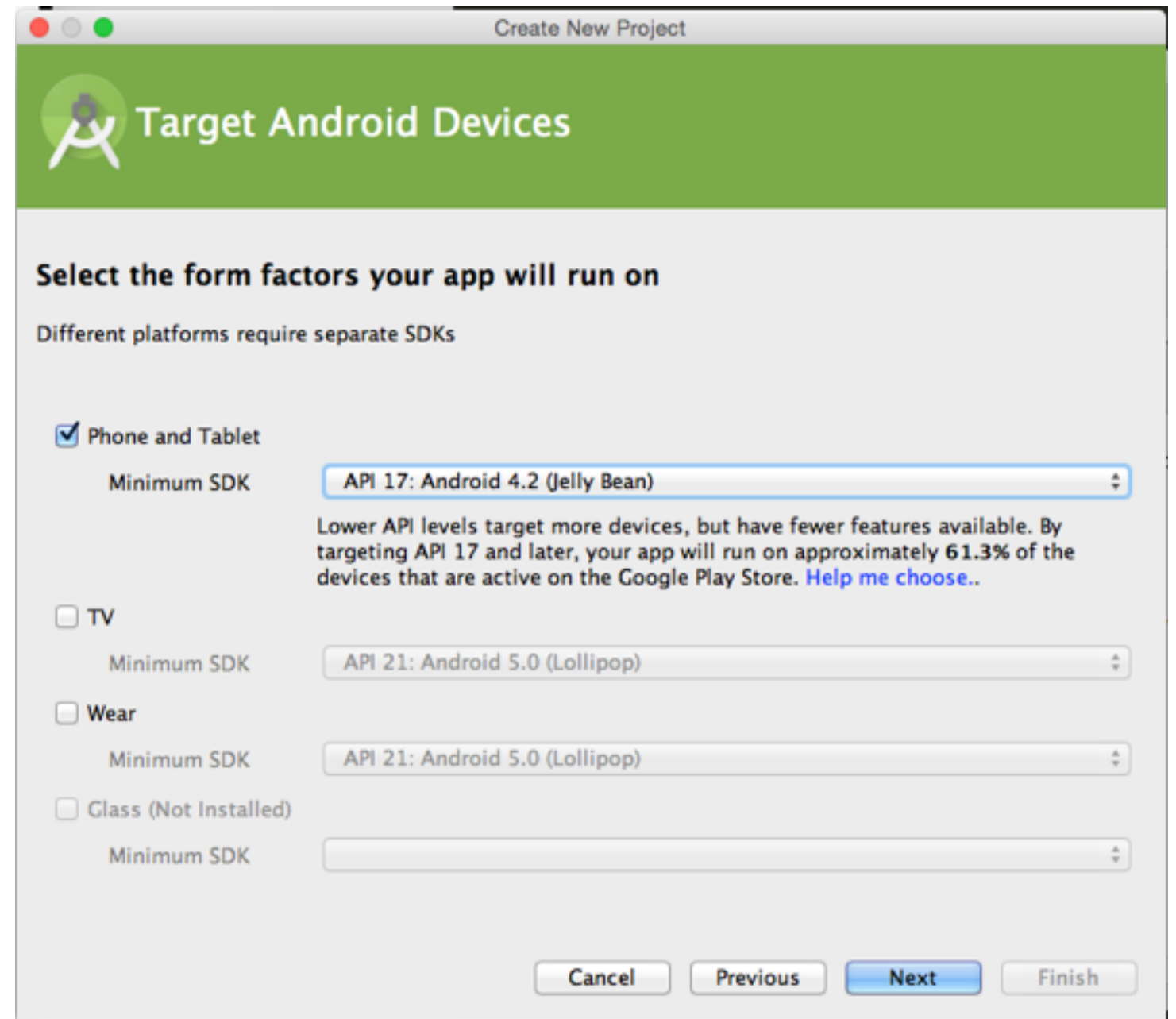


Click to browse to alternative project location



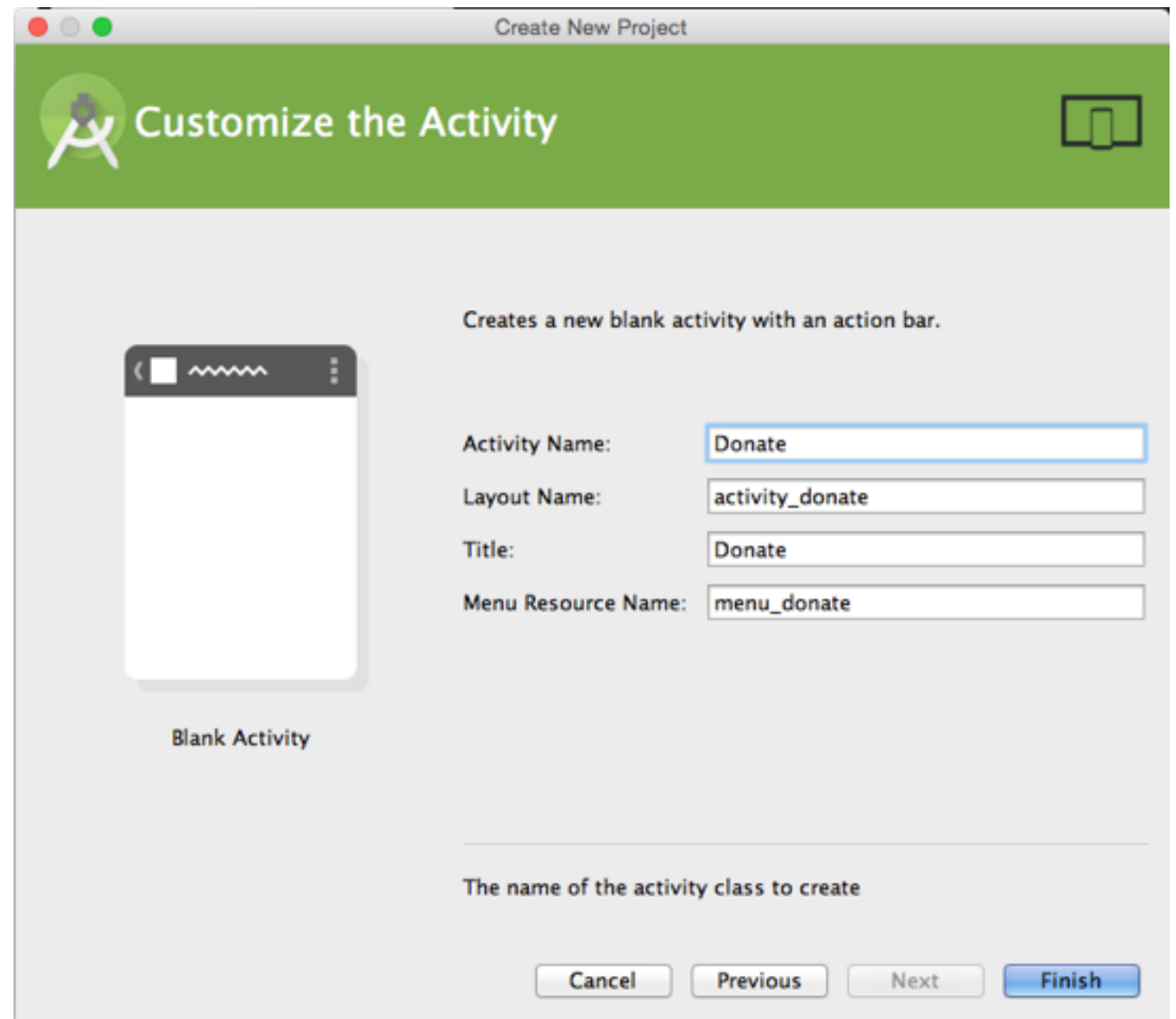
# Creating an Android Project

- Here we target the android devices that we intend developing for.
- We chose Phone & Tablet and a minimum SDK API 17 (Jelly Bean).
- Our app will run only on devices specified with an API level 17 or greater.



