Mobile Application Development

Produced by

Dr. Siobhán Drohan (sdrohan@wit.ie)
Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

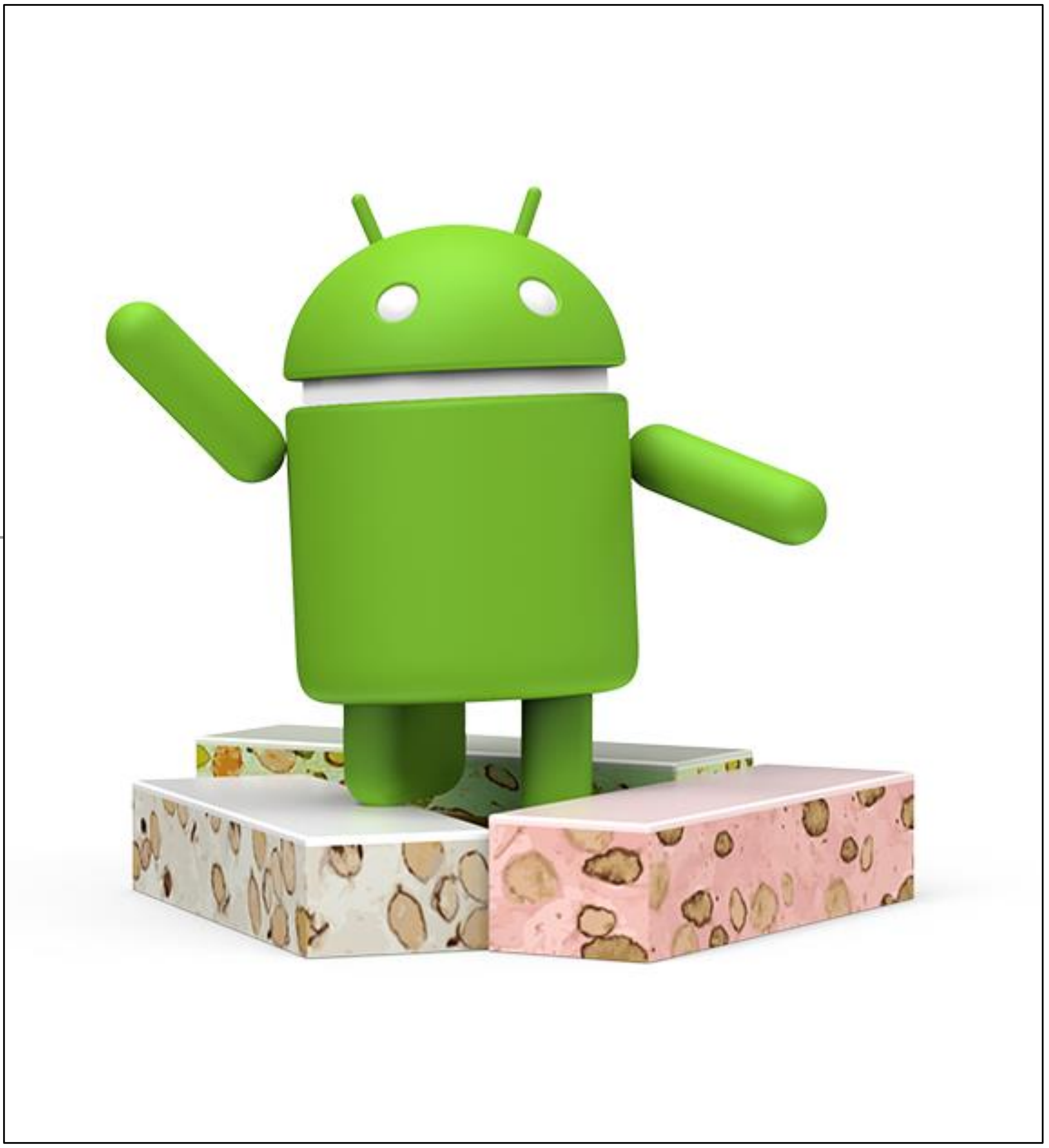
http://elearning.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

Event Handling

…and a brief recap of Interfaces



Interfaces Recap

What is an interface?

- Writing an interface is similar to writing a class.
  - But a class describes the attributes and behaviours of an object.
  - And an interface contains behaviours that a class implements.

http://www.tutorialspoint.com/java/java_interfaces.htm

What is an interface?

- An interface is:
    - a reference type in Java
    - similar(ish) to a class,
    - a collection of abstract method signatures.
- A class implements an interface, thereby inheriting the abstract methods of the interface.

http://www.tutorialspoint.com/java/java_interfaces.htm

What is an interface?

- Along with abstract methods an interface may also contain:
  - constants i.e. final static fields
  - default methods
  - static methods
- Method bodies exist only for default methods and static methods.
- NOTE: Pre Java 8, Interfaces did not have static and default methods.

http://www.tutorialspoint.com/java/java_interfaces.htm

Syntax for an Interface

```
import java.lang.*;
//Any number of import statements

public interface NameOfInterface {
    //Any number of final, static fields
    //Any number of abstract method declarations
    //Any number of default and static method implementations
}
```

File name : NameOfInterface.java

http://www.tutorialspoint.com/java/java_interfaces.htm

Syntax for an Interface

File name : NameOfInterface.java

```
import java.lang.*;
//Any number of import statements

public interface NameOfInterface {
    //Any number of final, static fields
    //Any number of abstract method declarations
    //Any number of default and static method implementations
}
```

File name : IMammal.java

```
interface IMammal
{
    public void eat();
    public void travel();
 }
```

http://www.tutorialspoint.com/java/java_interfaces.htm

Implementing an Interface

- When a class implements an interface:
  - you can think of the class as signing a contract, agreeing to perform the specific behaviours of the interface.
- If a class does not perform all the behaviours of the interface, the class must declare itself as abstract.
- A class can implement more than one interface at a time.

http://www.tutorialspoint.com/java/java_interfaces.htm

Implementing an Interface

```
public class Mammal implements IMammal{
   public void eat(){
       System.out.println("Mammal eats");
   }

   public void travel(){
       System.out.println("Mammal travels");
   }

   public static void main(String args[]){
        Mammal m = new Mammal();
        m.eat();
        m.travel();
   }
}
```

Mammal.java

IMammal.java

```
interface IMammal
{
    void eat();
    void travel();
 }
```

http://www.tutorialspoint.com/java/java_interfaces.htm

Interfaces and Android

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:

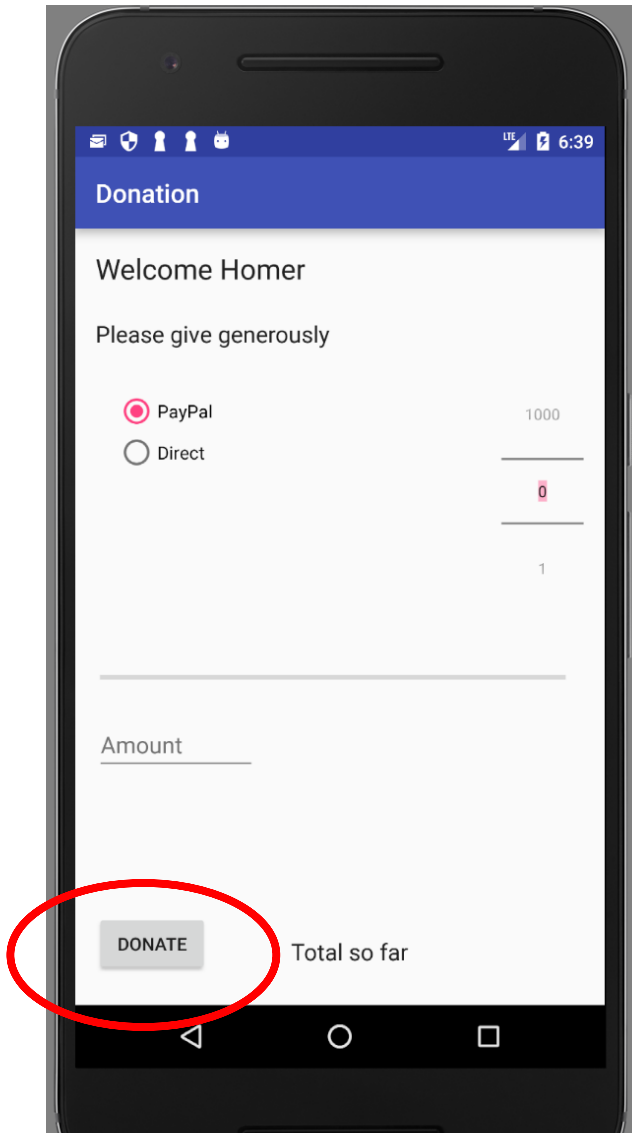
  1. Explicitly set in Resource File

  2. Using Listener Interface

  3. Using Anonymous Inner Class

- We need to master all three!

