warwick C<>ding

by Wesley Haigh

Lecture 2

Java, intents and button clicks

Recap

- Everything seen in an app are different **Activities.**
- Activities consist of a **Java** file and a **XML** file. There is also a sub XML file that is included in the main activity XML file.
- In the **res** folder there are several sub folders:
  - **/drawable** - where any images or backgrounds are kept.
  - **/layout** - where the layouts for activities are kept.
  - **/values** - where the strings, styles etc. are kept.
- XML is the language used to define how activities look. They have a similar syntax to HTML.
- To set properties of controls in XML, such as the height, you start it **android:*propertyName.***

Recap Exercise

- Remove the action bar (the bar at the top)
- Change the app's icon
- Add an EditText control to the layout. Name it password.
- Restructure the layout so that all elements (apart from the robot) are inside a LinearLayout. ← should have done this last lesson
- Your app should now look like what you see on the right

Java Files

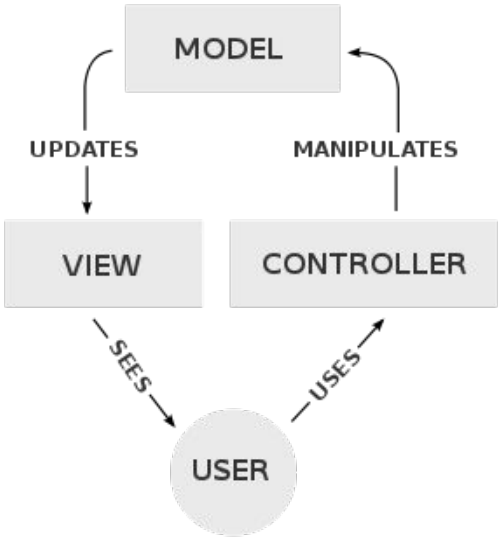
The heart of an app

Java

- Java is the programming language used in Android for all of the logic and backend processes.
- This next section will use a lot of **Object Oriented Programming** concepts and terminology.
- Every program works with data that can be described by entities (objects or events) from real life.
- For example we could have charity software that has objects such as donors and events such as payments etc.

Java: The Framework

- An important concept of Android is the MVC (Model-View-Controller) framework for activities.
- An important concept of Android is the MVC (Model-View-Controller) framework for activities.
- The **Model** is the java class that is part of the activity (in our case MainActivity.java)
- The **View/Controller** is the xml layout (activity_main.xml)



Java: Connecting Controls

- You can get hold of controls defined in the xml file in the java file.

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

    EditText password = (EditText) findViewById(R.id.password);
}
```

- This creates an **EditText** variable and finds the control (view) with id=password in the content view (activity_main.xml)
- Once you have the reference to the view you can start using it in your code.

Java: Buttons

- You can get a button reference in the same way

```
Button startButton = (Button) findViewById(R.id.startButton);
```

- Create a function called start (We will use this for our button click)

```
public void start(View view) {

}
```

- We are passing our function start a View for one of the purposes of adding button clicks

Java: Buttons

- There are 2 ways you can add an on click action to a button
  - Using the android:onClick property in xml

```
<Button
    android:id="@+id/startButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/red_button_selector"
    android:textColor="@android:color/white"
    android:textStyle="bold"
    android:textSize="20sp"
    android:onClick="start"
    android:paddingLeft="25dp"
```

This will call a function called “start” in the connected java file which accepts the arguments of a View.

Java: Buttons

- The other method is adding an onClickListener to the button object in java.

```
Button startButton = (Button) findViewById(R.id.startButton);
startButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //call start function here or put other code.
        start(null); //could also pass View v.
    }
});
```

The advantage of this is that it will always work, in some cases the onClick property in xml doesn’t work.

Java: Exercise

- Modify MainActivity.java so that when a user clicks on the start button the text entered into the EditText is displayed in some way.
- You could use a Toast or a TextView or a Dialog to display the text.

Java: My Style

- Have function called getXMLControls() that gets all relevant controls for that activity and store then in activity wide variables.
- Then all controls can be referenced in an function without having to call findViewById again or worry about having multiple references.

```
public class MainActivity extends AppCompatActivity {
    private EditText _password;
    private Button _startButton;

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

        getXMLControls();
    }

    private void getXMLControls() {
        _startButton = (Button) findViewById(R.id.startButton);
        _password = (EditText) findViewById(R.id.password);
    }
```

Java: Intent

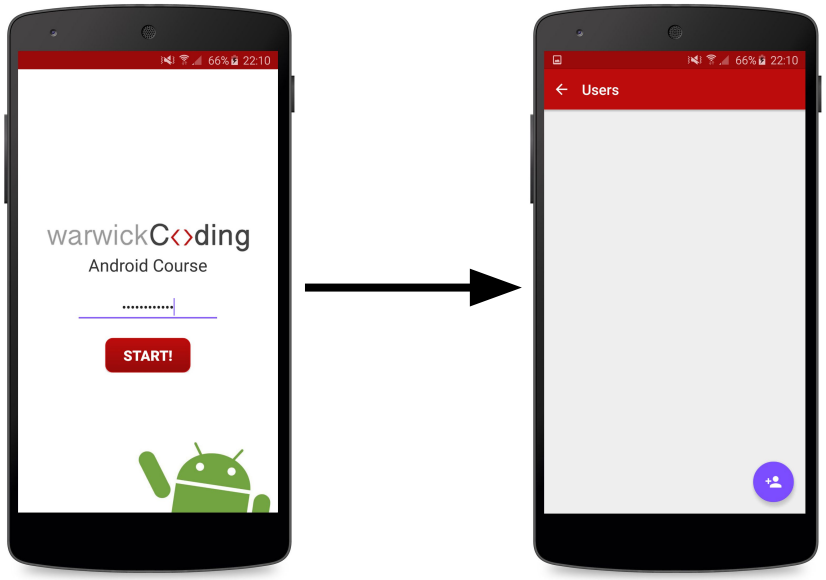
- Create a second activity called UsersActivity
- You can use **Intents** in java to move between activities. You make an “Intent” to open another activity on the phone.
- Modify the start button to make it start the UsersActivity

```
public void start(View view) {
    Intent intent = new Intent(this, UsersActivity.class);
    startActivity(intent);
}
```

- The arguments passed into the constructor are
    - The Context (**this**), this will always be the activity that is currently shown (normally **this**)
    - The java class of the activity we wish to go to (UserActivity.class)

End of Lecture Exercise

- Modify MainActivity.java so that the app only moves to UsersActivity when the user types a certain String into the password field (eg. your name). Any other String is rejected with a message.
- Add an image for the Floating Action Button on UsersActivity in the drawables folder and display it.



warwick

C<>ding