# Creating an Android Project

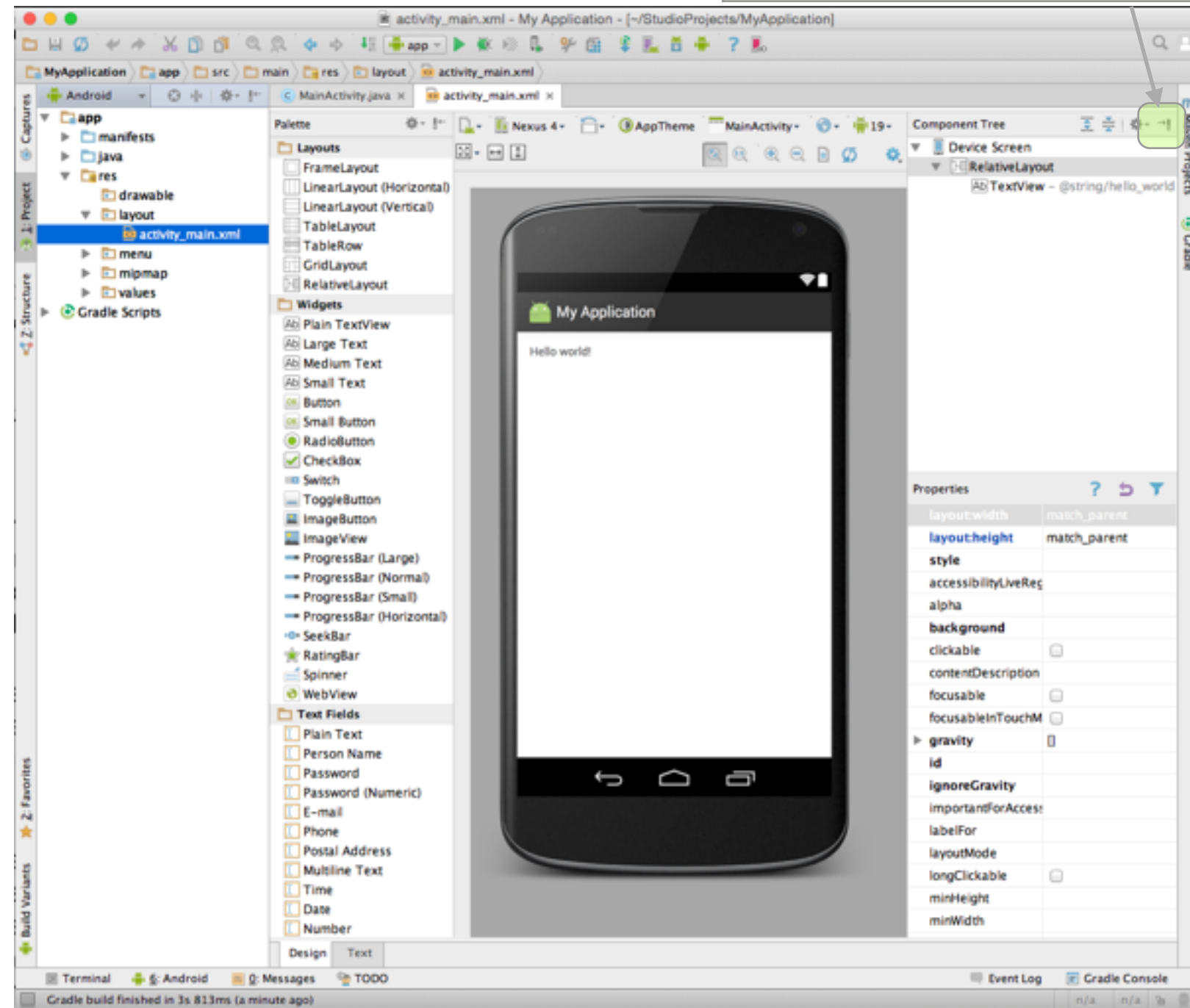
- In the final dialog of this wizard, name the activity subclass `Donate`
- The layout name will automatically update to `activity_donate` to reflect the activity's new name.
- The layout name reverses the order of the activity name, is all lowercase, and has underscores between words.
- This naming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



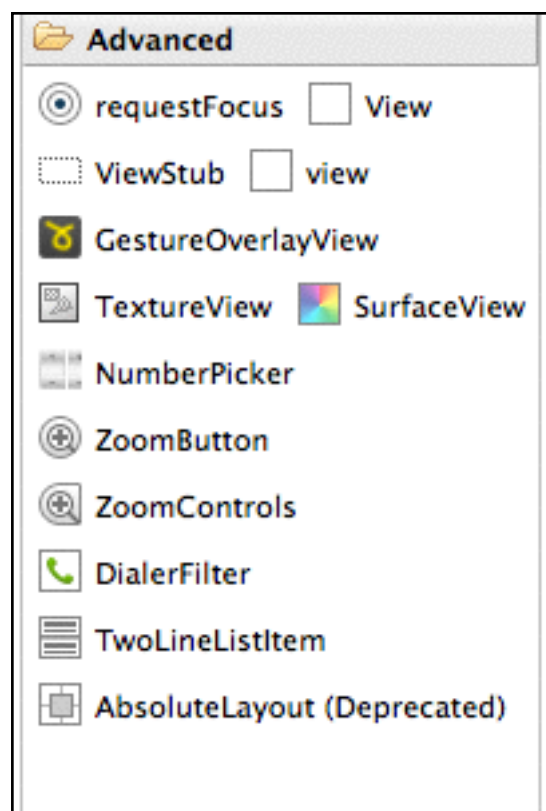
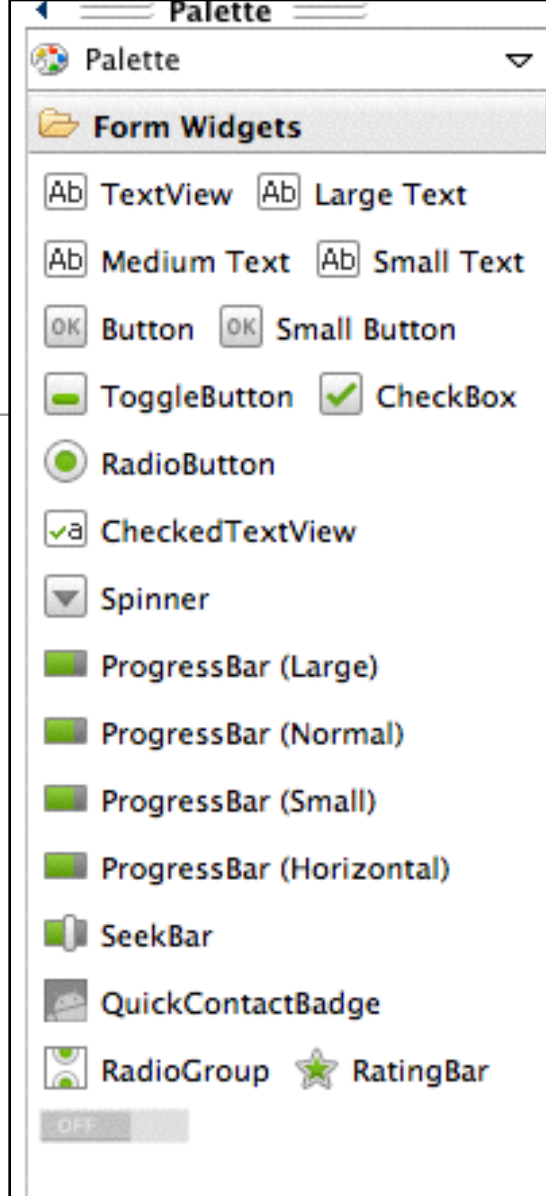
# Navigating Android Studio

Click icon to hide window

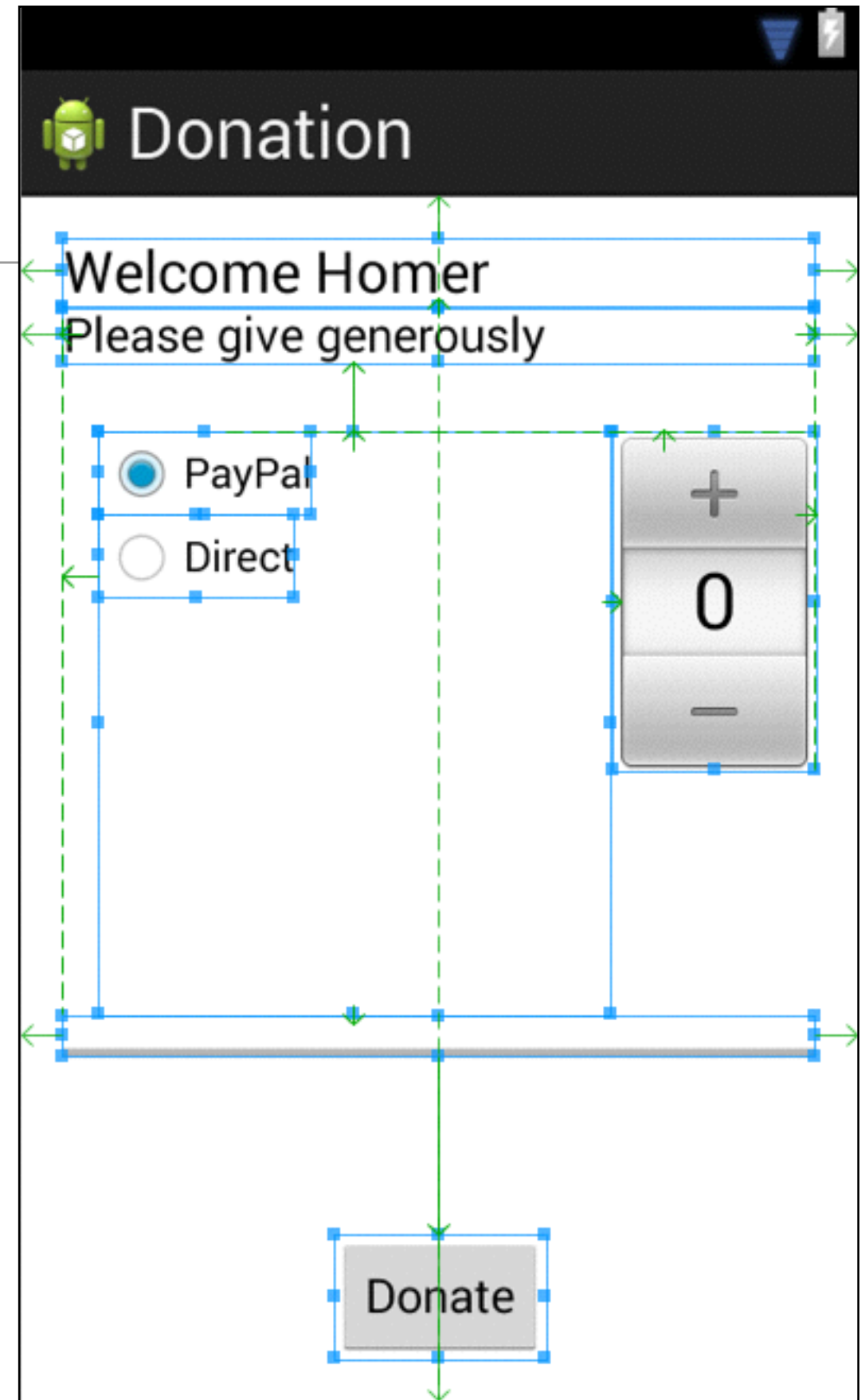
- Android Studio opens your project in the workbench window
- The different panes of the workbench window are called tool windows.
- The lefthand window is the Project Structure. From here you can manage the files associated with your project.
- The middle window is the editor. To get you started, Studio has opened activity\_main.xml in the editor.
- There are also windows on the righthand side and the bottom of the workbench. Close any windows on the righthand side by clicking *hide* icon.