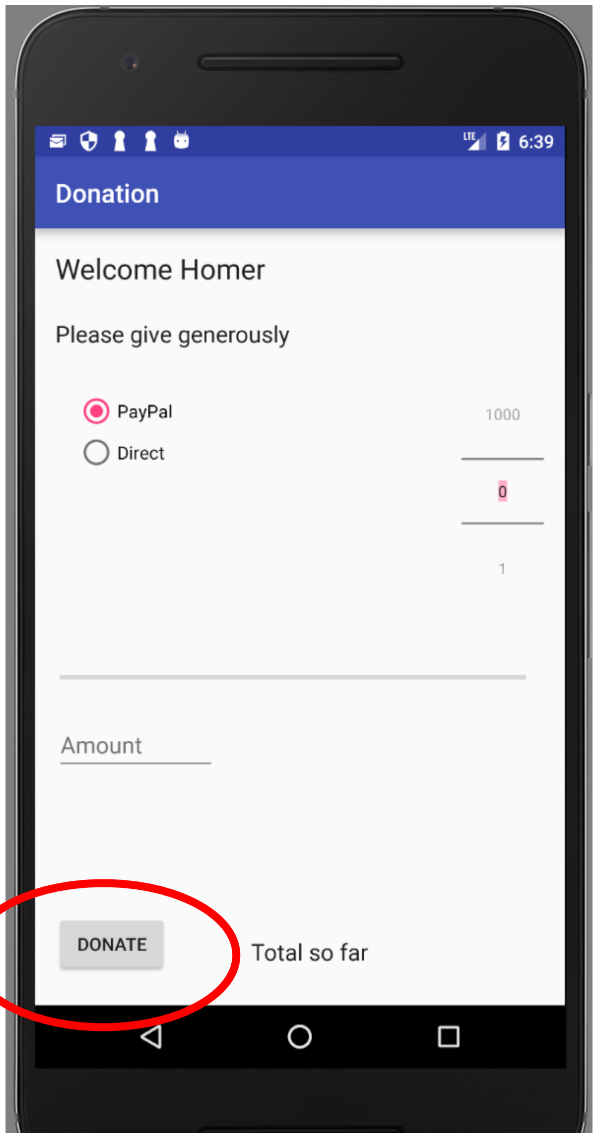
1. listeners explicitly set in a Resource File



1. listeners explicitly set in a Resource File

activity_donate.xml

```
<Button
    android:id="@+id/donateButton"
    android:layout_width="88dp"
    android:layout_height="48dp"
    android:layout_marginBottom="24dp"
    android:text="@string/donateButton"
    app:layout_constraintBottom_toBottomOf="parent"
    android:onClick="donateButtonPressed"
    android:layout_marginLeft="16dp"
    app:layout_constraintLeft_toLeftOf="parent" />
```



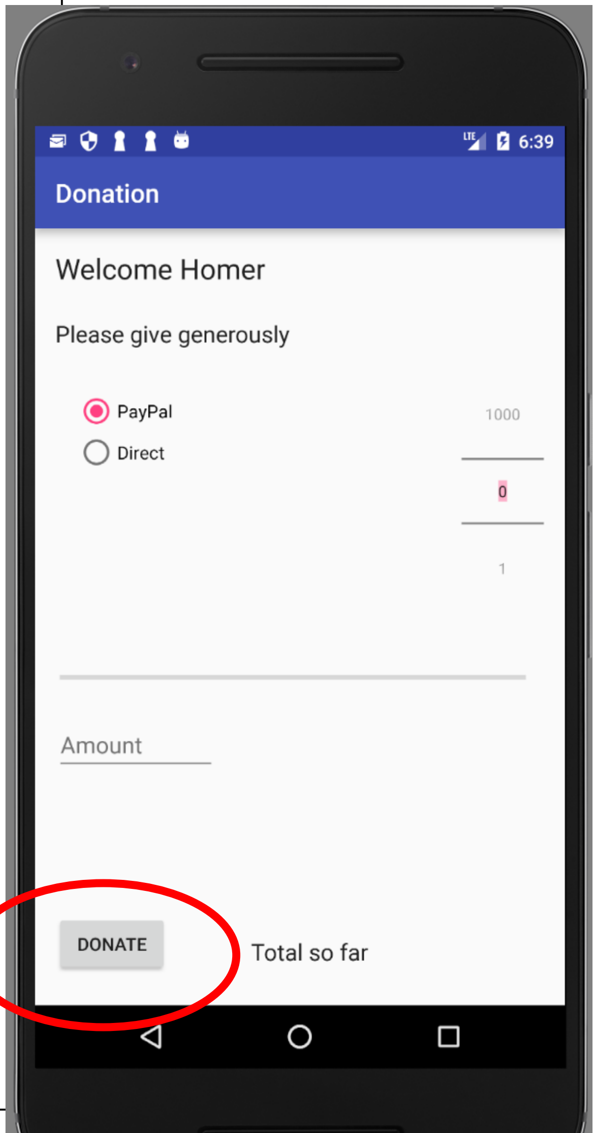
1. listeners explicitly set in a Resource File

activity_donate.xml

```
<Button
    android:id="@+id/donateButton"
    android:layout_width="88dp"
    android:layout_height="48dp"
    android:layout_marginBottom="24dp"
    android:text="@string/donateButton"
    app:layout_constraintBottom_toBottomOf="parent"
    android:onClick="donateButtonPressed"
    android:layout_marginLeft="16dp"
    app:layout_constraintLeft_toLeftOf="parent" />
```

