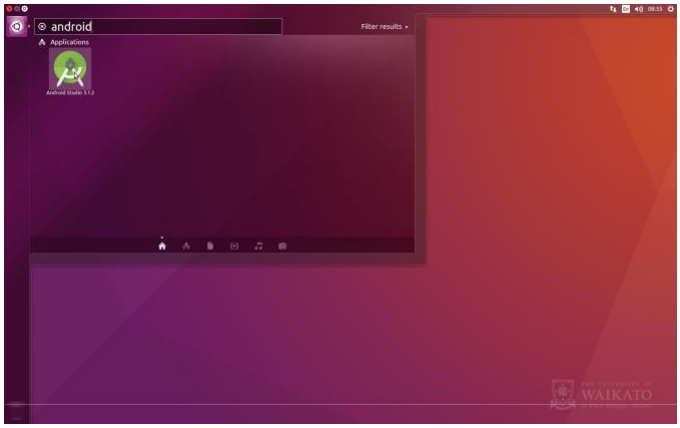
Android Exercises

Contents

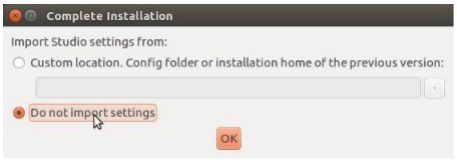
1 Running Android Studio

1. Start Android Studio via the launcher

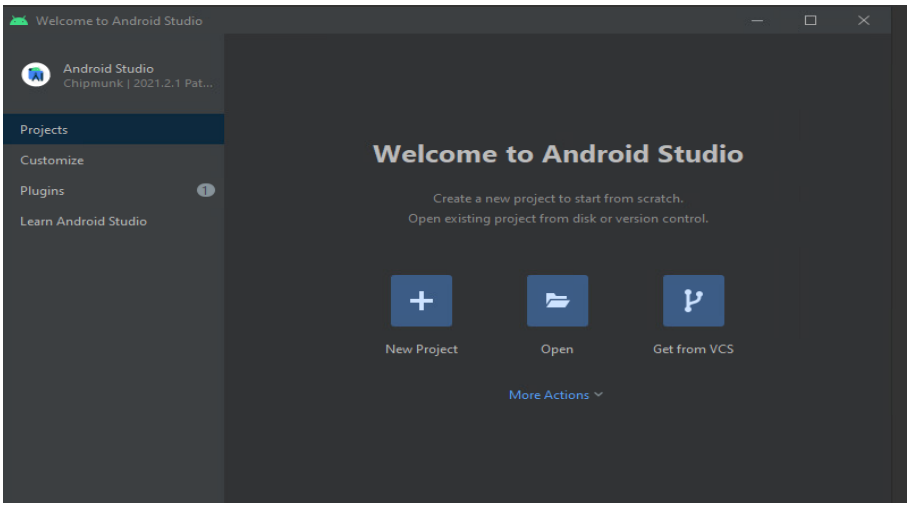


   A terminal window will open and print progress information. Do not close this terminal.

2. If asked, choose "Do not import settings".



3. In the Setup Wizard, you must close the "Updates" popup in order to proceed. The close button ("x") is only visible on mouse hover.

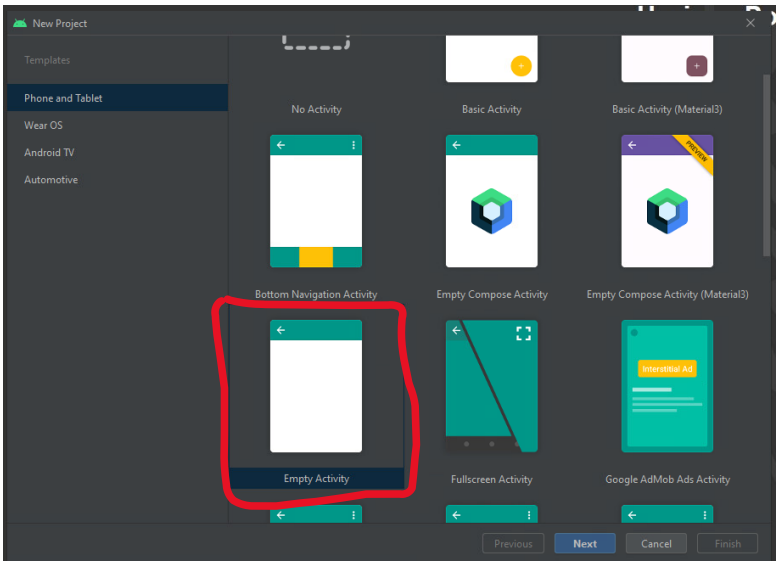


   Continue through the Setup Wizard, all the defaults are fine.