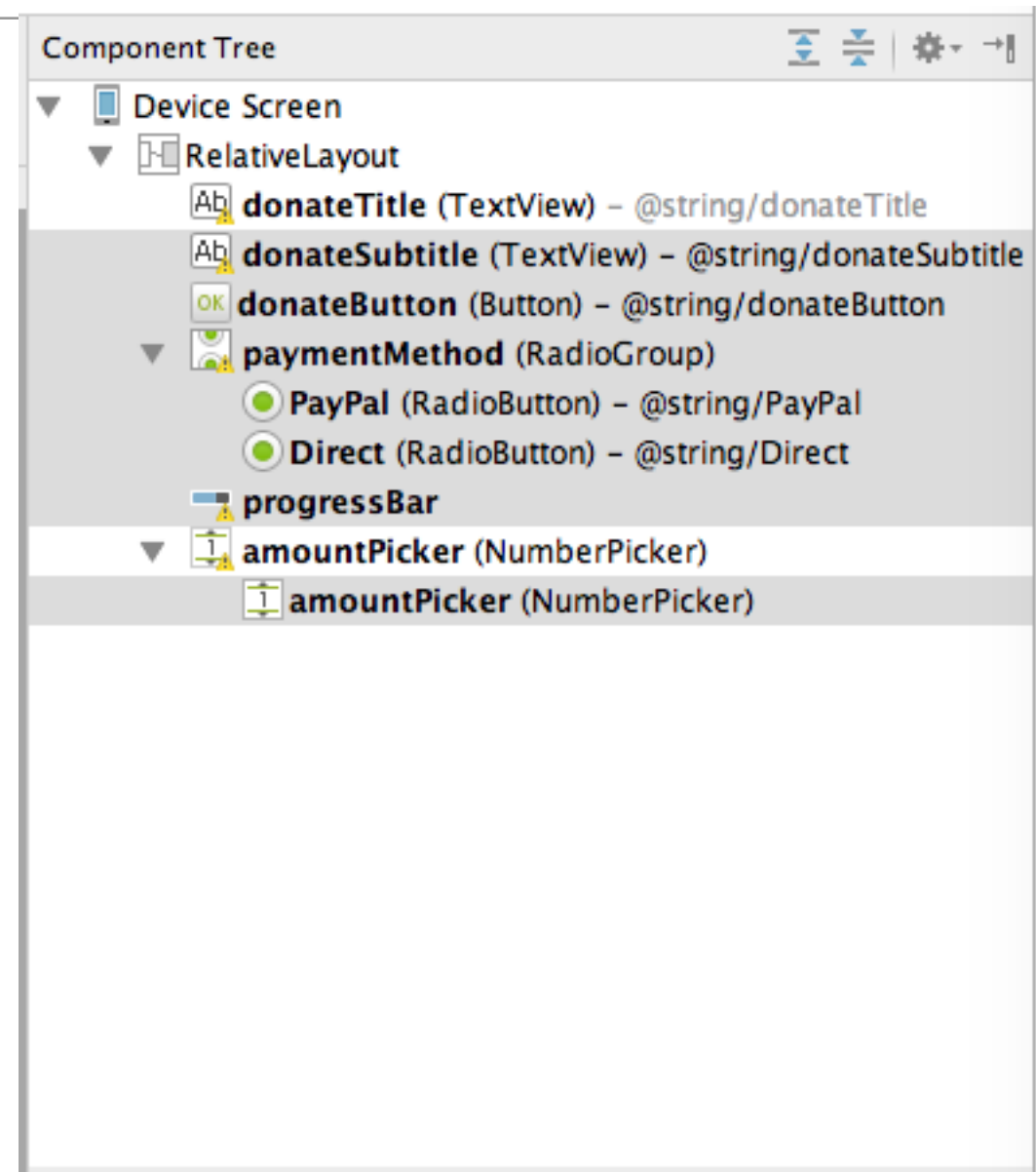
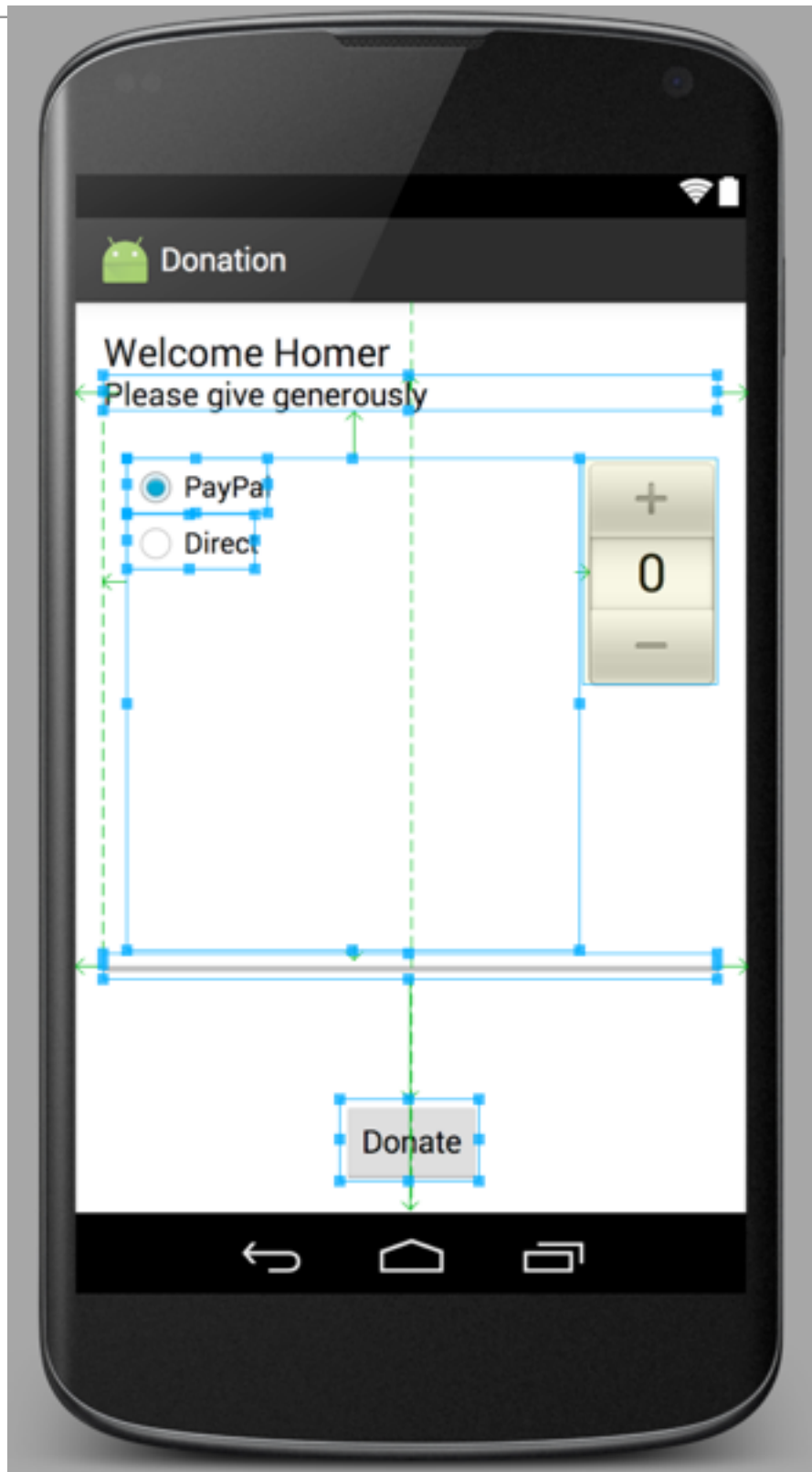
# Laying Out the User Interface



