

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



Activities & App Navigation





Agenda & Goals

- ❑ Using **Menu** Navigation and **MenuInflators**
- ❑ Switching from one Activity to another using **Intents**
- ❑ Creating and using basic **ArrayAdapters** and **ListView**s

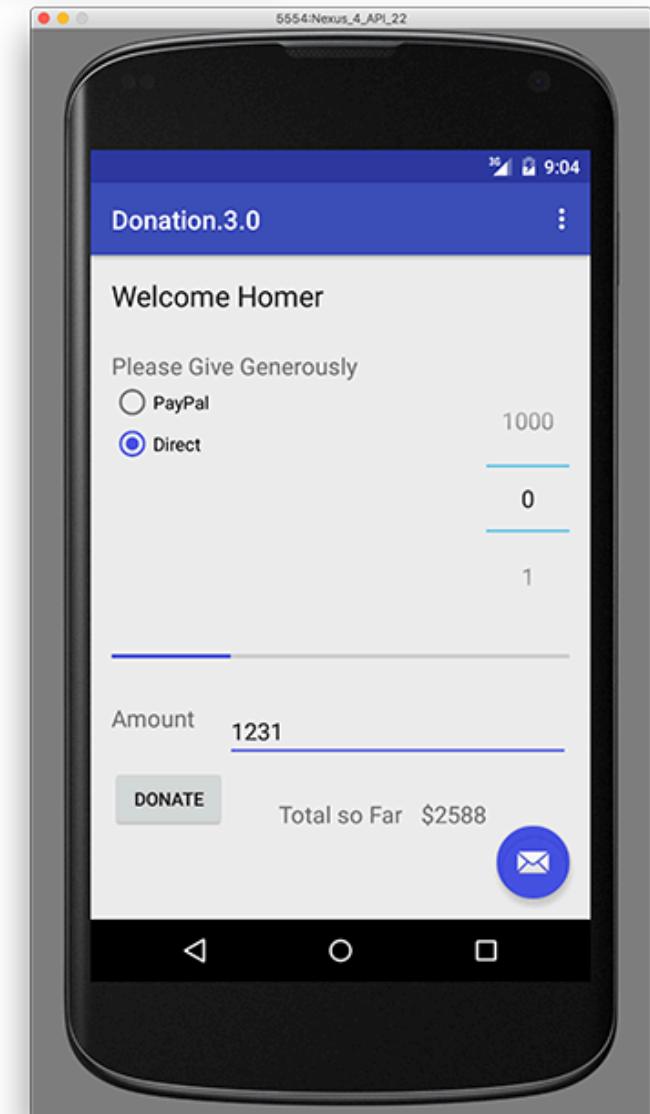


Case Study

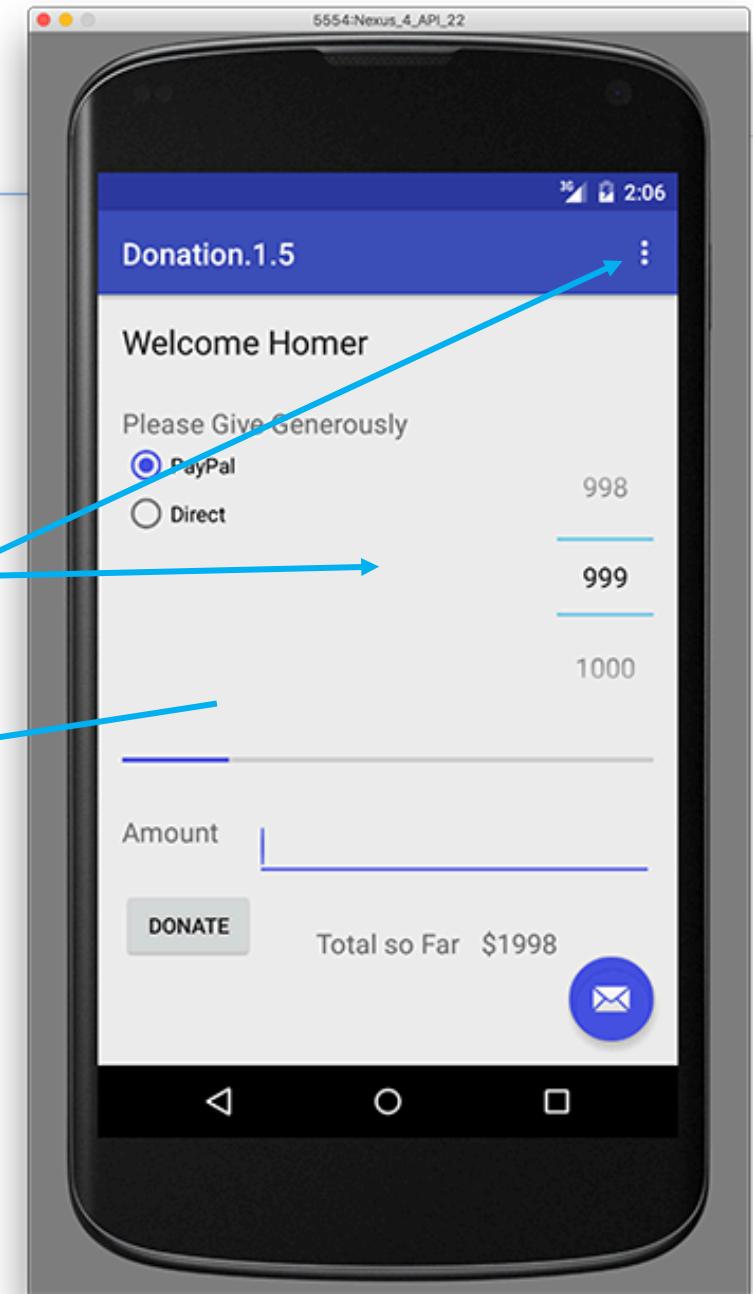
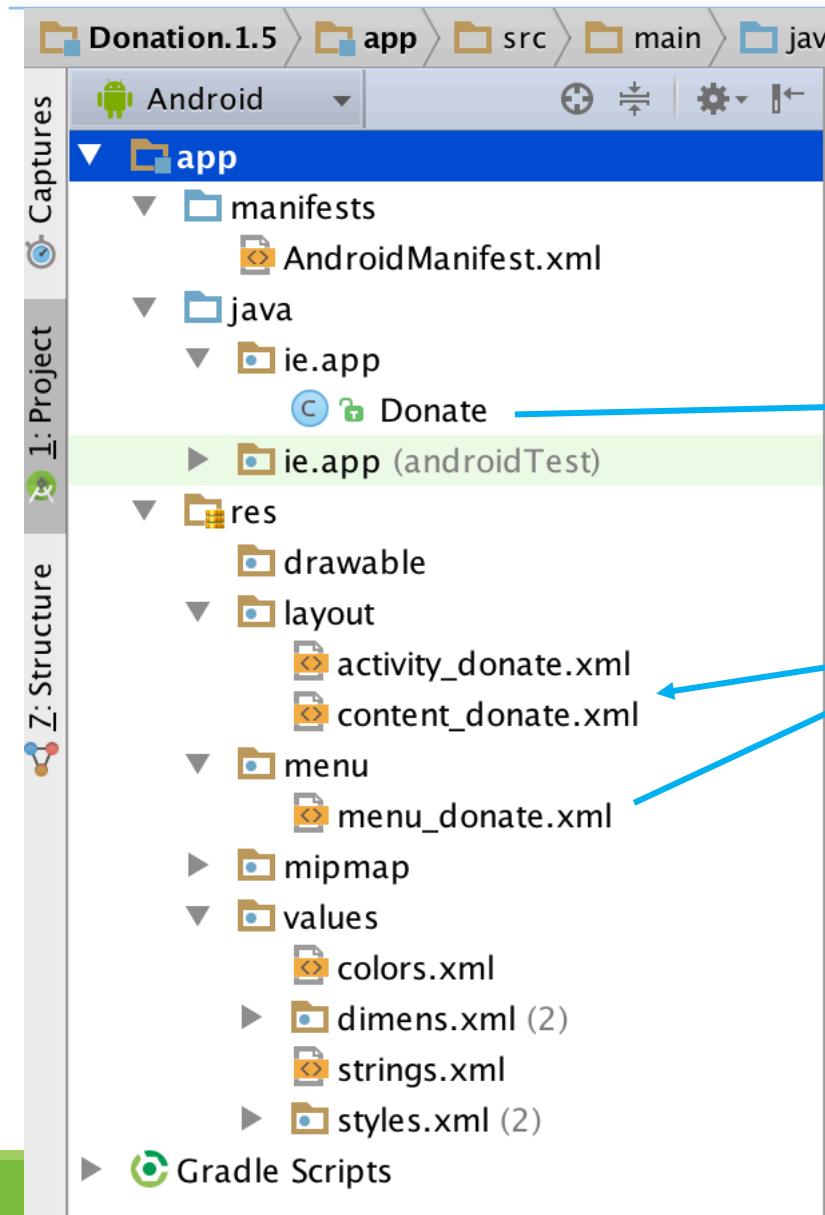
❑ **Donation** – an Android App to keep track of donations made to ‘*Homers Presidential Campaign*’.