2 Creating your First App

1. In the Welcome to Android Studio window, click Start a new Android Studio project. Or if you have a project opened, select File > New Project.

2. Choose Empty Activity



3. Specify parameters:

   Name: you can pick (in our case it's "HelloWorld")

   Package name: leave as default

   Save location: you can pick!

   - H: drive is your network drive, which you can access from any windows computer on the UoW network
   - Your "Documents" or "Home" folder used to point to your H: drive
   - BUT, it doesn't anymore! Now it points to a local folder that is only available on that specific machine
   - SO, if you are using Android Studio in the labs, make sure you save your projects in your H: drive rather than Documents or Home
   - You can now find your H: drive under "This PC"

   Language: Java

   Minimum API level: Android 6.0 (Marshmallow)

   Leave everything else as default

4. Click "Finish"
   … be patient, wait for the progress at the bottom of the page to finish!

   Note!
   • The first time you build with Android Studio, it will download a bunch of dependencies
   • This will take a long time!
   • Especially if you are on the lab computers.
   • Be patient!

3 Running your First App

Run on the emulator

Android studio includes the ability to run your application on an emulated device. You can create a new emulator as follows:

Tools -> Device manager -> Create virtual device

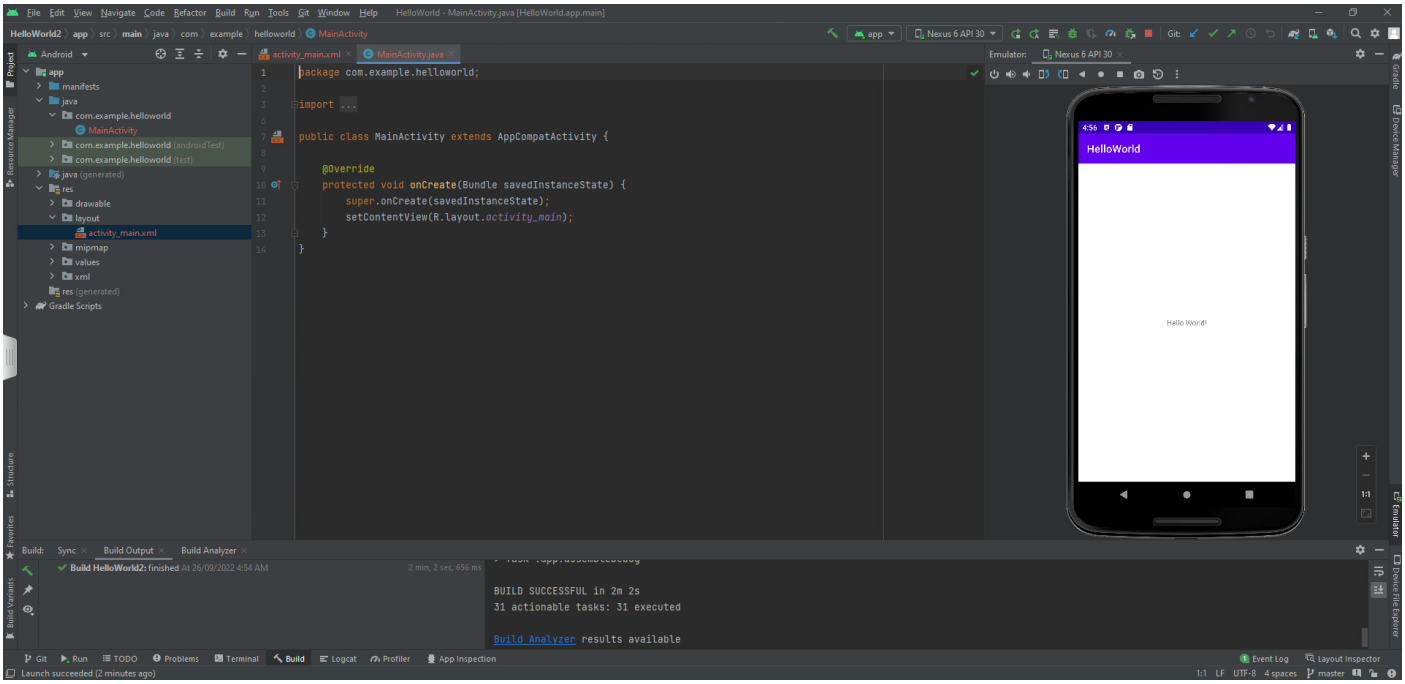
Choose any virtual device which is pre-installed on the lab machines

You can run your application on the emulator to:

- Test your functionality, i.e. check that your app runs as expected

Test your layout on multiple devices with different shapes and sizes

You will get something like this:



Using a physical phone

You can run your Android app on your own physical device!