- TextView
- Button
- RadioGroup
- ProgressBar
- NumberPicker



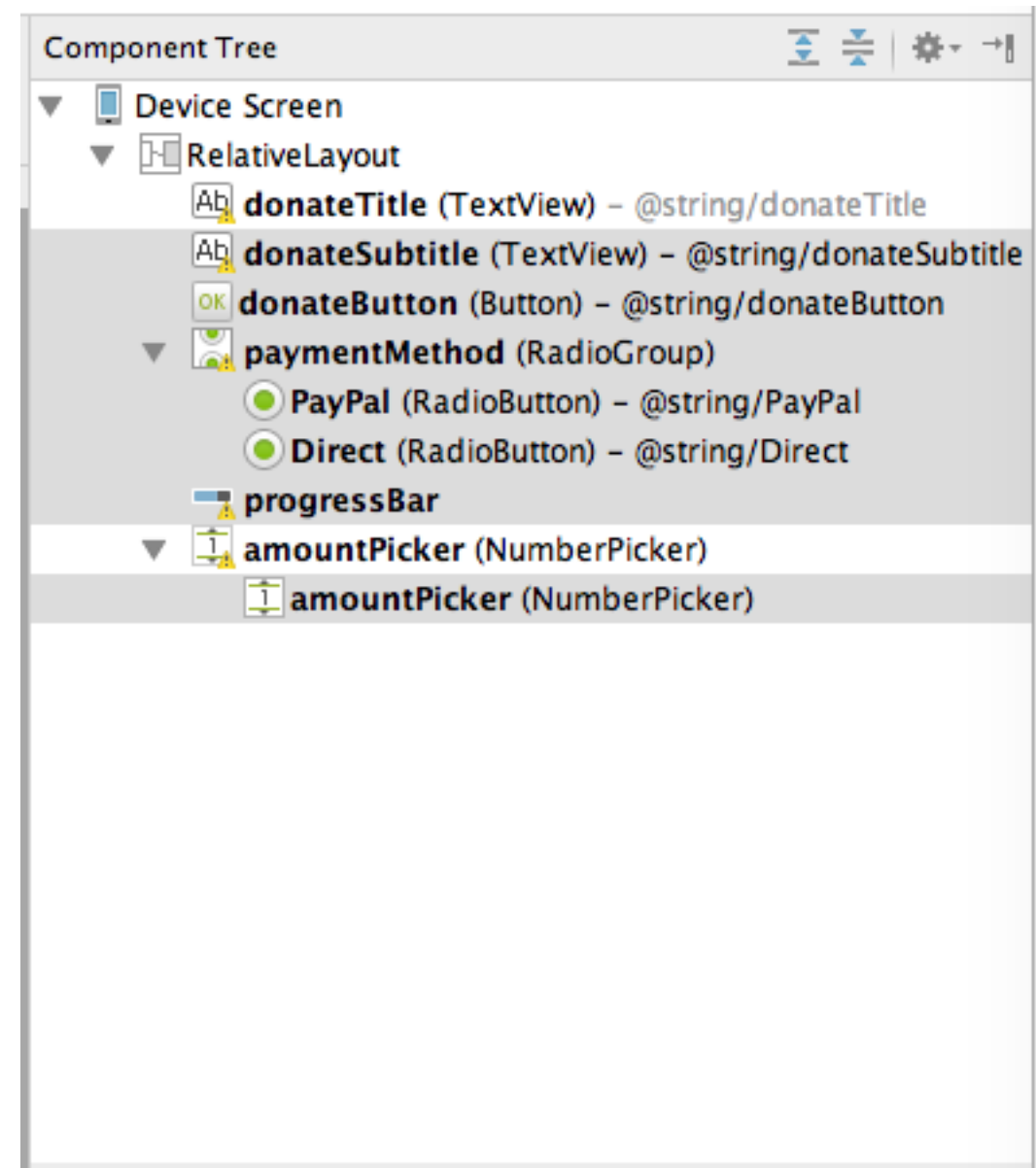
# The view hierarchy





# The view hierarchy

- RelativeLayout is the root
- It has 6 child nodes
  - 2 TextViews
  - 1 Push Button
  - 1 Number Picker
  - 1 Progress bar
  - 1 Radio Group
    - which has 2 child node RadioButtons



# the View 'Source'

```
<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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Donate" >

    <TextView
        android:id="@+id/donateTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:text="@string/donateTitle"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/donateSubtitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/donateTitle"
        android:text="@string/donateSubtitle"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <Button
        android:id="@+id/donateButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:onClick="donateButtonPressed"
        android:text="@string/donateButton" />
```

```
<RadioGroup
    android:id="@+id/paymentMethod"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/progressBar"
    android:layout_alignLeft="@+id/donateSubtitle"
    android:layout_below="@+id/donateSubtitle"
    android:layout_marginLeft="14dp"
    android:layout_marginTop="26dp"
    android:layout_toLeftOf="@+id/amountPicker" >

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

    <RadioButton
        android:id="@+id/Direct"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Direct" />

</RadioGroup>

<ProgressBar
    android:id="@+id/progressBar"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/donateButton"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_marginBottom="67dp" />

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

</RelativeLayout>
```

# Widget attributes

```
<TextView
    android:id="@+id/donateTitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:text="@string/donateTitle"
    android:textAppearance="?android:attr/textAppearanceLarge" />
```

- 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



# 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:id="@+id/paymentMethod"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/progressBar"
    android:layout_alignLeft="@+id/donateSubtitle"
    android:layout_below="@+id/donateSubtitle"
    android:layout_marginLeft="14dp"
    android:layout_marginTop="26dp"
    android:layout_toLeftOf="@+id/amountPicker" >

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

    <RadioButton
        android:id="@+id/Direct"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Direct" />

</RadioGroup>
```

# String Resources File

---

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Donation</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>

</resources>
```

- 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.

```

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/apk/res/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Donate" >

```

```

<TextView
    android:id="@+id/donateTitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:text="@string/donateTitle"
    android:textAppearance="?android:attr/textAppearanceLarge" />

```

```

<TextView
    android:id="@+id/donateSubtitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/donateTitle"
    android:text="@string/donateSubtitle"
    android:textAppearance="?android:attr/textAppearanceMedium" />

```

```

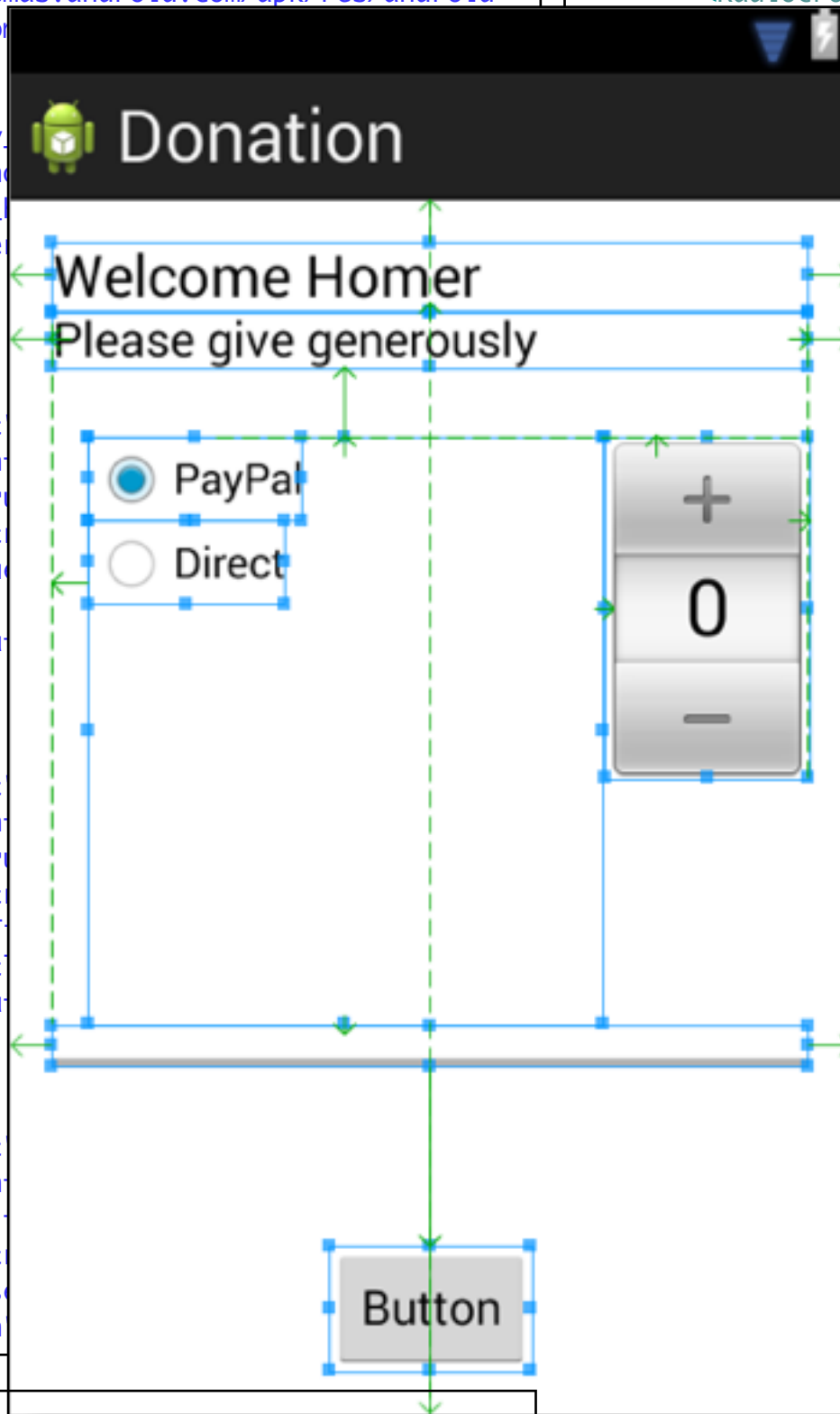
<Button
    android:id="@+id/donateButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:onClick="donateButtonPressed"
    android:text="@string/donateButton" />

