UNIVERSITY OF ALBERTA
FACULTY OF SCIENCE
Department of Computing Science

SOFTWARE DESIGN AND ARCHITECTURE

CAPSTONE ASSIGNMENT 1.4 TUTORIAL

Contents

This tutorial walks you through most the steps involved in converting the items only application into an application that has contacts. The steps in this tutorial are:

Table of Contents

You do not necessarily have to go through all these steps manually, you could opt to start this assignment from the peer review 4 starter code base. If you would like to opt to simply use the Peer Review 4 starter code base, you must still visit steps in the tutorial:

1. Clear the App Memory
18. Create and Implement the Contact Class
19. Create and Implement ContactList Class
20. Run the app

There are hints in these steps, so they are definitely worth checking out!

Step 1. Clear the App Memory

If you already have a previous version of SharingApp on your emulator, it is a good idea to clear the app's data.

Click the play button to run the app.

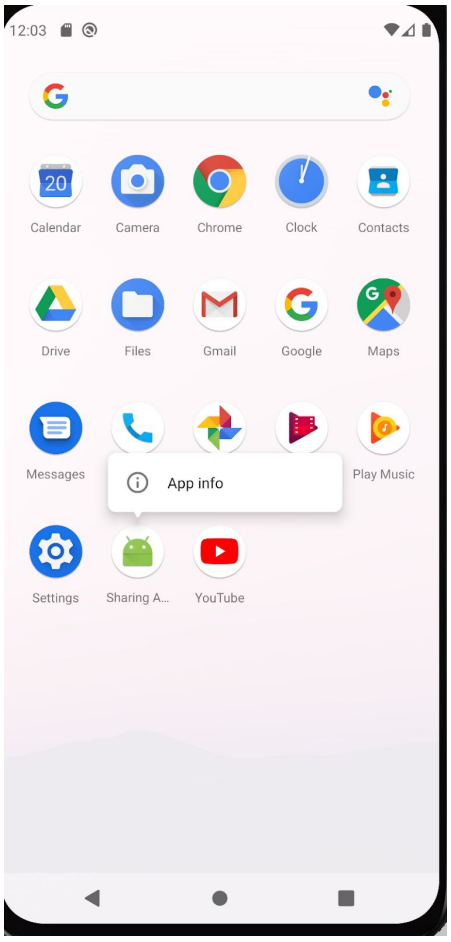
Be patient, the emulator may take a few minutes to load.

If the app launches and doesn't crash -- great! You are done. Apparently the changes you made to the app did not have an effect on the data being stored.

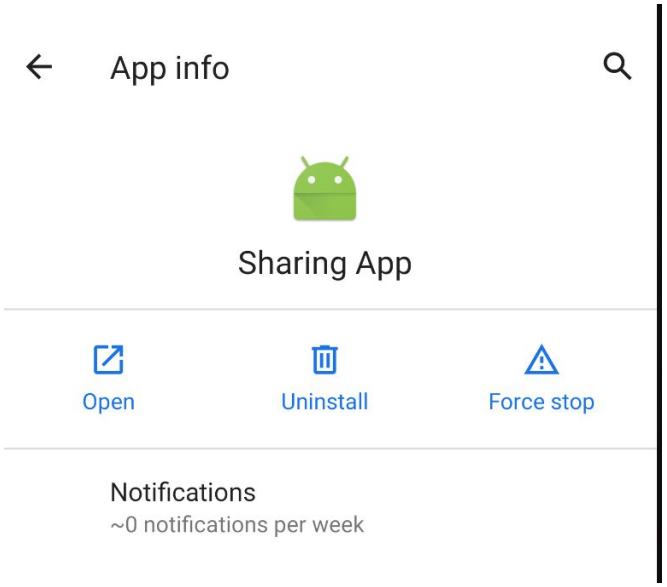
If it does crash -- don't worry. A message will appear

to inform you that the app has crashed. Click **OK**.

Then, swipe up and find the Sharing **App. Long click** on the icon to see the **App info**



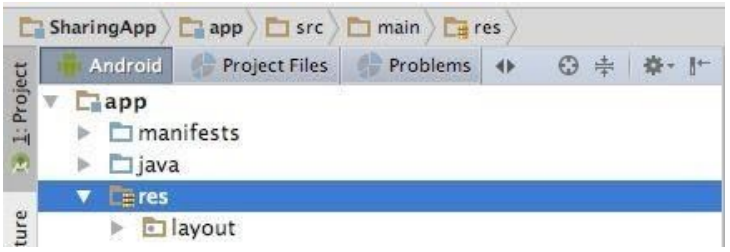
Click **App Info,** and then select **Uninstall**



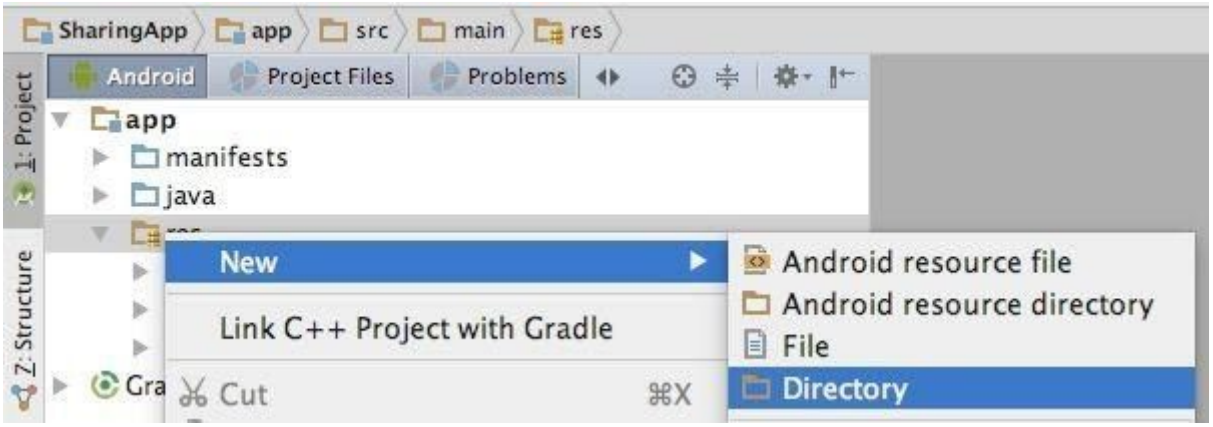
Now the app is uninstalled and all the previously stored data has been erased. The next time you run your app it shouldn't crash… unless you have a different error.

Step 2. Add a Menu

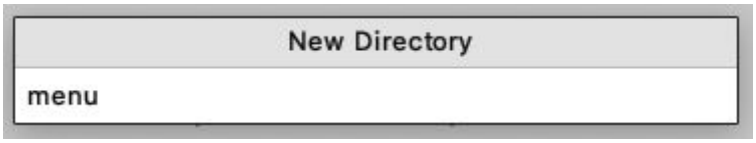
Locate the **res** folder in the project.



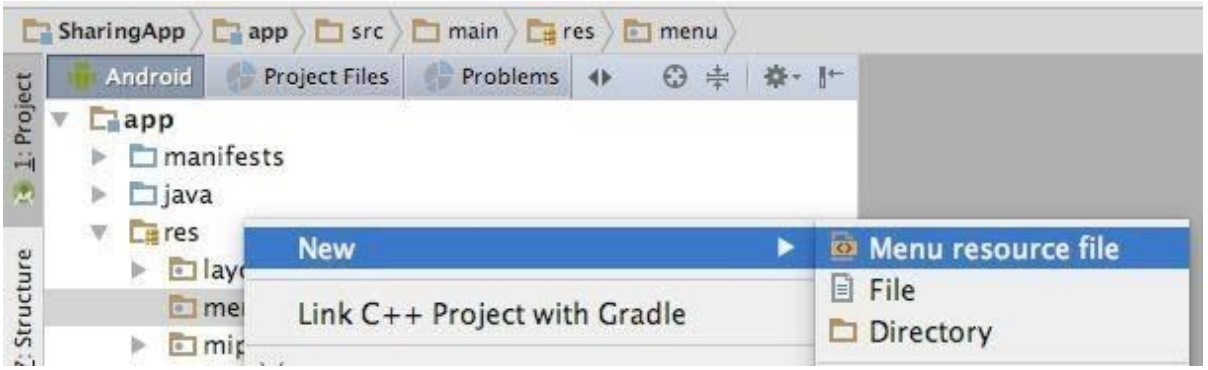
Add a **menu** folder to the **res** folder by right clicking on the **res** folder, then clicking **New** -> **Directory**



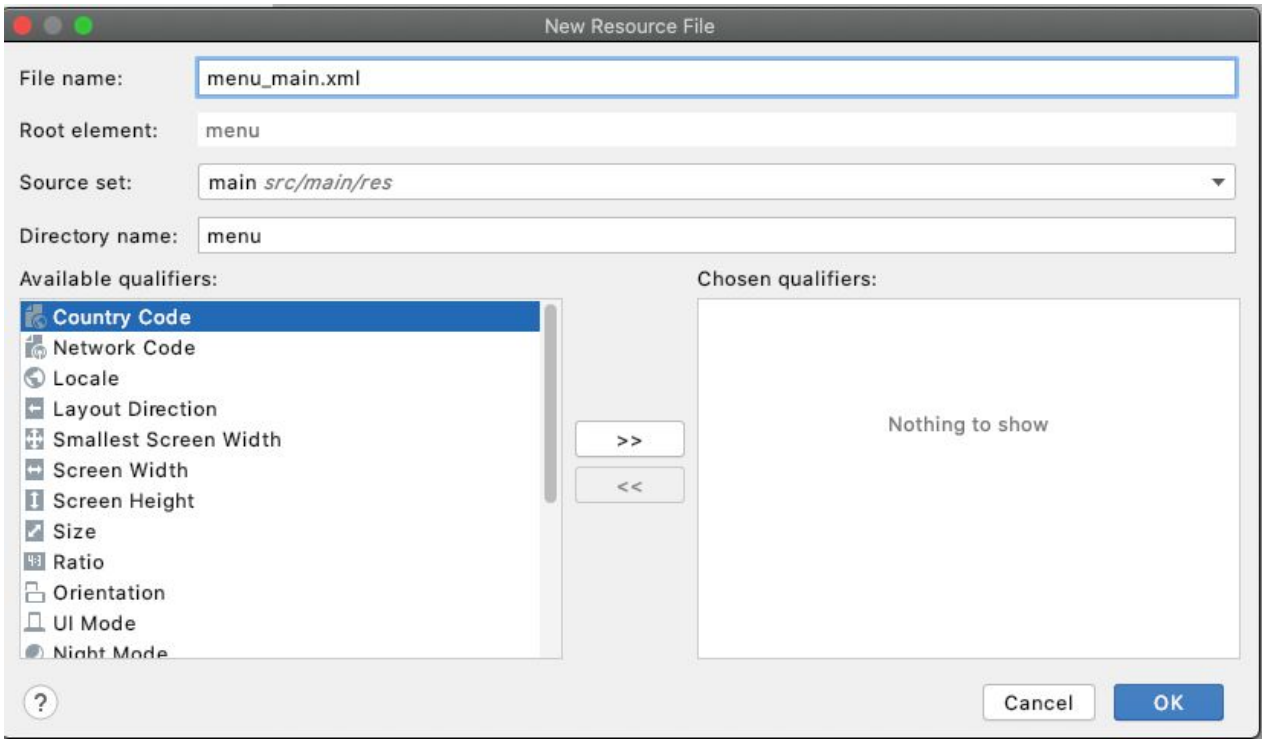
Name the directory **menu**. Hit **Enter**:



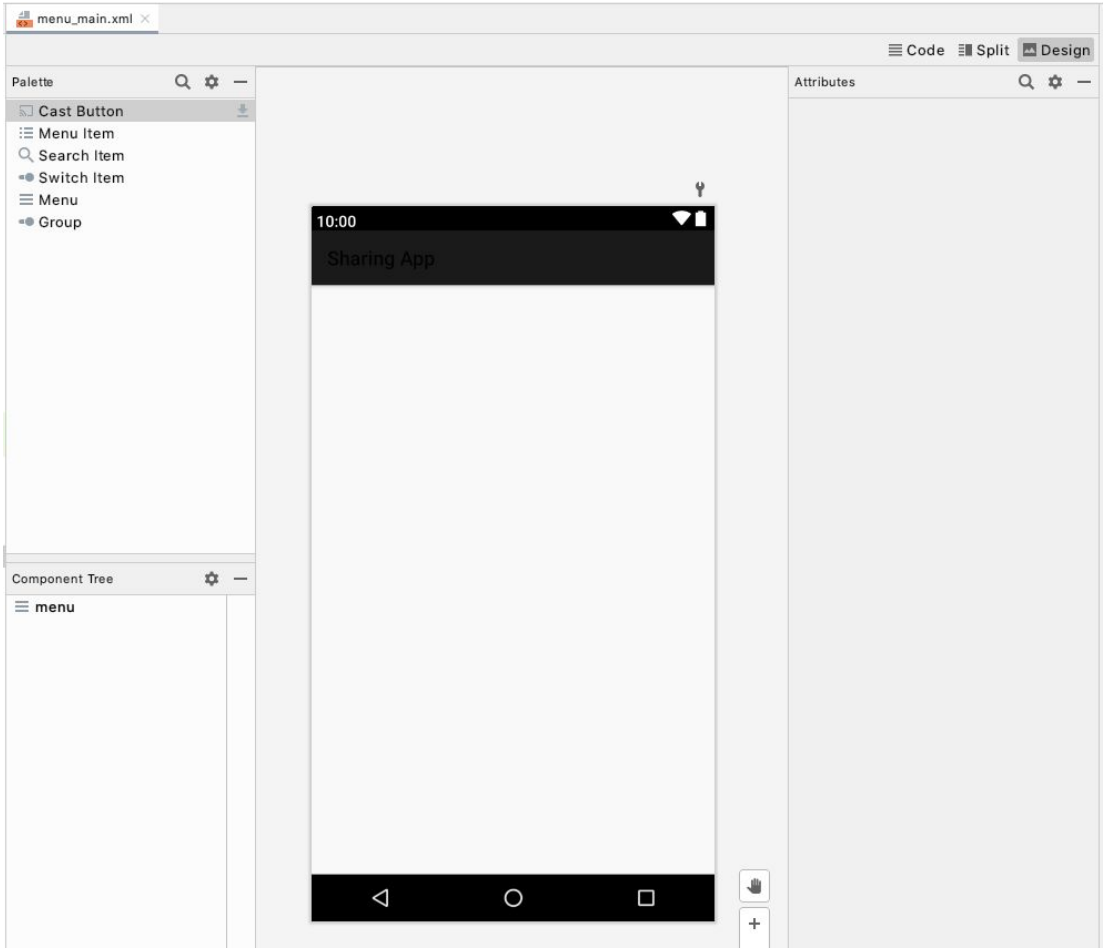
Add a menu resource file to the **menu** folder by right clicking on the **menu** folder, then clicking **New** → **Menu resource file**.



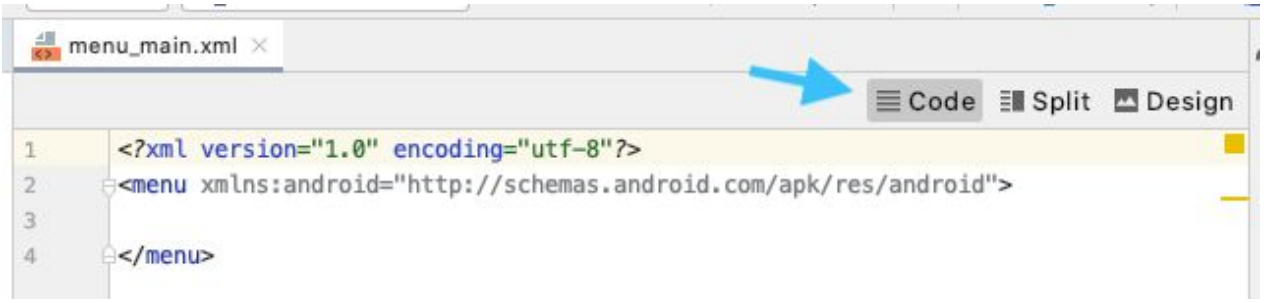
Call the resource file **menu_main.xml**. Click **OK**.



The **menu_main.xml** will be displayed in **Design** mode.



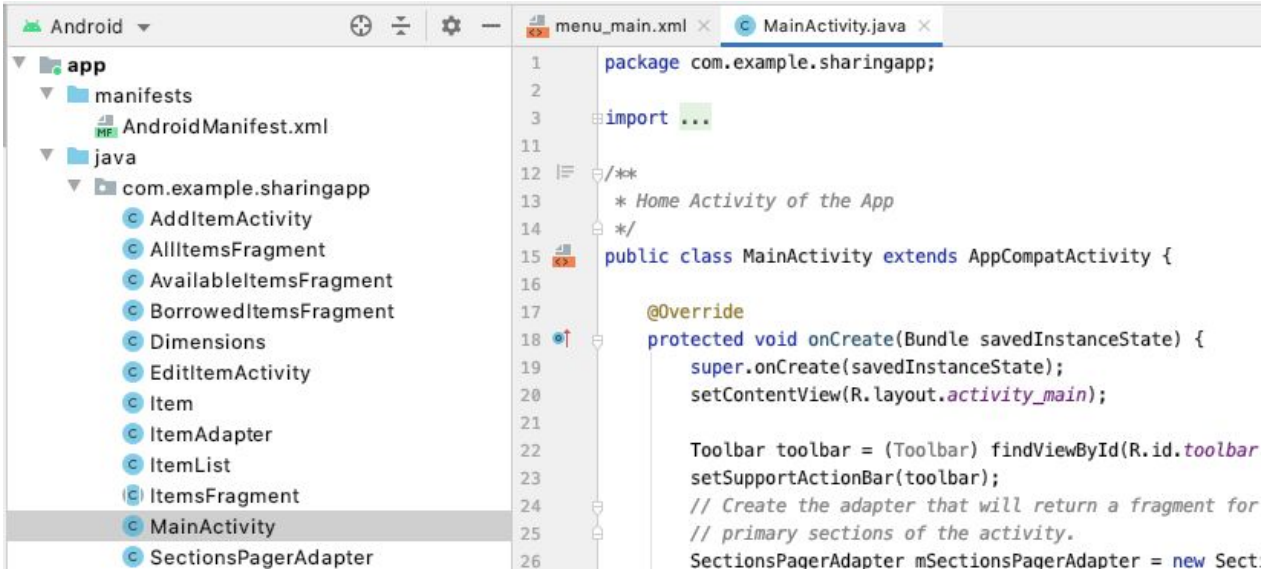
However, we want to work in **Code** mode. To switch modes click the Text tab near the bottom of the screen.



To implement the menu resource file replace contents of **menu_main.xml** with:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.example.sharingapp.MainActivity">
    <item
        android:id="@+id/contacts"
        android:title="@string/contacts" />
</menu>
```

Next, locate **MainActivity** within the project. Double click on **MainActivity** to open it.



Add two new imports:

```
import android.view.Menu;
import android.view.MenuItem;
```

We need to add the following two methods to **MainActivity:**

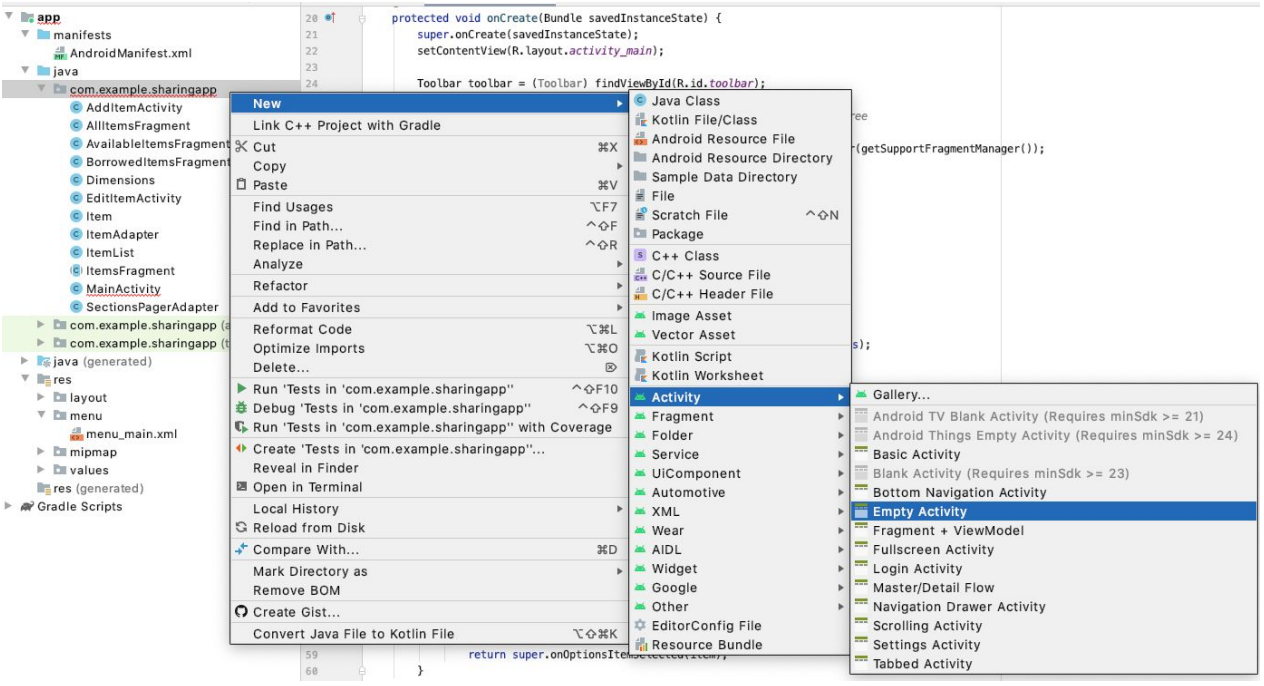
- **onCreateOptionsMenu()**: called when **MainActivity** is started. This method links the menu resource file **menu_main.xml** to **MainActivity** and “inflates” the menu.
- **onOptionsItemSelected()**: called when the user selects an option from the menu. In this application, this code handles what happens when the user selects the “Contacts” option from the menu -- **ContactsActivity** is started.