There are a couple of things you will need to do:

- Enable “Developer Options” on your phone

Settings -> About Phone -> Software Information -> Build Number -> tab a bunch of times

- Enable USB debugging on your phone

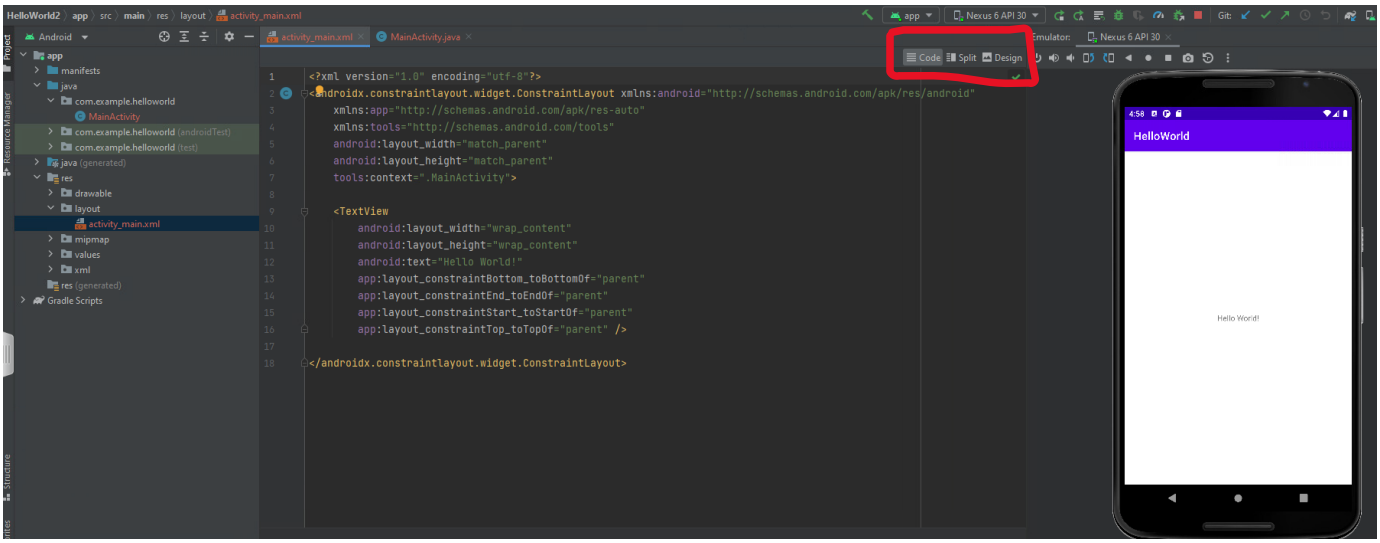
Settings -> Developer options -> Enable USB debugging

- Connect your phone to your PC/laptop using a USB cable

Find your phone in Android Studio and click “Run”

4 Creating a User Interface

1. Select the file app/res/layout/activity_main.xml in the Project window on the left.
2. There are two ways to create your app's user interface: the graphical editor (Design) and the XML editor (code). You can switch between using the tabs as shown in the image below:.

