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

Android Persistence

Agenda & Goals

- ❑ Be aware of the different approaches to data persistence in Android Development
- ❑ Be able to work with the **`SQLiteOpenHelper`** & **`SQLiteDatabase`** classes to implement an SQLite database on an Android device (to manage our Donations)
- ❑ Be able to work with **`SharedPreferences`** to manage our Login & Register screens

Main Idea – why do we need Persistence?

- Android can shut down and restart your app
  - When you rotate the screen
  - When you change languages
  - When your app is in background and Android is short on memory
  - When you hit the Back button
- Problem
  - You risk losing user changes and data
- Solutions ??

Solutions

- ❑ Android provides several options for you to save persistent application data.
- ❑ The solution you choose depends on your specific needs, such as whether the data should be private to your application or accessible to other applications (and the user) and how much space your data requires.
- ❑ Android provides a way for you to expose your private data to other applications — with a `Content Provider`.
  - A content provider is an optional component that exposes read/write access to your application data, subject to whatever restrictions you want to impose.

Data Storage Solutions *

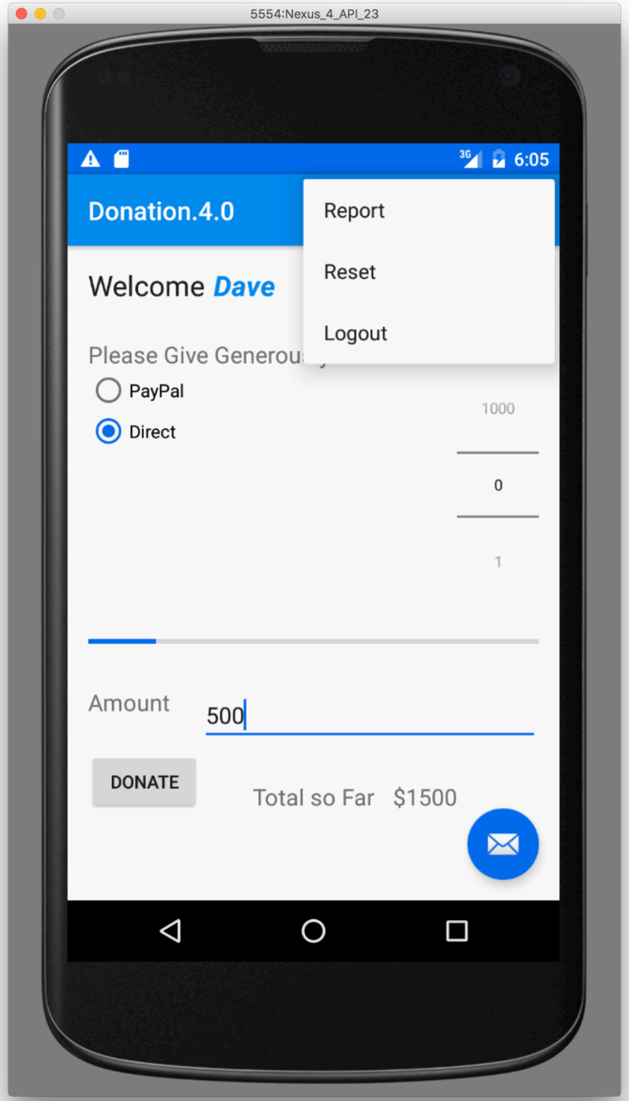
- Shared Preferences
  - Store private primitive data in key-value pairs.
- Internal Storage
  - Store private data on the device memory.
- External Storage
  - Store public data on the shared external storage.
- SQLite Databases
  - Store structured data in a private database.
- Network Connection
  - Store data on the web with your own network server.

Data Storage Solutions *

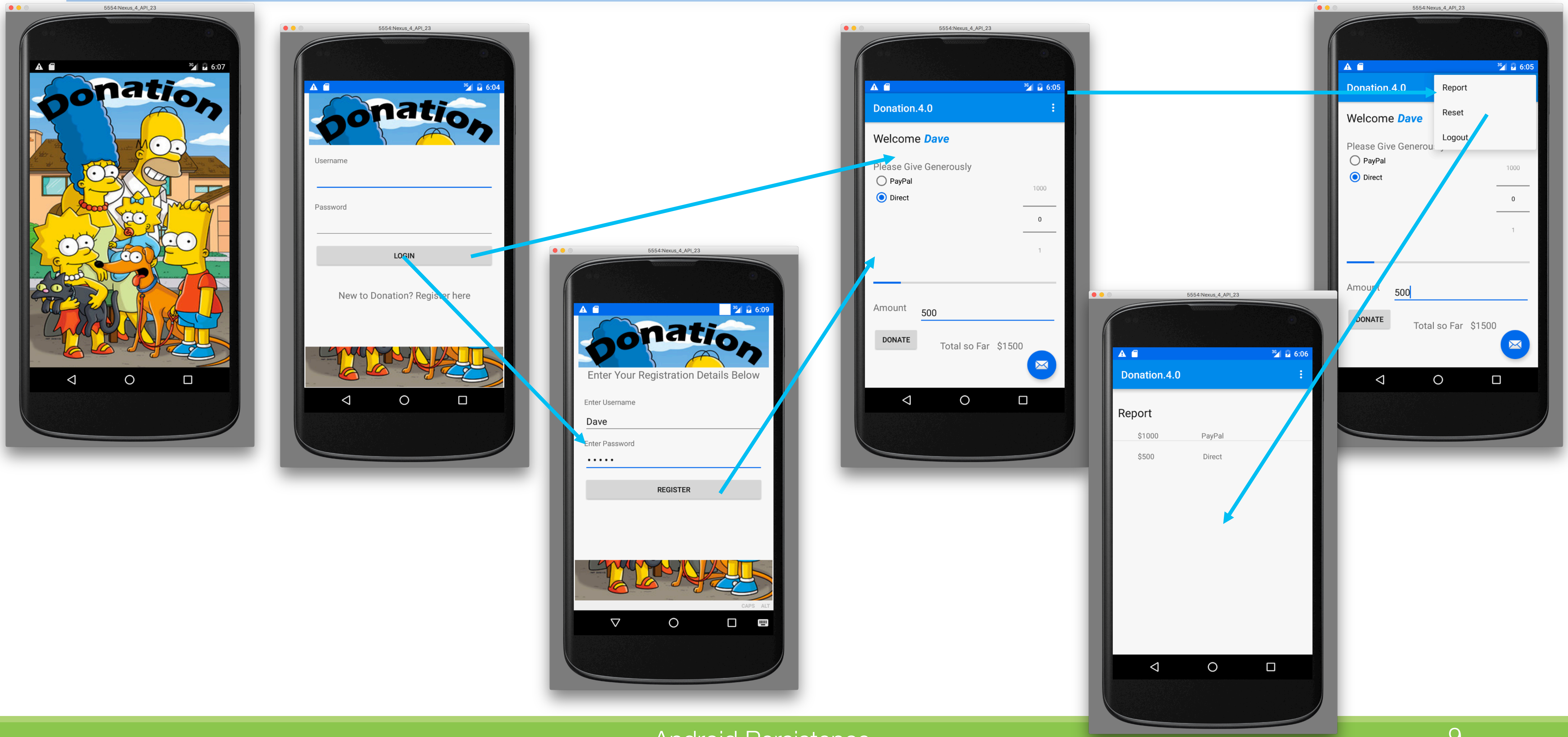
- Bundle Class
  - A mapping from String values to various `Parcelable` types and functionally equivalent to a standard `Map`.
  - Does not handle Back button scenario. App restarts from scratch with no saved data in that case.
- File
  - Use `java.io.*` to read/write data on the device's internal storage.

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



Ultimate Case Study – Donation 4.0



Enter Username: Dave

Enter Password: •••••

Amount: 500

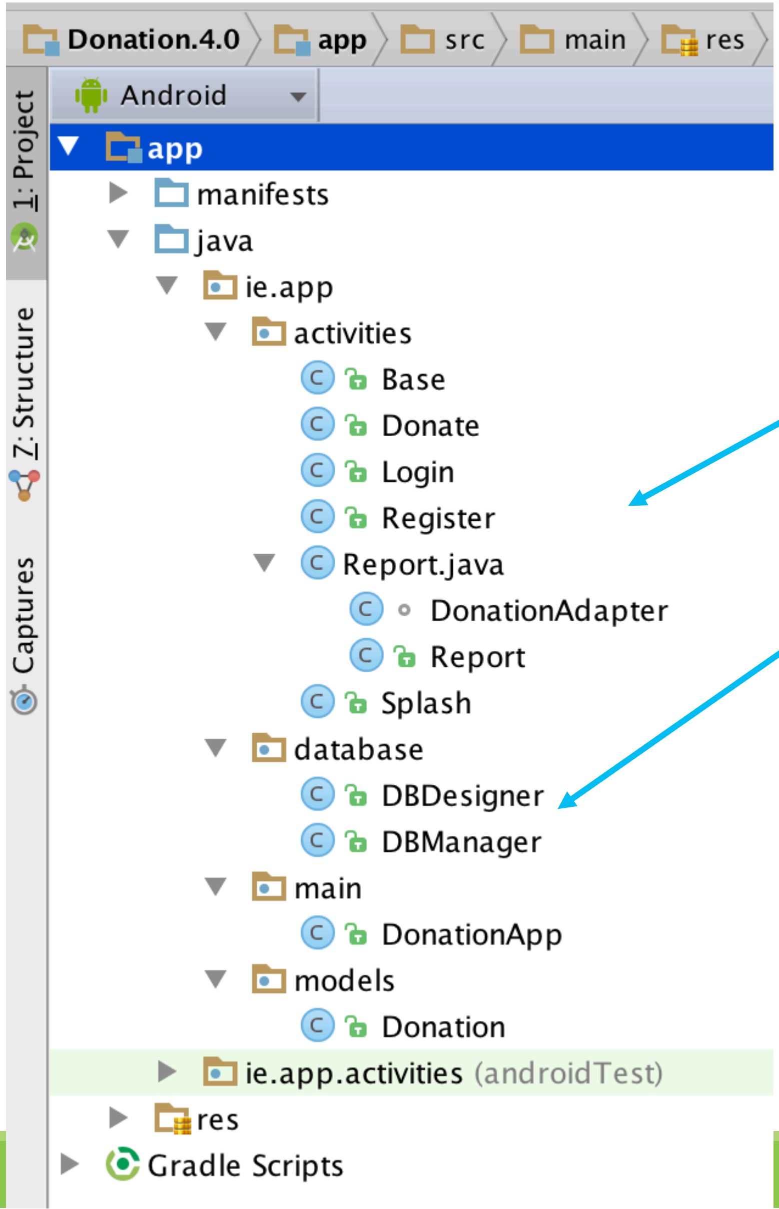
Report

$1000	PayPal
$500	Direct

Donation.4.0

Using an SQLite Database

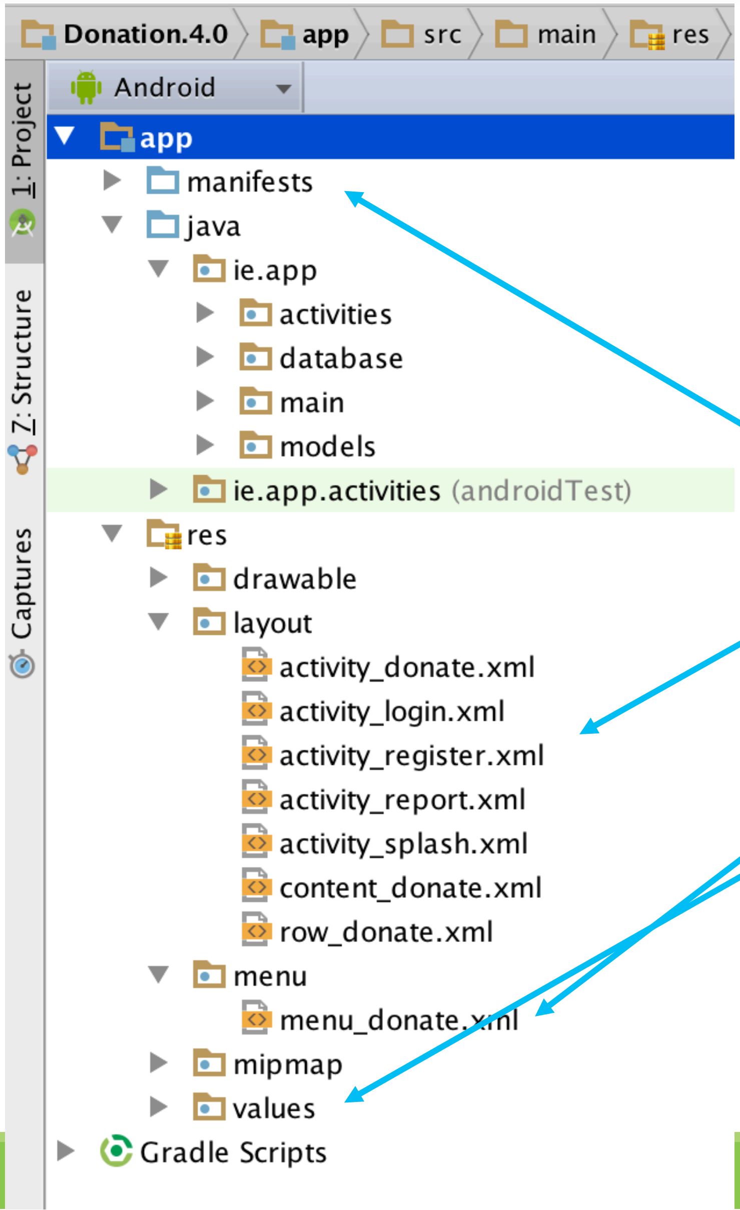
Donation 4.0 – Project Structure (Classes)



```
▼ app
  ▶ manifests
  ▼ java
    ▼ ie.app
      ▼ activities
          Base
          Donate
          Login
          Register
        ▼ Report.java
            DonationAdapter
            Report
          Splash
      ▼ database
          DBDesigner
          DBManager
      ▼ main
          DonationApp
      ▼ models
          Donation
    ▶ ie.app.activities (androidTest)
  ▶ res
▶ Gradle Scripts
```