```
@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_main, menu);//Menu Resource, Menu
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.contacts:
            Intent intent = new Intent(this, ContactsActivity.class);
            startActivity(intent);
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

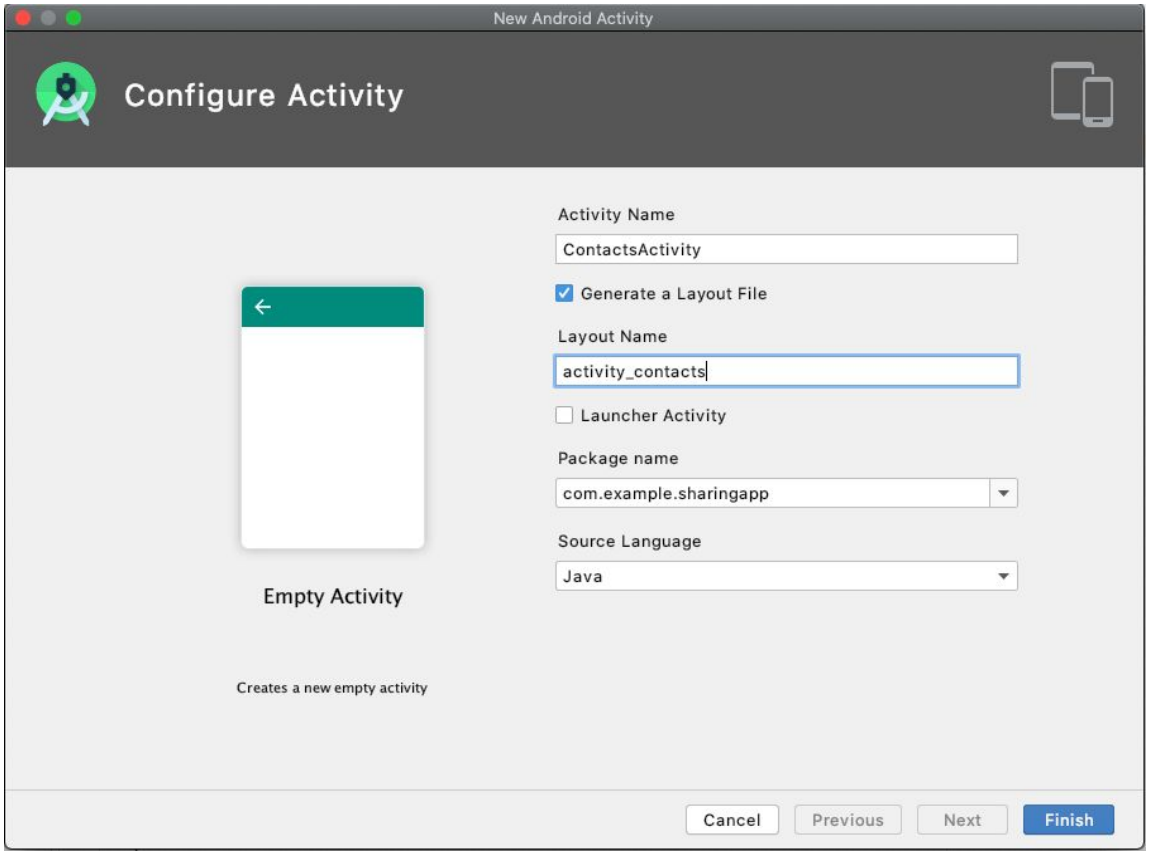
Notice that **ContactsActivity.class** is in **red**. This is because we have not created a **ContactsActivity** class yet. This error will disappear in the next part of the tutorial when we create the **ContactsActivity** class.

Step 3. Create the ContactsActivity Class

Right click on the **com.example.sharingapp** folder, then click **New** → **Activity** → **Empty Activity**.



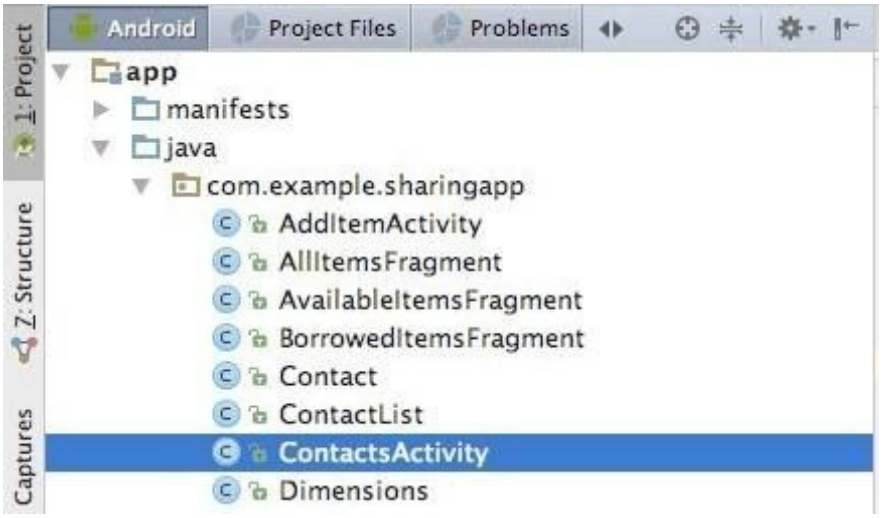
Name the activity **ContactsActivity** and the resource file **activity_contacts**. Then click **Finish**.



As a result:

- A new layout resource in the **layout** folder called **activity_contacts.xml** is created.

- A new activity class called **ContactsActivity** is created.



- You can also see that in **AndroidManifest.xml** file:

the line

```
13        <activity android:name=".ContactsActivity"></activity>
```

is **automatically added** to the **AndroidManifest.xml** file to link **ContactsActivity** to all the other activities in the app.

Step 4. Implement the ContactsActivity Layout Resource File, activity_contacts.xml

In the previous step we created **activity_contacts.xml**, the layout resource file corresponding to **ContactsActivity**.

The user stories explain that **ContactsActivity** will display a list of contacts. To achieve this we need to update the resource file, **activity_contacts.xml**, to add an ImageButton, which when clicked will start **ContactsActivity**.

Replace the current contents of **activity_contacts.xml** with:

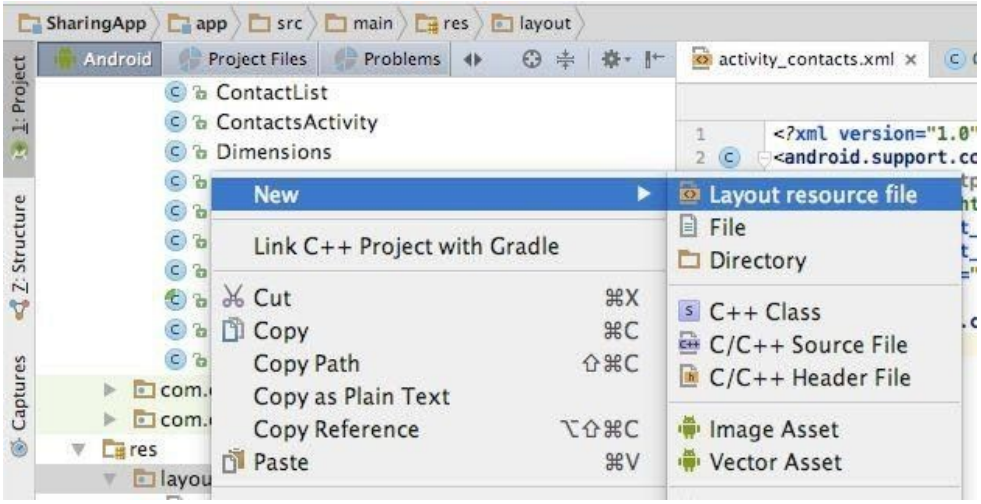
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="16dp"
   android:paddingLeft="16dp"
   android:paddingRight="16dp"
   android:paddingTop="16dp"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   tools:context="com.example.sharingapp.ContactsActivity">

   <ImageButton
       android:id="@+id/imageButton"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_gravity="end"
       android:onClick="addContactActivity"
       android:background="@color/colorPrimary"
       app:srcCompat="@android:drawable/ic_input_add"
       tools:layout_editor_absoluteX="8dp"
       tools:layout_editor_absoluteY="0dp" />

   <ListView
       android:id="@+id/my_contacts"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       tools:layout_editor_absoluteX="8dp"
       tools:layout_editor_absoluteY="75dp" />