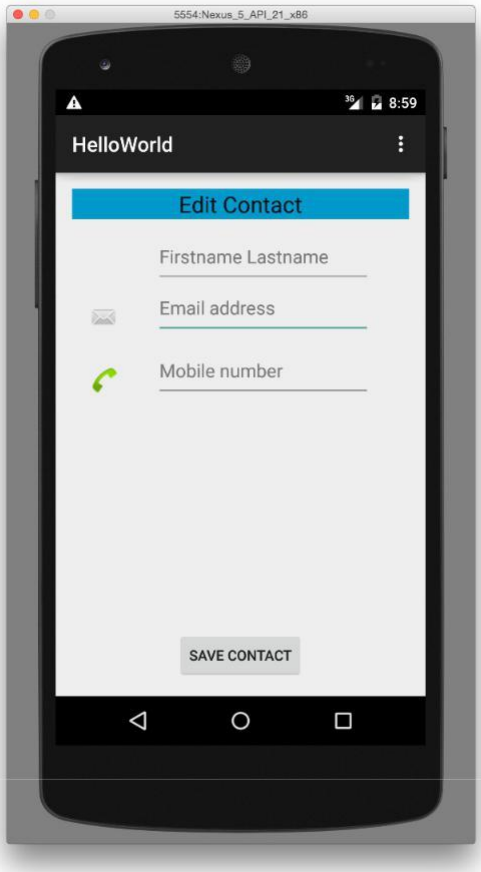


3. The default layout created by Android Studio is a ConstraintLayout, which is new and rather complex. For these exercises it is recommended to use RelativeLayout. Switch to the Text editor and replace the contents with the following:

```
<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="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".MainActivity">

</RelativeLayout>
```

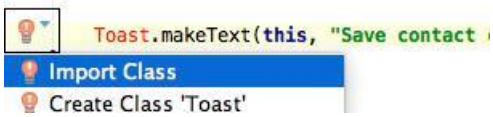
4. Use the graphical editor to create the layout below. You will have to decide which UI elements to use and investigate how to set their attributes. If you get stuck you can look at the XML provided at the end.



5. Now make the button respond to a click/tap. Read http://developer.android.com/guide/topics/ui/controls/button.html#HandlingEvents.
6. Set the onClick attribute of the button to saveContact in activity_main.xml.
7. In the Project window, open app/java/comp575.helloworld/MainActivity.java.
8. Create a method called saveContact and add this statement to the method:

```
public void saveContact(View view) {
    Toast.makeText(this, "Save contact clicked", Toast.LENGTH_SHORT).show();
}
```

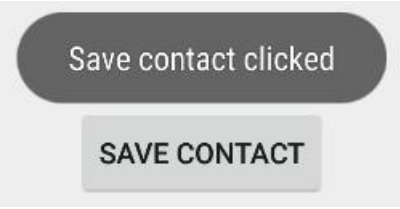
You may have issues where Android Studio cannot resolve a symbol, in this case either Toast or View. To fix this the class needs to be imported. You can do



or write the following at the top of the file after thepackage statement:

```
import android.widget.Toast;
import android.view.View;
```

9. When you run the app and click the button you will see a brief popup message.



Note: The onCreate method in MainActivity.java is the first method called when your app starts. You'll see it has the statement

```
setContentView(R.layout.activity_main);
```

Behind the scenes this reads the UI layout defined inactivity_main.xml then creates and initialise the UI elements as Java objects so that they can be displayed on the screen.

5 Reading Input Values

You need to find the EditText objects to retrieve what values the user entered into them. Read about usingfindViewById in http://developer.android.com/guide/topics/resources/accessing-resources.html.

1. Insert these statements into the saveContact method:

```
public void saveContact(View view) {
        EditText nameField = (EditText) findViewById(R.id.name);
        String name = nameField.getText().toString();
        Toast.makeText(this, "Saved contact for " + name,
                        Toast.LENGTH_SHORT).show();
}
```

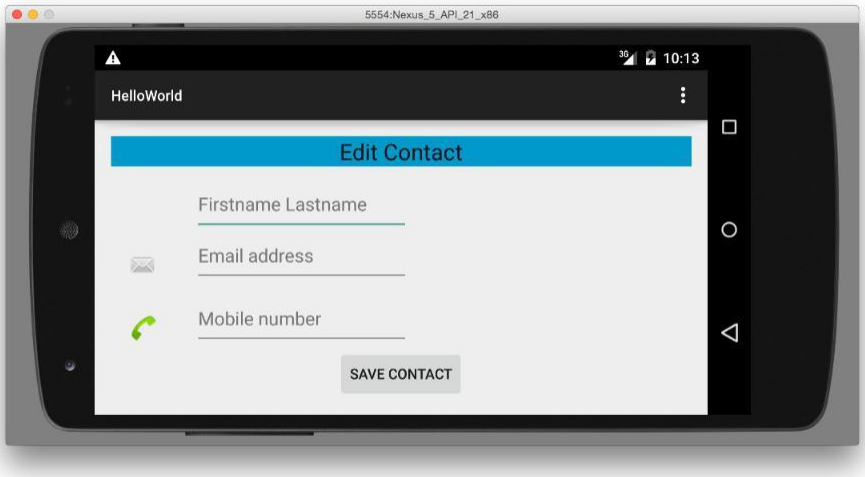
Note: R.id.name is defined in the XML file by

```
android:id="@+id/name"
```

2. Now add statements to get the phone number and email address.
3. Modify the Toast.makeText statement to display all the details of the contact in one message.