- 9 java source files
  - Our Database classes
- 3 xml layouts
- 1 xml menu
- 6 xml files for resources
- 1 xml ‘configuration’ file

Donation 4.0 – Project Structure (Resources)



Donation.4.0 › app › src › main › res

Android

```
▼ app
  ▶ manifests
  ▼ java
    ▼ ie.app
      ▶ activities
      ▶ database
      ▶ main
      ▶ models
    ▶ ie.app.activities (androidTest)
  ▼ res
    ▶ drawable
    ▼ layout
        activity_donate.xml
        activity_login.xml
        activity_register.xml
        activity_report.xml
        activity_splash.xml
        content_donate.xml
        row_donate.xml
    ▼ menu
        menu_donate.xml
    ▶ mipmap
    ▶ values
▶ Gradle Scripts
```

- 9 java source files
  - Our Database classes
- 7 xml layouts
- 1 xml menu
- 6 xml files for resources
- 1 xml 'configuration' file

Idea

- Goal
  - Enhance **Donation.3.0** by managing the Donations in an SQLite Database.
- Approach
  - Implement/extend specific classes to add the database functionality to the app.

Database Programming in Android *

- ❑ Android provides full support for `SQLite` databases. Any databases you create will be accessible by name to any class in the application, but not outside the application.
- ❑ The recommended method to create a new SQLite database is to create a subclass of `SQLiteOpenHelper` and override the `onCreate()` method, in which you can execute a SQLite command to create tables in the database. For example:

```java
public class DictionaryOpenHelper extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 2;
    private static final String DICTIONARY_TABLE_NAME = "dictionary";
    private static final String DICTIONARY_TABLE_CREATE =
                "CREATE TABLE " + DICTIONARY_TABLE_NAME + " (" +
                KEY_WORD + " TEXT, " +
                KEY_DEFINITION + " TEXT);";

    DictionaryOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DICTIONARY_TABLE_CREATE);
    }
}
```

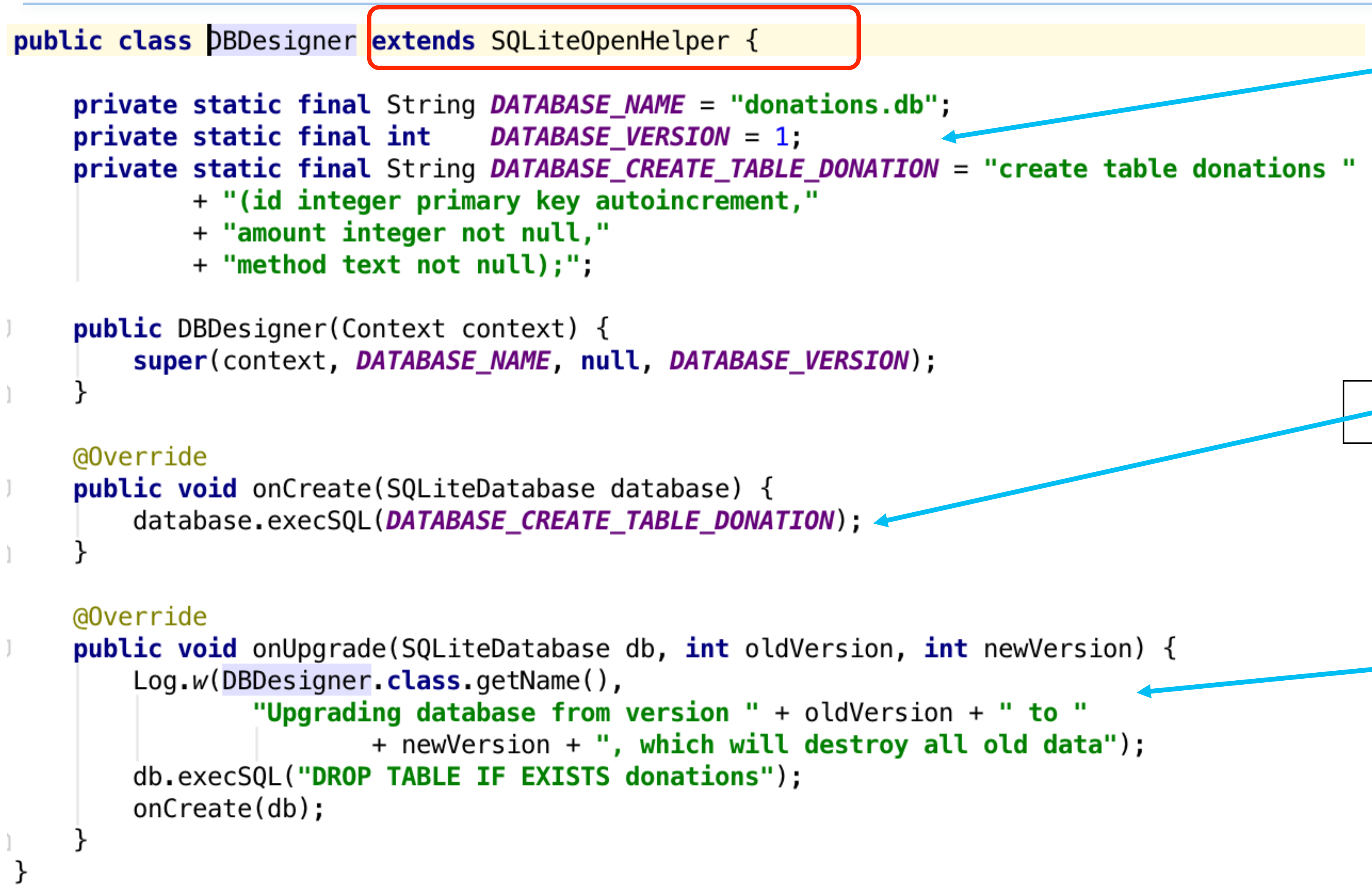
Database Programming in Android *

- You can then get an instance of your `SQLiteOpenHelper` implementation using the constructor you've defined. To write to and read from the database, call `getWritableDatabase()` and `getReadableDatabase()`, respectively. These both return a `SQLiteDatabase` object that represents the database and provides methods for `SQLite` operations.
- You can execute `SQLite` queries using the `SQLiteDatabase` `query()` methods, which accept various query parameters, such as the table to query, the projection, selection, columns, grouping, and others. For complex queries, such as those that require column aliases, you should use `SQLiteQueryBuilder`, which provides several convenient methods for building queries.
- Every SQLite query will return a `Cursor` that points to all the rows found by the query. The Cursor is always the mechanism with which you can navigate results from a database query and read rows and columns.

Database Programming in Android

- With `SQLite`, the database is a simple disk file. All of the data structures making up a relational database - tables, views, indexes, etc. - are within this file
- RDBMS is provided through the api classes so it becomes part of your app
- You can use the SQL you learned in a database module
- You should use DB best practices
  - Normalize data
  - Encapsulate database info in helper or wrapper classes
  - Don’t store files (e.g. images or audio), Instead just store the path string

Donation – DBDesigner *



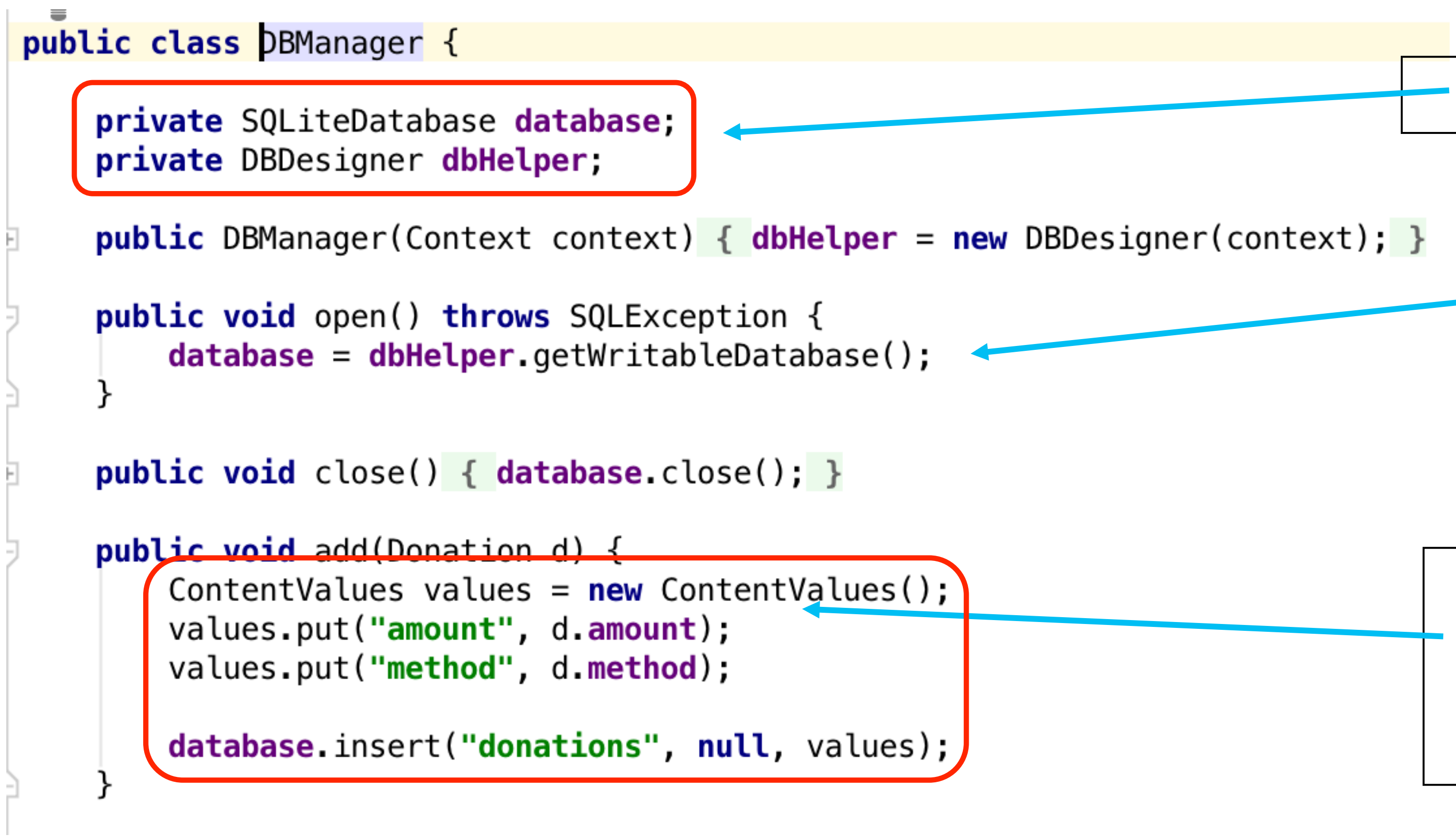
```java
public class DBDesigner extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "donations.db";
    private static final int    DATABASE_VERSION = 1;
    private static final String DATABASE_CREATE_TABLE_DONATION = "create table donations "
            + "(id integer primary key autoincrement,"
            + "amount integer not null,"
            + "method text not null);";

    public DBDesigner(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase database) {
        database.execSQL(DATABASE_CREATE_TABLE_DONATION);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        Log.w(DBDesigner.class.getName(),
                "Upgrading database from version " + oldVersion + " to "
                        + newVersion + ", which will destroy all old data");
        db.execSQL("DROP TABLE IF EXISTS donations");
        onCreate(db);
    }
}
```