</LinearLayout>
```

Note that since we haven't implemented the "onClick" method, it will remain red until we complete it.

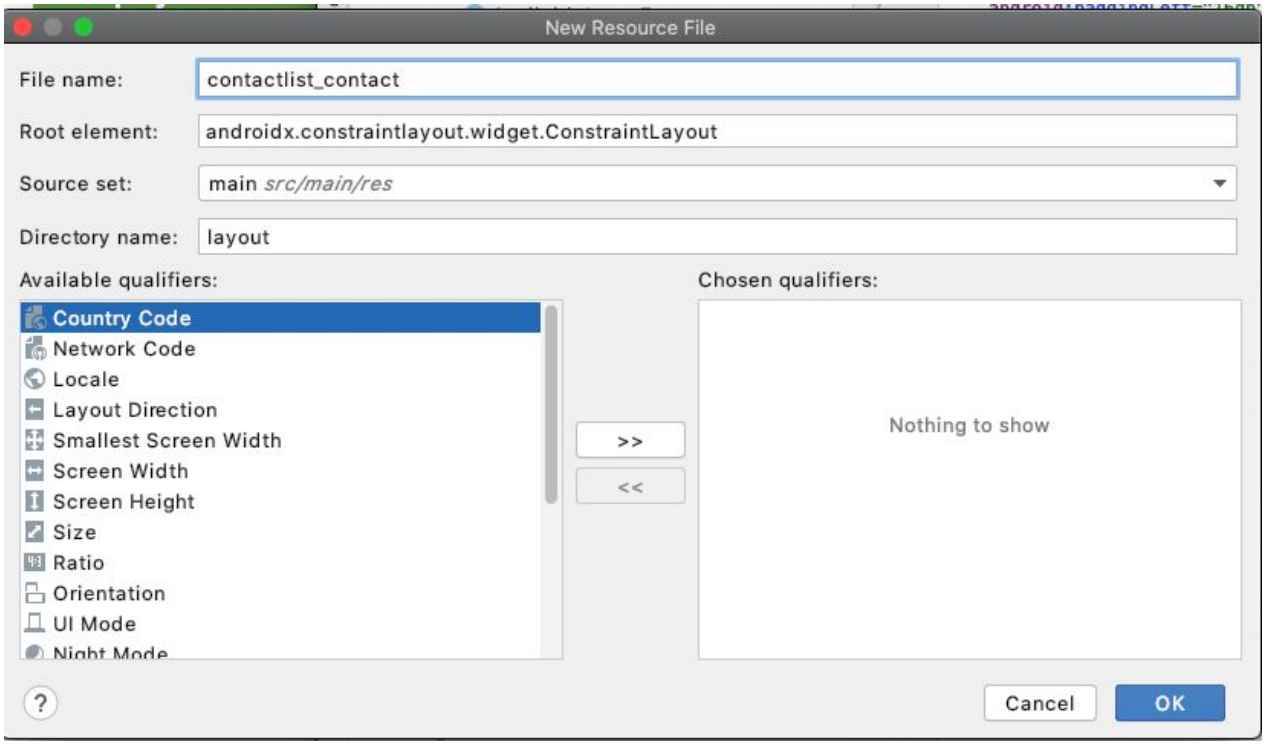
Step 5. Create and Implement a Layout Resource for a Contact Display in the List of Contacts: contactlist_contact.xml

Recall the user stories explain that **ContactsActivity** will display a list of contacts.

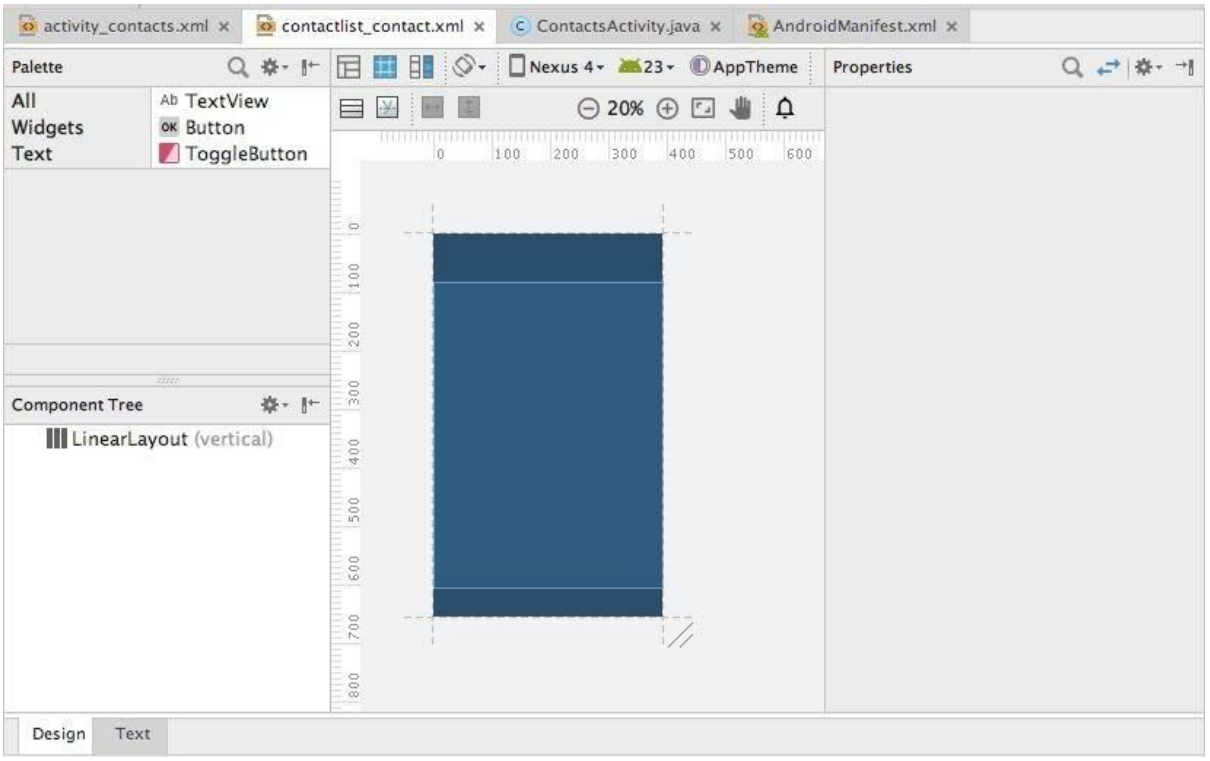
Each contact that appears in the list will display the same information: username and email address. Each contact displayed will make use of the same layout resource file. We can create this resource file by right clicking on the **layout** folder:



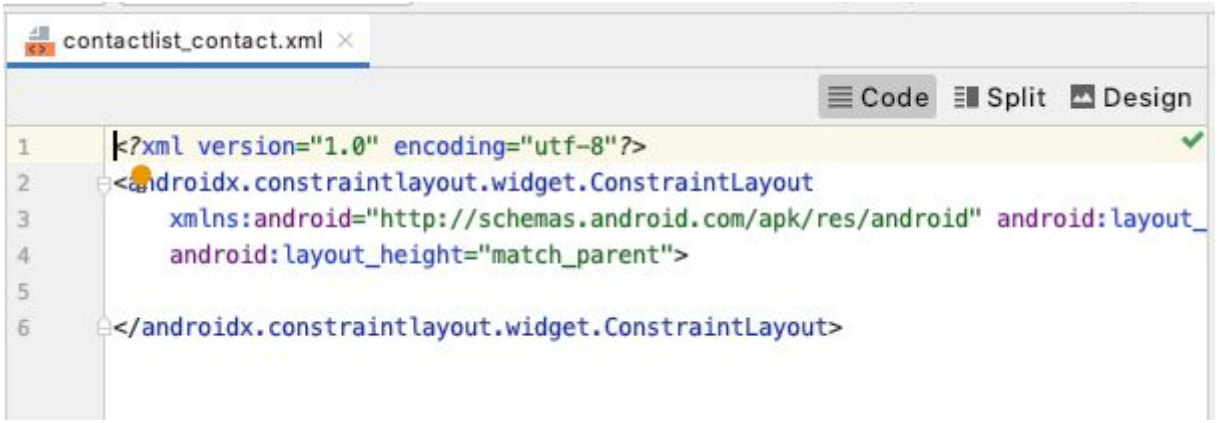
Name this new resource **contactlist_contact**. Click **OK**.



After clicking **OK**, this opens the **contactlist_contact.xml** resource in **Design** mode.



To edit this file we need to click the **code** tab.



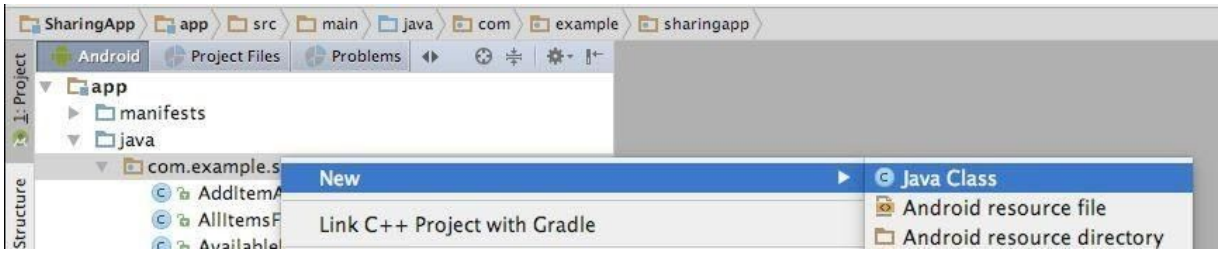
Now we replace the original contents of **contactlist_contact.xml** with the following. This code specifies how each contact will look, including which properties will be visible, how they will appear and where they will appear.

```
<?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="wrap_content"
    android:orientation="vertical">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="center_horizontal">

        <ImageView
            android:id="@+id/contacts_image_view"
            android:layout_width="60dp"
            android:layout_height="60dp"
            android:scaleType="centerCrop"
            android:layout_marginBottom="5dp"
            android:layout_marginLeft="5dp"
            android:layout_marginRight="10dp"
            android:layout_marginTop="5dp"
            android:background="@color/colorPrimary"
            android:label="@string/image_icon"/>

        <LinearLayout
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center_horizontal">

            <TextView
                android:id="@+id/username_tv"
                android:layout_marginTop="10dp"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:hint="@string/title_hint"/>

            <TextView
                android:id="@+id/email_tv"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:hint="@string/status_hint"/>

        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

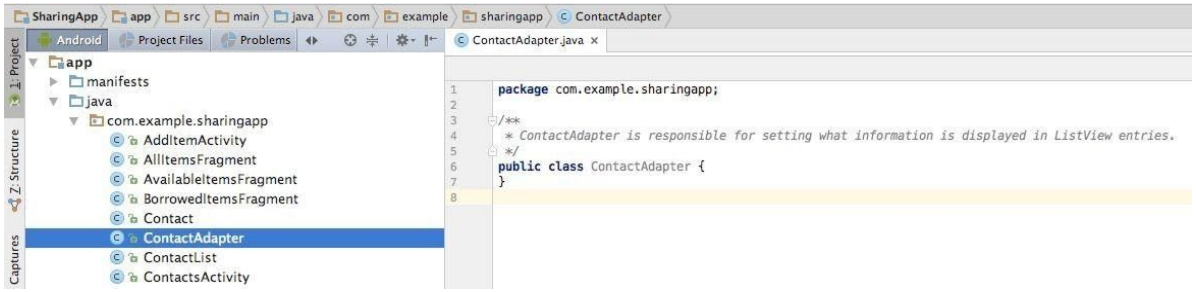
Step 6. Create and Implement the ContactAdapter Class

Now that we have our layout resource file for each contact that will appear in the contact list, we need to link this **contactlist_contact.xml** to the **Contact** model using a custom adapter: **ContactAdapter**.