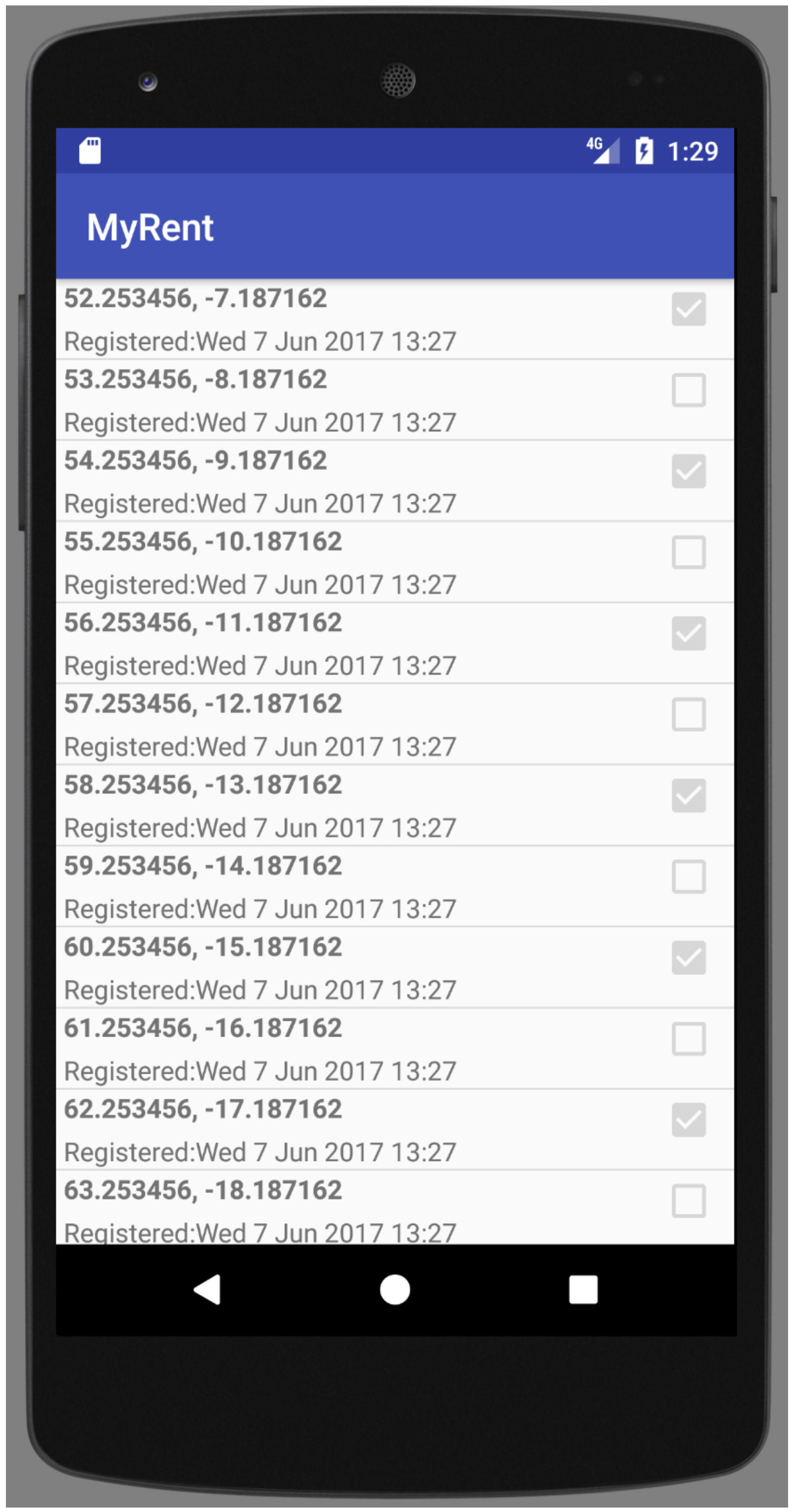
Donate.java

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

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

   protected void onCreate(Bundle savedInstanceState) {
         //code omitted
    }

   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 (totalDonated > target) {
            Toast toast = Toast.makeText(this, "Target Exceeded!", Toast.LENGTH_SHORT);
            toast.show();
            Log.v("Donate","Target Exceeded: " + totalDonated);
        }
        else  {
            totalDonated = totalDonated + donatedAmount;
            progressBar.setProgress(totalDonated);
            Log.v("Donate", amountPicker.getValue() + " donated by " + method
                          + "\nCurrent total " + totalDonated);
        }

        String totalDonatedStr = "$" + totalDonated;
        amountTotal.setText(totalDonatedStr);
    }
}
```



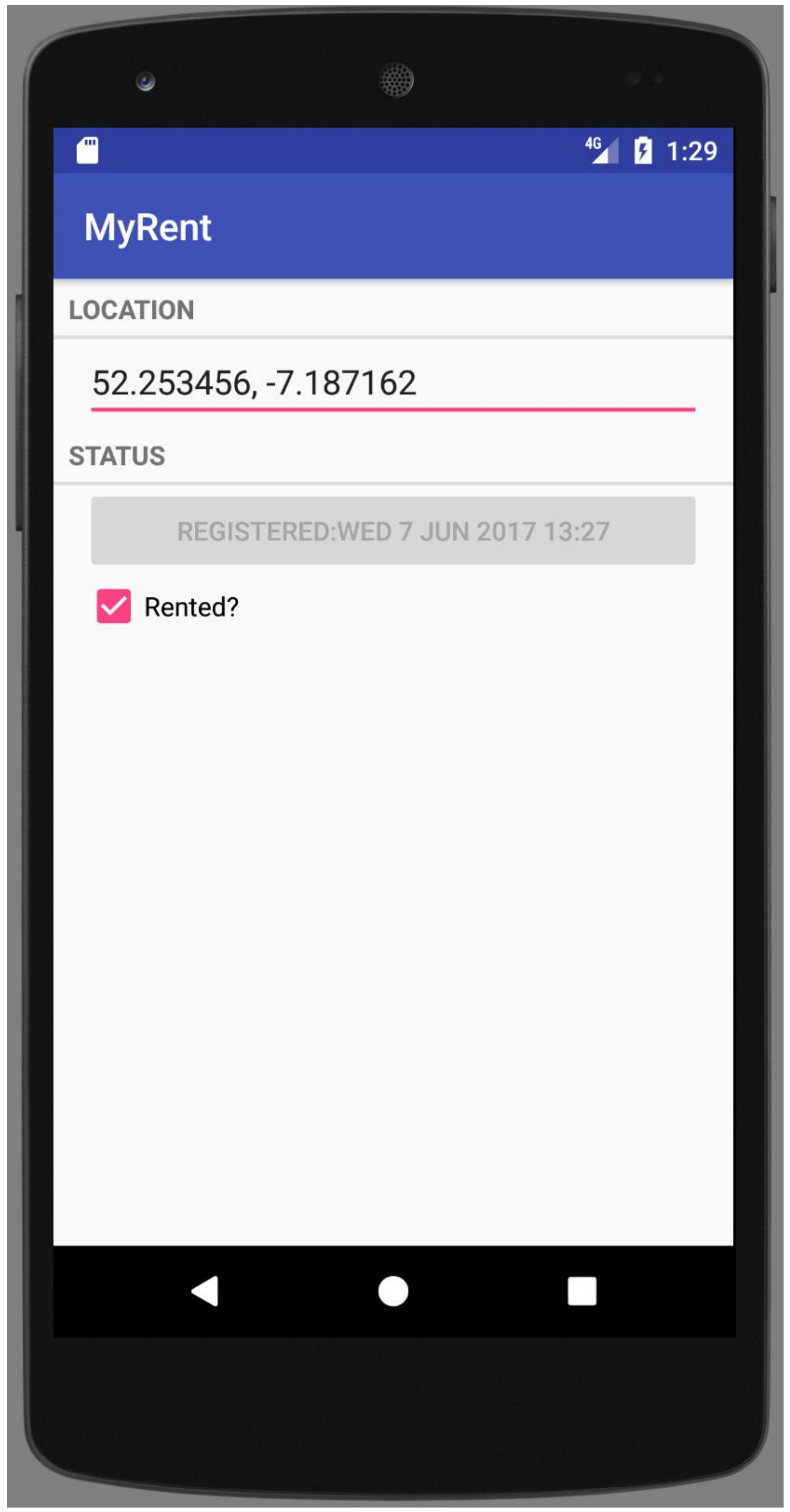
2. Using the Listener Interface

MyRent V02



MyRent

52.253456, -7.187162
Registered:Wed 7 Jun 2017 13:27
53.253456, -8.187162
Registered:Wed 7 Jun 2017 13:27
54.253456, -9.187162
Registered:Wed 7 Jun 2017 13:27
55.253456, -10.187162
Registered:Wed 7 Jun 2017 13:27
56.253456, -11.187162
Registered:Wed 7 Jun 2017 13:27
57.253456, -12.187162
Registered:Wed 7 Jun 2017 13:27
58.253456, -13.187162
Registered:Wed 7 Jun 2017 13:27
59.253456, -14.187162
Registered:Wed 7 Jun 2017 13:27
60.253456, -15.187162
Registered:Wed 7 Jun 2017 13:27
61.253456, -16.187162
Registered:Wed 7 Jun 2017 13:27
62.253456, -17.187162
Registered:Wed 7 Jun 2017 13:27
63.253456, -18.187162
Registered:Wed 7 Jun 2017 13:27



MyRent

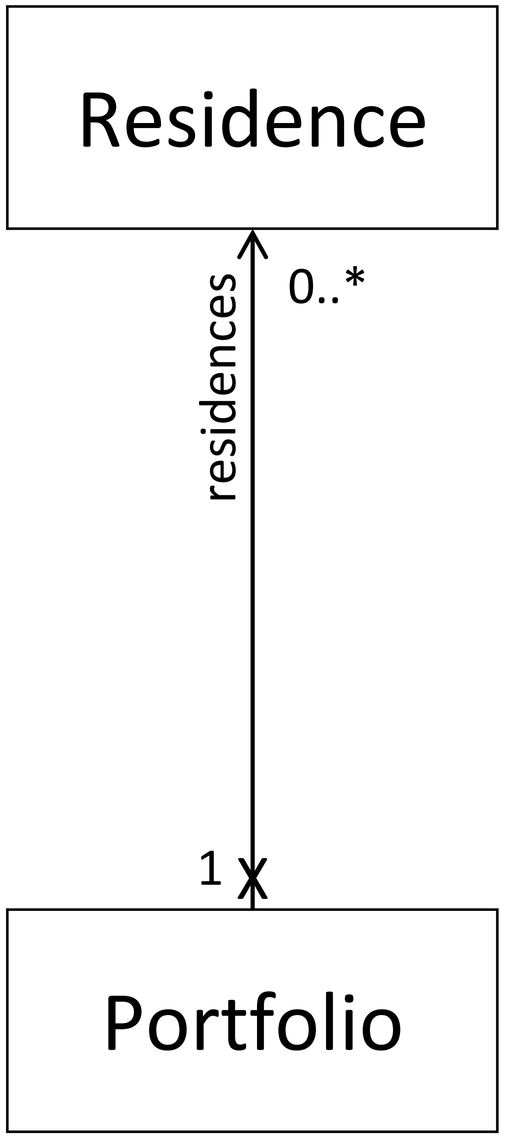
LOCATION

52.253456, -7.187162

STATUS

REGISTERED:WED 7 JUN 2017 13:27

Rented?

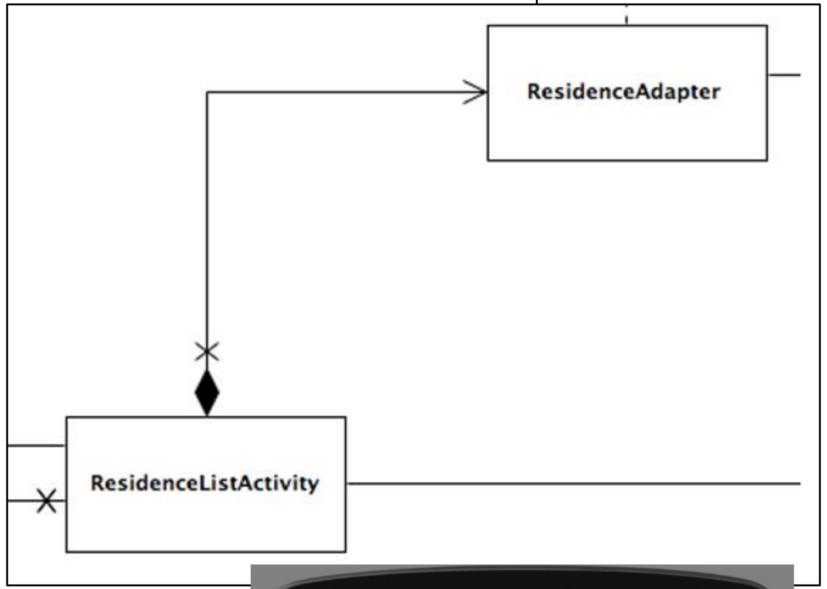


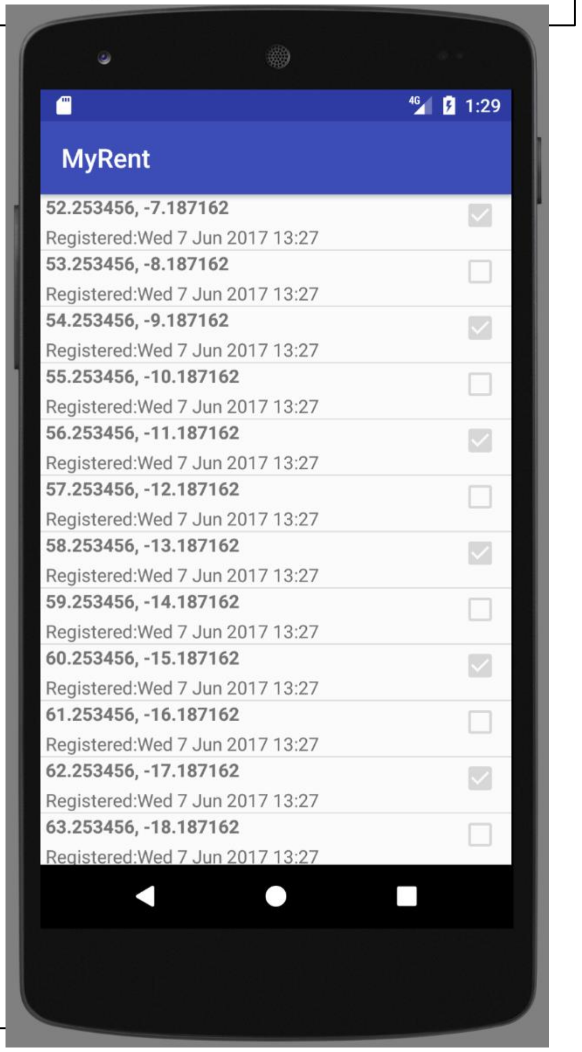
Application Object