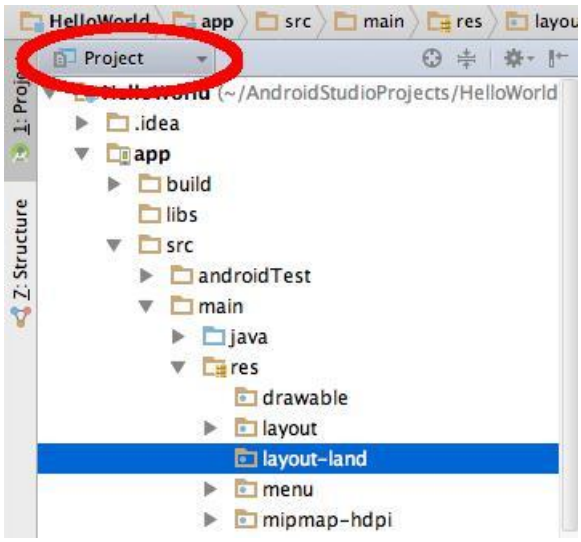
6 Handling Device Rotation

Unlike desktop systems, it is easy to rotate a mobile device between portrait and landscape orientations. Rotate the emulator using the buttons in the panel on the right to check that your layout is reasonable in both orientations. In landscape you are likely to have a lot of blank screen space:



We could use the space more effectively by showing a list of contacts on the left and the contact editor on the right. This would mean that we have different UI layouts for portrait and landscape. This is reasonably easy in Android.

1. Change the Project window from Android view to Project view. Then right click on HelloWorld/app/src/main/res and create a new directory called layout-land (must be that name).



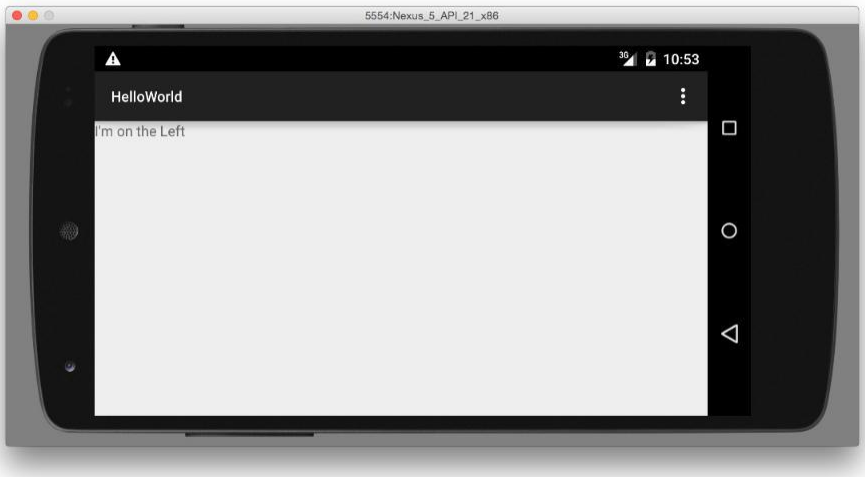
2. Right click on layout-land and create a new Layout resource file (New > Layout resource file)
   File name: activity_main
   Root element: LinearLayout

   Now you have two layout files with the same name, but in different layout folders.

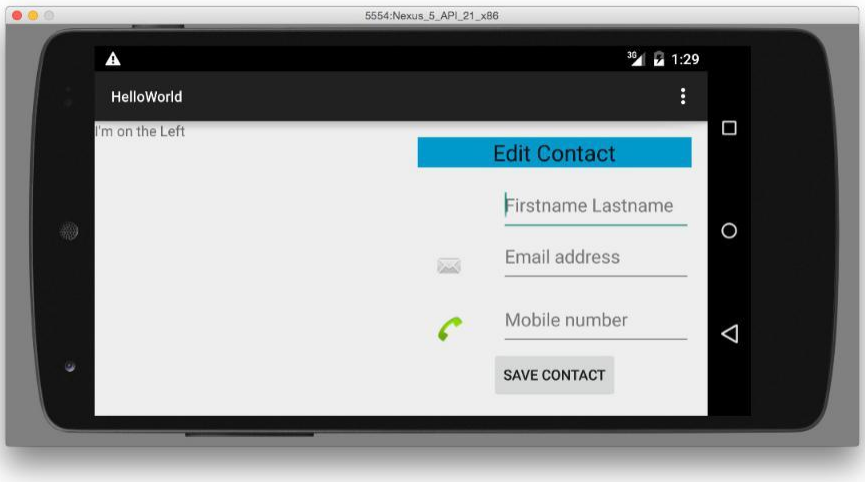
3. Now run your app and see what happens when you change between portrait and landscape.