Create the new **ContactAdapter** class by right clicking on the **com.example.sharingapp** folder and selecting **New → Java Class**.



Name the new class **ContactAdapter**. Click **OK**.



Replace the contents of **ContactAdapter** with the following code (continues onto next page):

```
package com.example.sharingapp;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import java.util.ArrayList;

/**
* ContactAdapter is responsible for what information is displayed in ListView
entries.
*/
public class ContactAdapter extends ArrayAdapter<Contact> {
```

```
    private LayoutInflater inflater;
    private Context context;

    public ContactAdapter(Context context, ArrayList<Contact> contacts) {
        super(context, 0, contacts);
        this.context = context;
        this.inflater = LayoutInflater.from(context);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // getItem(position) gets the "contact" at "position" in the "contacts" ArrayList
        // (where "contacts" is a parameter in the ContactAdapter creator as seen above ^^)
        Contact contact = getItem(position);

        String username = "Username: " + contact.getUsername();
        String email = "Email: " + contact.getEmail();

        // Check if an existing view is being reused, otherwise inflate the view.
        if (convertView == null) {
            convertView = inflater.from(context).inflate(R.layout.contactlist_contact, parent, false);
        }

        TextView username_tv = (TextView) convertView.findViewById(R.id.username_tv);
        TextView email_tv = (TextView) convertView.findViewById(R.id.email_tv);
        ImageView photo = (ImageView) convertView.findViewById(R.id.contacts_image_view);

        photo.setImageResource(android.R.drawable.ic_menu_gallery);

        username_tv.setText(username);
        email_tv.setText(email);

        return convertView;
    }
}
```

Alternatively, you can also use the content of this gist to copy the code.

Notice that everything related to the **Contact** model is shown in **red**. Don't worry about this now. When you create and implement the **Contact** and **ContactList** classes these errors should go away.

Step 7. Implement the ContactsActivity Class

Now that we have our contact list related layout resources in place, we can flesh out **ContactsActivity**. Double click on the **ContactsActivity** class to open it. Replace the contents of the file with the following code (continues onto next page):

```
package com.example.sharingapp;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
import java.util.ArrayList;

/**
* Displays a list of all contacts
*/
public class ContactsActivity extends AppCompatActivity {

   private ContactList contact_list = new ContactList();
   private ListView my_contacts;
   private ArrayAdapter<Contact> adapter;
   private Context context;
   private ItemList item_list = new ItemList();
   private ContactList active_borrowers_list = new ContactList();

   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_contacts);

       context = getApplicationContext();
       contact_list.loadContacts(context);
       item_list.loadItems(context);

       my_contacts = (ListView) findViewById(R.id.my_contacts);
       adapter = new ContactAdapter(ContactsActivity.this,
contact_list.getContacts());
```

```
        my_contacts.setAdapter(adapter);
        adapter.notifyDataSetChanged();

        // When contact is long clicked, this starts EditContactActivity
        my_contacts.setOnItemLongClickListener(new
android.widget.AdapterView.OnItemLongClickListener() {

            @Override
            public boolean onItemLongClick(AdapterView<?> parent, View view, int
pos, long id) {

                Contact contact = adapter.getItem(pos);

                ArrayList<Contact> active_borrowers =
item_list.getActiveBorrowers();
                active_borrowers_list.setContacts(active_borrowers);

                // Prevent contact from editing an "active" borrower.
                if (active_borrowers_list != null) {
                    if (active_borrowers_list.hasContact(contact)) {
                        CharSequence text = "Cannot edit or delete active
borrower!";
                        int duration = Toast.LENGTH_SHORT;
                        Toast.makeText(context, text, duration).show();
                        return true;
                    }
                }

                contact_list.loadContacts(context); // Must load contacts again
here
                int meta_pos = contact_list.getIndex(contact);

                Intent intent = new Intent(context, EditContactActivity.class);
                intent.putExtra("position", meta_pos);
                startActivity(intent);

                return true;
            }
        });
    }

    @Override
    protected void onStart() {
        super.onStart();
```

```
        context = getApplicationContext();
        contact_list.loadContacts(context);

        my_contacts = (ListView) findViewById(R.id.my_contacts);
        adapter = new ContactAdapter(ContactsActivity.this,
contact_list.getContacts());
        my_contacts.setAdapter(adapter);
        adapter.notifyDataSetChanged();
    }


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

Alternatively, you can also use the content of this gist to copy the code.

Notice that everything related to the **Contact** model and **ContactList** class is shown in **red**. Don't worry about this now. When you create and implement the **Contact** and **ContactList** classes these errors should go away. However, there are also some additional errors here that are not directly related to this, we will deal with them shortly.

Step 8. Update the Item Class

Next we need to update the **Item** class. Double click on the **Item** class to edit the file. In the original app, the borrower was stored as a **String**:

```
private String borrower;
```

However, the updated UML class diagram indicates that the borrower should be stored as a **Contact**. This means we need to replace the above line of code to:

```
private Contact borrower;
```

This change in type (from **String** to **Contact**) requires that several **Item** methods are updated. Replace the **Item()**, **setBorrower()** and **getBorrower()** methods with the following:

```
public Item(String title, String maker, String description, Dimensions
dimensions, Bitmap image,
           String id) {
   this.title = title;
   this.maker = maker;
   this.description = description;
   this.dimensions = dimensions;
   this.status = "Available";
   this.borrower = null;
   addImage(image);

   if (id == null){
       setId();
   } else {
       updateId(id);
   }
}

public void setBorrower(Contact borrower) {
   this.borrower = borrower;
}
```

```
public Contact getBorrower() {
   return borrower;
}
```

Notice that everything related to the **Contact** model is shown in red. Don't worry about this now. When you create and implement the **Contact** and **ContactList** classes these errors should go away.

Step 9. Update the ItemList Class

According to the updated UML class diagram we need to implement a new method in **ItemList**, **getActiveBorrowers()**. Double click on the **ItemList** class to edit the

file. Add the following method to the **ItemList** class:

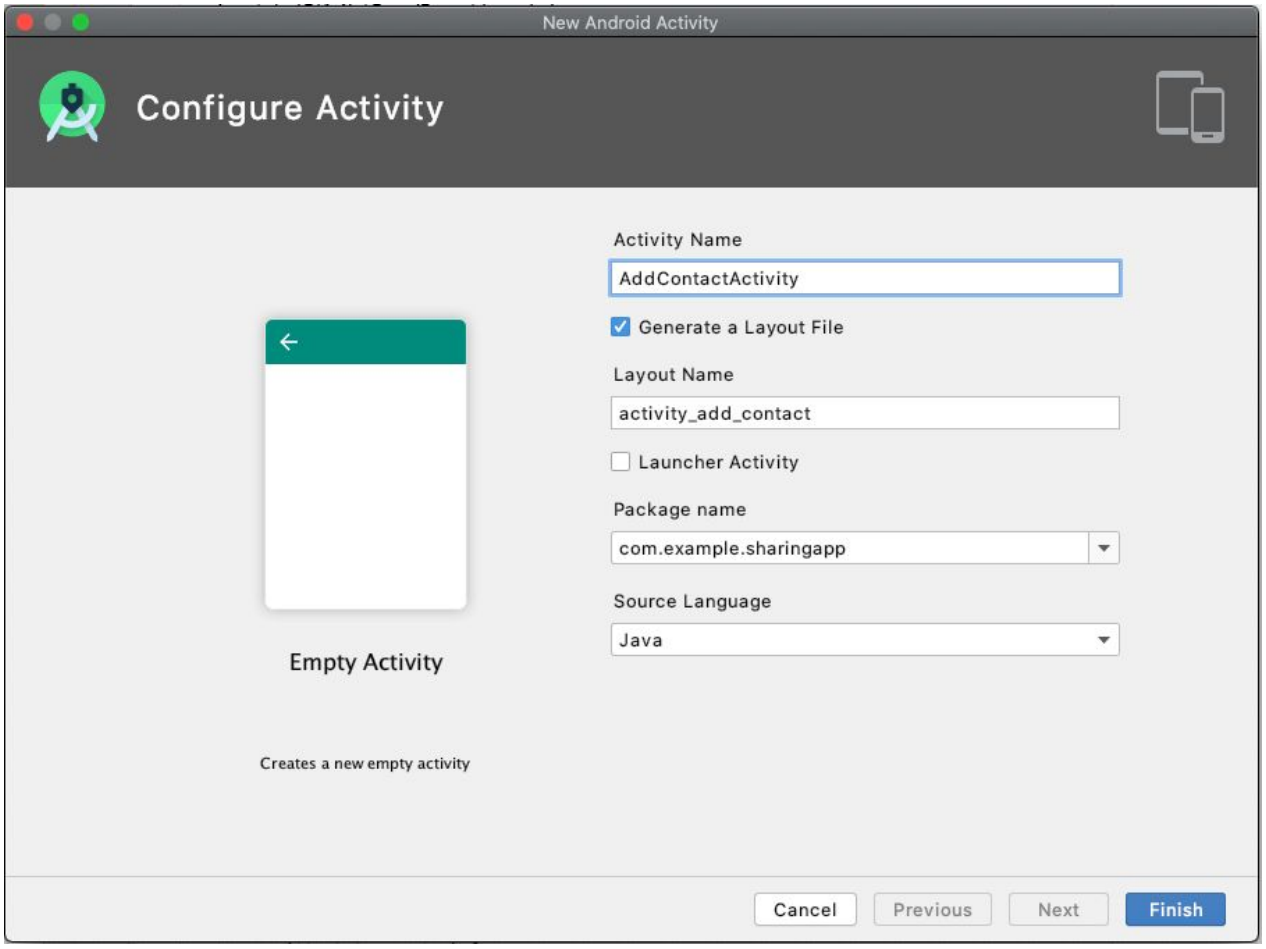
```
public ArrayList<Contact> getActiveBorrowers() {

   ArrayList<Contact> active_borrowers = new ArrayList<Contact>();
   for (Item i : items) {
       Contact borrower = i.getBorrower();
       if (borrower != null) {
           active_borrowers.add(borrower);
       }
   }
   return active_borrowers;
}
```

Notice that everything related to the **Contact** model is shown in red. Don't worry about this now. When you create and implement the **Contact** and **ContactList** classes these errors should go away.

Step 10. Create the AddContactActivity Class

Right click on the com.example.sharingapp folder then click **New** → **Activity** → **Empty Activity**.

Name the activity **AddContactActivity** and the resource file **activity_add_contact**. Then click **Finish**.



As a result:

- A new layout resource in the **layout** folder called **activity_add_contact.xml** is created. We will take a closer look at **activity_add_contact.xml** soon.
- A new activity class called **AddContactActivity** is created. We will revisit **AddContactActivity** later in the tutorial.
- And the line