```java
package org.wit.myrent.app;

import org.wit.myrent.models.Portfolio;
import android.app.Application;
import static org.wit.android.helpers.LogHelpers.info;

public class MyRentApp extends Application
{
    public Portfolio portfolio;

    @Override
    public void onCreate()
    {
        super.onCreate();
        portfolio = new Portfolio();

        info(this, "MyRent app launched");
    }
}
```

We use the application object to manage the list of residences i.e. an object of Portfolio.

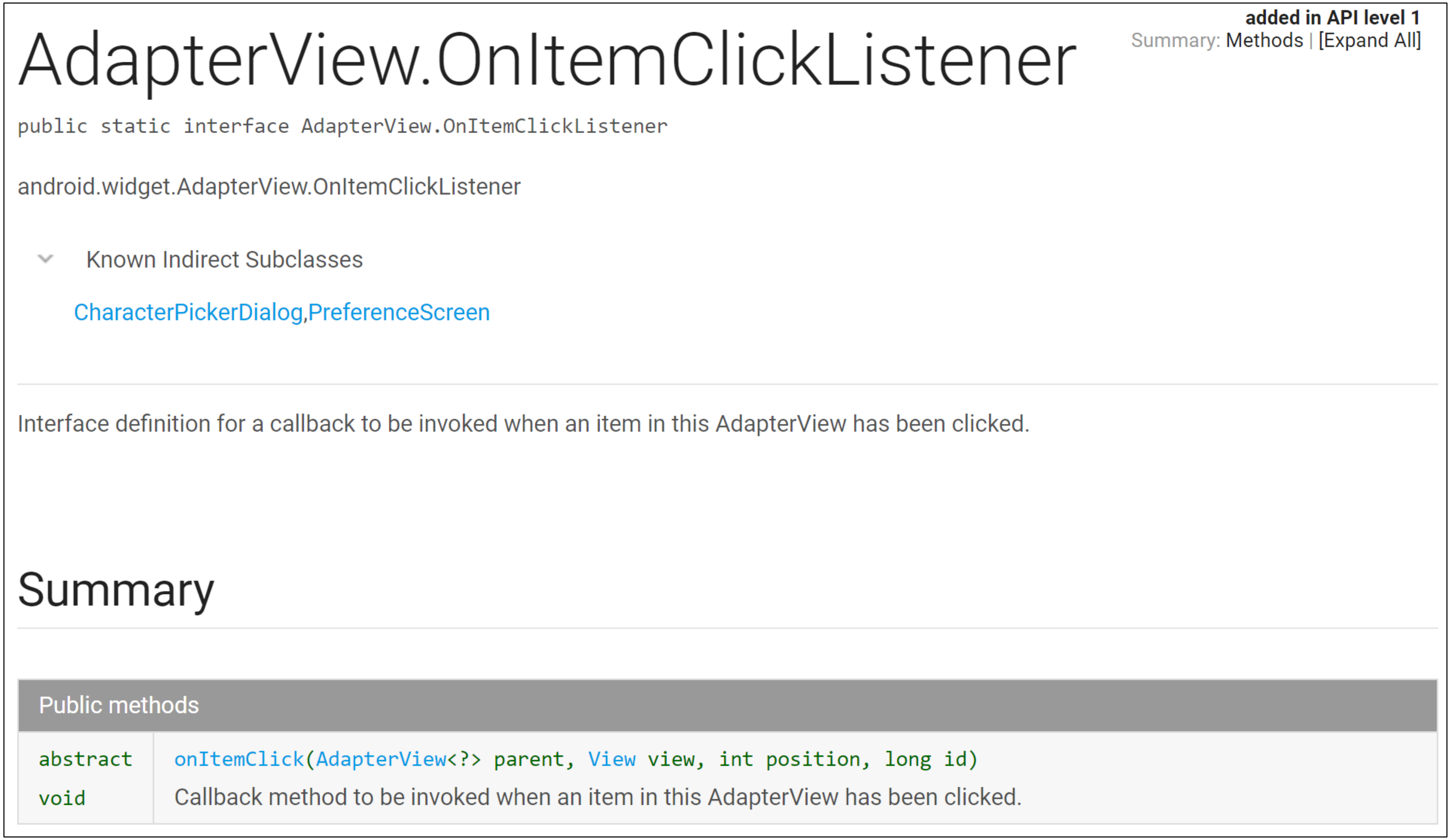
```
public class ResidenceListActivity extends AppCompatActivity implements AdapterView.OnItemClickListener
{
    private ListView listView;
    private Portfolio portfolio;
    private ResidenceAdapter adapter;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setTitle(R.string.app_name);
        setContentView(R.layout.activity_residence_list);