```

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Donation</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>
</resources>

```



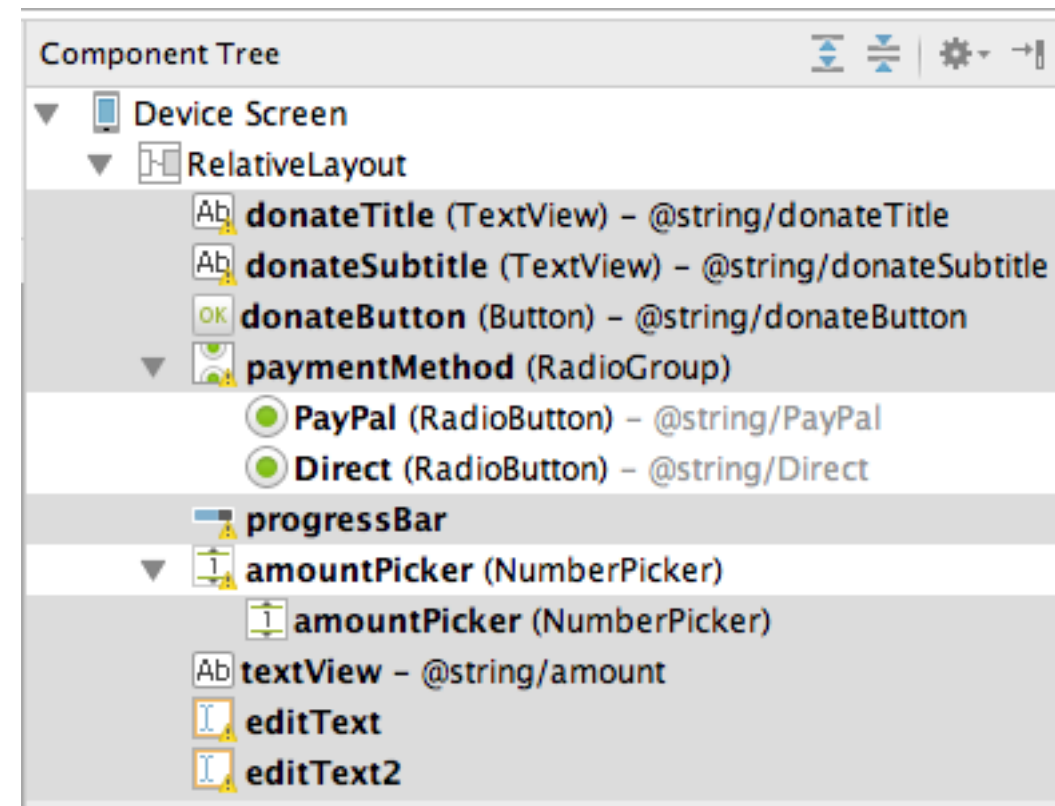
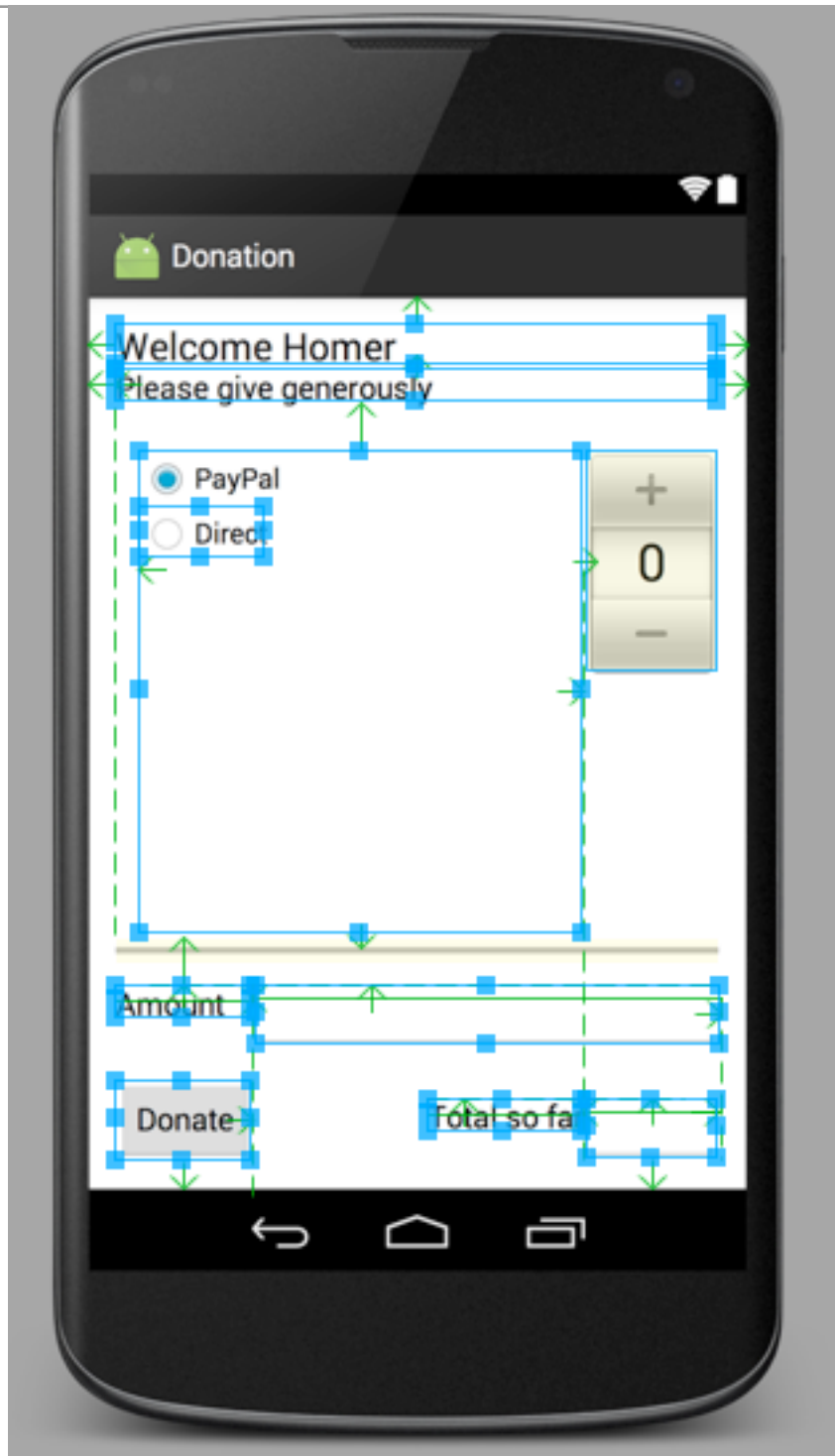
```