```
<activity android:name=".AddContactActivity" />
```

is automatically added to the **AndroidManifest.xml** file to link **AddContactActivity** to all the other activities in the app

Step 11. Implement the AddContactActivity Layout Resource File, activity_add_contact.xml

In the previous step we created **activity_add_contact.xml**, the layout resource file corresponding to **AddContactActivity**.
Next, we need to replace the current contents of **activity_add_contact.xml** with the following code (continues onto next page):

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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="com.example.sharingapp.AddContactActivity"
   android:orientation="vertical">

   <LinearLayout
       android:orientation="horizontal"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:layout_gravity="center_horizontal"
       android:layout_marginTop="5dp">

       <TextView
           android:id="@+id/username_tv"
           android:layout_width="104dp"
           android:layout_height="wrap_content"
           android:text="@string/username_hint"
           android:gravity="center_vertical"
           android:textAppearance="@android:style/TextAppearance.Medium"
/>

       <EditText
           android:id="@+id/username"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:hint="@string/username_hint"
           android:inputType="text"
           android:textAppearance="@android:style/TextAppearance.Medium"
           android:maxLength="24"/>

   </LinearLayout>

   <LinearLayout
       android:orientation="horizontal"
```

```
        android:layout_width="fill_parent"
        android:layout_gravity="center"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp">

        <TextView
            android:id="@+id/email_tv"
            android:layout_width="104dp"
            android:layout_height="wrap_content"
            android:text="@string/email_hint"
            android:gravity="center_vertical"
            android:textAppearance="@android:style/TextAppearance.Medium"
/>

        <EditText
            android:id="@+id/email"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:hint="@string/email_hint"
            android:inputType="textEmailAddress"
            android:textAppearance="@android:style/TextAppearance.Medium"
            android:maxLength="24"/>

    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_gravity="center"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp">

        <Button
            android:id="@+id/save_button"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.25"
            android:onClick="saveContact"
            android:text="@string/save"
            android:layout_gravity="center_horizontal"

android:textAppearance="@android:style/TextAppearance.Medium"/>

    </LinearLayout>
</LinearLayout>
```

Alternatively, you can also use the content of this gist to copy the content. Note that since we haven't implemented the "onClick" method, it will remain red until we complete it.

Step 12. Implement the AddContactActivity Class

Now that we have its corresponding layout resource in place, we can flesh out **AddContactActivity**.

Double click on the **AddContactActivity** class to open it. Replace the contents of the file with following code (continues onto next page):

```
package com.example.sharingapp;

import android.content.Context;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
/**
* Add a new contact
*/
public class AddContactActivity extends AppCompatActivity {

   private ContactList contact_list = new ContactList();
   private Context context;
   private EditText username;
   private EditText email;

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

       username = (EditText) findViewById(R.id.username);
       email = (EditText) findViewById(R.id.email);

       context = getApplicationContext();
       contact_list.loadContacts(context);
   }

   public void saveContact(View view) {

       String username_str = username.getText().toString();
       String email_str = email.getText().toString();

       if (username_str.equals("")) {
           username.setError("Empty field!");
           return;
       }

       if (email_str.equals("")) {
```

```
            email.setError("Empty field!");
            return;
        }

        if (!email_str.contains("@")){
            email.setError("Must be an email address!");
            return;
        }

        if (!contact_list.isUsernameAvailable(username_str)){
            username.setError("Username already taken!");
            return;
        }

        Contact contact = new Contact(username_str, email_str, null);
        contact_list.addContact(contact);
        contact_list.saveContacts(context);
        // End AddContactActivity
        finish();
    }
}
```

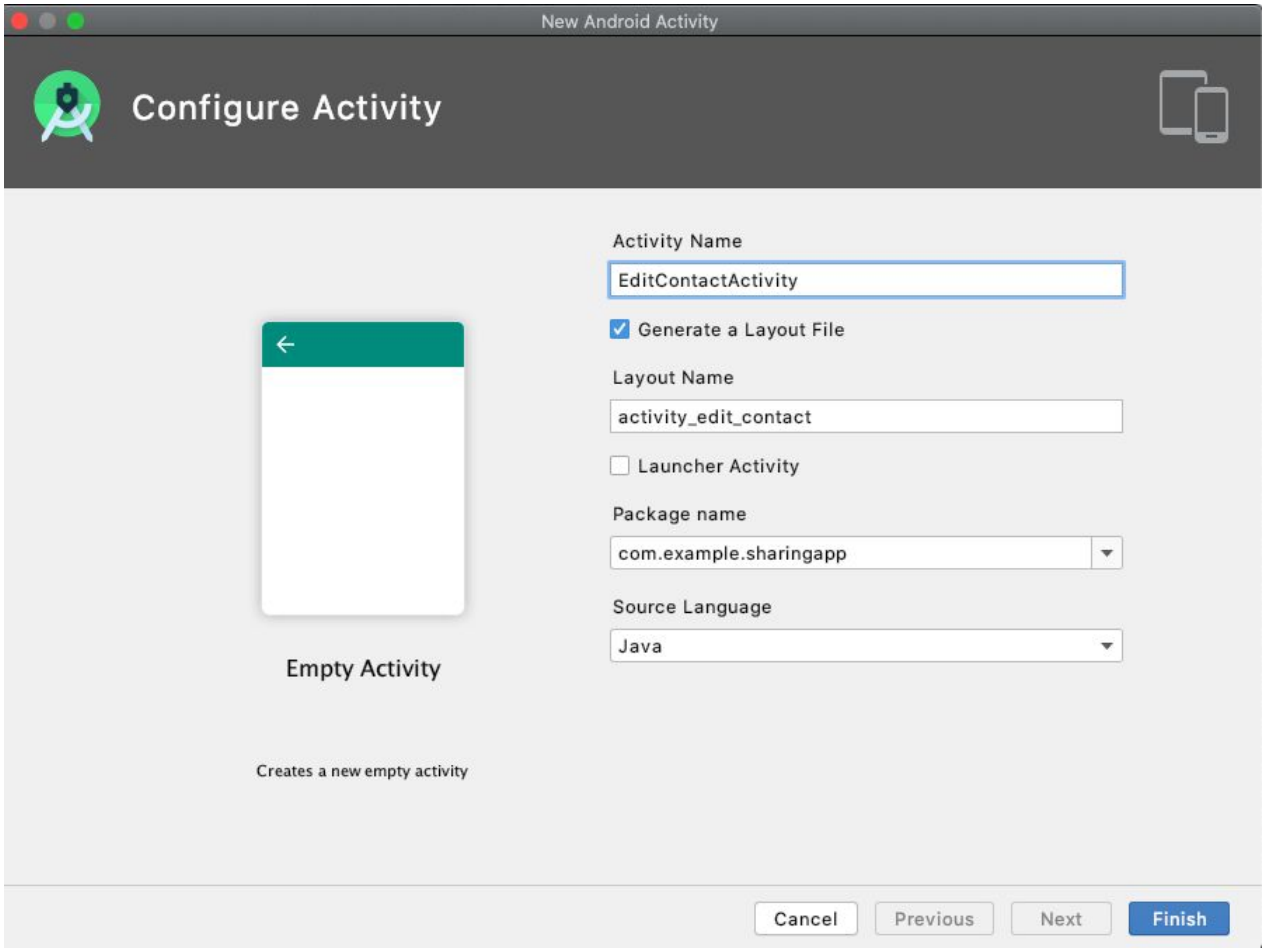
Alternatively, you can also use the content of this gist to copy the code.

Notice that everything related to the **Contact** model or **ContactList** is shown in **red**. Don't worry about this now. When you create and implement the **Contact** and **ContactList** classes these errors should go away.

Step 13. Create the EditContactActivity Class

Next, we will create the **EditContactActivity**. Right click on the **com.example.sharingapp** folder then click **New** → **Activity** → **Empty Activity**

Name the activity **EditContactActivity** and the resource file **activity_edit_contact**.
Then click **Finish**.



As a result:

- A new layout resource in the layout folder called **activity_edit_contact.xml** is created. We will take a closer look at **activity_edit_contact.xml** soon.
- A new activity class called **EditContactActivity** is created. We will revisit **EditContactActivity** later in the tutorial.
  And the line:

```
<activity android:name=".EditContactActivity" />
```

is automatically added to the AndroidManifest.xml file to link **EditContactActivity** to all the other activities in the app.

Step 14. Implement the EditContactActivity layout resource file, activity_edit_contact.xml

In the previous step we created **activity_edit_contact.xml**, the layout resource file corresponding to **EditContactActivity**. Replace the current contents of **activity_edit_contacts.xml** with the following code (continues onto next page):

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
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="com.example.sharingapp.EditContactActivity"
   android:orientation="vertical">

   <LinearLayout
       android:orientation="horizontal"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:layout_gravity="center_horizontal"
       android:layout_marginTop="5dp">

       <TextView
           android:id="@+id/username_tv"
           android:layout_width="104dp"
           android:layout_height="wrap_content"
           android:gravity="center_vertical"
           android:text="@string/username_hint"

android:textAppearance="@android:style/TextAppearance.Medium" />

       <EditText
           android:id="@+id/username"
           android:layout_width="fill_parent"
```

```
            android:layout_height="wrap_content"
            android:hint="@string/title_hint"
            android:inputType="text"

android:textAppearance="@android:style/TextAppearance.Medium"
            android:maxLength="24" />

    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_gravity="center"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp">

        <TextView
            android:id="@+id/email_tv"
            android:layout_width="104dp"
            android:layout_height="wrap_content"
            android:gravity="center_vertical"
            android:text="@string/email_hint"

android:textAppearance="@android:style/TextAppearance.Medium" />

        <EditText
            android:id="@+id/email"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:hint="@string/email_hint"
            android:inputType="textEmailAddress"
            android:maxLength="24"

android:textAppearance="@android:style/TextAppearance.Medium" />

    </LinearLayout>

    <LinearLayout
```

```
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_gravity="center"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp">