4. Open layout-land/activity_main.xml in Text mode and change the orientation attribute of LinearLayout to horizontal.

5. Add a TextView to give something like this, then run your app.



Now we want to use the portrait view within the landscape view, to give something similar to the following.



6. In res/layout copy activity_main.xml by right clicking.
7. Now right click on res/layout and select paste. This will prompt you to name the new file. Name itcontact_editor.xml. This means we have a separate file with the view for editing contacts.
8. Edit res/layout/activity_main.xml so it contains the following:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <include layout="@layout/contact_editor" />
</LinearLayout>
```

This will now include our contact_editor.xml layout. Run the app to see that the portrait view is still the same.

9. To include the contact editor view in landscape, we must editres/layout-land/activity_main.xml to be the following:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:orientation="vertical">

        <TextView
            android:id="@+id/textViewLeft"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="I'm on the Left" />
    </LinearLayout>

    <include
        layout="@layout/contact_editor"
        android:layout_width="fill_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:orientation="vertical" />
</LinearLayout>
```

By setting android:layout_width to fill_parent and setting android:layout_weight to the same value, we can position two linear layout views next to each other with equal widths. Try change the weights to different value and see how the ratio changes.

10. Run the app and change rotations. Try editing a text field and see what happens when you rotate the device.

7 Using a List View

Now that we have the left side free when in landscape, we are going to use it for showing a list of all the previously added contacts.

1. First we need to change the TextView to a ListView. Editres/layout-land/activity_main.xml. Replace TextView with ListView, then remove android:text and change the id to @+id/contactsListView.

```
<ListView
    android:id="@+id/contactsListView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

2. Now we are going to define an object to represent a contact. Create a new Java class in src/main/java/comp575.helloworld called Contact (as kind Class).

```
package comp575.helloworld;

public class Contact {
    public String name;
    public String email;
    public String mobile;

    public Contact(String name, String email, String mobile) {
        this.name = name;
        this.email = email;
        this.mobile = mobile;
    }

    public String toString() {
        return name;
    }
}
```

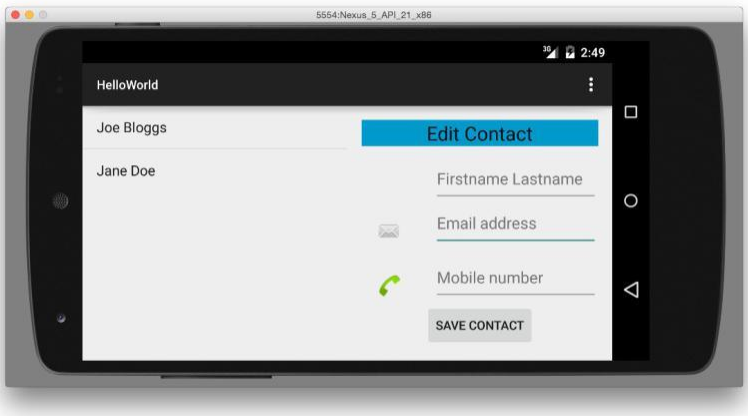
3. Edit src/main/java/MainActivity and add the following 3 lines to the top of the MainActivity class:

```
public class MainActivity extends ActionBarActivity {
    private ArrayList<Contact> contacts = new ArrayList<Contact>(); private
    ArrayAdapter<Contact> adapter; private ListView contactListView;
```

4. Anything we add to the contacts ArrayList, we want to go into the ListView shown on the screen. This is done with the ArrayAdapter. Edit the onCreate method in src/main/java/MainActivity.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // Setup Adapter
    contactListView = (ListView) findViewById(R.id.contactsListView); adapter = new
    ArrayAdapter<Contact>(this,
            android.R.layout.simple_list_item_1, contacts);
    contactListView.setAdapter(adapter);
    // Add some Contacts
    contacts.add(new Contact("Joe Bloggs",
            "joe@bloggs.co.nz", "021123456"));
    contacts.add(new Contact("Jane Doe",
            "jane@doe.co.nz", "022123456"));
}
```