Our Table & Column names (SQL)

Creating the Table (or Tables)

Drop the Table (if we change the schema)

Donation – DBManager *



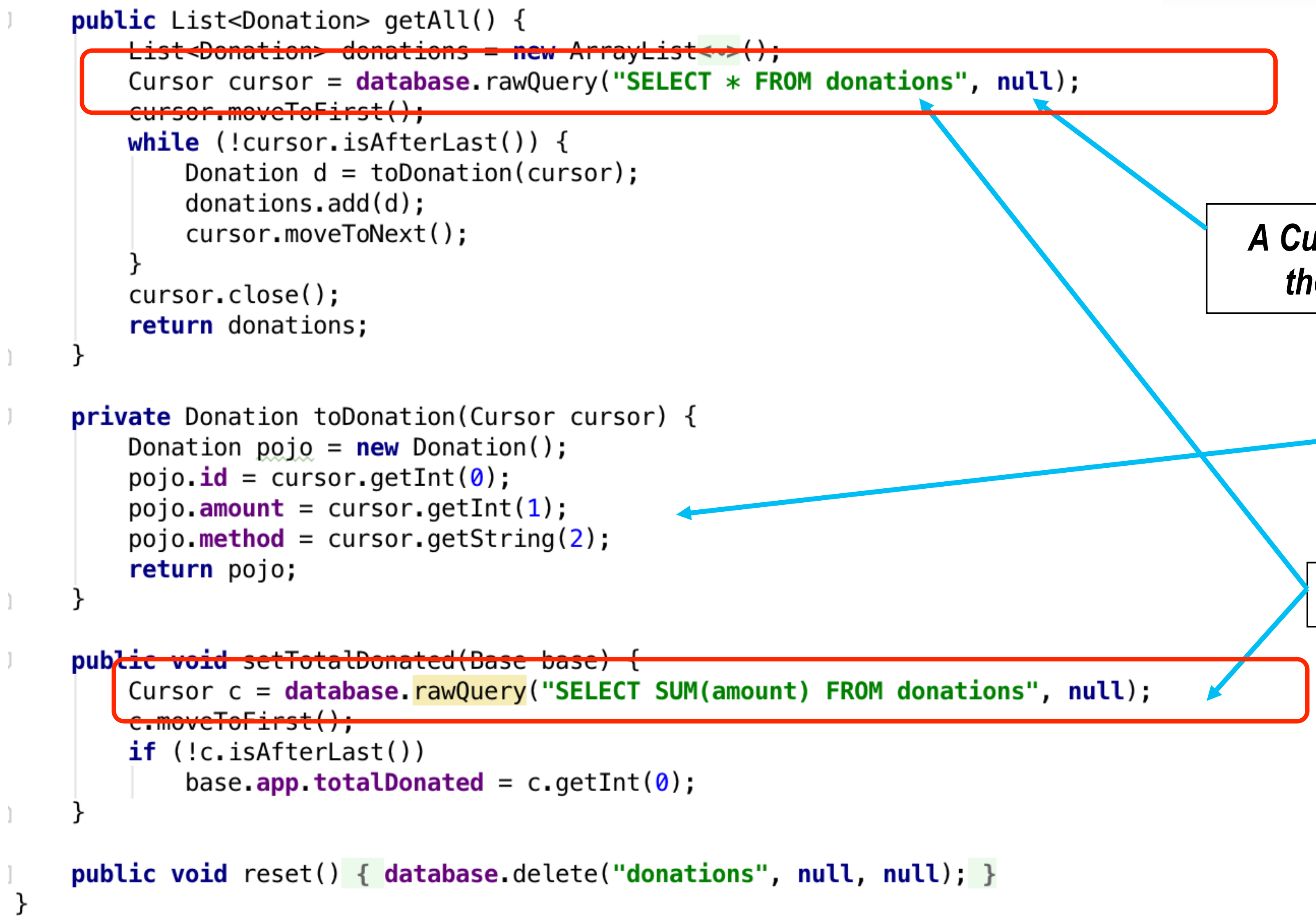
```java
public class DBManager {

    private SQLiteDatabase database;
    private DBDesigner dbHelper;

    public DBManager(Context context) { dbHelper = new DBDesigner(context); }

    public void open() throws SQLException {
        database = dbHelper.getWritableDatabase();
    }

    public void close() { database.close(); }

    public void add(Donation d) {
        ContentValues values = new ContentValues();
        values.put("amount", d.amount);
        values.put("method", d.method);

        database.insert("donations", null, values);
    }
```

Our database reference

Returns a reference to the database created from our SQL string

ContentValues are key/value pairs that are used when inserting/updating databases. Each ContentValue object corresponds to one row in a table

Donation – DBManager *



```
public List<Donation> getAll() {
    List<Donation> donations = new ArrayList<>();
    Cursor cursor = database.rawQuery("SELECT * FROM donations", null);
    cursor.moveToFirst();
    while (!cursor.isAfterLast()) {
        Donation d = toDonation(cursor);
        donations.add(d);
        cursor.moveToNext();
    }
    cursor.close();
    return donations;
}

private Donation toDonation(Cursor cursor) {
    Donation pojo = new Donation();
    pojo.id = cursor.getInt(0);
    pojo.amount = cursor.getInt(1);
    pojo.method = cursor.getString(2);
    return pojo;
}

public void setTotalDonated(Base base) {
    Cursor c = database.rawQuery("SELECT SUM(amount) FROM donations", null);
    c.moveToFirst();
    if (!c.isAfterLast())
        base.app.totalDonated = c.getInt(0);
}

public void reset() { database.delete("donations", null, null); }
}
```

A Cursor provides random read-write access to the resultset returned by a database query

This method 'converts' a Cursor object into a Donation Object

You can execute standard SQL if you like?

Other Cursor Functions

- moveToPrevious
- getCount
- getColumnIndexOrThrow
- getColumnName
- getColumnNames
- moveToPosition
- getPosition

Donation.4.0

Using the Application Object

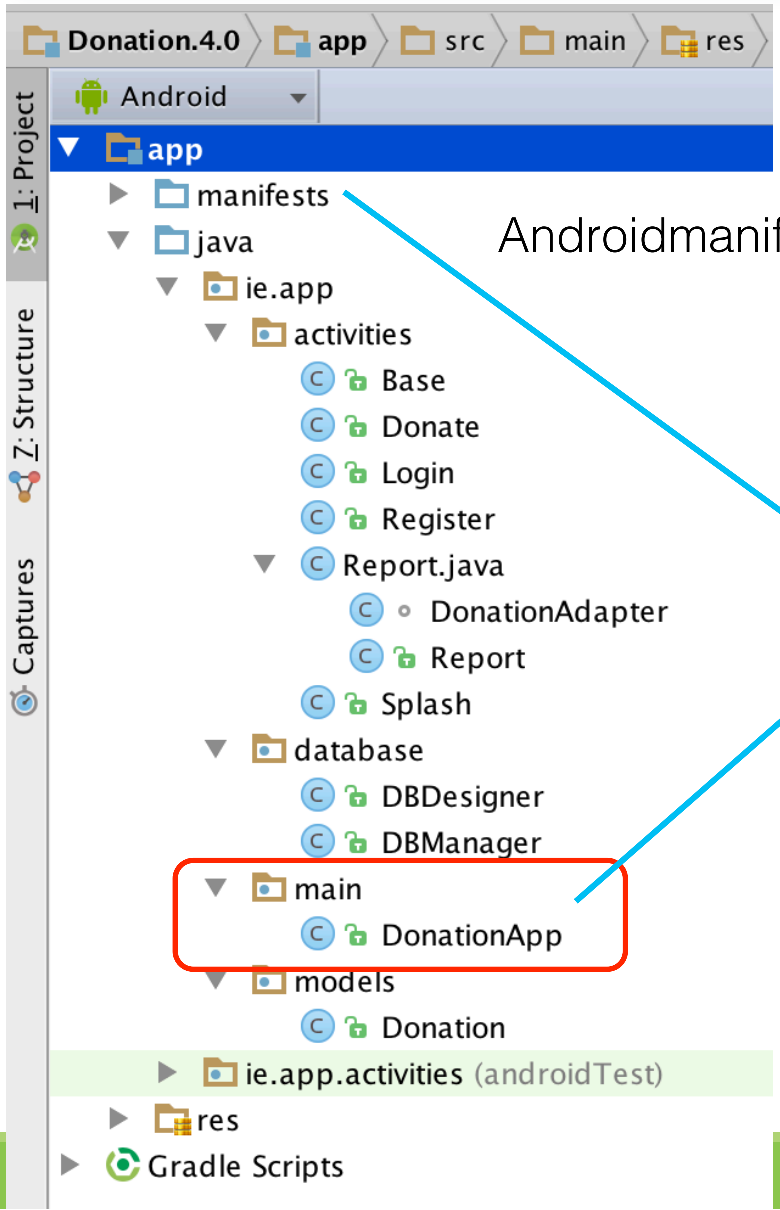
Maintaining Global Application State

- ❑ Sometimes you want to store data, like global variables which need to be accessed from multiple Activities – sometimes everywhere within the application. In this case, the Application object will help you.
- ❑ Activities come and go based on user interaction
- ❑ Application objects can be a useful 'anchor' for an android app
- ❑ You can use it to hold information shared by all activities

Application Object Callbacks

- ❑ onConfigurationChanged( ) Called by the system when the device configuration changes while your component is running.
- ❑ onCreate( ) Called when the application is starting, before any other application objects have been created.
- ❑ onLowMemory( ) This is called when the overall system is running low on memory, and would like actively running processes to tighten their belts.
- ❑ onTerminate( ) This method is for use in emulated process environments. It will never be called on a production Android device, where processes are removed by simply killing them; no user code (including this callback) is executed when doing so.