        <Button
            android:id="@+id/save_edited_user_button"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_weight="0.25"
            android:onClick="saveContact"
            android:text="@string/save"

android:textAppearance="@android:style/TextAppearance.Medium" />

        <Button
            android:id="@+id/delete_edited_user_button"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_weight="0.25"
            android:onClick="deleteContact"
            android:text="@string/delete"

android:textAppearance="@android:style/TextAppearance.Medium" />

    </LinearLayout>
</LinearLayout>
```

Alternatively, you can also use the content of this gist to copy the code. Note that since we haven't implemented the "onClick" methods, they will remain red until we complete them.

Step 15. Implement the EditContactActivity Class

Now that we have its corresponding layout resource in place, we can flesh out **EditContactActivity**.

Double click on the **EditContactActivity** class to open it. Replace the contents of the file with the following (continues onto next page):

```
package com.example.sharingapp;

import android.content.Context;
import android.content.Intent;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

/**
* Editing a pre-existing contact consists of deleting the old contact and adding
a new contact with the old
* contact's id.
* Note: You will not be able contacts which are "active" borrowers
*/
public class EditContactActivity extends AppCompatActivity {

   private ContactList contact_list = new ContactList();
   private Contact contact;
   private EditText email;
   private EditText username;
   private Context context;

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

       context = getApplicationContext();
       contact_list.loadContacts(context);

       Intent intent = getIntent();
       int pos = intent.getIntExtra("position", 0);

       contact = contact_list.getContact(pos);

       username = (EditText) findViewById(R.id.username);
       email = (EditText) findViewById(R.id.email);

       username.setText(contact.getUsername());
       email.setText(contact.getEmail());
   }
```

```
    public void saveContact(View view) {

        String email_str = email.getText().toString();

        if (email_str.equals("")) {
            email.setError("Empty field!");
            return;
        }

        if (!email_str.contains("@")){
            email.setError("Must be an email address!");
            return;
        }

        String username_str = username.getText().toString();
        String id = contact.getId(); // Reuse the contact id

        // Check that username is unique AND username is changed (Note: if
username was not changed
        // then this should be fine, because it was already unique.)
        if (!contact_list.isUsernameAvailable(username_str) &&
!(contact.getUsername().equals(username_str))) {
            username.setError("Username already taken!");
            return;
        }

        Contact updated_contact = new Contact(username_str, email_str, id);

        contact_list.deleteContact(contact);
        contact_list.addContact(updated_contact);
        contact_list.saveContacts(context);

        // End EditContactActivity
        finish();
    }

    public void deleteContact(View view) {

        contact_list.deleteContact(contact);
        contact_list.saveContacts(context);

        // End EditContactActivity
        finish();
    }
}
```

Alternatively, you can also use the content of this gist to copy the code.

Notice that everything related to the **Contact** model or **ContactList** class is shown in **red**. Don't worry about this now. When you create and implement the **Contact** and **ContactList** classes these errors should go away

Step 16. Update the EditItemActivity Class

Because changes were made to the **ItemList** model to store the borrower as a Contact now, instead of as a String, we need to update **EditItemActivity**.

Double click on **EditItemActivity** to edit the file. Replace the current contents of **EditItemActivity** with the following code (continues on several pages):

```
package com.example.sharingapp;

import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.provider.MediaStore;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.Switch;
import android.widget.TextView;

/**
* Editing a pre-existing item consists of deleting the old item and adding a new
item with the old
* item's id.
* Note: invisible EditText is used to setError for status. For whatever reason we
cannot .setError to
* the status Switch so instead an error is set to an "invisible" EditText.
*/
public class EditItemActivity extends AppCompatActivity{

    private ItemList item_list = new ItemList();
    private Item item;
    private Context context;

    private ContactList contact_list = new ContactList();

    private Bitmap image;
    private int REQUEST_CODE = 1;
```

```
    private ImageView photo;

    private EditText title;
    private EditText maker;
    private EditText description;
    private EditText length;
    private EditText width;
    private EditText height;
    private Spinner borrower_spinner;
    private TextView  borrower_tv;
    private Switch status;
    private EditText invisible;

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

        title = (EditText) findViewById(R.id.title);
        maker = (EditText) findViewById(R.id.maker);
        description = (EditText) findViewById(R.id.description);
        length = (EditText) findViewById(R.id.length);
        width = (EditText) findViewById(R.id.width);
        height = (EditText) findViewById(R.id.height);
        borrower_spinner = (Spinner) findViewById(R.id.borrower_spinner);
        borrower_tv = (TextView) findViewById(R.id.borrower_tv);
        photo = (ImageView) findViewById(R.id.image_view);
        status = (Switch) findViewById(R.id.available_switch);
        invisible = (EditText) findViewById(R.id.invisible);

        invisible.setVisibility(View.GONE);

        context = getApplicationContext();
        item_list.loadItems(context);
        contact_list.loadContacts(context);

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_spinner_dropdown_item,
contact_list.getAllUsernames());
        borrower_spinner.setAdapter(adapter);

        Intent intent = getIntent();   // Get intent from ItemsFragment
        int pos = intent.getIntExtra("position", 0);
```

```
        item = item_list.getItem(pos);

        Contact contact = item.getBorrower();
        if (contact != null){
            int contact_pos = contact_list.getIndex(contact);
            borrower_spinner.setSelection(contact_pos);
        }

        title.setText(item.getTitle());
        maker.setText(item.getMaker());
        description.setText(item.getDescription());

        Dimensions dimensions = item.getDimensions();

        length.setText(dimensions.getLength());
        width.setText(dimensions.getWidth());
        height.setText(dimensions.getHeight());

        String status_str = item.getStatus();
        if (status_str.equals("Borrowed")) {
            status.setChecked(false);
        } else {
            borrower_tv.setVisibility(View.GONE);
            borrower_spinner.setVisibility(View.GONE);
        }

        image = item.getImage();
        if (image != null) {
            photo.setImageBitmap(image);
        } else {
            photo.setImageResource(android.R.drawable.ic_menu_gallery);
        }
    }

    public void addPhoto(View view) {
        Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        if (intent.resolveActivity(getPackageManager()) != null) {
            startActivityForResult(intent, REQUEST_CODE);
        }
    }

    public void deletePhoto(View view) {
```

```
        image = null;
        photo.setImageResource(android.R.drawable.ic_menu_gallery);
    }

    @Override
    protected void onActivityResult(int request_code, int result_code, Intent
intent){
        if (request_code == REQUEST_CODE && result_code == RESULT_OK){
            Bundle extras = intent.getExtras();
            image = (Bitmap) extras.get("data");
            photo.setImageBitmap(image);
        }
    }

    public void deleteItem(View view) {

        item_list.deleteItem(item);
        item_list.saveItems(context);

        // End EditItemActivity
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }

    public void saveItem(View view) {

        String title_str = title.getText().toString();
        String maker_str = maker.getText().toString();
        String description_str = description.getText().toString();
        String length_str = length.getText().toString();
        String width_str = width.getText().toString();
        String height_str = height.getText().toString();
        Contact contact = null;
        if (!status.isChecked()) {
            String borrower_str = borrower_spinner.getSelectedItem().toString();
            contact = contact_list.getContactByUsername(borrower_str);
        }

        Dimensions dimensions = new Dimensions(length_str, width_str, height_str);

        if (title_str.equals("")) {
            title.setError("Empty field!");
            return;
```

```
        }

        if (maker_str.equals("")) {
            maker.setError("Empty field!");
            return;
        }

        if (description_str.equals("")) {
            description.setError("Empty field!");
            return;
        }

        if (length_str.equals("")) {
            length.setError("Empty field!");
            return;
        }

        if (width_str.equals("")) {
            width.setError("Empty field!");
            return;
        }

        if (height_str.equals("")) {
            height.setError("Empty field!");
            return;
        }

        String id = item.getId(); // Reuse the item id
        Item updated_item = new Item(title_str, maker_str, description_str,
dimensions, image, id);

        boolean checked = status.isChecked();
        if (!checked) {
            updated_item.setStatus("Borrowed");
            updated_item.setBorrower(contact);
        }

        item_list.deleteItem(item);
        item_list.addItem(updated_item);
        item_list.saveItems(context);

        // End EditItemActivity
        Intent intent = new Intent(this, MainActivity.class);
```

```
        startActivity(intent);
    }

    /**
     * Checked = Available
     * Unchecked = Borrowed
     */
    public void toggleSwitch(View view){
        if (status.isChecked()) {
            // Means was previously borrowed, switch was toggled to available
            borrower_spinner.setVisibility(View.GONE);
            borrower_tv.setVisibility(View.GONE);
            item.setBorrower(null);
            item.setStatus("Available");

        } else {
            // Means not borrowed
            if (contact_list.getSize()==0){
                // No contacts, need to add contacts to be able to add a borrower.
                invisible.setEnabled(false);
                invisible.setVisibility(View.VISIBLE);
                invisible.requestFocus();
                invisible.setError("No contacts available! Must add borrower to
contacts.");
                status.setChecked(true); // Set switch to available

            } else {
                // Means was previously available
                borrower_spinner.setVisibility(View.VISIBLE);
                borrower_tv.setVisibility(View.VISIBLE);
            }
        }
    }
}
```

Alternatively, you can also use the content of this gist to copy the code

Notice that everything related to the **Contact** model is shown in **red**. Don't worry about this now. When you create and implement the **Contact** and **ContactList** classes these errors should go away. However, there are also some additional errors here that are not directly related to this.

EditItemActivity.java

```
40      private EditText height;
41      private Spinner borrower_spinner;
42      private TextView  borrower_tv;
43      private Switch status;
44      private EditText invisible;
45
46      @Override
47      protected void onCreate(Bundle savedInstanceState) {
48          super.onCreate(savedInstanceState);
49          setContentView(R.layout.activity_edit_item);
50
51          title = (EditText) findViewById(R.id.title);
52          maker = (EditText) findViewById(R.id.maker);
53          description = (EditText) findViewById(R.id.description);
54          length = (EditText) findViewById(R.id.length);
55          width = (EditText) findViewById(R.id.width);
56          height = (EditText) findViewById(R.id.height);
57          borrower_spinner = (Spinner) findViewById(R.id.borrower_spinner);
58          borrower_tv = (TextView) findViewById(R.id.borrower_tv);
59          photo = (ImageView) findViewById(R.id.image_view);
60          status = (Switch) findViewById(R.id.available_switch);
61          invisible = (EditText) findViewById(R.id.invisible);
62
```

In the next step we will update **EditItemActivity**'s corresponding layout resource file to include these new layout items.

Step 17. Update the EditItemActivity layout resource file, activity_edit_item.xml

In the previous step we updated **EditItemActivity**. Now we need to update its corresponding layout resource file, **activity_edit_item.xml**.

Double click on **activity_edit_item** to edit the file. Replace the current contents of **activity_edit_item.xml** with the following code (continues on several pages):

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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="com.example.sharingapp.EditItemActivity"
    android:orientation="vertical">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="5dp">

        <TextView
            android:id="@+id/title_tv"
            android:layout_width="104dp"
            android:layout_height="wrap_content"
            android:text="@string/title_hint"
            android:gravity="center_vertical"
            android:textAppearance="@android:style/TextAppearance.Medium" />

        <EditText
            android:id="@+id/title"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:hint="@string/title_hint"
            android:inputType="text"
            android:textAppearance="@android:style/TextAppearance.Medium"
            android:maxLength="24"/>

    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
```

```
        android:layout_gravity="center"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp">

        <TextView
            android:id="@+id/maker_tv"
            android:layout_width="104dp"
            android:layout_height="wrap_content"
            android:text="@string/maker_hint"
            android:gravity="center_vertical"
            android:textAppearance="@android:style/TextAppearance.Medium" />

        <EditText
            android:id="@+id/maker"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:hint="@string/maker_hint"
            android:inputType="text"
            android:textAppearance="@android:style/TextAppearance.Medium"
            android:maxLength="24"/>

    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_gravity="center"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp">

        <TextView
            android:id="@+id/description_tv"
            android:layout_width="104dp"
            android:layout_height="wrap_content"
            android:text="@string/description_hint"
            android:gravity="center_vertical"
            android:textAppearance="@android:style/TextAppearance.Medium"/>

        <EditText
            android:id="@+id/description"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:hint="@string/description_hint"
            android:inputType="text"
            android:textAppearance="@android:style/TextAppearance.Medium"
            android:maxLength="24"/>

    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
```

```
        android:layout_gravity="center"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp">

        <TextView
            android:id="@+id/dimensions_tv"
            android:layout_width="104dp"
            android:layout_height="wrap_content"
            android:text="@string/dimensions_hint"
            android:gravity="center_vertical"
            android:textAppearance="@android:style/TextAppearance.Medium"/>

        <EditText
            android:id="@+id/length"
            android:layout_width="64dp"
            android:layout_height="wrap_content"
            android:hint="@string/length_hint"
            android:inputType="numberDecimal"
            android:textAppearance="@android:style/TextAppearance.Medium"
            android:maxLength="24"/>

        <EditText
            android:id="@+id/width"
            android:layout_width="64dp"
            android:layout_height="wrap_content"
            android:hint="@string/width_hint"
            android:inputType="numberDecimal"
            android:textAppearance="@android:style/TextAppearance.Medium"
            android:maxLength="24" />

        <EditText
            android:id="@+id/height"
            android:layout_width="64dp"
            android:layout_height="wrap_content"
            android:hint="@string/height_hint"
            android:inputType="numberDecimal"
            android:textAppearance="@android:style/TextAppearance.Medium"
            android:maxLength="24" />
    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_gravity="center"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp">

        <Switch
            android:id="@+id/available_switch"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true"
```

```
        android:onClick="toggleSwitch"
        android:showText="true"
        android:text="@string/status_hint"
        android:textAppearance="@android:style/TextAppearance.Medium"
        android:textOff="@string/toggle_borrowed"
        android:textOn="@string/toggle_available" />


    <EditText
        android:id="@+id/invisible"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:textAppearance="@android:style/TextAppearance.Medium" />

</LinearLayout>


<LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="5dp">

    <TextView
        android:id="@+id/borrower_tv"
        android:layout_width="104dp"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:text="@string/borrower_hint"
        android:textAppearance="@android:style/TextAppearance.Medium"/>

    <Spinner
        android:id="@+id/borrower_spinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/btn_dropdown"
        android:spinnerMode="dropdown" />

</LinearLayout>

<ImageView
    android:id="@+id/image_view"
    android:layout_width="fill_parent"
    android:layout_height="0dp"
    android:layout_weight="0.5"
    android:gravity="center"
    android:layout_marginTop="5dp"
    android:layout_gravity="center_horizontal"
    android:background="@color/colorPrimary"
    android:label="@string/image_icon"/>
```

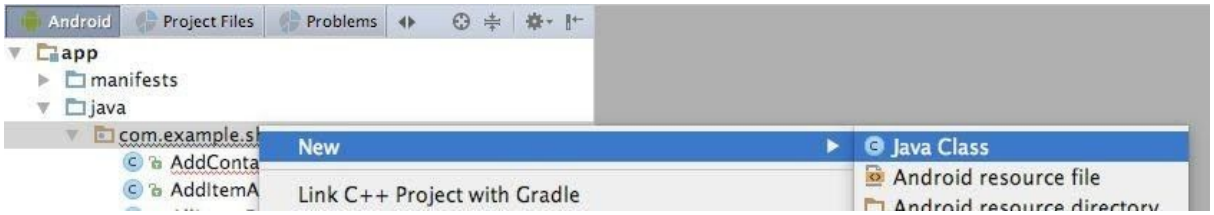
```
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="5dp">

        <ImageButton
            android:id="@+id/add_image_button"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:onClick="addPhoto"
            android:layout_gravity="center"
            android:background="@android:drawable/ic_menu_camera" />


        <ImageButton
            android:id="@+id/cancel_image_button"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:onClick="deletePhoto"
            android:layout_gravity="center"
            android:background="@android:drawable/ic_menu_close_clear_cancel"
/>

        <Button
            android:id="@+id/delete_item"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:gravity="center"
            android:onClick="deleteItem"
            android:text="@string/delete_item"
            android:textAppearance="@android:style/TextAppearance.Medium"/>

    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_gravity="center"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp">

        <Button
            android:id="@+id/save_button"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.25"
            android:onClick="saveItem"
            android:text="@string/save"
```

```
            android:layout_gravity="center_horizontal"
            android:textAppearance="@android:style/TextAppearance.Medium"/>

    </LinearLayout>
</LinearLayout>
```

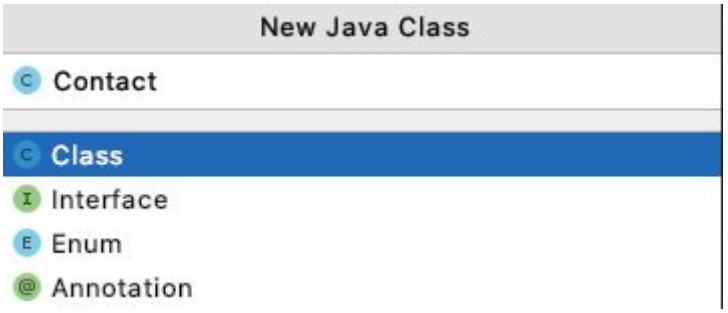
Alternatively, you can also use the content of this gist to copy the code.

Step 18. Create and Implement the Contact Class

Create a new class by right-clicking on the **com.example.sharingapp** folder. **New** → **Java Class**.

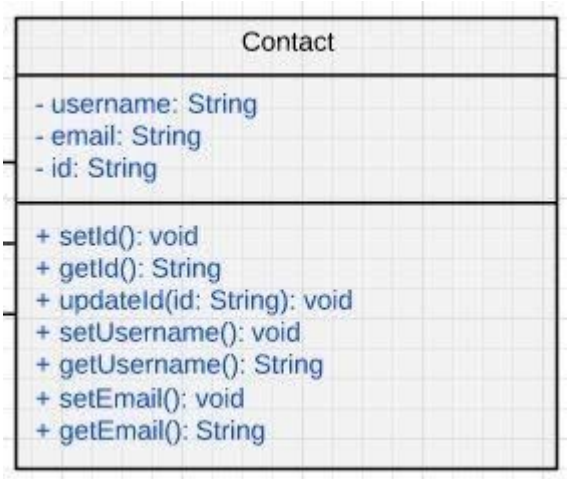


Name the class **Contact**. Hit **Enter**.



This creates an empty **Contact** class.

Now, it's your turn to Implement the **Contact** methods provided in the Updated UML Class diagram. Make sure you implement **all the attributes and methods** in the **Contact** class.



Hints

The implementation of the **Contact** constructor is:

```
public Contact(String username, String email, String id) {
   this.username = username;
   this.email = email;

   if (id == null){
       setId();
   } else {
       updateId(id);
   }
}
```

And the **setId()** and **updateId()** implementation is:

```
public void setId() {
   this.id = UUID.randomUUID().toString();
}

public void updateId(String id){
   this.id = id;
}
```

Notice that **UUID** is red and when you hover over it it gives you an error message, “Cannot resolve symbol ‘UUID’ ”.

To fix this you need to import Java support for UUIDs. To do this, add the following import line to the top of the **Contact** class.

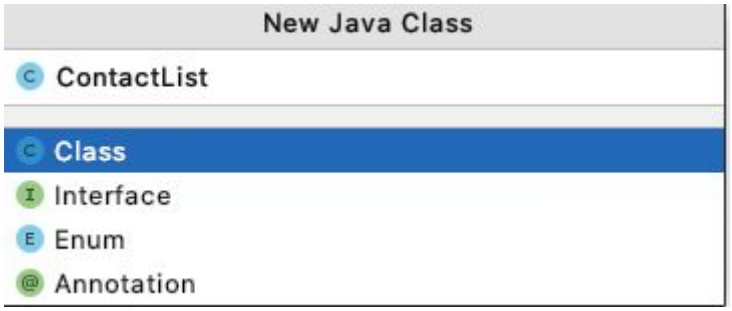
```
import java.util.UUID;
```

Now, it’s your turn to implement the rest of the attributes and methods!

Step 19. Create and Implement the ContactList Class

As previously done, create a new class by right-clicking on the **com.example.sharingapp** folder. **New → Java Class**

Name the class **ContactList**. Hit **Enter**.



This creates an empty **ContactList** class.
Now, implement the **ContactList** methods provided in the Updated UML Class diagram. Make sure you implement **all the attributes and methods** in the **ContactList** class.

Hints

The implementation of the **ContactList** constructor is:

```
public ContactList() {
    contacts = new ArrayList<Contact>();
}
```

The implementation of many of the methods in **ContactList** are analogous to those methods in **ItemList**. For example, the methods **loadContacts()** and **saveContacts()** are **completely analogous** to the **loadItems()** and **saveItems()** methods in the **ItemList** class.

If you use some of the code from the ItemList class, you may notice that you get a lot of import errors. Recall that you can click the red text and press **alt** and **enter** at the same time to import the necessary things. Alternatively, you can copy the imports from the **ItemList** class file to the top of your **ContactList** class file.

Step 20. Run the App

Assuming you have correctly implemented the **Contact** and **ContactList** classes, and do not have any remaining imports to be added to your project, at this point you should be able to run the app by clicking the **play** button.

Be patient! It may take a few minutes to launch SharingApp.