<RadioGroup
    android:id="@+id/paymentMethod"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/progressBar"
    android:layout_alignLeft="@+id/donateSubtitle"
    android:layout_below="@+id/donateSubtitle"
    android:layout_marginLeft="14dp"
    android:layout_marginTop="26dp"
    android:layout_toLeftOf="@+id/amountPicker" >
    <Button
        android:id="@+id/PayPal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="@string/PayPal" />
    <Button
        android:id="@+id/Direct"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Direct" />
</RadioGroup>
<ProgressBar
    android:id="@+id/progressBar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_marginBottom="67dp" />
<AmountPicker
    android:id="@+id/amountPicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@+id/donateSubtitle"
    android:layout_alignTop="@+id/paymentMethod" />
</RelativeLayout>

```

# Previewing the layout

# Additional Widgets and Textfields



- Keep track of the Component Tree view
- Give appropriate names to each control

```
public class Donate extends Activity
{
```

```
//...
```

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

    paymentMethod = (RadioGroup) findViewById(R.id.paymentMethod);
    progressBar = (ProgressBar) findViewById(R.id.progressBar);
    amountPicker = (NumberPicker) findViewById(R.id.amountPicker);

    amountPicker.setMinValue(0);
    amountPicker.setMaxValue(1000);
    progressBar.setMax(10000);
}
```

```
//...
```

```
}
```

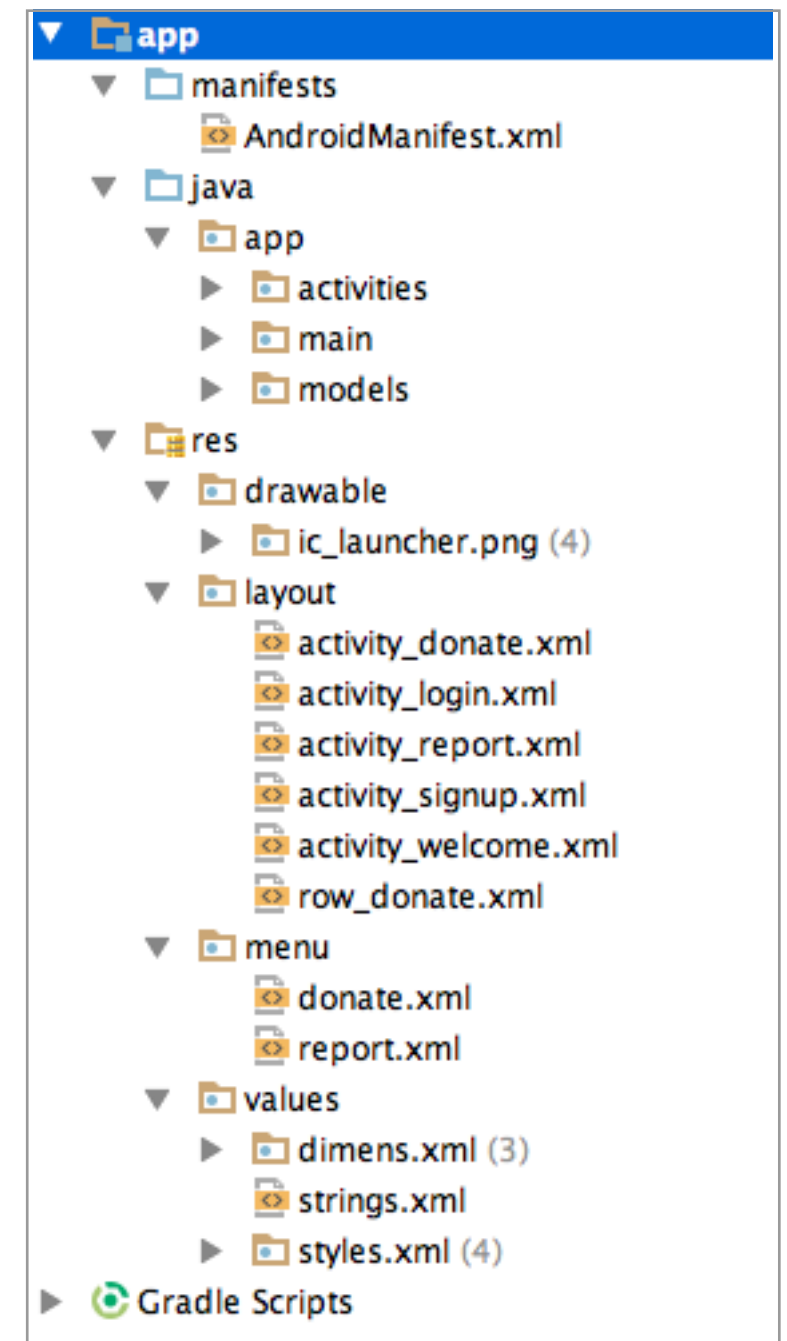
## **public void setContentView(int layoutResID)**

- 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.

- 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 to manage. To get the activity its user interface, you call the following Activity method:

# 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
- Never edit or modify this file, it is automatically updated as new resources are added/edited.

```
public final class R
{
    //...
    public static final class id
    {
        public static final int Direct = 0x7f080006;
        public static final int PayPal = 0x7f080005;
        public static final int action_settings = 0x7f08000c;
        public static final int amountLabel = 0x7f080009;
        public static final int amountPicker = 0x7f080004;
        public static final int amountText = 0x7f080008;
        public static final int amountTotal = 0x7f08000a;
        public static final int donateButton = 0x7f080007;
        public static final int donateSubtitle = 0x7f080001;
        public static final int donateTitle = 0x7f080000;
        public static final int paymentMethod = 0x7f080002;
        public static final int progressBar = 0x7f080003;
        public static final int totalLabel = 0x7f08000b;
    }
    public static final class layout
    {
        public static final int activity_donate = 0x7f030000;
    }
    public static final class menu
    {
        public static final int donate = 0x7f070000;
    }
    public static final class string
    {
        public static final int Direct = 0x7f050006;
        public static final int PayPal = 0x7f050005;
        public static final int action_settings = 0x7f050001;
        public static final int amount = 0x7f050007;
        public static final int amountSoFarLabel = 0x7f050009;
        public static final int app_name = 0x7f050000;
        public static final int donateButton = 0x7f050004;
        public static final int donateSubtitle = 0x7f050003;
        public static final int donateTitle = 0x7f050002;
        public static final int initialAmount = 0x7f050008;
    }
    //...
}
```

```

public class Donate extends Activity
{
    private int          totalDonated = 0;

    private RadioGroup   paymentMethod;
    private ProgressBar  progressBar;
    private NumberPicker amountPicker;

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

        paymentMethod = (RadioGroup) findViewById(R.id.paymentMethod);
        progressBar   = (ProgressBar) findViewById(R.id.progressBar);
        amountPicker  = (NumberPicker) findViewById(R.id.amountPicker);

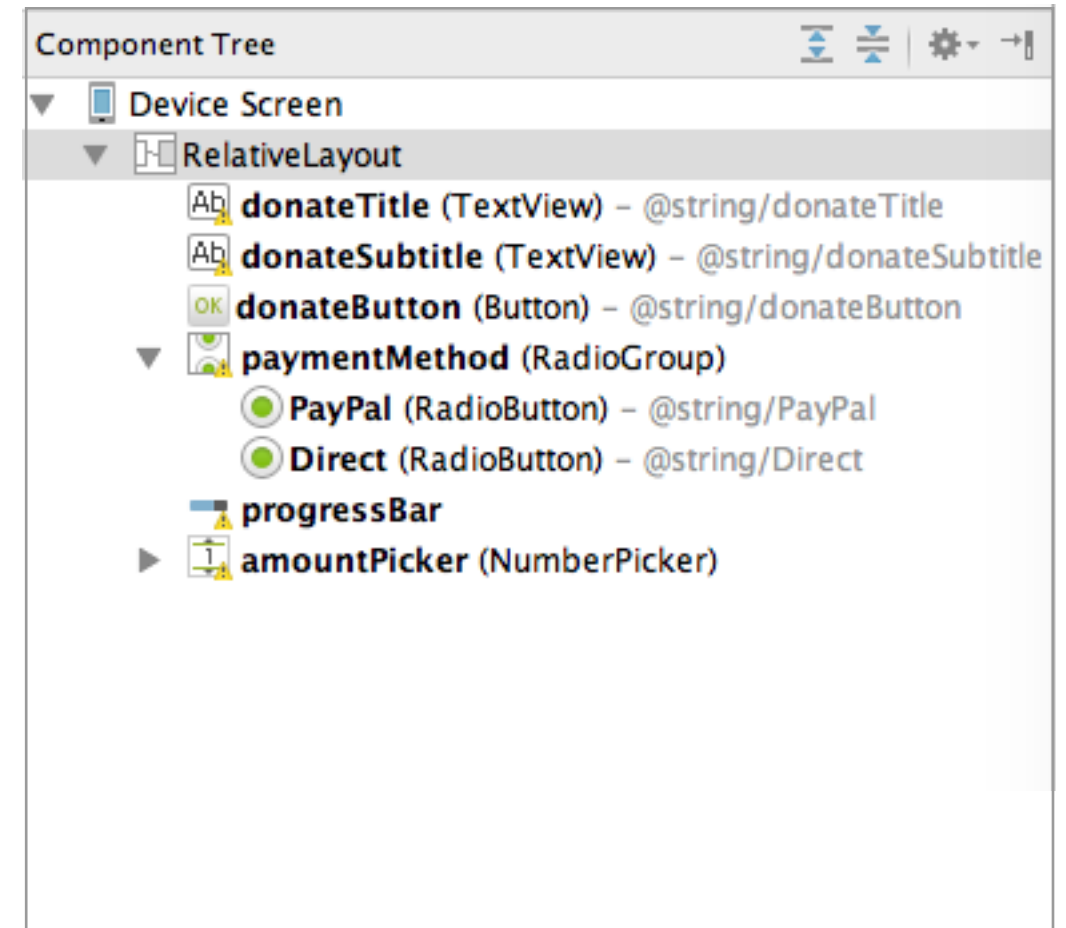
        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
        progressBar.setMax(10000);
    }

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

    public void donateButtonPressed (View view)
    {
        totalDonated = totalDonated + amountPicker.getValue();
        String method = paymentMethod.getCheckedRadioButtonId() == R.id.PayPal ? "PayPal" : "Direct";
        progressBar.setProgress(totalDonated);

        Log.v("Donate", amountPicker.getValue() + " donated by " + method + "\nCurrent total " + totalDonated);
    }
}