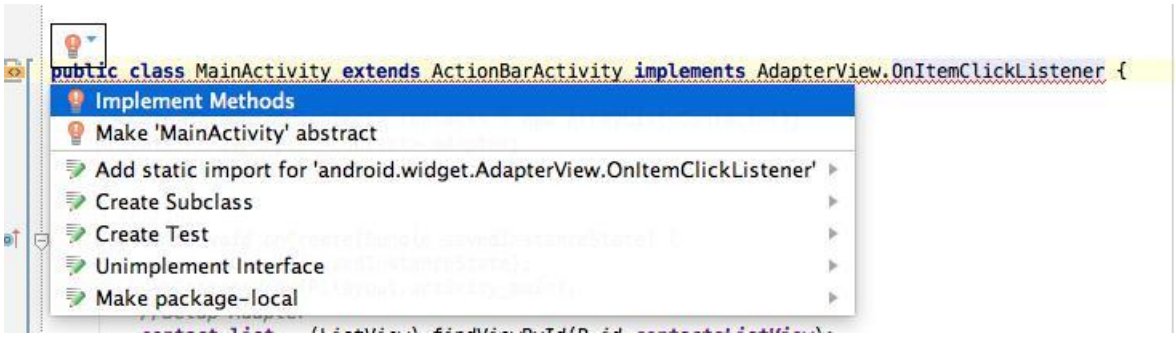
5. Try rotating the device to portrait. If the app crashes, edit theonCreate method to allow the app to work in portrait mode. Check the Logcat window to see the exception.
   Hint: R.id.contactsListView does not exist in portrait view.



8 Updating the Contact List

Now we will add a contact to the list of the left when the users clicks Save Contact.

1. Change the saveContact method in src/main/java/MainActivity so that the new contact is added to the ArrayList.
2. Run the app and see if the ListView is updated. If not, it is because the ArrayAdapter does not know the ArrayList of contacts has changed. Read http://developer.android.com/reference/android/widget/ArrayAdapter.html to find how to do this (there are a few ways).
3. Edit the MainActivity class so that it implements AdapterView.OnItemClickListener. This will cause an error and a lightbulb should appear. Use it to implement the missing onItemClick method.



4. In the onCreate method after we set the adapter, we need to set the on item click listener.

```
contactListView.setAdapter(adapter);
contactListView.setOnItemClickListener(this);
```

5. Now we can edit the onItemClick method so that is shows us which item was clicked.

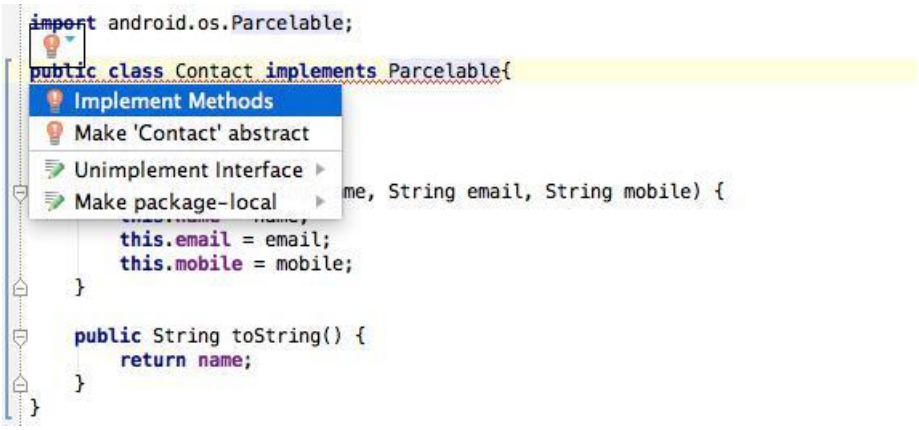
```
Contact contact = (Contact) parent.getAdapter().getItem(position);
Toast.makeText(parent.getContext(), "Clicked " + contact,
        Toast.LENGTH_SHORT).show();
```

6. Write some code in the onItemClick method to show the contact in the edit contact form.
7. In the saveContact method, update the contact's information if the person already exists. There are a few ways to check if the contact is in the ArrayList. The first is to loop through the ArrayList and manually look for the contact. Another way would be to override the equals method in the Contact class so that it checks if the names are equal. This allows you to use the built-in contains method in the ArrayList.