❑ App Features

- Accept donation via number picker or typed amount
- Keep a running total of donations
- Display report on donation amounts and types
- Display running total on progress bar



Case Study so far – Donation.1.5 *



Case Study so far – Donation.1.5

```
public class Donate extends AppCompatActivity {

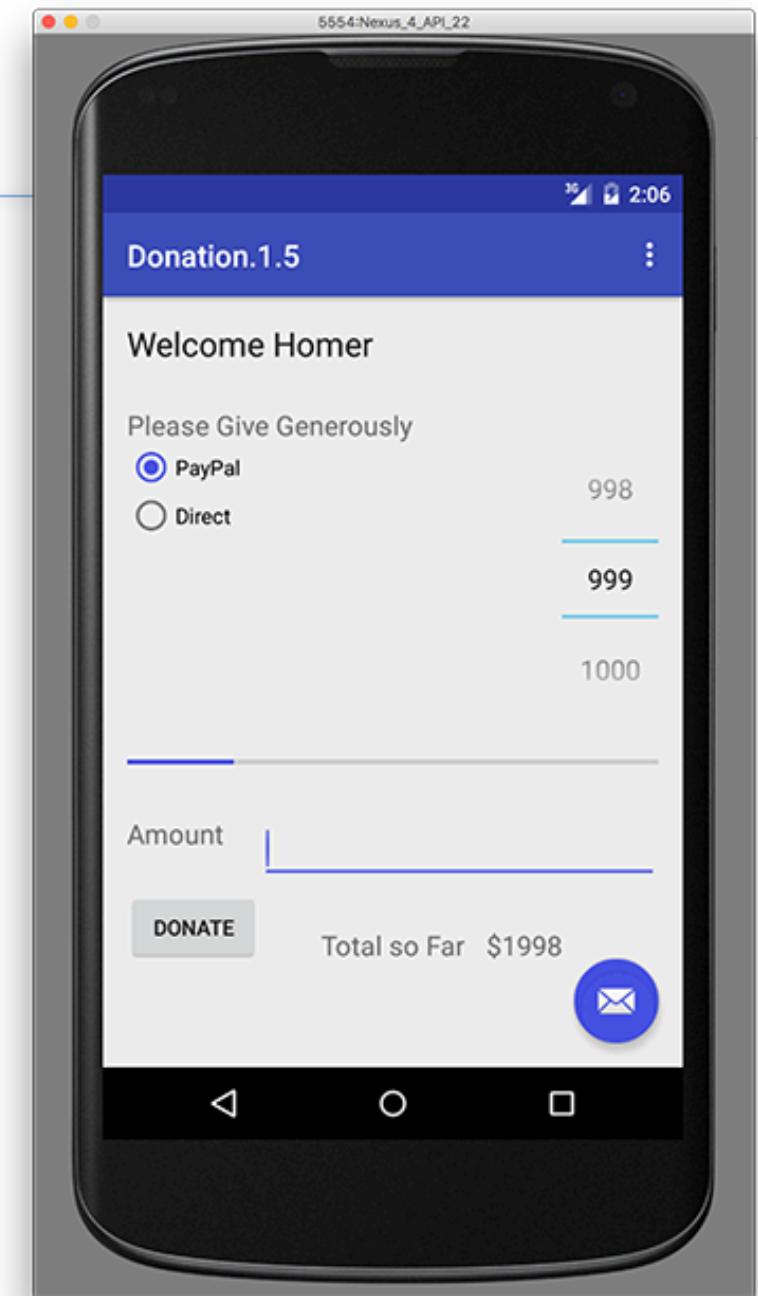
    private Button donateButton;
    private RadioGroup paymentMethod;
    private ProgressBar progressBar;
    private NumberPicker amountPicker;
    private EditText amountText;
    private TextView amountTotal;

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

    @Override
    protected void onCreate(Bundle savedInstanceState) {...}

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {...}
```