```





```

public class Donate extends Activity
{
    private int        totalDonated = 0;

    private RadioGroup  paymentMethod;
    private ProgressBar progressBar;
    private NumberPicker amountPicker;

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

        paymentMethod = (RadioGroup) findViewById(R.id.paymentMethod);
        progressBar = (ProgressBar) findViewById(R.id.progressBar);
        amountPicker = (NumberPicker) findViewById(R.id.amountPicker);

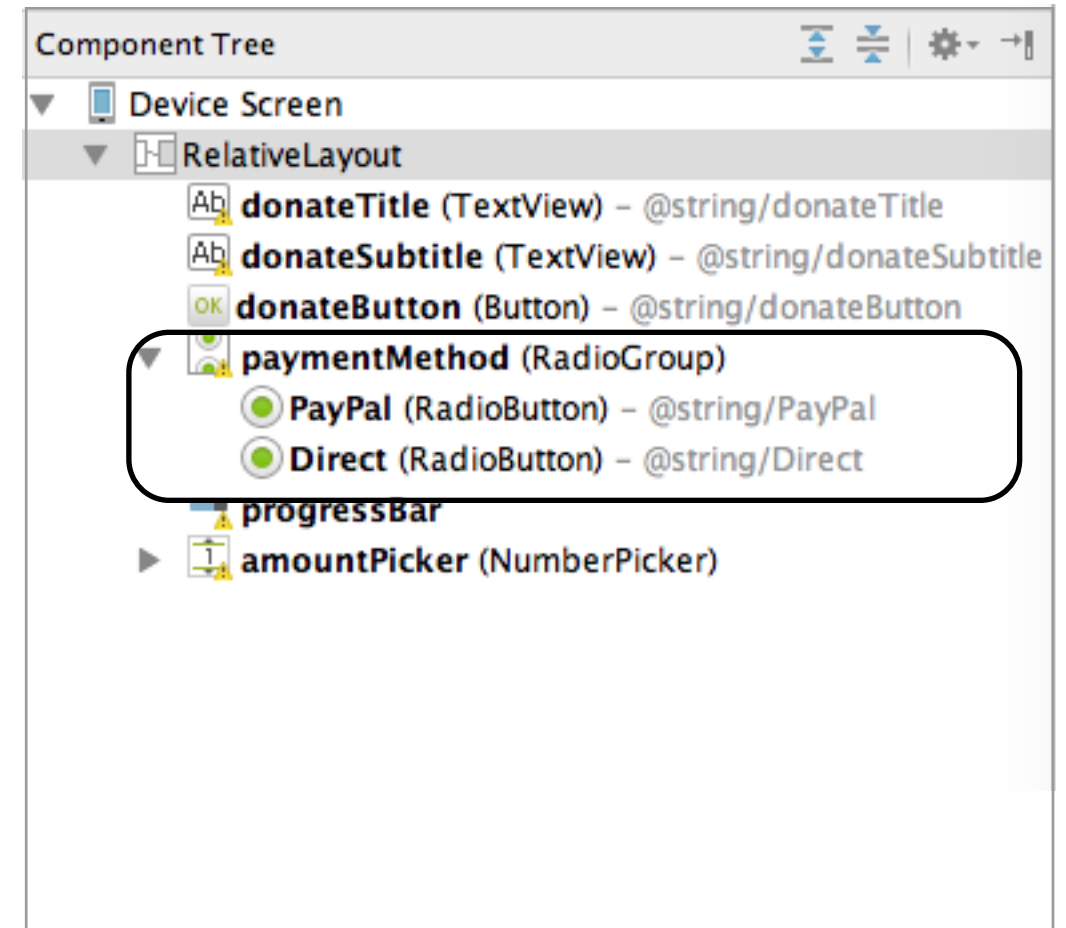
        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
        progressBar.setMax(10000);
    }

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

    public void donateButtonPressed (View view)
    {
        totalDonated = totalDonated + amountPicker.getValue();
        String method = paymentMethod.getCheckedRadioButtonId() == R.id.PayPal ? "PayPal" : "Direct";
        progressBar.setProgress(totalDonated);

        Log.v("Donate", amountPicker.getValue() + " donated by " + method + "\nCurrent total " + totalDonated);
    }
}

```



```

public class Donate extends Activity
{
    private int        totalDonated = 0;

    private RadioGroup  paymentMethod;
    private ProgressBar progressBar;
    private NumberPicker amountPicker;

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

        paymentMethod = (RadioGroup) findViewById(R.id.paymentMethod);
        progressBar   = (ProgressBar) findViewById(R.id.progressBar);
        amountPicker  = (NumberPicker) findViewById(R.id.amountPicker);

        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
        progressBar.setMax(10000);
    }

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

    public void donateButtonPressed (View view)
    {
        totalDonated = totalDonated + amountPicker.getValue();
        String method = paymentMethod.getCheckedRadioButtonId() == R.id.PayPal ? "PayPal" : "Direct";
        progressBar.setProgress(totalDonated);

        Log.v("Donate", amountPicker.getValue() + " donated by " + method + "\nCurrent total " + totalDonated);
    }
}

```



# 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 Listener Interface
  3. Using Anonymous Inner Class
- We need to master all three.

# Explicitly set in Resource File

---

```
<Button
    android:id="@+id/donateButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:onClick="donateButtonPressed"
    android:text="@string/donateButton" />
```

```
public class Donate extends Activity
{
    //...

    public void donateButtonPressed (View view)
    {
        totalDonated = totalDonated + amountPicker.getValue();
        String method = paymentMethod.getCheckedRadioButtonId() == R.id.PayPal ? "PayPal" : "Direct";
        progressBar.setProgress(totalDonated);

        Log.v("Donate", amountPicker.getValue() + " donated by " + method + "\nCurrent total " + totalDonated);
    }
}
```

# 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.

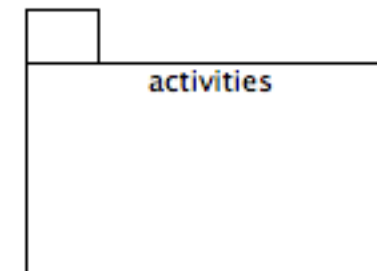
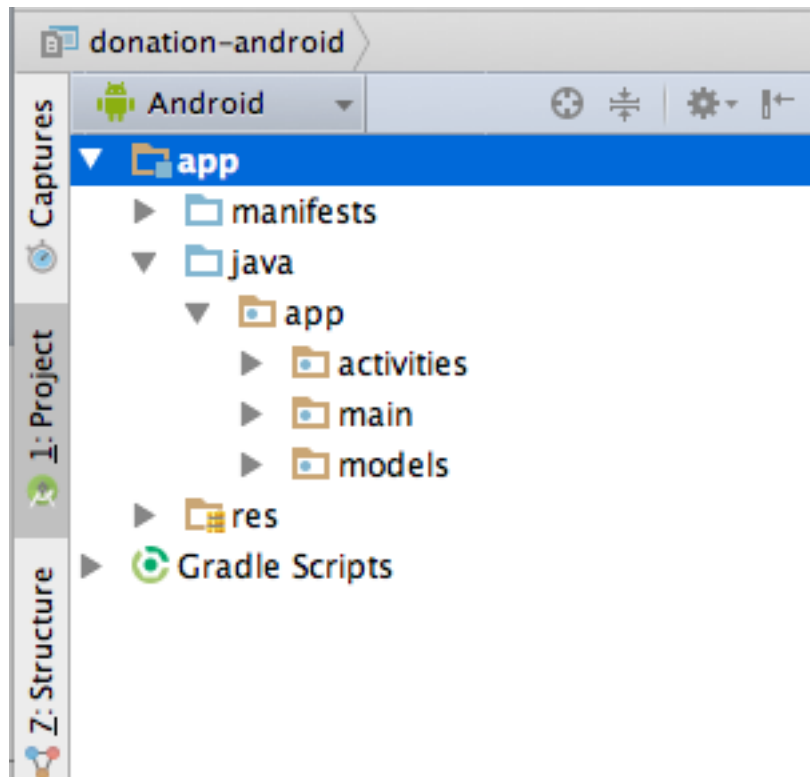
```
if (totalDonated > target)
{
    Toast toast = Toast.makeText(this, "Target Exceeded!", Toast.LENGTH_SHORT);
    toast.show();
    Log.v("Donate", "Target Exceeded: " + totalDonated);
}
```