        listView = (ListView) findViewById(R.id.residenceList);
        MyRentApp app = (MyRentApp) getApplication();
        portfolio = app.portfolio;

        adapter = new ResidenceAdapter(this, portfolio.residences);
        listView.setAdapter(adapter);
        listView.setOnItemClickListener(this);
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Residence residence = adapter.getItem(position);
        Intent intent = new Intent(this, ResidenceActivity.class);
        intent.putExtra("RESIDENCE_ID", residence.id);
        startActivity(intent);
    }

    @Override
    public void onResume()
    {
        super.onResume();
        adapter.notifyDataSetChanged();
    }
}
```





MyRent

52.253456, -7.187162 Registered:Wed 7 Jun 2017 13:27	✓
53.253456, -8.187162 Registered:Wed 7 Jun 2017 13:27	☐
54.253456, -9.187162 Registered:Wed 7 Jun 2017 13:27	✓
55.253456, -10.187162 Registered:Wed 7 Jun 2017 13:27	☐
56.253456, -11.187162 Registered:Wed 7 Jun 2017 13:27	✓
57.253456, -12.187162 Registered:Wed 7 Jun 2017 13:27	☐
58.253456, -13.187162 Registered:Wed 7 Jun 2017 13:27	✓
59.253456, -14.187162 Registered:Wed 7 Jun 2017 13:27	☐
60.253456, -15.187162 Registered:Wed 7 Jun 2017 13:27	✓
61.253456, -16.187162 Registered:Wed 7 Jun 2017 13:27	☐
62.253456, -17.187162 Registered:Wed 7 Jun 2017 13:27	✓
63.253456, -18.187162 Registered:Wed 7 Jun 2017 13:27	☐



added in API level 1
Summary: Methods | [Expand All]

AdapterView.OnItemClickListener

```
public static interface AdapterView.OnItemClickListener
```

android.widget.AdapterView.OnItemClickListener

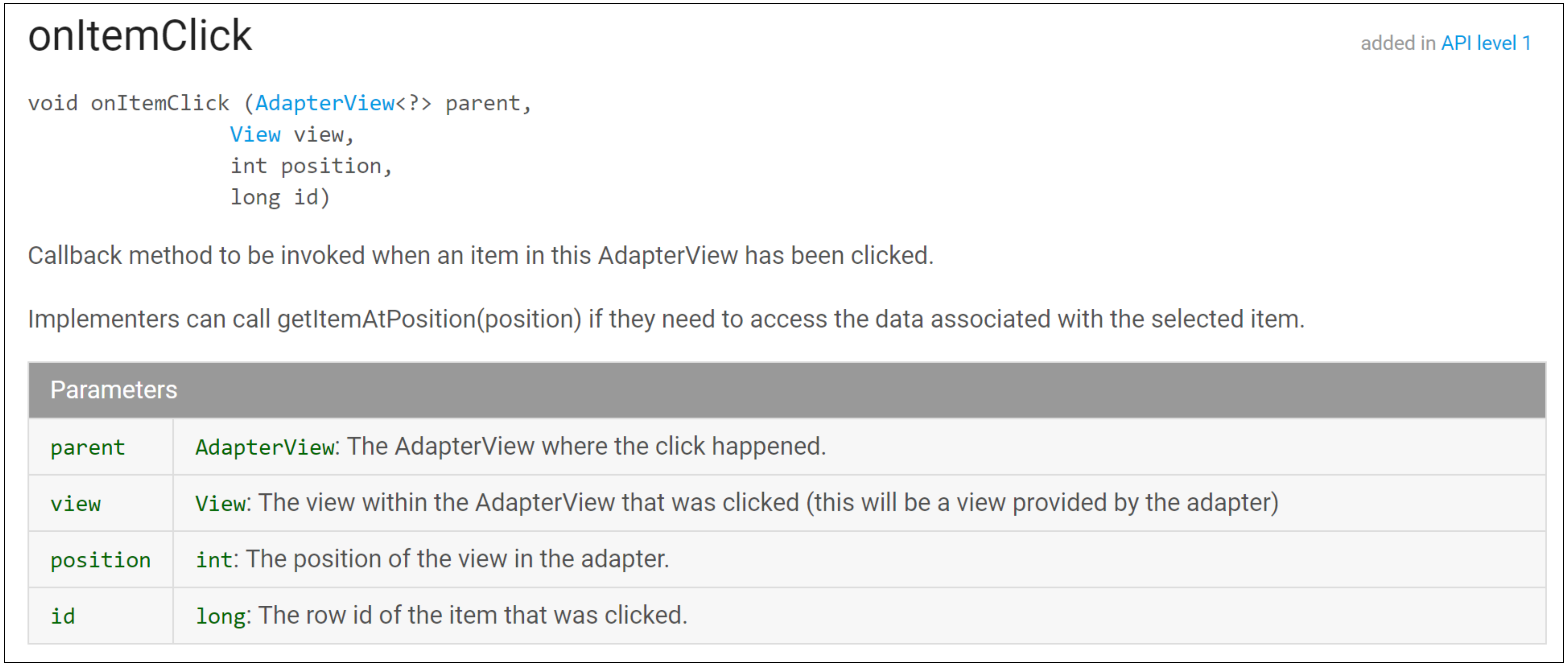
Known Indirect Subclasses

CharacterPickerDialog,PreferenceScreen

Interface definition for a callback to be invoked when an item in this AdapterView has been clicked.

Summary

Public methods	
`abstract void`	`onItemClick(AdapterView<?> parent, View view, int position, long id)` Callback method to be invoked when an item in this AdapterView has been clicked.



onItemClick

added in API level 1