The Application Object *


Donation.4.0 › app › src › main › res

Android

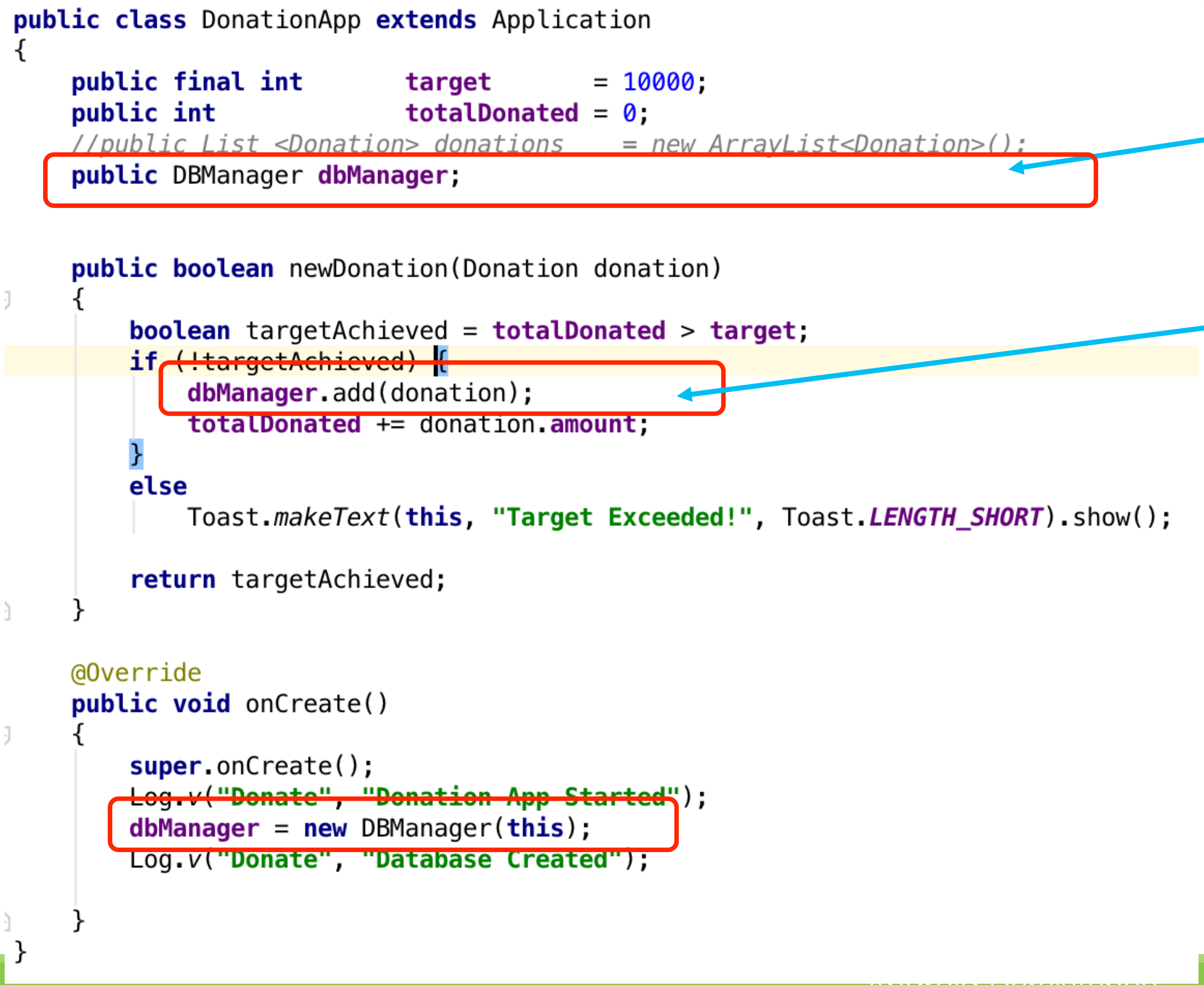
- app
  - manifests
  - java
    - ie.app
      - activities
        - Base
        - Donate
        - Login
        - Register
        - Report.java
          - DonationAdapter
          - Report
        - Splash
      - database
        - DBDesigner
        - DBManager
      - main
        - DonationApp
      - models
        - Donation
    - ie.app.activities (androidTest)
  - res
- Gradle Scripts

Androidmanifest.xml

```java
import android.app.Application;

public class DonationApp extends Application
{
  @Override
  public void onCreate()
  {
    super.onCreate();
    Log.v("Donation", "Donation App Started");
  }
}
```

```xml
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Donation.4.0"
    android:supportsRtl="true"
    android:theme="@style/AppTheme"
    android:name="ie.app.main.DonationApp">
```

Donation 4.0 – DonationApp Application Object *



```
public class DonationApp extends Application
{
    public final int        target        = 10000;
    public int              totalDonated = 0;
    //public List <Donation> donations     = new ArrayList<Donation>();
    public DBManager dbManager;


    public boolean newDonation(Donation donation)
    {
        boolean targetAchieved = totalDonated > target;
        if (!targetAchieved) {
            dbManager.add(donation);
            totalDonated += donation.amount;
        }
        else
            Toast.makeText(this, "Target Exceeded!", Toast.LENGTH_SHORT).show();

        return targetAchieved;
    }

    @Override
    public void onCreate()
    {
        super.onCreate();
        Log.v("Donate", "Donation App Started");
        dbManager = new DBManager(this);
        Log.v("Donate", "Database Created");

    }
}
```

Our Database Reference

Adding a Donation

Refactor existing Activities/Classes

- ❑ In order to make full use of our Database and Application object we need to refactor the classes in the project.
- ❑ This will form part of the Practical Lab (Lab 5) but we'll have a quick look now at some of the refactoring that needs to be done.
- ❑ We also add in a new Menu option to 'Reset' the database once the target has been reached and keep track of the current total donated if the app is shut down are restarted.

Donation 4.0 – `Base` (Refactored, extract) *

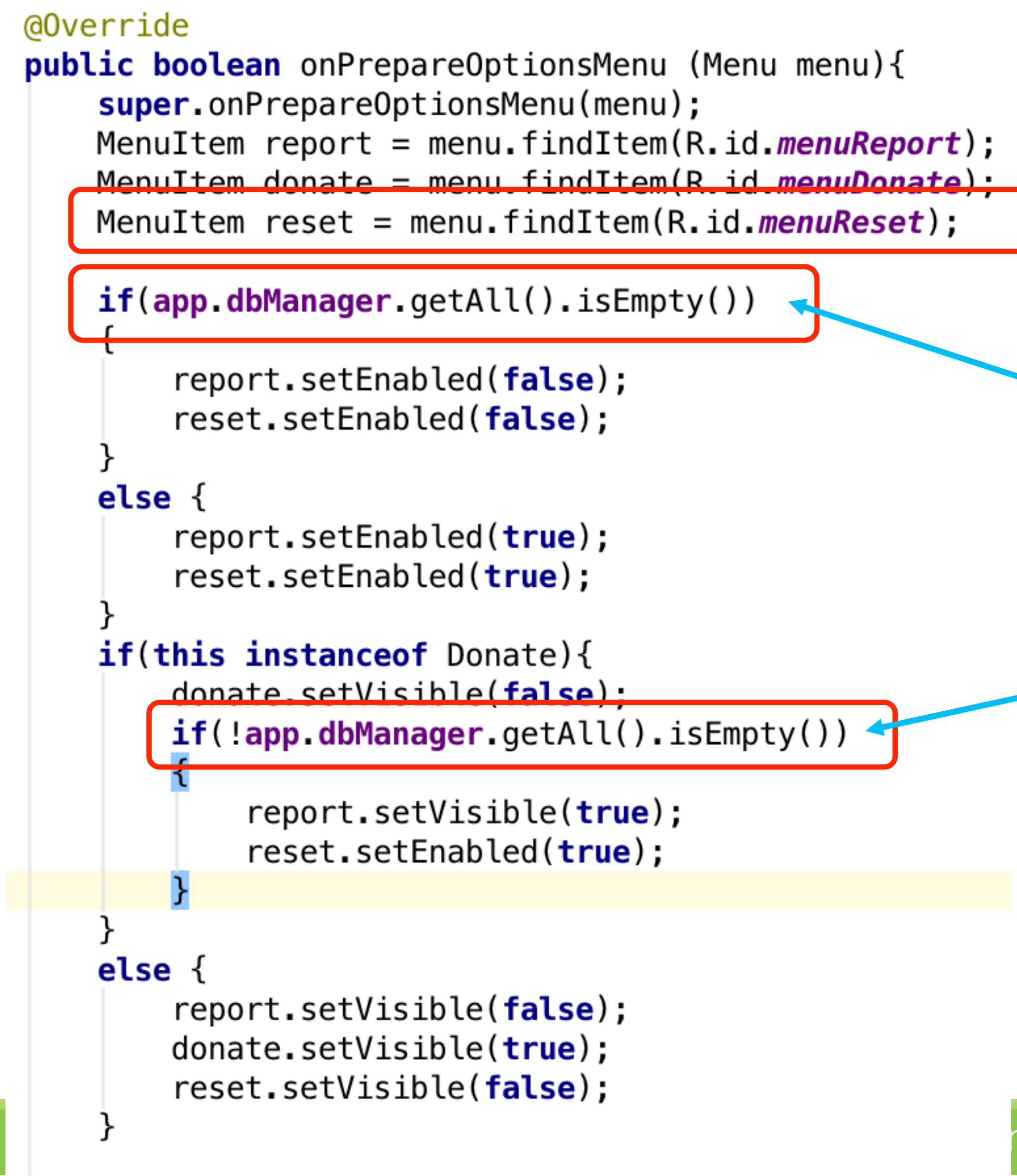
```
public class Base extends AppCompatActivity {

    public DonationApp app;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        app = (DonationApp) getApplication();

        app.dbManager.open();
        app.dbManager.setTotalDonated(this);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        app.dbManager.close();
    }
```

Our DonationApp reference

'Binding' to our Application Object

Invoking the Application wide methods

Donation 4.0 – Base (Refactored, extract) *



```
@Override
public boolean onPrepareOptionsMenu (Menu menu){
    super.onPrepareOptionsMenu(menu);
    MenuItem report = menu.findItem(R.id.menuReport);
    MenuItem donate = menu.findItem(R.id.menuDonate);
    MenuItem reset = menu.findItem(R.id.menuReset);

    if(app.dbManager.getAll().isEmpty())
    {
        report.setEnabled(false);
        reset.setEnabled(false);
    }
    else {
        report.setEnabled(true);
        reset.setEnabled(true);
    }
    if(this instanceof Donate){
        donate.setVisible(false);
        if(!app.dbManager.getAll().isEmpty())
        {
            report.setVisible(true);
            reset.setEnabled(true);
        }
    }
    else {
        report.setVisible(false);
        donate.setVisible(true);
        reset.setVisible(false);
    }
```

Our new 'Reset' menu option

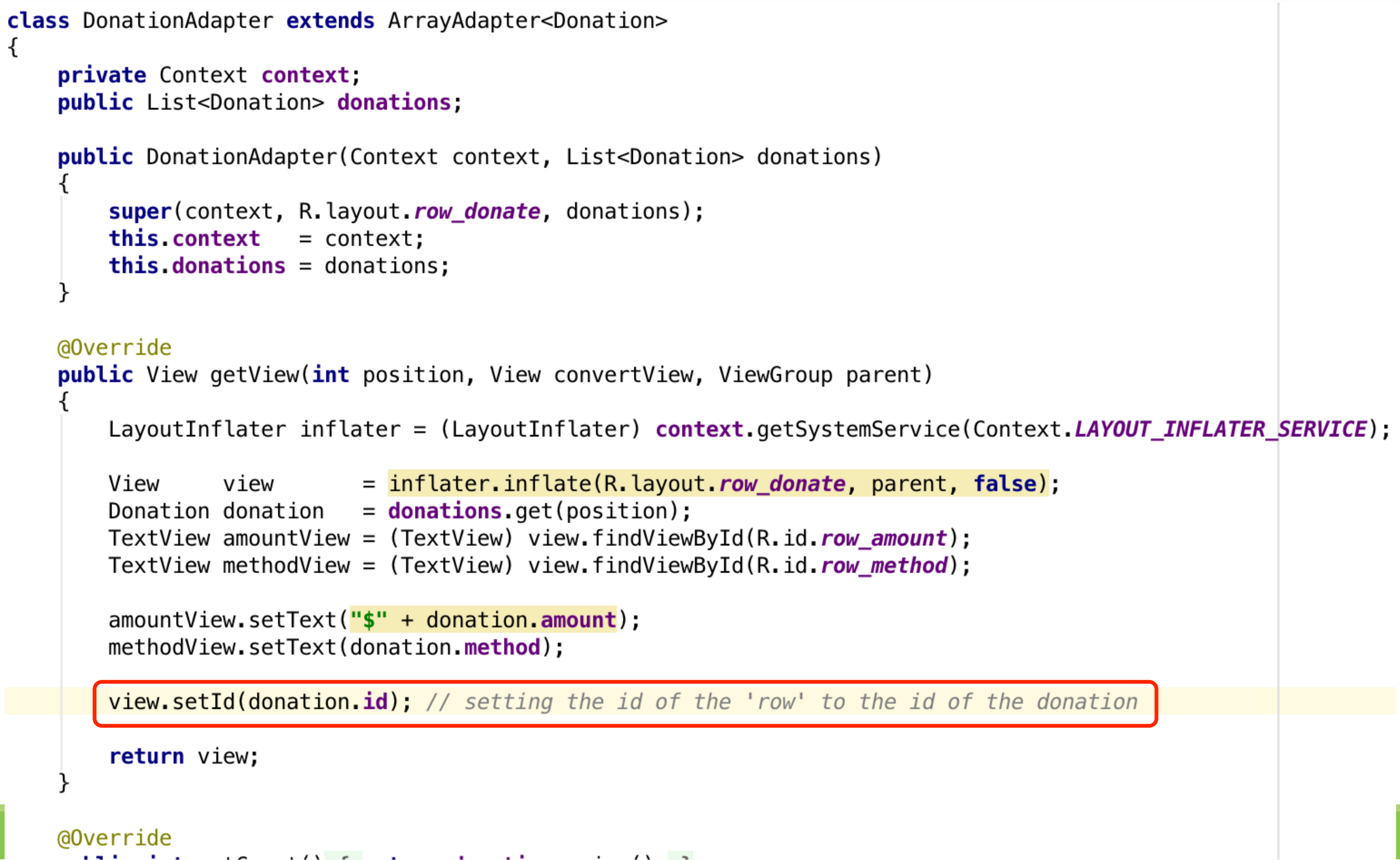
Checking the database

Donation 4.0 – **Donate** (Refactored) *

```
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 (donatedAmount > 0)
    {
        app.newDonation(new Donation(donatedAmount, method));
        progressBar.setProgress(app.totalDonated);
        String totalDonatedStr = "$" + app.totalDonated;
        amountTotal.setText(totalDonatedStr);
    }
}

@Override
public void reset(MenuItem item)
{
    app.dbManager.reset();
    app.totalDonated = 0;
    amountTotal.setText("$" + app.totalDonated);
}
```