9 Saving Instance on Rotation

You may have noticed that when you rotate the device, the ListView resets back to the default contacts we add in the onCreate method. For example if you save a new contact, it will not appear if you rotate the device from landscape to portrait and back again. We can fix this by adding the ArrayList to the bundle used for saving the apps state.

1. First, we need to make our Contact object (src/main/java/Contact) implement the Parcelable interface. Use the lightbulb helper to implement the missing methods.



2. The describeContents method can be left default (return 0).

3. In order to save the object, we must put it within a parcel. To do this we write all of the contacts string variables to the parcel like so:

```
@Override
public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeString(email);
        dest.writeString(mobile);
}
```

4. We then add a new constructor which takes a Parcel object, and we read each string from the parcel. Note we must do it in the same order as when we wrote them to the parcel.

```
public Contact(Parcel in) {
        name = in.readString();
        email = in.readString();
        mobile = in.readString();
}
```

5. We also need to provide a static field called CREATOR which the system will use to create contacts from parcels:

```
public static final Creator<Contact> CREATOR = new Creator<Contact>() { @Override

        public Contact createFromParcel(Parcel in) {
                return new Contact(in);
        }

        @Override
        public Contact[] newArray(int size) {
                return new Contact[size];
        }
};
```

6. In our MainActivity class (src/main/java/MainActivity) we must override the method which saves the apps state, and save the contacts ArrayList. Add the following to the class.

```
@Override
public void onSaveInstanceState(Bundle savedState) {
        savedState.putParcelableArrayList("contacts", contacts);
        super.onSaveInstanceState(savedState);
}
```

7. The last thing needed is to restore the ArrayList. This can be done in theonCreate method. We check if there is a saved instance state, and get the ParcelableArrayList from the bundle.

```
if (savedInstanceState != null) {
        for (Parcelable contact : savedInstanceState.getParcelableArrayList( "contacts")) {

                contacts.add((Contact) contact);
        }
} else {
        contacts.add(new Contact("Joe Bloggs", "joe@bloggs.co.nz",
                        "021123456"));
        contacts.add(new Contact("Jane Doe", "jane@doe.co.nz",
                        "022123456"));
}
```

10 activity_main.xml for Section 4

```
<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="16dp"
        android:paddingLeft="16dp"
        android:paddingRight="16dp"
        android:paddingTop="16dp"
        tools:context=".MainActivity">

        <TextView
                android:id="@+id/page_title"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_alignParentLeft="true"
                android:layout_alignParentTop="true"
                android:layout_centerHorizontal="false"
                android:background="@android:color/holo_blue_dark"
                android:gravity="center_horizontal"
                android:text="Edit Contact"
                android:textAppearance="?android:attr/textAppearanceLarge" />

        <EditText
                android:id="@+id/name"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```

```
            android:layout_alignLeft="@+id/email"
            android:layout_below="@+id/page_title"
            android:layout_marginTop="20dp"
            android:ems="10"
            android:hint="Firstname Lastname"
            android:inputType="textPersonName" />

        <ImageView
            android:id="@+id/email_icon"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_below="@+id/name"
            android:layout_marginLeft="16dp"
            android:layout_marginTop="20dp"
            android:src="@android:drawable/sym_action_email" />

        <EditText
            android:id="@+id/email"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBottom="@+id/email_icon"
            android:layout_marginLeft="35dp"
            android:layout_marginTop="20dp"
            android:layout_toRightOf="@+id/email_icon"
            android:ems="10"
            android:hint="Email address"
            android:inputType="textEmailAddress">

            <requestFocus />
        </EditText>

        <ImageView
            android:id="@+id/phone_icon"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBottom="@+id/mobile"
            android:layout_alignLeft="@+id/email_icon"
            android:layout_marginTop="20dp"
            android:src="@android:drawable/sym_action_call" />

        <EditText
            android:id="@+id/mobile"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignLeft="@+id/email"
            android:layout_below="@+id/email"
            android:layout_marginTop="20dp"
            android:ems="10"
            android:hint="Mobile number"
            android:inputType="phone" />

        <Button
            android:id="@+id/btnSubmit"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_centerHorizontal="true"
            android:onClick="saveContact"
            android:text="Save contact" />
</RelativeLayout>
```