```
void onItemClick (AdapterView<?> parent,
                View view,
                int position,
                long id)
```

Callback method to be invoked when an item in this AdapterView has been clicked.

Implementers can call getItemAtPosition(position) if they need to access the data associated with the selected item.

Parameters	
`parent`	`AdapterView`: The AdapterView where the click happened.
`view`	`View`: The view within the AdapterView that was clicked (this will be a view provided by the adapter)
`position`	`int`: The position of the view in the adapter.
`id`	`long`: The row id of the item that was clicked.

ResidenceListActivity

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
  Residence residence = adapter.getItem(position);
  Intent intent = new Intent(this, ResidenceActivity.class);
  intent.putExtra("RESIDENCE_ID", residence.id);
  startActivity(intent);
}
```

onItemClick

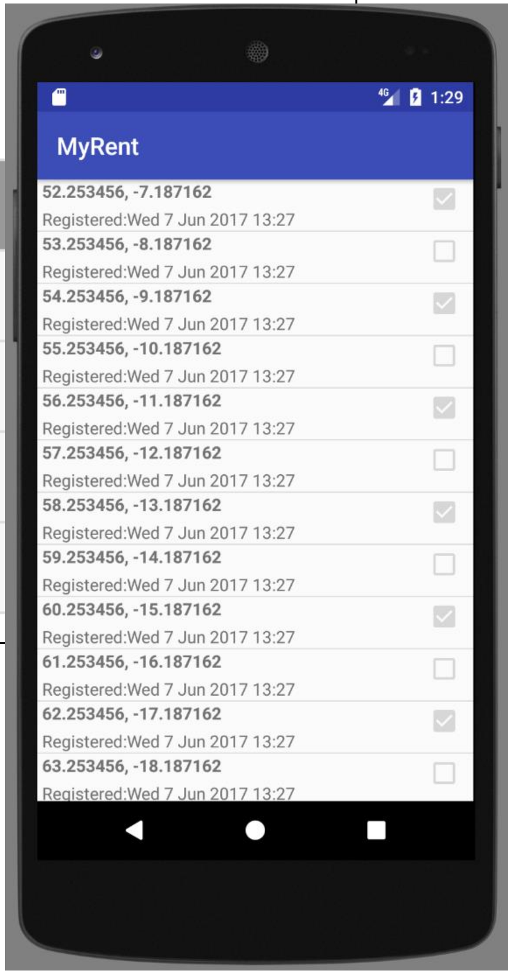
added in API level 1

```
void onItemClick (AdapterView<?> parent,
                View view,
                int position,
                long id)
```

Callback method to be invoked when an item in this AdapterView has been clicked.

Implementers can call getItemAtPosition(position) if they need to access the data associated with the selected item.

Parameters	
`parent`	`AdapterView`: The AdapterView where the click happened.
`view`	`View`: The view within the AdapterView that was clicked (this will be a view provided by the adapter)
`position`	`int`: The position of the view in the adapter.
`id`	`long`: The row id of the item that was clicked.

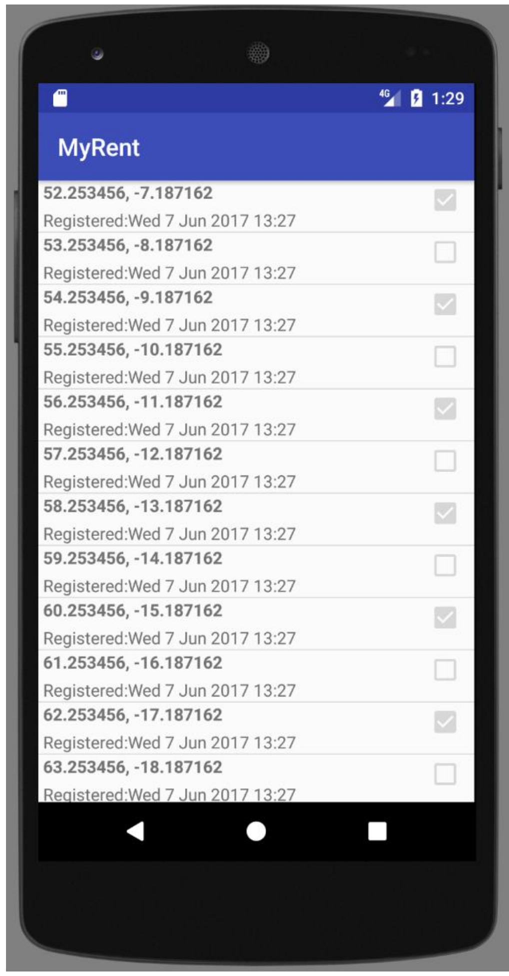


52.253456, -7.187162 Registered:Wed 7 Jun 2017 13:27	✓
53.253456, -8.187162 Registered:Wed 7 Jun 2017 13:27	☐
54.253456, -9.187162 Registered:Wed 7 Jun 2017 13:27	✓
55.253456, -10.187162 Registered:Wed 7 Jun 2017 13:27	☐
56.253456, -11.187162 Registered:Wed 7 Jun 2017 13:27	✓
57.253456, -12.187162 Registered:Wed 7 Jun 2017 13:27	☐
58.253456, -13.187162 Registered:Wed 7 Jun 2017 13:27	✓
59.253456, -14.187162 Registered:Wed 7 Jun 2017 13:27	☐
60.253456, -15.187162 Registered:Wed 7 Jun 2017 13:27	✓
61.253456, -16.187162 Registered:Wed 7 Jun 2017 13:27	☐
62.253456, -17.187162 Registered:Wed 7 Jun 2017 13:27	✓
63.253456, -18.187162 Registered:Wed 7 Jun 2017 13:27	☐