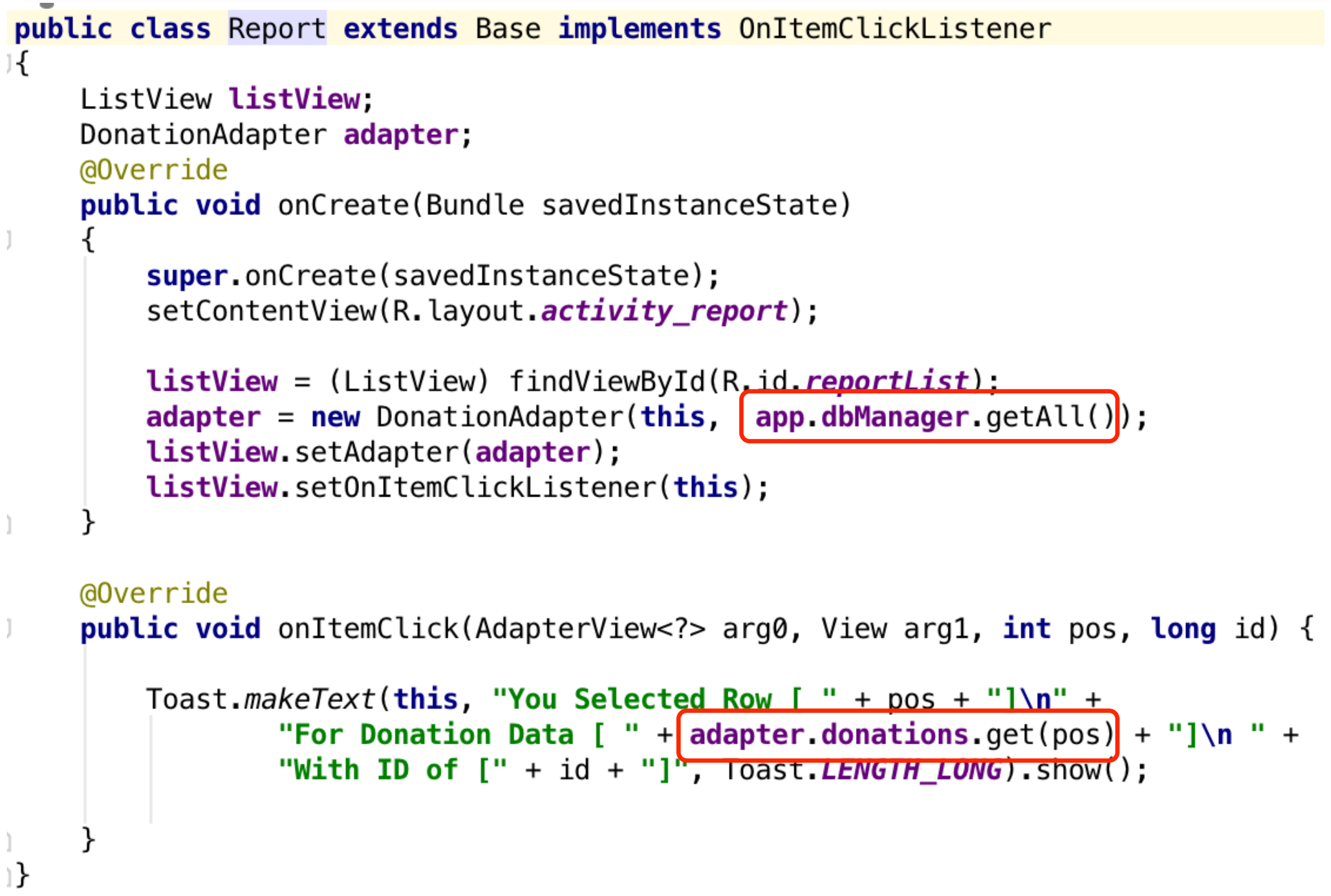
Invoking the Application wide methods

Donation 4.0 – DonationAdapter *



```
class DonationAdapter extends ArrayAdapter<Donation>
{
    private Context context;
    public List<Donation> donations;

    public DonationAdapter(Context context, List<Donation> donations)
    {
        super(context, R.layout.row_donate, donations);
        this.context   = context;
        this.donations = donations;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent)
    {
        LayoutInflater inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        View     view       = inflater.inflate(R.layout.row_donate, parent, false);
        Donation donation   = donations.get(position);
        TextView amountView = (TextView) view.findViewById(R.id.row_amount);
        TextView methodView = (TextView) view.findViewById(R.id.row_method);

        amountView.setText("$" + donation.amount);
        methodView.setText(donation.method);

        view.setId(donation.id); // setting the id of the 'row' to the id of the donation

        return view;
    }

    @Override
```

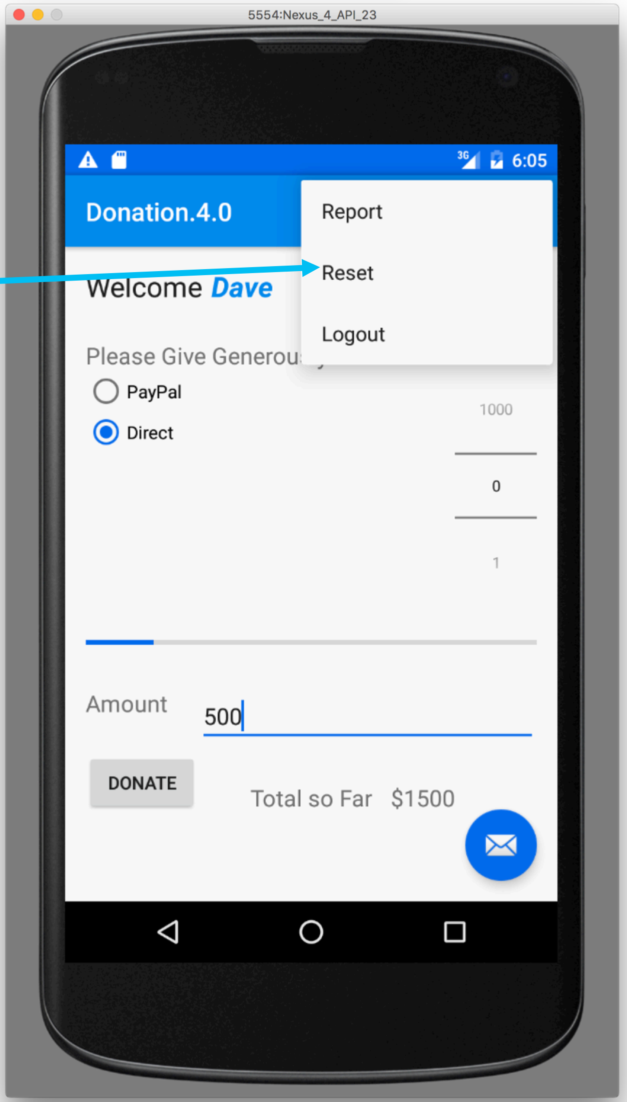
Donation 4.0 – **Report**



```
public class Report extends Base implements OnItemClickListener
{
    ListView listView;
    DonationAdapter adapter;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_report);

        listView = (ListView) findViewById(R.id.reportList);
        adapter = new DonationAdapter(this, app.dbManager.getAll());
        listView.setAdapter(adapter);
        listView.setOnItemClickListener(this);
    }

    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int pos, long id) {

        Toast.makeText(this, "You Selected Row [ " + pos + "]\n" +
                "For Donation Data [ " + adapter.donations.get(pos) + "]\n " +
                "With ID of [" + id + "]", Toast.LENGTH_LONG).show();

    }
}
```

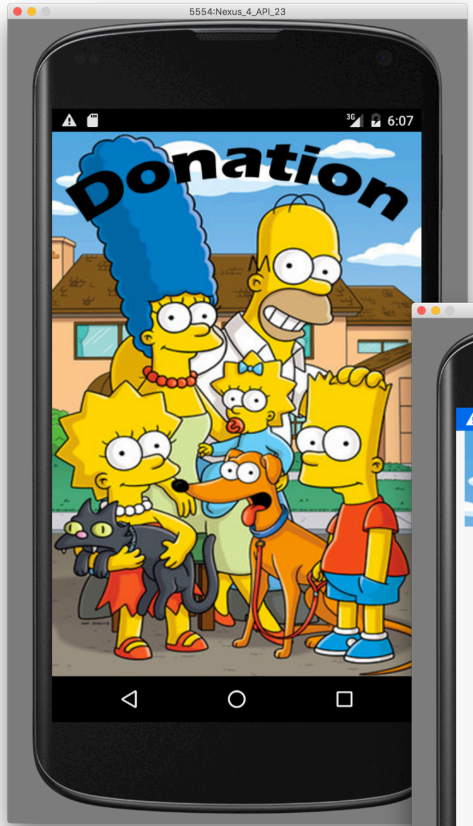
Donation 4.0 – 'Reset' Menu Item

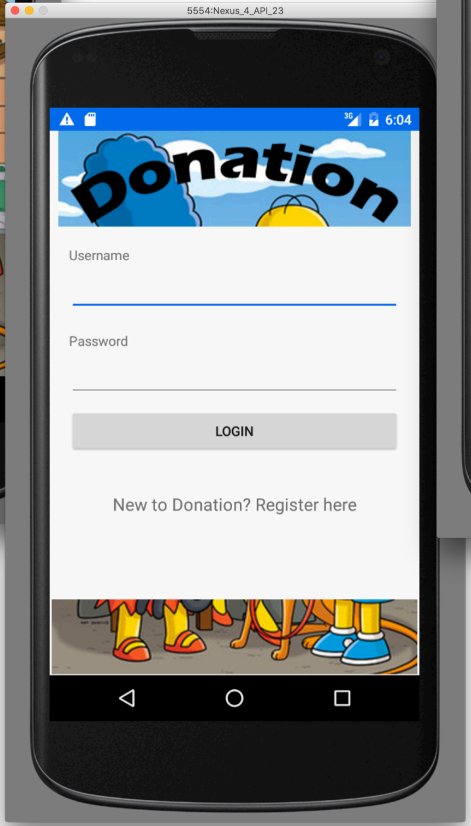
```
<item
   android:id="@+id/menuReset"
   android:orderInCategory="100"
   android:showAsAction="never"
   android:title="@string/menuReset"
   android:onClick="reset"/>
```

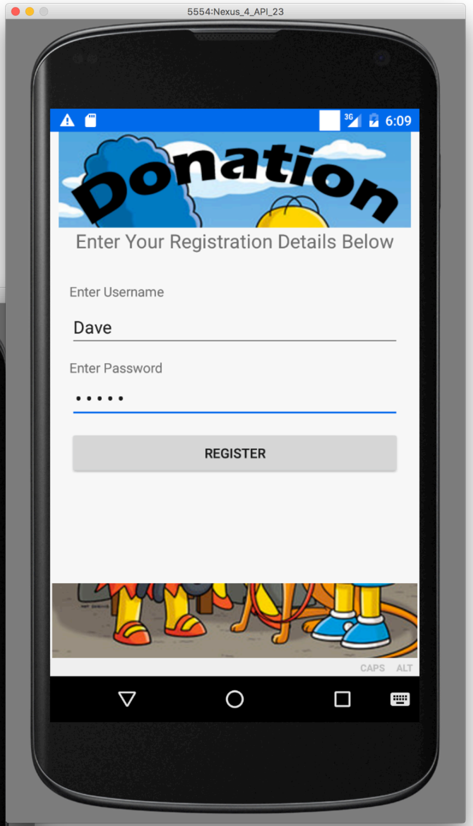


Using A Splash Screen,
Login Screen / Register Screen
&
Application Object

What do we want exactly?







- ❑ Display Splash Screen for a few seconds
- ❑ Display Login Screen
- ❑ Allow User to Register
- ❑ Only show Home Screen once valid details entered
- ❑ Logout Menu Option
- ❑ AND Manage our DB via Application Object

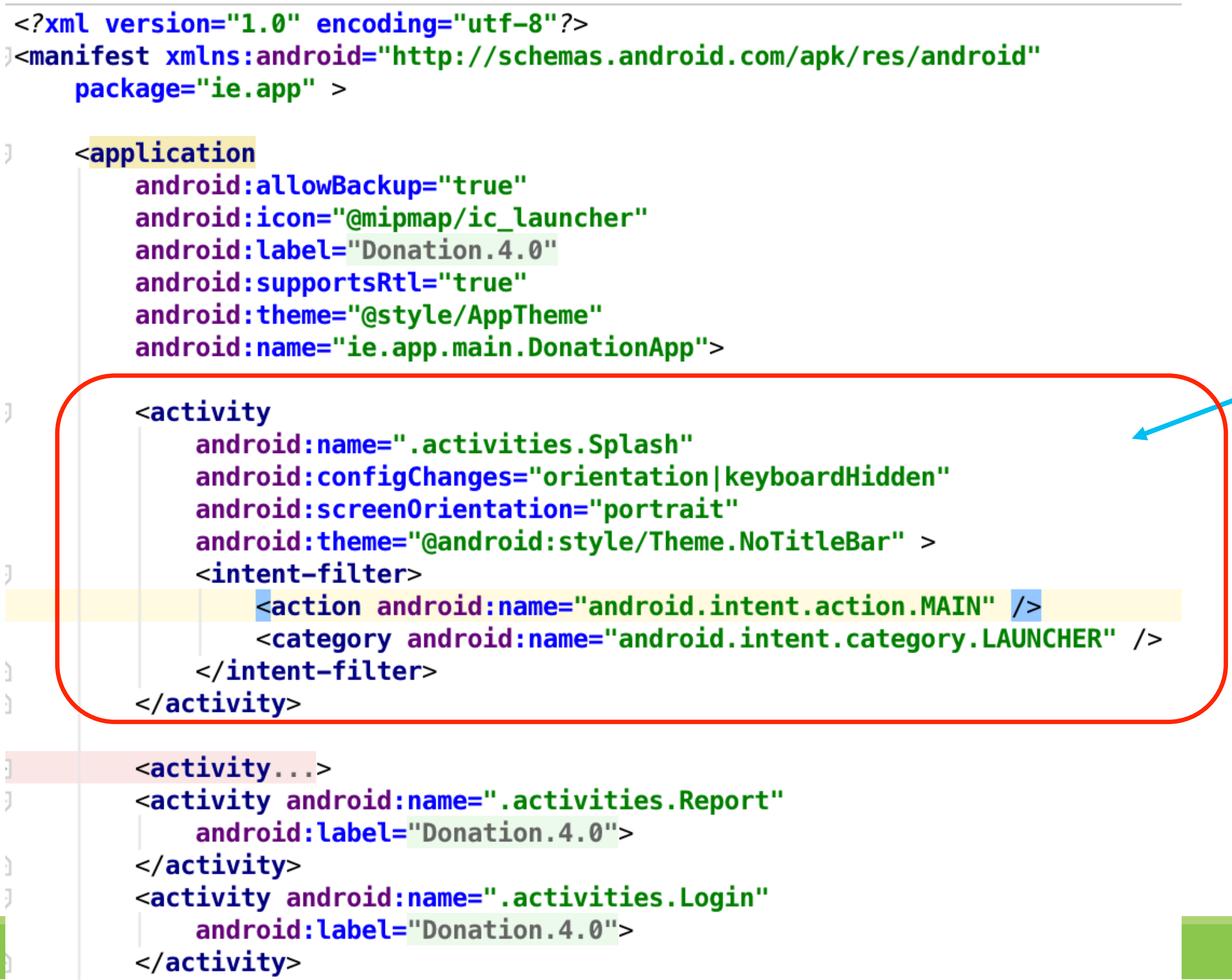
Donation 4.0 - Splash

```java
public class Splash extends Activity {
    // used to know if the back button was pressed in the splash screen activity
    // and avoid opening the next activity
    private boolean              mIsBackButtonPressed;
    private static final int     SPLASH_DURATION = 2000; // 2 seconds

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        Handler handler = new Handler();
        // run a thread after 2 seconds to start the home screen
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                // make sure we close the splash screen so the user
                // won't come back when it presses back key
                finish();

                if (!mIsBackButtonPressed) {
                    // start the home screen if the back button wasn't pressed already
                    Splash.this.startActivity(new Intent(Splash.this, Login.class));
                }
            }
        }, SPLASH_DURATION); // time in milliseconds to delay call to run()
    }

    @Override
   public void onBackPressed() {
        // set the flag to true so the next activity won't start up
        mIsBackButtonPressed = true;
```

Handler object associated with single thread

Start Login Screen via Intent

Update Manifest File



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ie.app" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Donation.4.0"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:name="ie.app.main.DonationApp">

        <activity
            android:name=".activities.Splash"
            android:configChanges="orientation|keyboardHidden"
            android:screenOrientation="portrait"
            android:theme="@android:style/Theme.NoTitleBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity...>
        <activity android:name=".activities.Report"
            android:label="Donation.4.0">
        </activity>
        <activity android:name=".activities.Login"
            android:label="Donation.4.0">
        </activity>
```

Activity to Launch

Using SharedPreferences

SharedPreferences (1)

- Three forms:
  - Share across all components in an application
    - getSharedPreferences(“SomeString”,Activity.MODE_PRIVATE);
  - Store only data needed by this Activity
    - getPreferences(Activity.MODE_PRIVATE);
  - Store only data needed by this Activity when Activity becomes inactive (but not when finished)
    - Eg. Orientation change from portrait to landscape
    - use *Bundle* in onSaveInstanceState / onRestoreInstanceState / onCreate

SharedPreferences (2)

- ❑ Create your SharedPreferences instance

```
SharedPreferences settings
   = this.getSharedPreferences("Demo", MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
```

- ❑ Add data in the form : <String Key,String Value>

```
editor.putString("name", "value");
editor.commit();
```

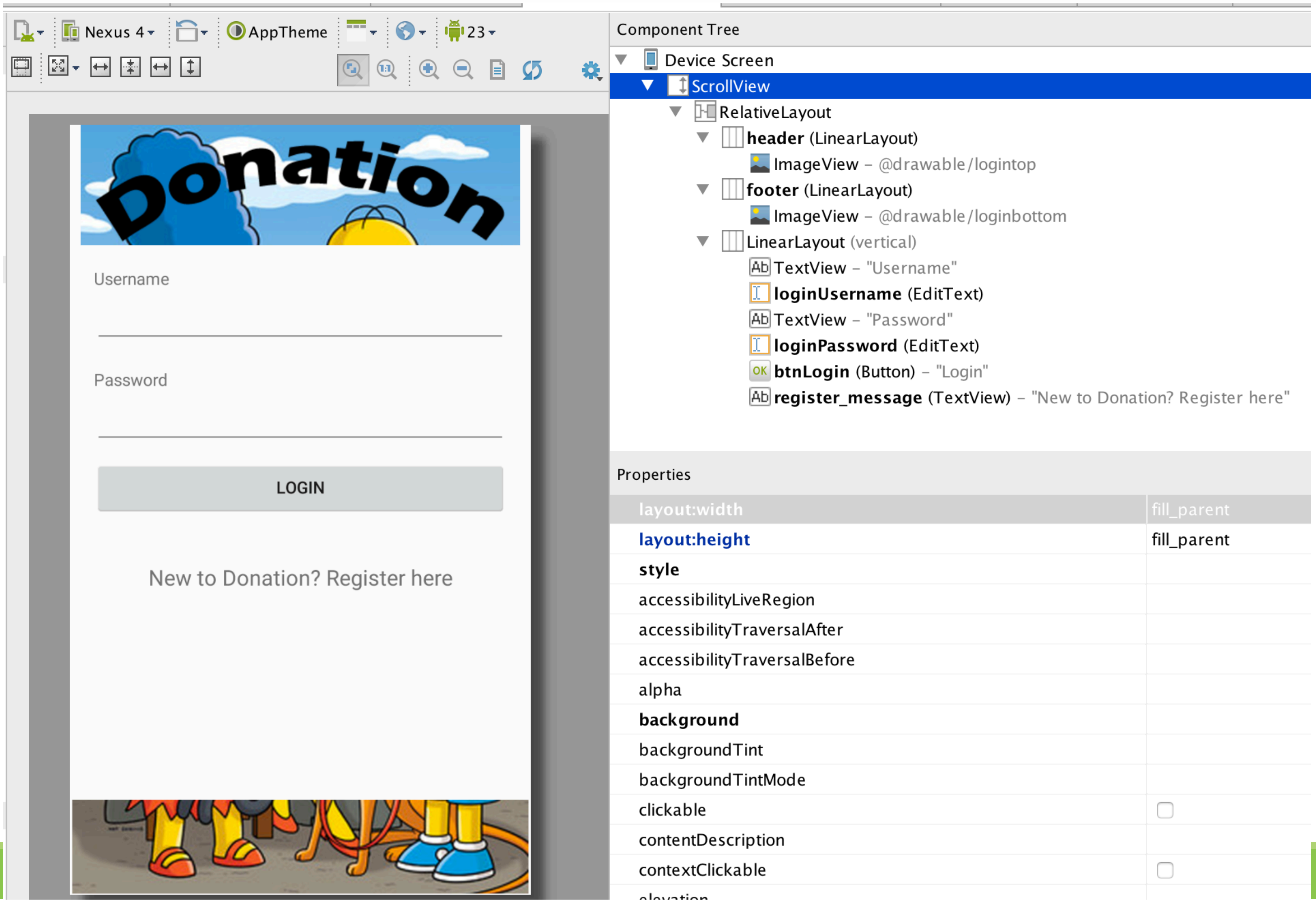
- ❑ Use ‘Key’ to get ‘Value’

```
String str = settings.getString("name", "defaultValue");
```

- ❑ Reset the preferences (clear)

```
editor.clear().commit();
```

Donation 4.0 – activity_login.xml



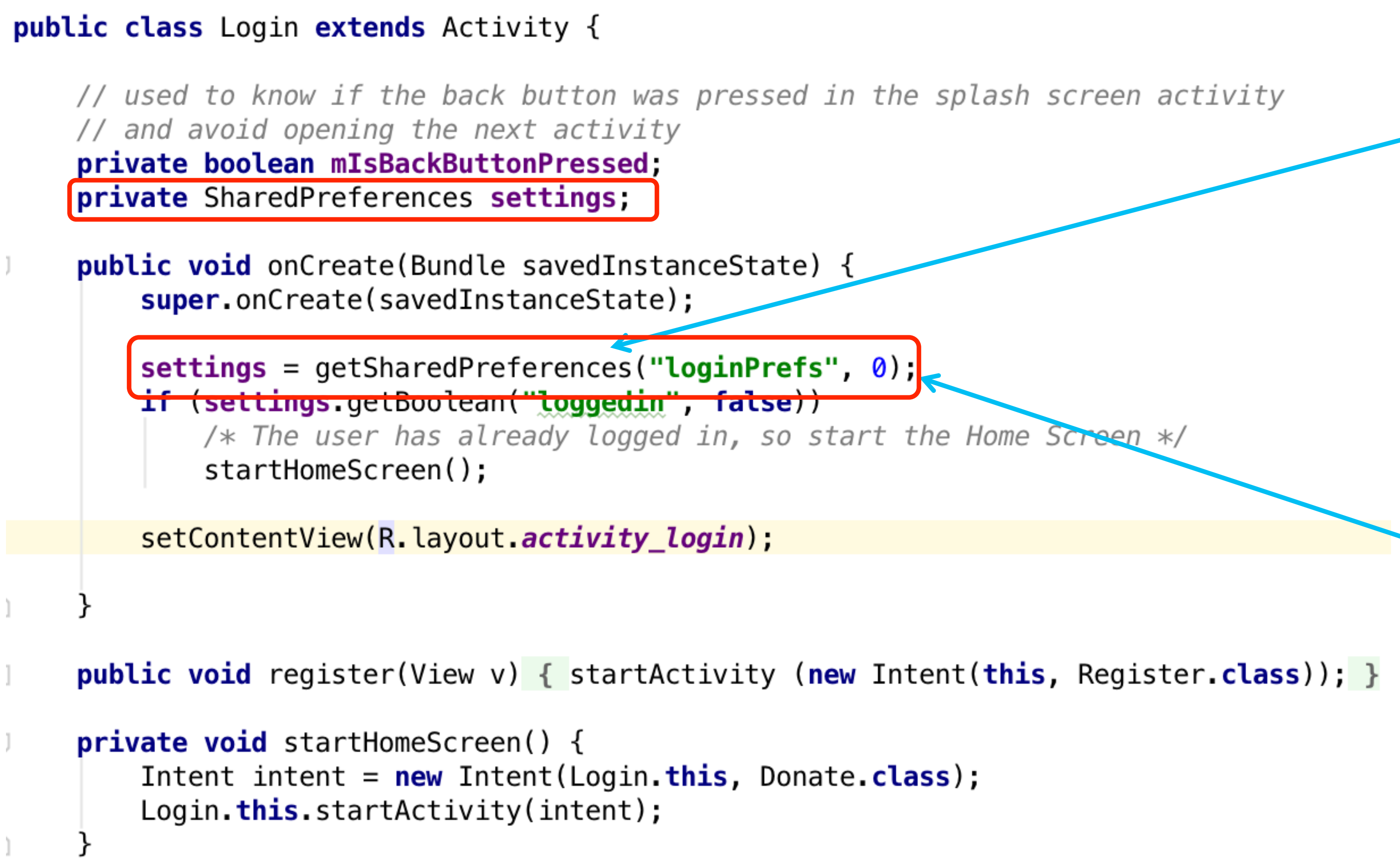
Component Tree

```
▼ Device Screen
  ▼ ScrollView
    ▼ RelativeLayout
      ▼ header (LinearLayout)
          ImageView – @drawable/logintop
      ▼ footer (LinearLayout)
          ImageView – @drawable/loginbottom
      ▼ LinearLayout (vertical)
          TextView – "Username"
          loginUsername (EditText)
          TextView – "Password"
          loginPassword (EditText)
          btnLogin (Button) – "Login"
          register_message (TextView) – "New to Donation? Register here"
```

Properties

layout:width	fill_parent
layout:height	fill_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfter	
accessibilityTraversalBefore	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	☐
contentDescription	
contextClickable	☐

Donation 4.0 – Login *

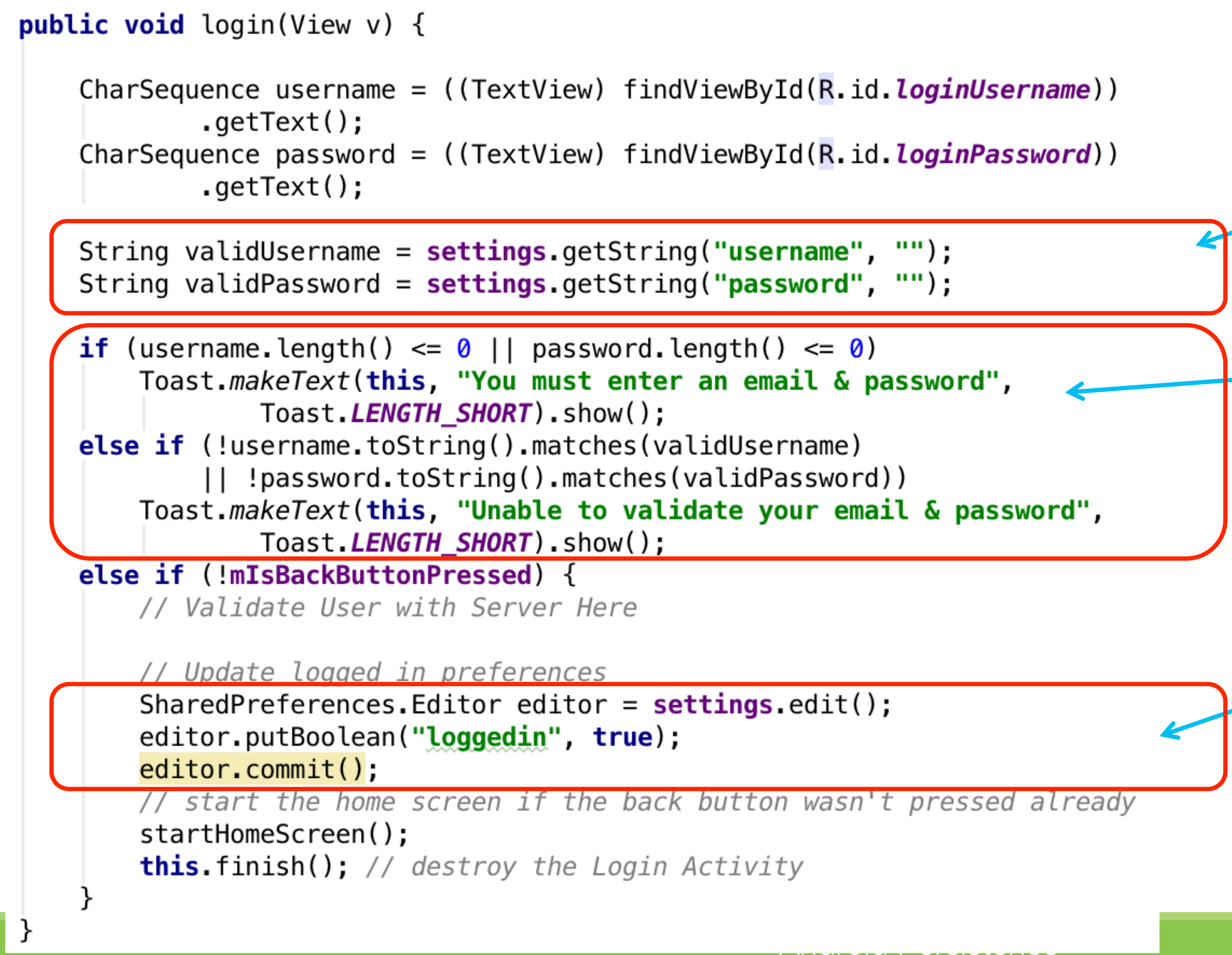


```
public class Login extends Activity {

    // used to know if the back button was pressed in the splash screen activity
    // and avoid opening the next activity
    private boolean mIsBackButtonPressed;
    private SharedPreferences settings;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        settings = getSharedPreferences("loginPrefs", 0);
        if (settings.getBoolean("loggedin", false))
            /* The user has already logged in, so start the Home Screen */
            startHomeScreen();

        setContentView(R.layout.activity_login);

    }

    public void register(View v) { startActivity (new Intent(this, Register.class)); }

    private void startHomeScreen() {
        Intent intent = new Intent(Login.this, Donate.class);
        Login.this.startActivity(intent);
    }
```

Load your Preferences

Default value

Donation – Login *



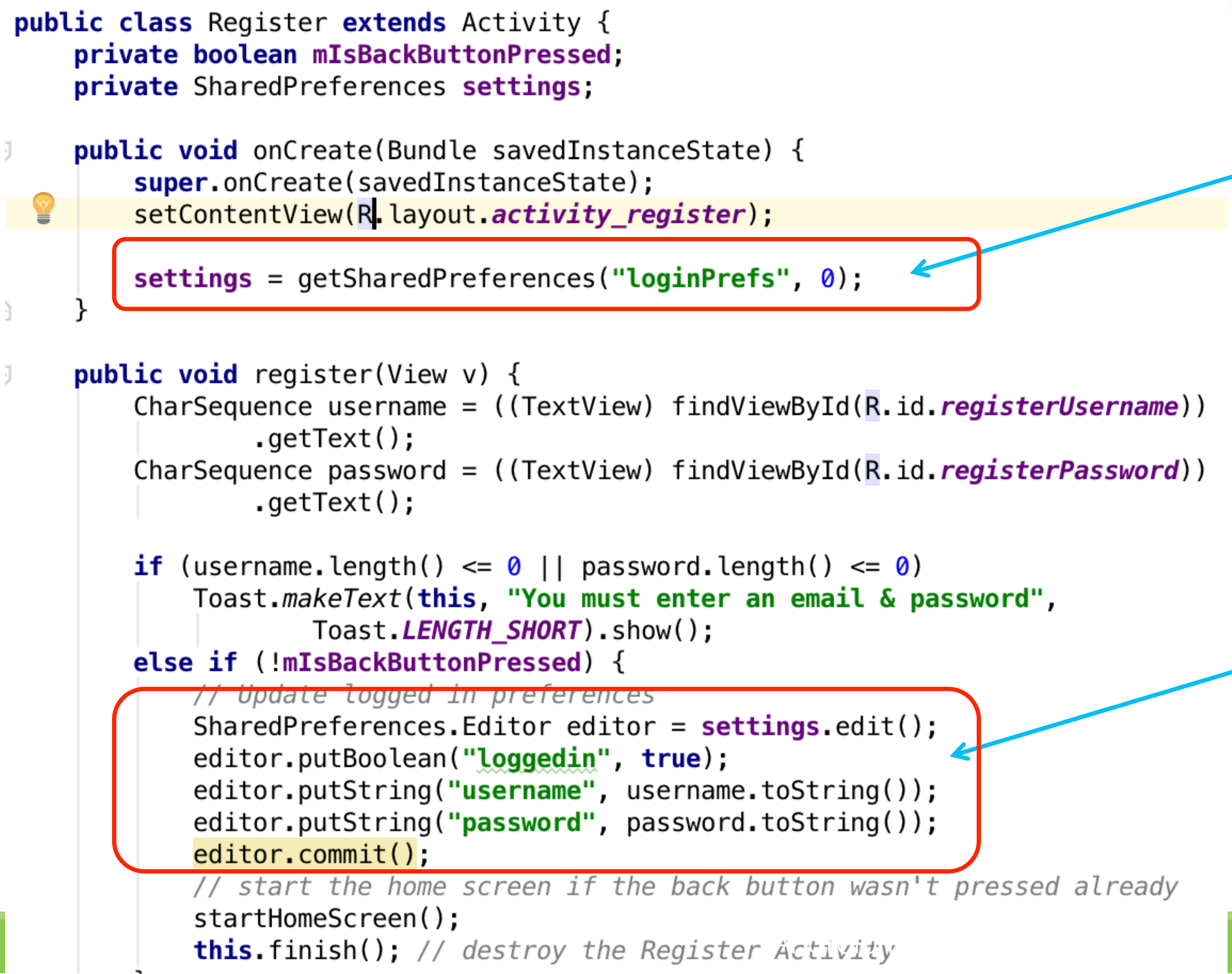
```
public void login(View v) {

    CharSequence username = ((TextView) findViewById(R.id.loginUsername))
            .getText();
    CharSequence password = ((TextView) findViewById(R.id.loginPassword))
            .getText();

    String validUsername = settings.getString("username", "");
    String validPassword = settings.getString("password", "");

    if (username.length() <= 0 || password.length() <= 0)
        Toast.makeText(this, "You must enter an email & password",
                Toast.LENGTH_SHORT).show();
    else if (!username.toString().matches(validUsername)
            || !password.toString().matches(validPassword))
        Toast.makeText(this, "Unable to validate your email & password",
                Toast.LENGTH_SHORT).show();
    else if (!mIsBackButtonPressed) {
        // Validate User with Server Here

        // Update logged in preferences
        SharedPreferences.Editor editor = settings.edit();
        editor.putBoolean("loggedin", true);
        editor.commit();
        // start the home screen if the back button wasn't pressed already
        startHomeScreen();
        this.finish(); // destroy the Login Activity
    }
}
```

Retrieving existing details

Verifying entered details

Update Preferences with data

Donation 4.0 – Register *



```
public class Register extends Activity {
    private boolean mIsBackButtonPressed;
    private SharedPreferences settings;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        settings = getSharedPreferences("loginPrefs", 0);
    }

    public void register(View v) {
        CharSequence username = ((TextView) findViewById(R.id.registerUsername))
                .getText();
        CharSequence password = ((TextView) findViewById(R.id.registerPassword))
                .getText();

        if (username.length() <= 0 || password.length() <= 0)
            Toast.makeText(this, "You must enter an email & password",
                    Toast.LENGTH_SHORT).show();
        else if (!mIsBackButtonPressed) {
            // Update logged in preferences
            SharedPreferences.Editor editor = settings.edit();
            editor.putBoolean("loggedin", true);
            editor.putString("username", username.toString());
            editor.putString("password", password.toString());
            editor.commit();
            // start the home screen if the back button wasn't pressed already
            startHomeScreen();
            this.finish(); // destroy the Register Activity
```

Retrieving existing prefs file

Update Preferences with new data

Donation 4.0 – Logout method (in `Base`)

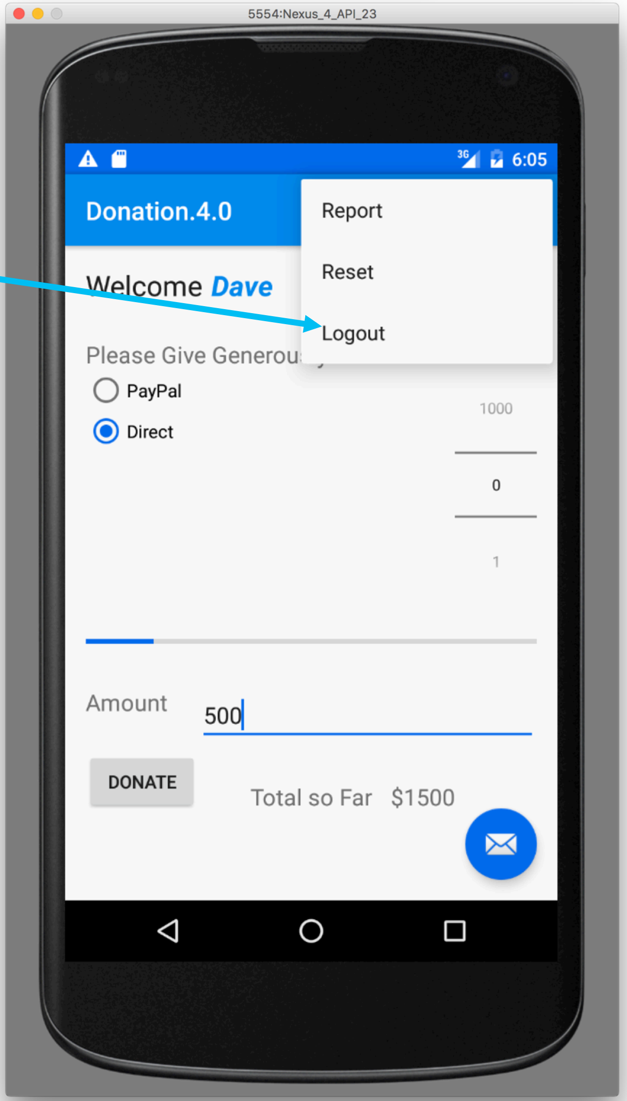
Resetting 'loggedin' to false

```java
public void logout(MenuItem item) {
    SharedPreferences.Editor editor = getSharedPreferences("loginPrefs", 0).edit();
    editor.putBoolean("loggedin", false);
    editor.commit();

    startActivity(new Intent(Base.this,Login.class)
            .setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK));
    finish();
}
```

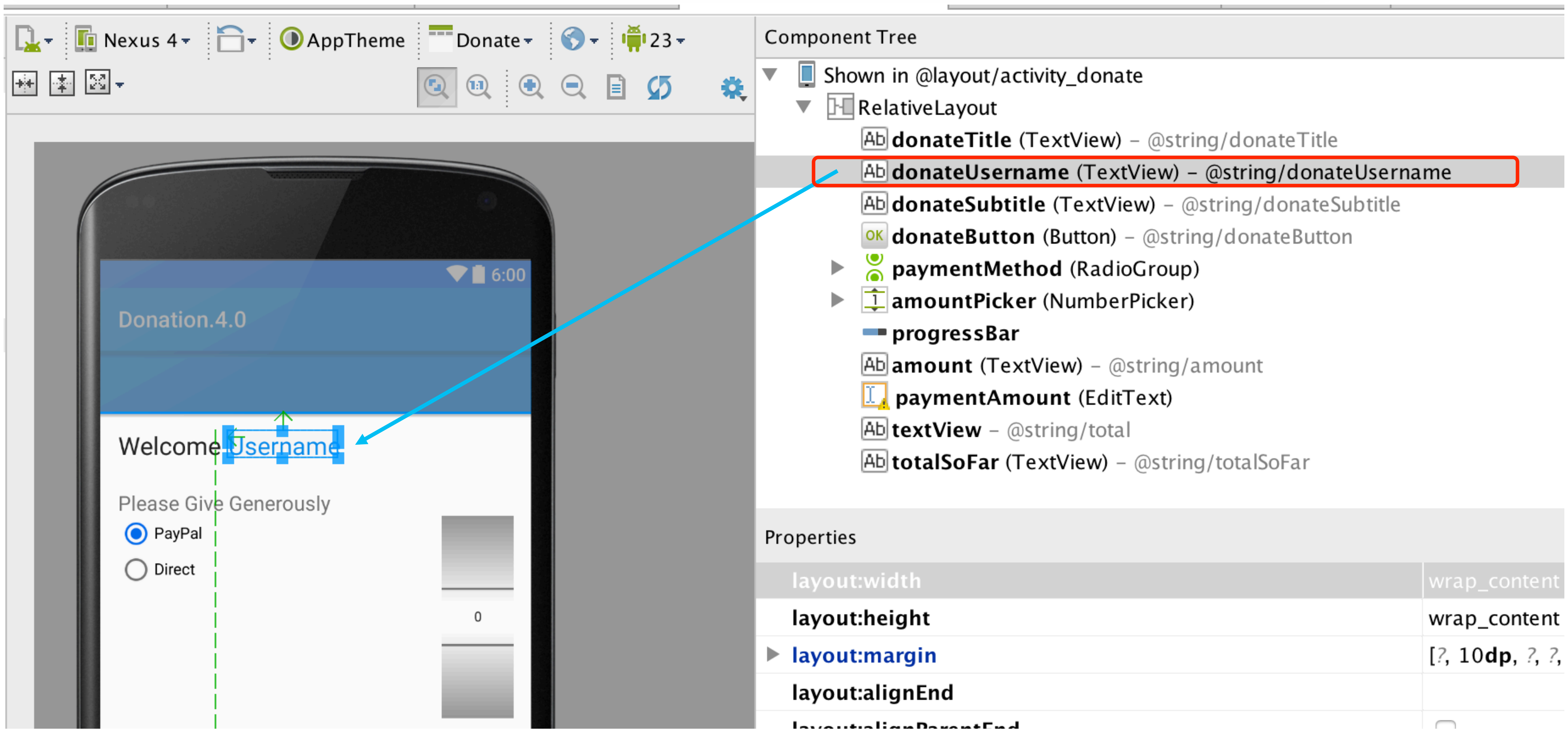
Returning to the 'Login' screen

Donation 4.0 – 'Logout' Menu Item

```
<item
    android:id="@+id/menuLogout"
    android:title="@string/menuLogout"
    android:orderInCategory="100"
    android:showAsAction="never"
    android:onClick="logout"/>
```



Donation – activity_donate.xml



Component Tree

- Shown in @layout/activity_donate
  - RelativeLayout
    - **donateTitle** (TextView) – @string/donateTitle
    - **donateUsername** (TextView) – @string/donateUsername
    - **donateSubtitle** (TextView) – @string/donateSubtitle
    - **donateButton** (Button) – @string/donateButton
    - **paymentMethod** (RadioGroup)
    - **amountPicker** (NumberPicker)
    - **progressBar**
    - **amount** (TextView) – @string/amount
    - **paymentAmount** (EditText)
    - **textView** – @string/total
    - **totalSoFar** (TextView) – @string/totalSoFar

Properties

layout:width	wrap_content
layout:height	wrap_content
layout:margin	[?, 10dp, ?, ?,
layout:alignEnd	

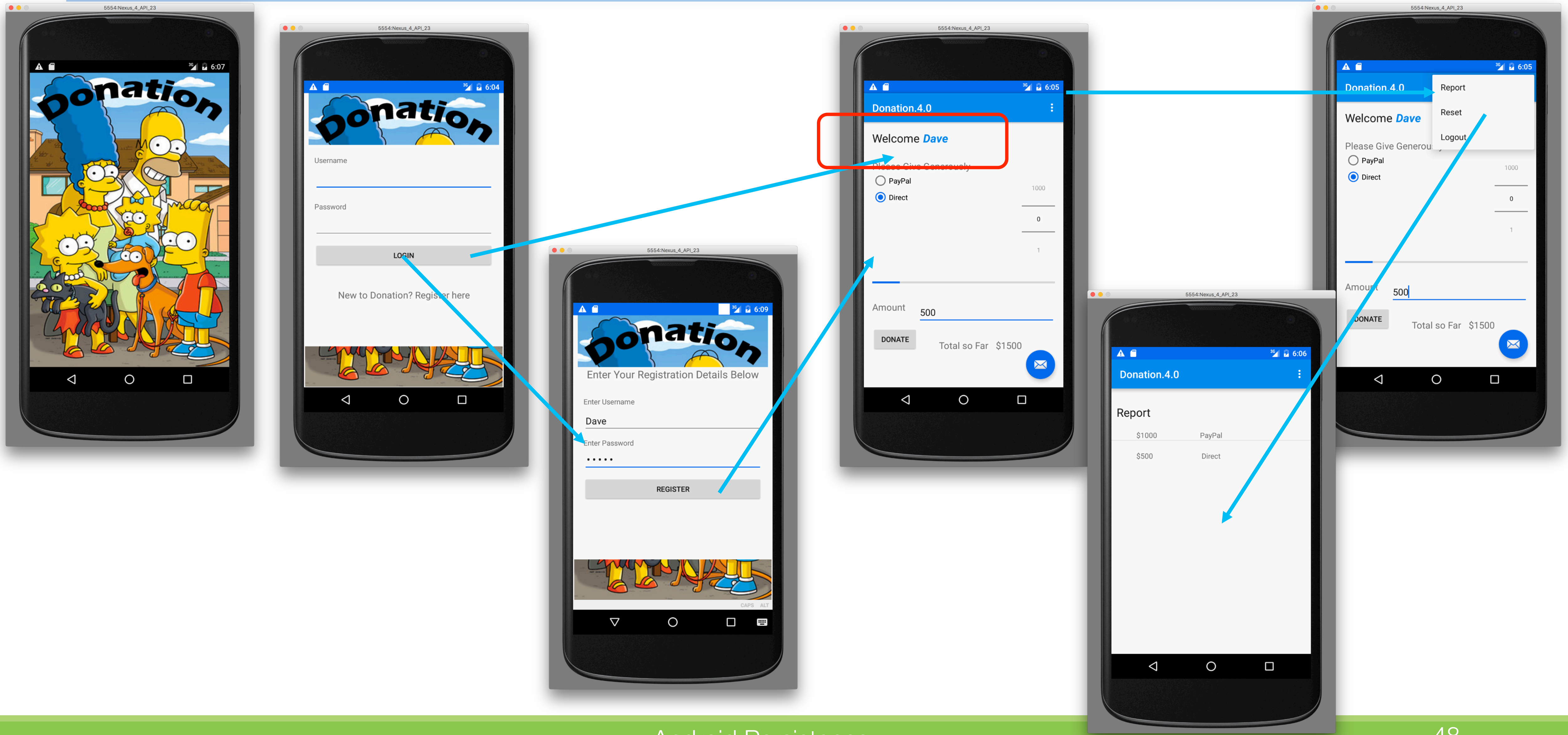
Donation 4.0 – Donate ‘onCreate()’ extract *

Retrieving existing Username from Prefs file

```
settings = getSharedPreferences("loginPrefs", 0);
String username = settings.getString("username", "");

TextView donateUsername = (TextView) findViewById(R.id.donateUsername);

SpannableString spanString = new SpannableString(username);
//spanString.setSpan(new UnderlineSpan(), 0, spanString.length(), 0);
spanString.setSpan(new StyleSpan(Typeface.BOLD), 0, spanString.length(), 0);
spanString.setSpan(new StyleSpan(Typeface.ITALIC), 0, spanString.length(), 0);
donateUsername.setText(spanString);
```

Applying some ‘styling’ and updating our TextView

End Result – Donation 4.0



More Reading

- JavaDoc: Activity
  - http://developer.android.com/reference/android/app/Activity.html
    - Introductory parts give lots of details
- Chapters
  - Handling Activity Lifecycle Events *and*
  - Handling Rotation
    - From *The Busy Coder's Guide to Android Development* by Mark Murphy.
      - http://commonsware.com/Android/

Questions?