Donate button event handler

```
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);
    }

    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();
    }

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



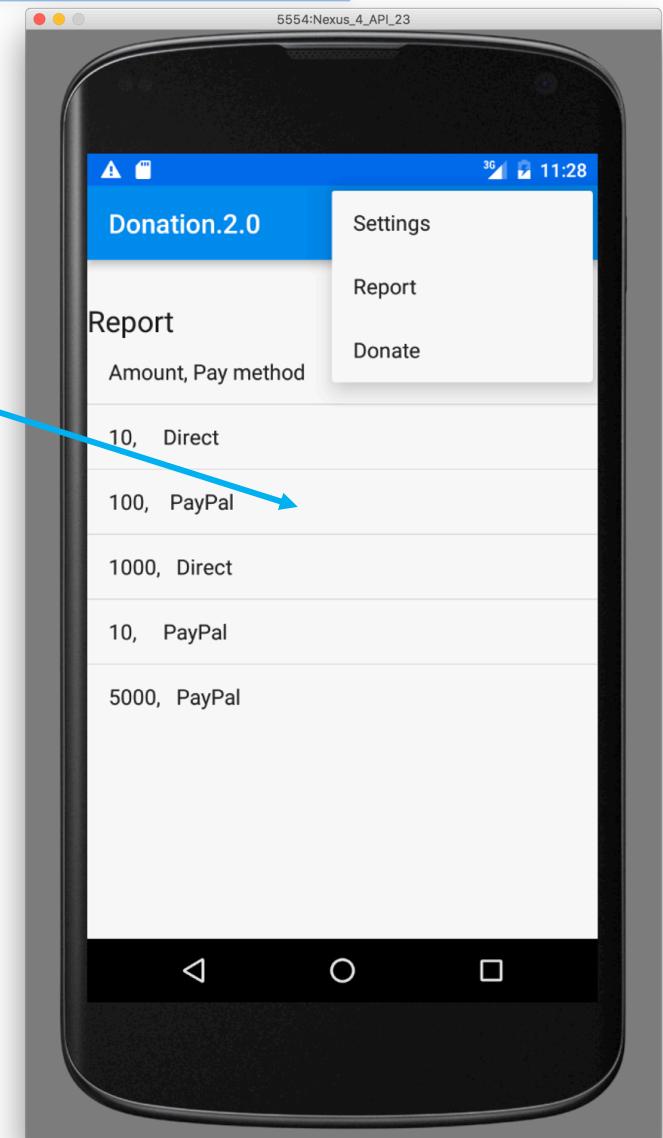
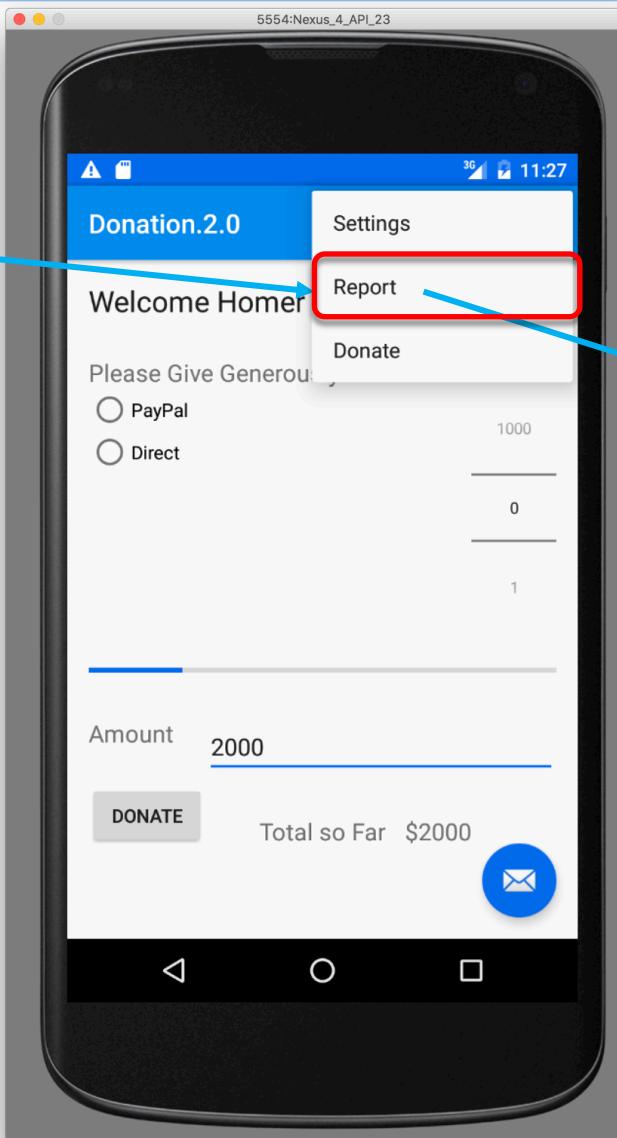
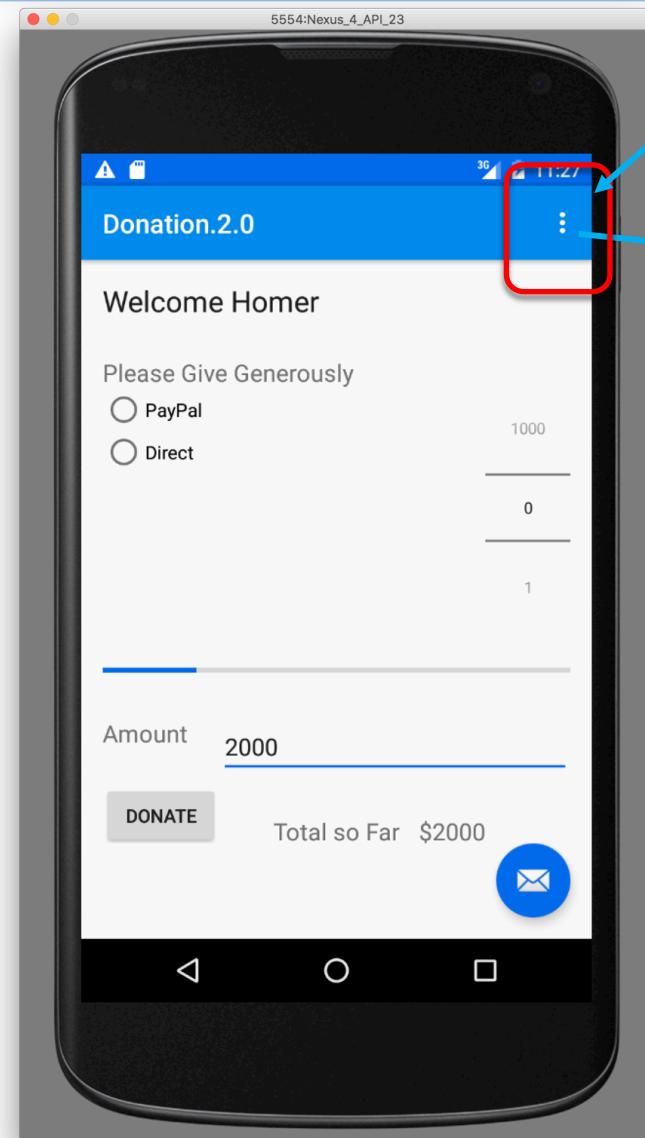
Donation.2.0

Using Menus & Intents



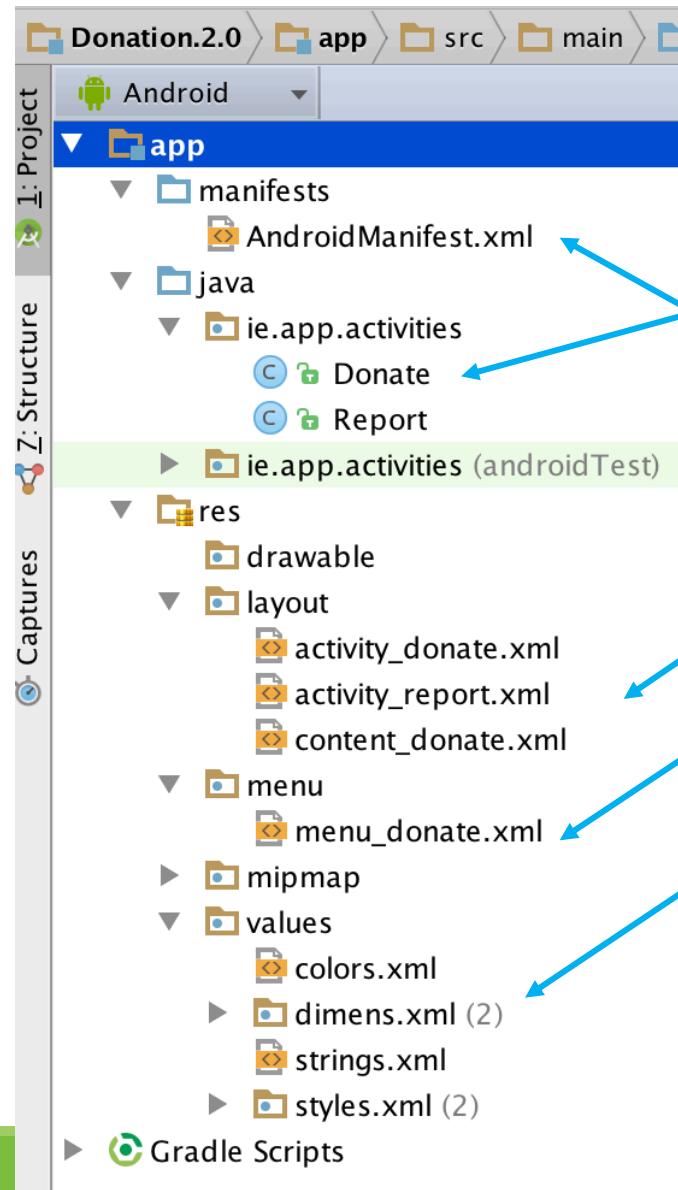
Donation 2.0

User Selects Menu





Donation 2.0 – Project Structure



- 2 java source files
- 3 xml layouts
- 1 xml menu
- 6 xml files for resources
- 1 xml ‘configuration’ file



Using Menus

Part 1