Models



# Project Structure

---

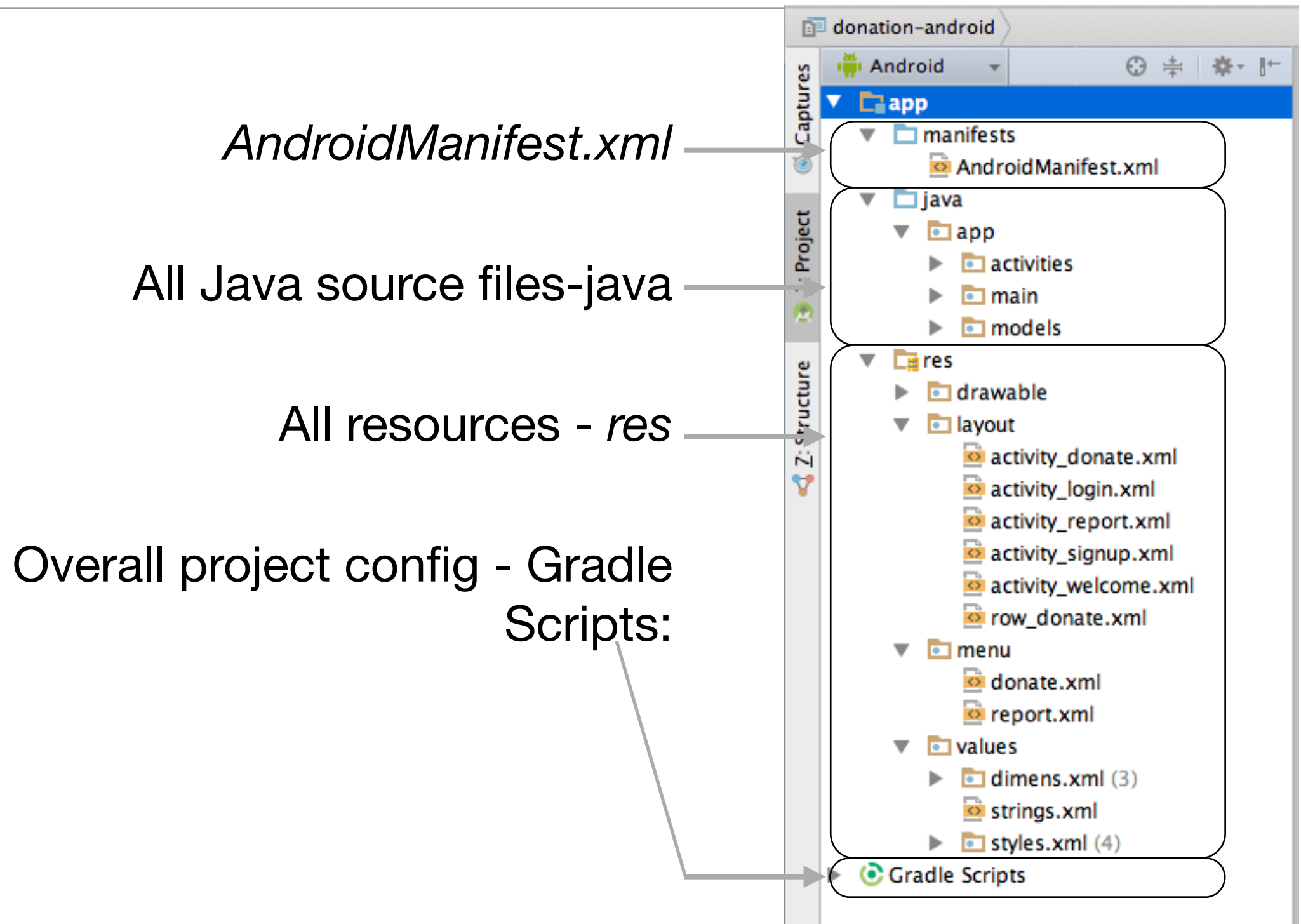


# High Level External Dependencies

---



# Project Structure - Detail



# Model?

```
public class Donate extends Activity
{
    private int          totalDonated = 0;
    private int          target = 10000;

    private RadioGroup   paymentMethod;
    private ProgressBar  progressBar;
    private NumberPicker amountPicker;
    private TextView     amountText;
    private TextView     amountTotal;

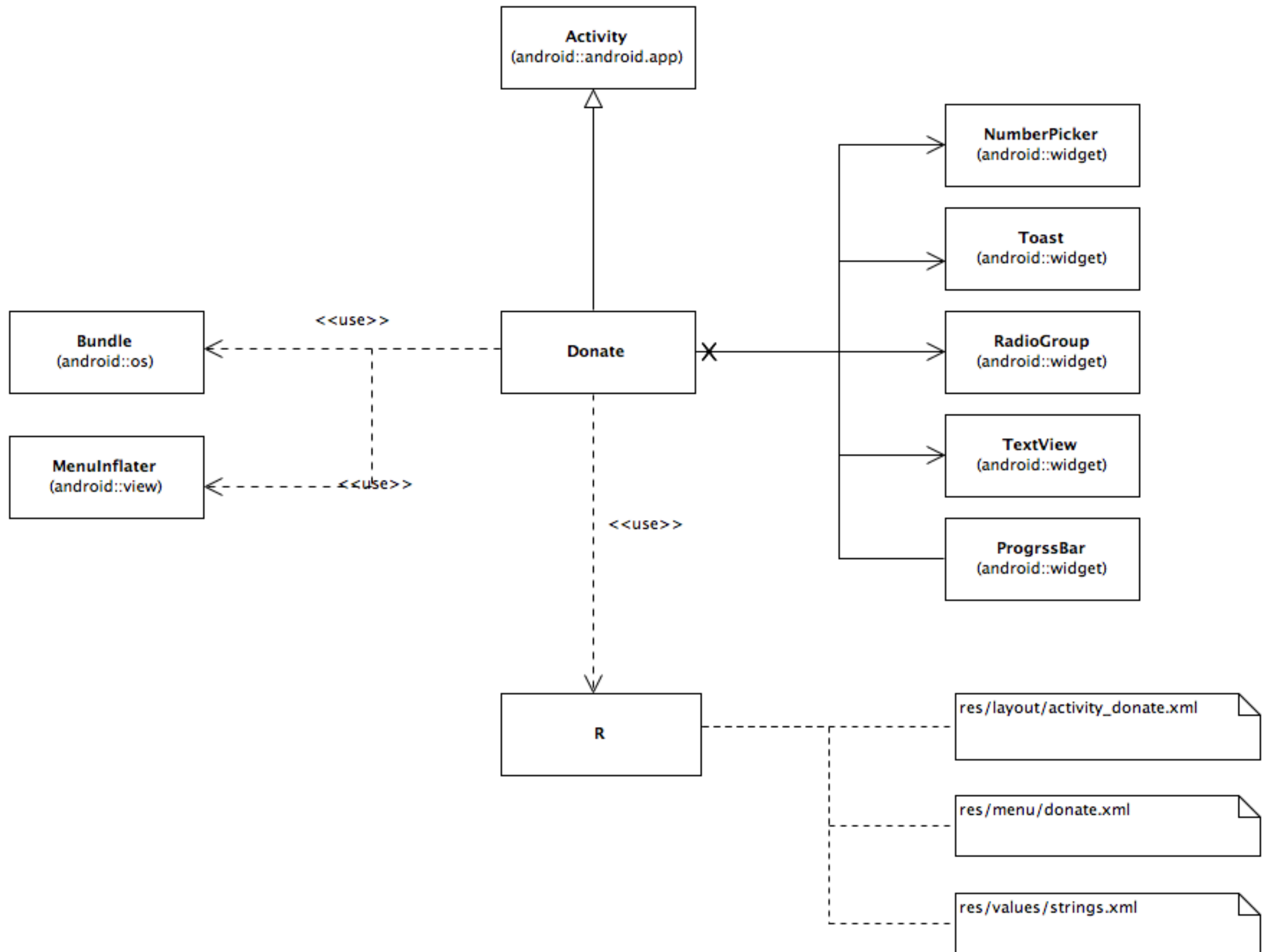
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_donate);

        ...
    }

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

    public void donateButtonPressed (View view)
    {
        ...
    }
}
```

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





Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>