ResidenceListActivity

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
  Residence residence = adapter.getItem(position);
  Intent intent = new Intent(this, ResidenceActivity.class);
  intent.putExtra("RESIDENCE_ID", residence.id);
  startActivity(intent);
}
```

1. Retrieve the Residence object by its position in the list
2. Create a new Intent to start ResidenceActivity class.
   - Before starting it, put the ID of the object we retrieved into the 'extra' information passed to the intent.


1:29

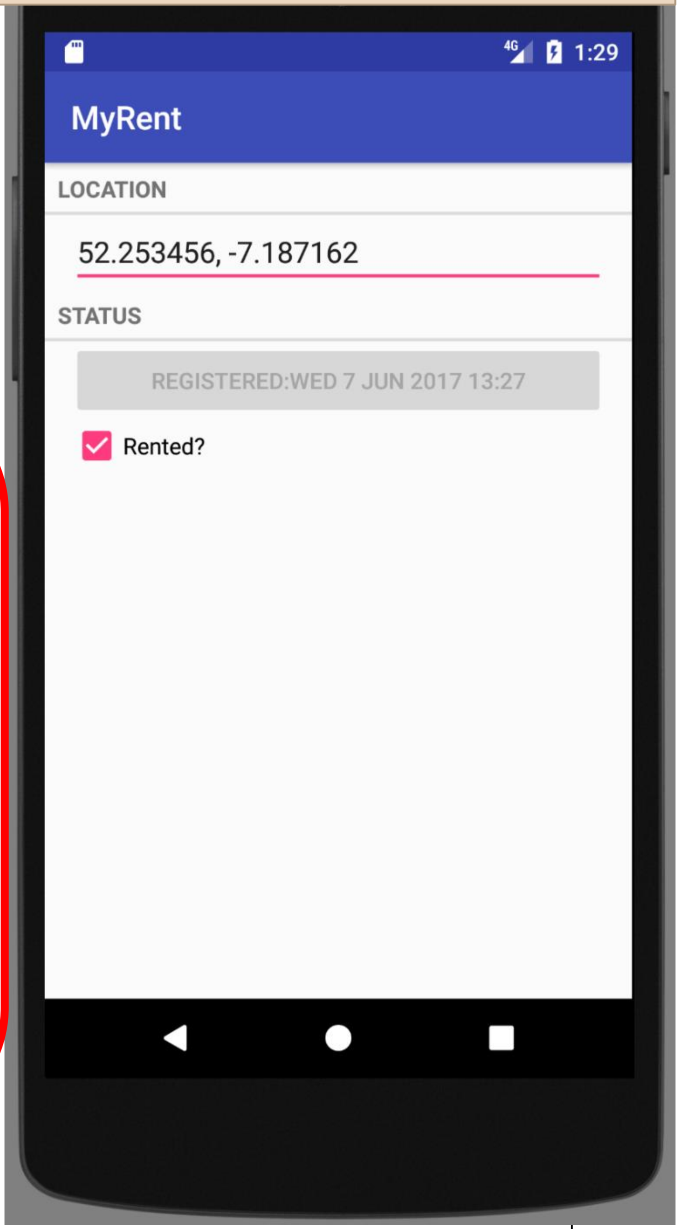
MyRent

52.253456, -7.187162 ✓
Registered:Wed 7 Jun 2017 13:27
53.253456, -8.187162 ☐
Registered:Wed 7 Jun 2017 13:27
54.253456, -9.187162 ✓
Registered:Wed 7 Jun 2017 13:27
55.253456, -10.187162 ☐
Registered:Wed 7 Jun 2017 13:27
56.253456, -11.187162 ✓
Registered:Wed 7 Jun 2017 13:27
57.253456, -12.187162 ☐
Registered:Wed 7 Jun 2017 13:27
58.253456, -13.187162 ✓
Registered:Wed 7 Jun 2017 13:27
59.253456, -14.187162 ☐
Registered:Wed 7 Jun 2017 13:27
60.253456, -15.187162 ✓
Registered:Wed 7 Jun 2017 13:27
61.253456, -16.187162 ☐
Registered:Wed 7 Jun 2017 13:27
62.253456, -17.187162 ✓
Registered:Wed 7 Jun 2017 13:27
63.253456, -18.187162 ☐
Registered:Wed 7 Jun 2017 13:27

Note: An Intent is a messaging object you can use to request an action from another app component.

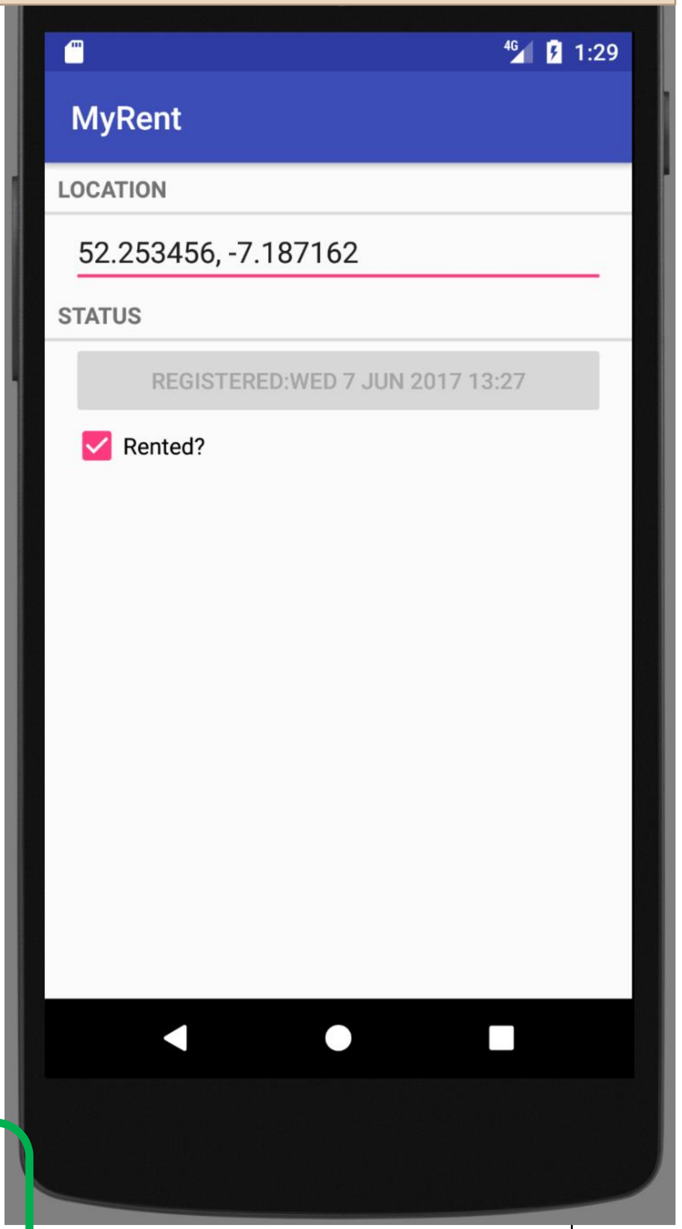
```
public class ResidenceActivity extends AppCompatActivity implements TextWatcher, OnCheckedChangeListener{

    private EditText geolocation;
    private Residence residence;
    private CheckBox rented;
    private Button dateButton;
    private Portfolio portfolio;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //omitted code
    }

    public void updateControls(Residence residence){
        //omitted code
    }

    @Override
    public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2)

    }

    @Override
    public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

    }

    @Override
    public void afterTextChanged(Editable editable) {
        //omitted code
    }

    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean isChecked)
        //omitted code
    }
}
```

ResidenceActivity



```java
public class ResidenceActivity extends AppCompatActivity implements TextWatcher, OnCheckedChangeListener{

    private EditText geolocation;
    private Residence residence;
    private CheckBox rented;
    private Button dateButton;
    private Portfolio portfolio;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //omitted code
    }

    public void updateControls(Residence residence){
        //omitted code
    }

    @Override
    public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2)

    }

    @Override
    public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {

    }

    @Override
    public void afterTextChanged(Editable editable) {
        //omitted code
    }

    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean isChecked)
        //omitted code
    }
}
```

ResidenceActivity

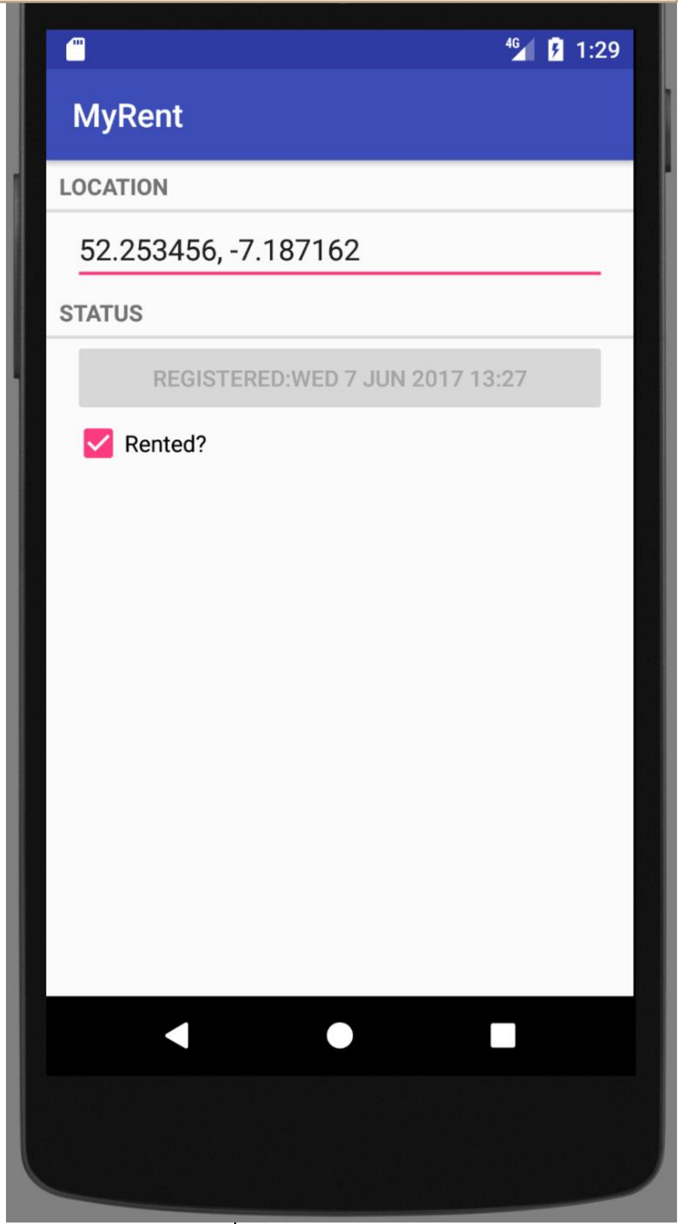


```
public class ResidenceActivity extends AppCompatActivity implements TextWatcher, OnCheckedChangeListener{

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

        geolocation = (EditText) findViewById(R.id.geolocation);
        residence   = new Residence();
        geolocation.addTextChangedListener(this);

        dateButton  = (Button)   findViewById(R.id.registration_date);
        dateButton.setEnabled(false);

        rented = (CheckBox) findViewById(R.id.isrented);
        rented.setOnCheckedChangeListener(this);