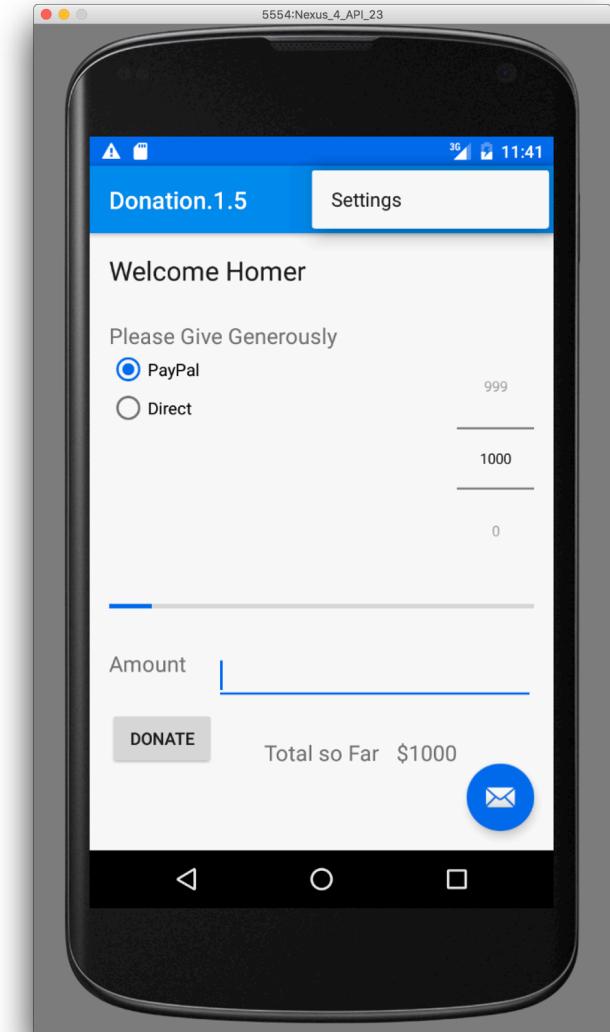
Menus

- ❑ Menus are a common user interface component in many types of applications.
- ❑ To provide a familiar and consistent user experience, you should use the [Menu](#) APIs to present user actions and other options in your activities.
- ❑ Beginning with Android 3.0 (API 11), Android-powered devices were no longer required to provide a dedicated **Menu** button
 - instead you provide an **action bar** to present common user actions.



Options Menu & Action Bar *

- ❑ The options menu is the primary collection of menu items for an activity.
 - It's where you should place actions that have a global impact on the app, such as "Donate", "Report" and "Settings" etc.
- ❑ If you're developing for Android 2.3 or lower, users can reveal the options menu panel by pressing the *Menu* button.
- ❑ On Android 3.0 and higher, items from the options menu are presented by the action bar as a combination of on-screen action items and overflow options.

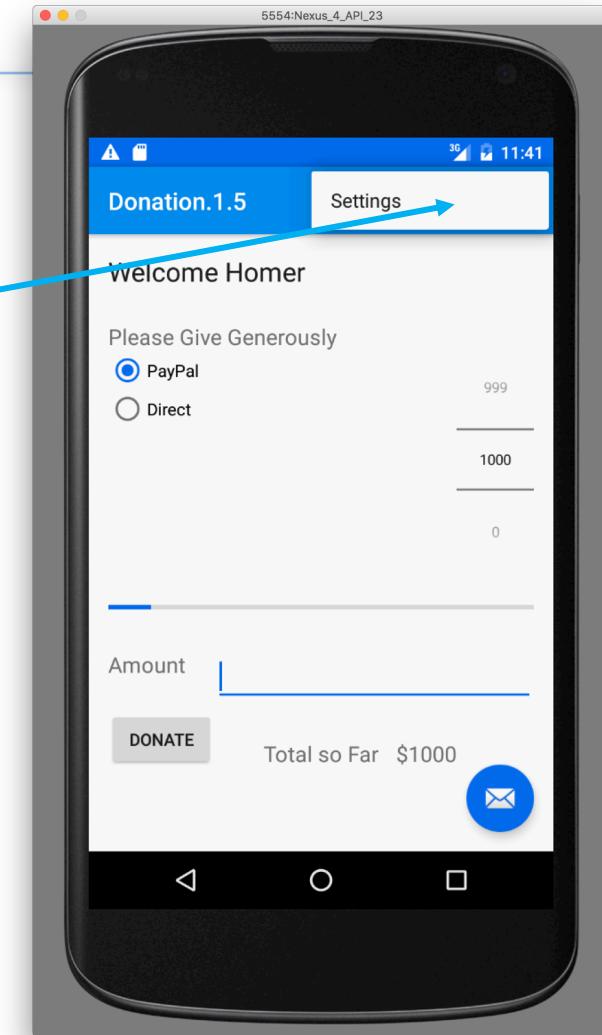
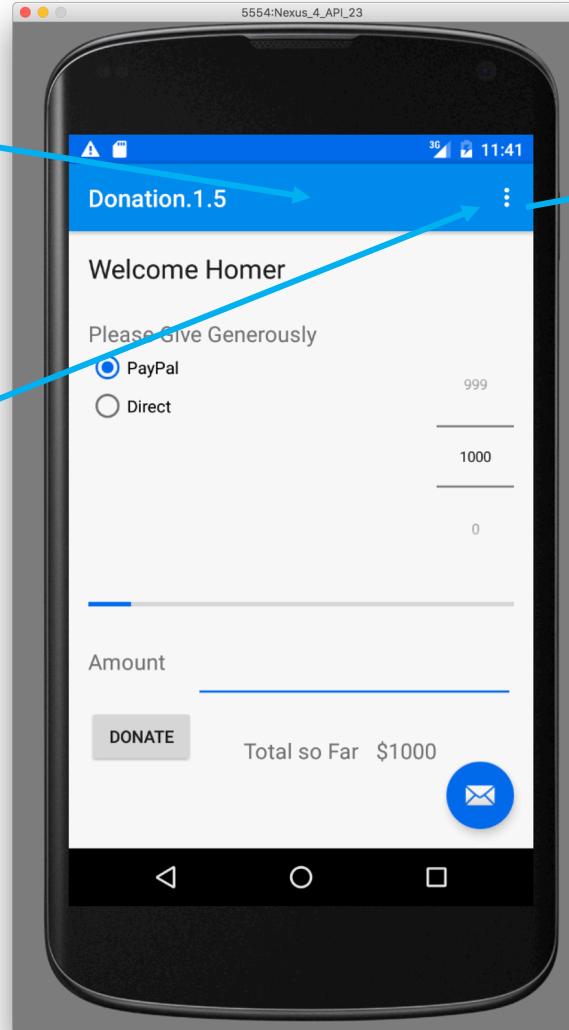




Menus in *Donation* (so far) *

Action Bar

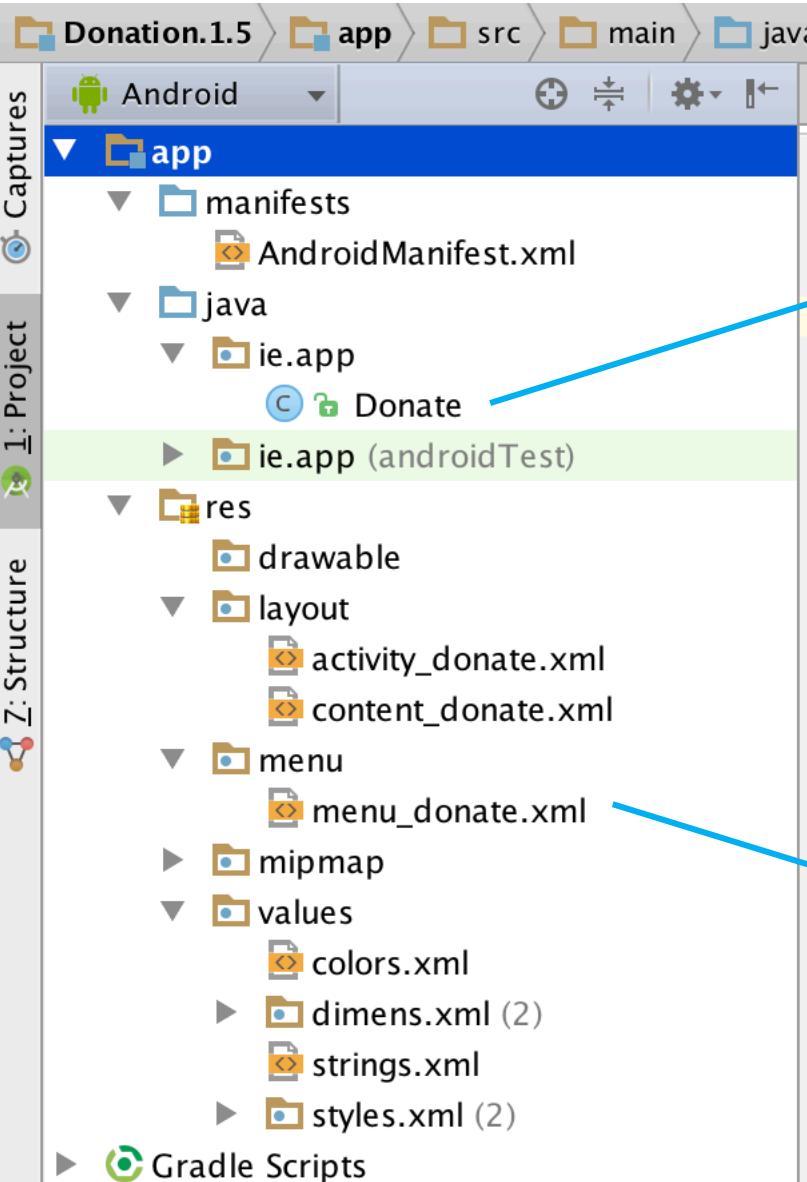
Pressing the “Menu” button on the emulator brings up a menu with the following entry





Menus in *Donation* (so far)

Menu Load



```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    // Inflate the menu; this adds items to the action bar if it is present.  
    getMenuInflater().inflate(R.menu.menu_donate, menu);  
    return true;  
}
```

inflate this resource as a 'Menu'
(creates the menu)

Menu Specification

```
<menu 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" tools:context=".Donate">  
  
    <item android:id="@+id/action_settings"  
          android:title="Settings"  
          android:orderInCategory="100"  
          app:showAsAction="never" />  
  
</menu>
```



Donation 1.5 Menu Event Handler

Check which ‘menu item’ was selected (by id)

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    // Handle action bar item clicks here. The action bar will  
    // automatically handle clicks on the Home/Up button, so long  
    // as you specify a parent activity in AndroidManifest.xml.  
    int id = item.getItemId();|  
  
    //noinspection SimplifiableIfStatement  
    if (id == R.id.action_settings) {  
        return true;  
    }  
  
    return super.onOptionsItemSelected(item);  
}
```



Donation 2.0 'New' Menu Item(s) *

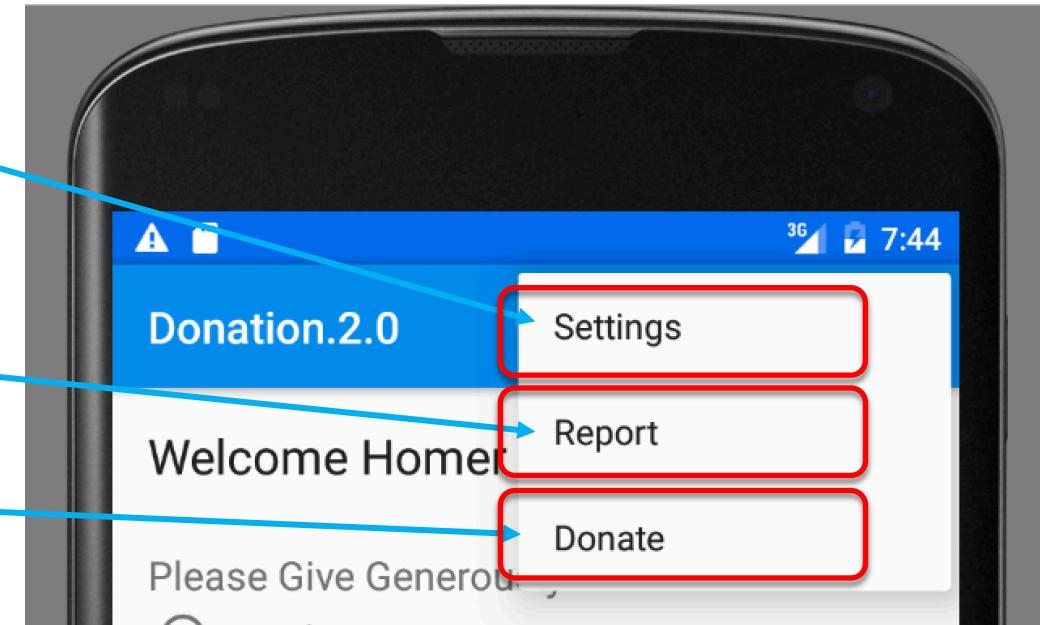
```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools" tools:context=".Donate">

    <item android:id="@+id/action_settings"
        android:title="Settings"
        android:orderInCategory="100"
        app:showAsAction="never" />

    <item
        android:id="@+id/menuReport"
        android:orderInCategory="100"
        android:title="Report"
        app:showAsAction="never"/>

    <item
        android:id="@+id/menuDonate"
        android:orderInCategory="100"
        android:title="Donate"
        app:showAsAction="never"/>

</menu>
```

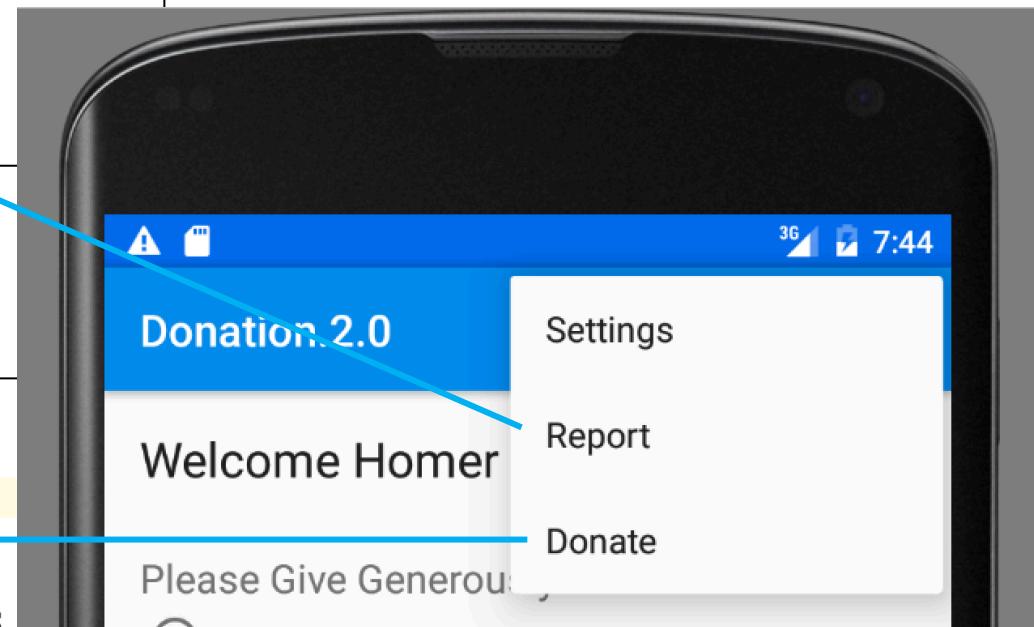




Donation 2.0 'New' Menu Event Handler(s)

```
@Override  
public boolean onOptionsItemSelected(MenuItem item)  
{  
    switch (item.getItemId())  
    {  
        case R.id.menuReport : startActivity (new Intent(this, Report.class));  
        break;  
    }  
    return super.onOptionsItemSelected(item);
```

Donate.java



```
@Override  
public boolean onOptionsItemSelected(MenuItem item)  
{  
    switch (item.getItemId())  
    {  
        case R.id.menuDonate : startActivity (new Intent(this, Donate.class));  
        break;  
    }  
    return super.onOptionsItemSelected(item);
```

Report.java



Using Intents



Recap on Android Components

- ❑ Three of the core components of an application — **activities**, **services**, and broadcast **receivers** — are activated through messages, called ***intents***
- ❑ In this section we'll look at using intents to start other activities, allowing us to 'switch between screens'
- ❑ The intent itself, an [Intent](#) object, is a passive data structure holding
 - an abstract description of an operation to be performed, or
 - a description of something that has happened and is being announced (broadcasts)



Switching Activities: General Approach

- ❑ Switch between Activities with Intents when
 - Main screen has buttons and/or menus to navigate to other Activities (your intent)
 - Return to original screen with “back” button (system intent)
- ❑ Syntax required to start new Activity
 - Java

```
Intent goToActivity = new Intent(this,OtherActivity.class);
startActivity(goToActivity);
```
 - XML
 - ◆ Requires an entry in `AndroidManifest.xml` (runtime error otherwise!)



Donation 2.0 - Report Layout *

The screenshot shows the Android Studio interface with the following components:

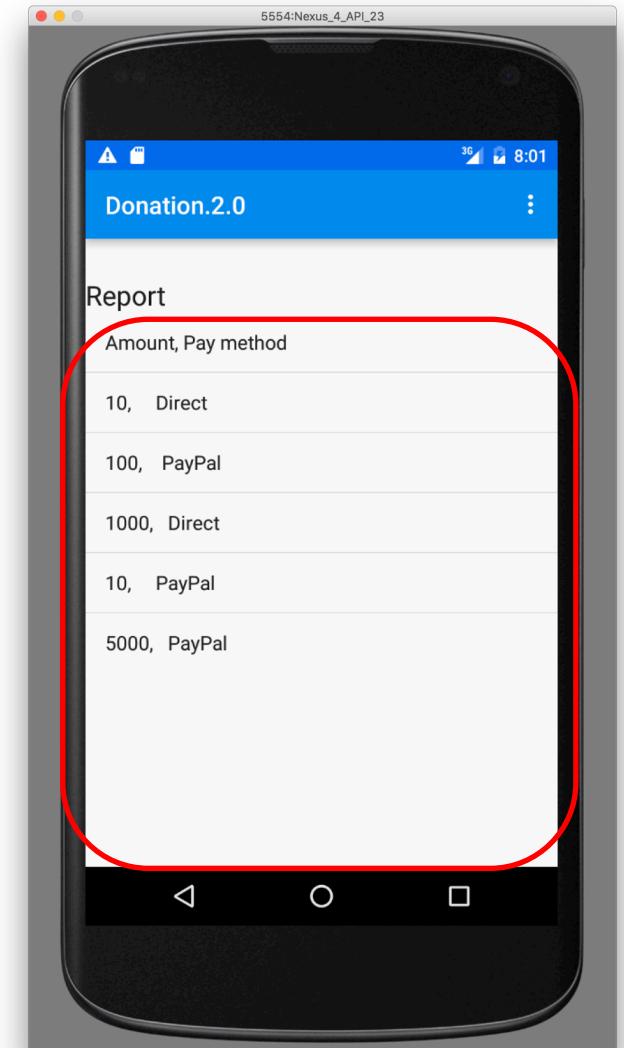
- Title Bar:** activity_report.xml x, Nexus 4, AppTheme, 23.
- Palette:** Layouts (FrameLayout, LinearLayout (Horizontal), LinearLayout (Vertical), TableLayout, TableRow, GridLayout, RelativeLayout) and Widgets (Plain TextView, Large Text, Medium Text, Small Text, Button, Small Button, RadioButton, CheckBox, Switch, ToggleButton, ImageButton, ImageView, ProgressBar (Large), ProgressBar (Normal), ProgressBar (Small), ProgressBar (Horizontal), SeekBar, RatingBar, Spinner, WebView).
- Preview:** A smartphone screen displaying the "Report" layout. The layout consists of a title bar with "Donation.2.0" and a list view below it. The list view contains 7 items, each with a main item and a sub-item. A red box highlights the entire list view area.
- Component Tree:** Device Screen > RelativeLayout containing reportTitle (TextView) and reportList (ListView). The reportList item is highlighted with a red box.
- Properties:** A list of XML attributes for the selected ListView component, including layout:width, layout:height, style, accessibilityLiveRegion, accessibilityTraversalAfter, accessibilityTraversalBefore, alpha, background, backgroundTint, backgroundTintMode, clickable, contentDescription, contextClickable, elevation, and f...

We will cover
ListViews in more
detail later on



Donation 2.0 – activity_report.xml *

```
activity_report.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical" android:layout_width="match_parent"
4      android:layout_height="match_parent">
5
6      <TextView
7          android:layout_width="wrap_content"
8          android:layout_height="wrap_content"
9          android:textAppearance="?android:attr/textAppearanceLarge"
10         android:text="Report"
11         android:id="@+id/reportTitle"
12         android:layout_marginLeft="0dp"
13         android:layout_marginTop="31dp"
14         android:layout_alignParentTop="true"
15         android:layout_alignParentStart="true"
16         android:layout_alignParentEnd="true" />
17
18      <ListView
19          android:layout_width="wrap_content"
20          android:layout_height="wrap_content"
21          android:id="@+id/reportList"
22          android:layout_below="@+id/reportTitle"
23          android:layout_alignParentStart="true" />
24  </RelativeLayout>
```





Donation 2.0 - Report Activity *

```
public class Report extends AppCompatActivity
{
    ListView listView;

    static final String[] numbers = new String[] {
        "Amount, Pay method",
        "10, Direct",
        "100, PayPal",
        "1000, Direct",
        "10, PayPal",
        "5000, PayPal"};
}

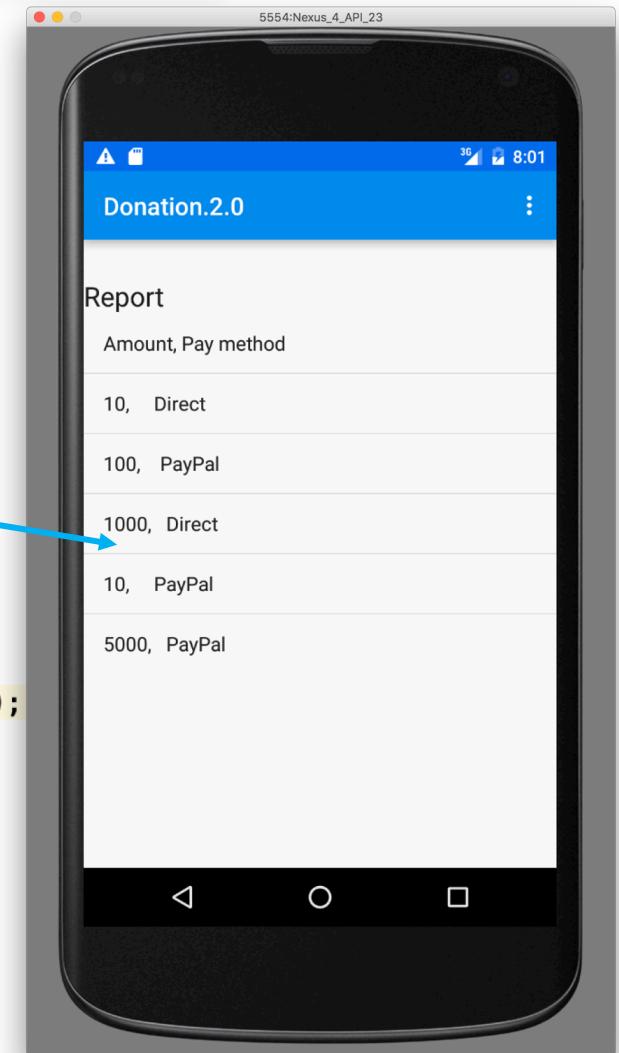
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_report);

    listView = (ListView) findViewById(R.id.reportList);
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, numbers);
    listView.setAdapter(adapter);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {...}

@Override
public boolean onOptionsItemSelected(MenuItem item)
{...}
}
```

Initially, Hardcoded values





Donation 2.0 - Report Activity

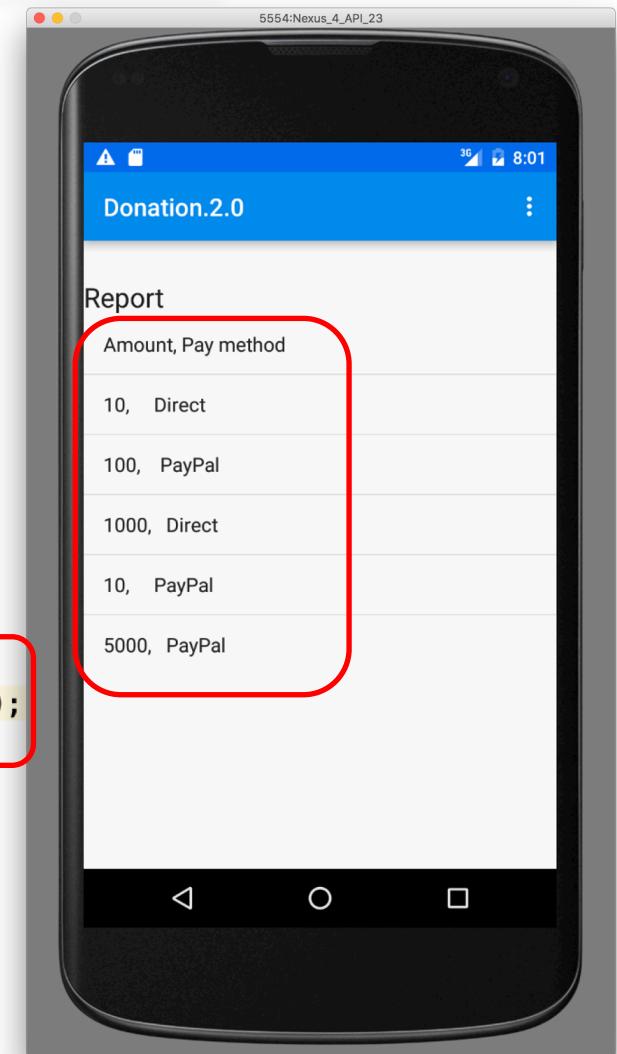
```
public class Report extends AppCompatActivity
{
    ListView listView;

    static final String[] numbers = new String[] {
        "Amount, Pay method",
        "10, Direct",
        "100, PayPal",
        "1000, Direct",
        "10, PayPal",
        "5000, PayPal"};
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_report);

        listView = (ListView) findViewById(R.id.reportList);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, numbers);
        listView.setAdapter(adapter);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {...}

    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {...}
}
```

Using an Adapter to
populate the ListView
(more on this in next section)





Filling an Adapter View with Data

- You can populate an `AdapterView` such as `ListView` or `GridView` by binding the `AdapterView` instance to an Adapter, which retrieves data from an external source and creates a View that represents each data entry.

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
                                         android.R.layout.simple_list_item_1,  
                                         numbers);
```

- The arguments for this constructor are:

- Your app `Context`
- The layout that contains a `TextView` for each string in the array
- The string array (`numbers`)

- Then simply call `setAdapter()` on your `ListView`:

```
listView = (ListView) findViewById(R.id.reportList);  
listView.setAdapter(adapter);
```

Donation.2.0



Summary

- ❑ We investigated using **Menus** for Navigation
- ❑ We saw how to Switch from one Activity to another using **Intents**
- ❑ And we had a quick look at creating and using basic **ArrayAdapters** and **ListViews** to display data in a list



Questions?



Thanks!