        MyRentApp app = (MyRentApp) getApplication();
        portfolio = app.portfolio;

        Long resId = (Long) getIntent().getExtras().getSerializable("RESIDENCE_ID");
        residence = portfolio.getResidence(resId);
        if (residence != null){
            updateControls(residence);
        }
    }

    @Override
    public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {
    }

    @Override
    public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
    }

    @Override
    public void afterTextChanged(Editable editable) {
        residence.setGeolocation(editable.toString());
    }

    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean isChecked) {
        Log.i(this.getClass().getSimpleName(), "rented Checked");
        residence.rented = isChecked;
    }

    //omitted code
}
```

ResidenceActivity

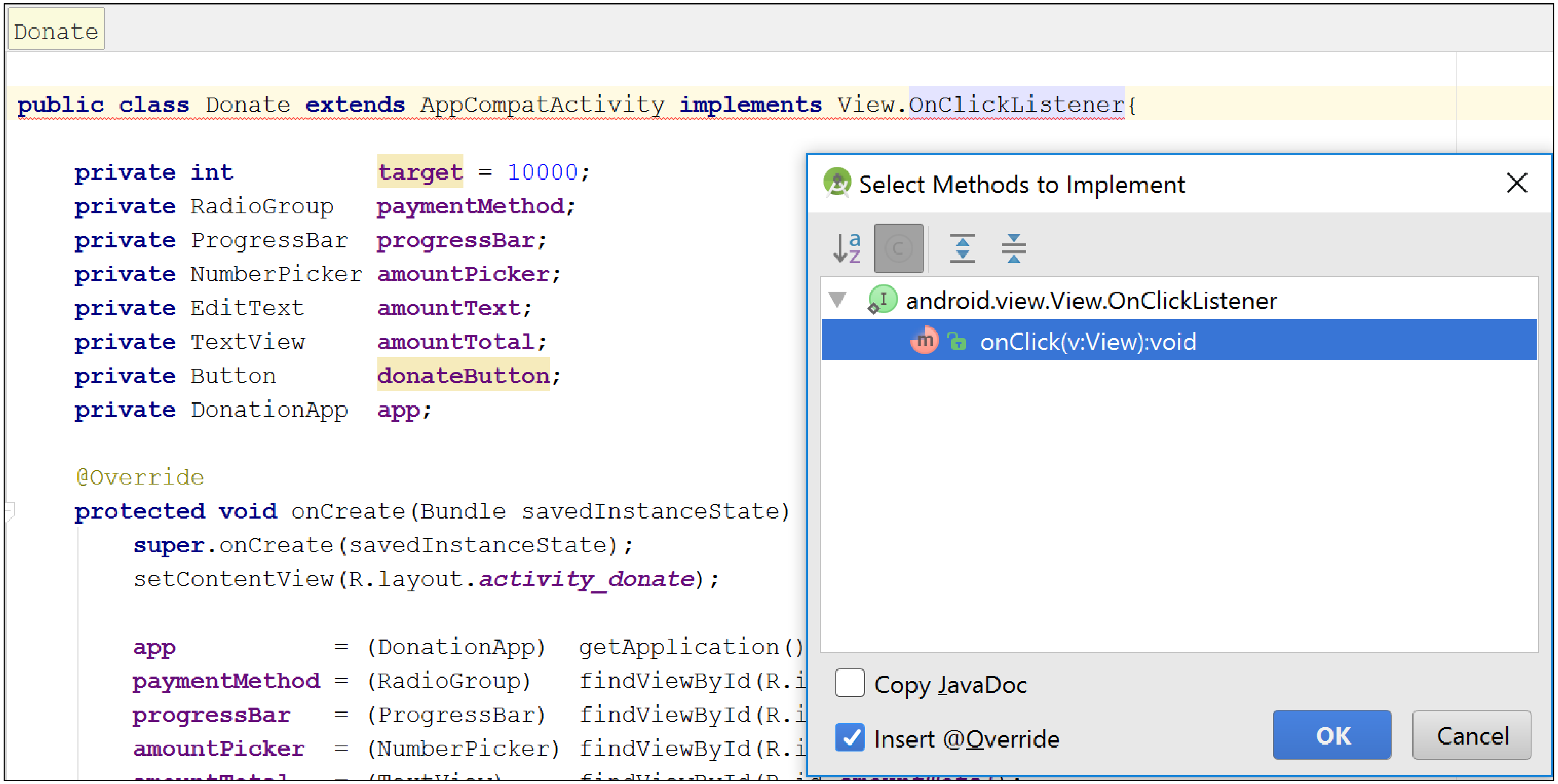
Registering the handlers

Event handler methods



3. Using Anonymous Inner Classes

Donate Button (using Listener Interface)



```
public class Donate extends AppCompatActivity implements View.OnClickListener{

    private int           target = 10000;
    private RadioGroup    paymentMethod;
    private ProgressBar   progressBar;
    private NumberPicker  amountPicker;
    private EditText      amountText;
    private TextView      amountTotal;
    private Button        donateButton;
    private DonationApp   app;

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

        app           = (DonationApp)   getApplication()
        paymentMethod = (RadioGroup)    findViewById(R.i
        progressBar   = (ProgressBar)   findViewById(R.i
        amountPicker  = (NumberPicker)  findViewById(R.i
```

Select Methods to Implement

- android.view.View.OnClickListener
  - onClick(v:View):void

[ ] Copy JavaDoc

[✓] Insert @Override

Donate Button (using Listener Interface)

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

    app           = (DonationApp)  getApplication();
    paymentMethod = (RadioGroup)   findViewById(R.id.paymentMethod);
    progressBar   = (ProgressBar)  findViewById(R.id.progressBar);
    amountPicker  = (NumberPicker) findViewById(R.id.amountPicker);
    amountTotal   = (TextView)     findViewById(R.id.amountTotal);
    amountText    = (EditText)     findViewById(R.id.amountText);
    donateButton  = (Button)       findViewById(R.id.donateButton);

    amountPicker.setMinValue(0);
    amountPicker.setMaxValue(1000);
    progressBar.setMax(target);

    donateButton.setOnClickListener(this);
}

@Override
public void onClick(View v) {
    donateButtonPressed(v);
}
```

Listener Callback Method

Registering the listener

```java
public class Donate extends AppCompatActivity {

    private int           target = 10000;
    private RadioGroup    paymentMethod;
    private ProgressBar   progressBar;
    private NumberPicker  amountPicker;
    private EditText      amountText;
    private TextView      amountTotal;
    private Button        donateButton;
    private DonationApp   app;

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

        app           = (DonationApp)  getApplication();
        paymentMethod = (RadioGroup)   findViewById(R.id.paymentMethod);
        progressBar   = (ProgressBar)  findViewById(R.id.progressBar);
        amountPicker  = (NumberPicker) findViewById(R.id.amountPicker);
        amountTotal   = (TextView)     findViewById(R.id.amountTotal);
        amountText    = (EditText)     findViewById(R.id.amountText);
        donateButton  = (Button)       findViewById(R.id.donateButton);

        amountPicker.setMinValue(0);
        amountPicker.setMaxValue(1000);
        progressBar.setMax(target);

        donateButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                donateButtonPressed(v);
            }
        });
    }

    //code omitted
}
```

- Can be rewritten using an anonymous inner class.
- No need to implement the listener at class level.
- Can only be done with functional interfaces (i.e. those with only one abstract method).

2. Implementing Listener Interface

```
public class Donate extends AppCompatActivity implements View.OnClickListener{

    //code omitted

    @Override
    protected void onCreate(Bundle savedInstanceState) {
       //code omitted
        donateButton.setOnClickListener(this);
   }

    @Override
    public void onClick(View v) {
        donateButtonPressed(v);
    }
}
```

3. Anonymous Inner Class

```
donateButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                donateButtonPressed(v);
            }
        });
```

Lambdas and functional interfaces

```java
donateButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                donateButtonPressed(v);
            }
        });
```

Lambda functions eliminate the need for anonymous classes when dealing with functional interfaces (i.e. with one method)

```java
donateButton.setOnClickListener((v) → { donateButtonPressed(v); });
```

Questions?

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit