UNIVERSITY OF ALBERTA
FACULTY OF SCIENCE
Department of Computing Science

SOFTWARE DESIGN AND ARCHITECTURE

Course 2 Capstone Peer Review 2.2 Tutorial

Course 2 Capstone Peer Review 2 Tutorial

This tutorial walks you through most of the steps involved in updating the starter code base to implement the MVC pattern. The steps in this tutorial are:

You do not necessarily have to go through all these steps manually, you could opt to start this assignment from the Peer Review 2 starter code base.

If you would like to opt to simply use the Peer Review 2 starter code base, you must still visit steps in the tutorial:

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

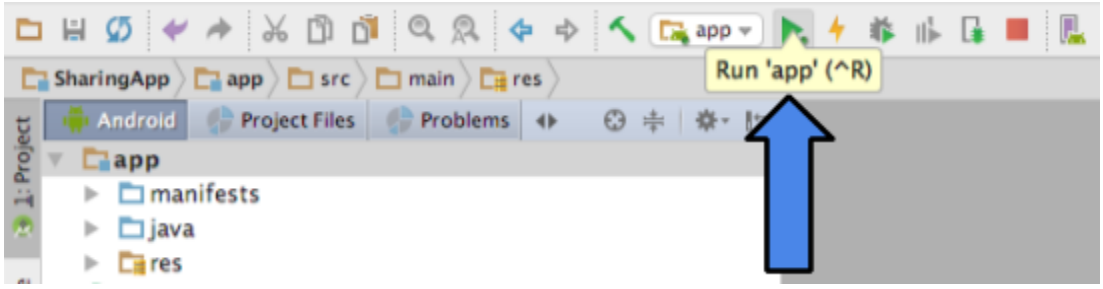
When you implement the MVC Pattern for this assignment, the features and functionality of the app should not change. By implementing this design pattern you are simply organizing the code so that there is an observer relationship between the views and the model, and so that there is a barrier between the views and the model, the controllers.

---

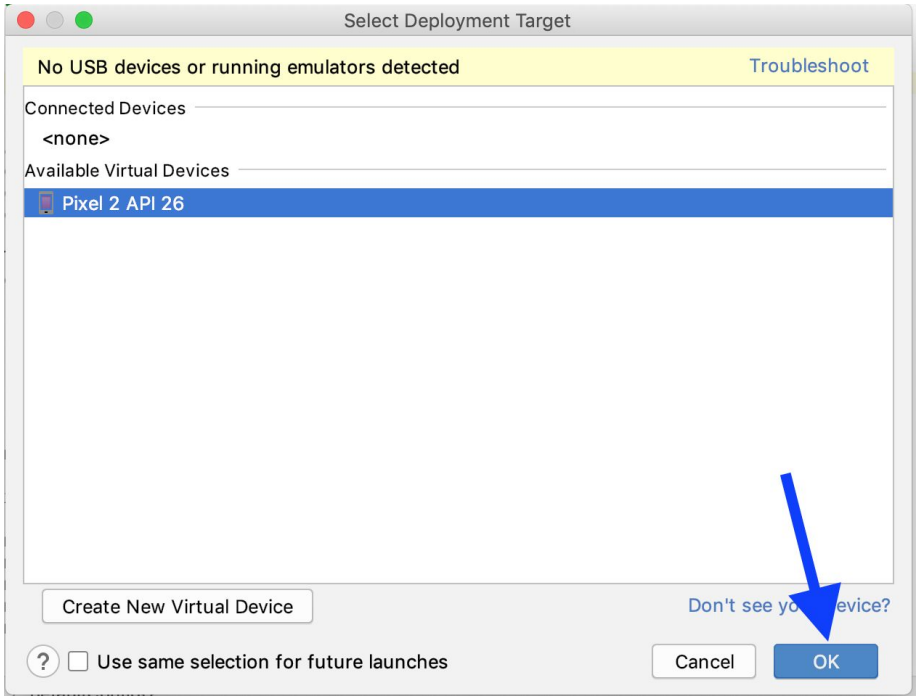
1. Clear the app memory

If you already have a previous version of SharingApp on your emulator it is a good idea to clear the app's data. If we don't clear the previously stored data then the app may crash due to changes made to the model.

After opening Android Studio click the **play button** to run the app.



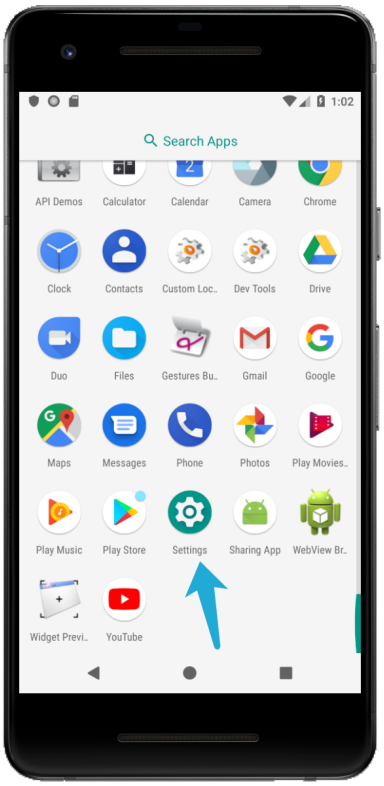
Select the emulator from the list and press **OK**.



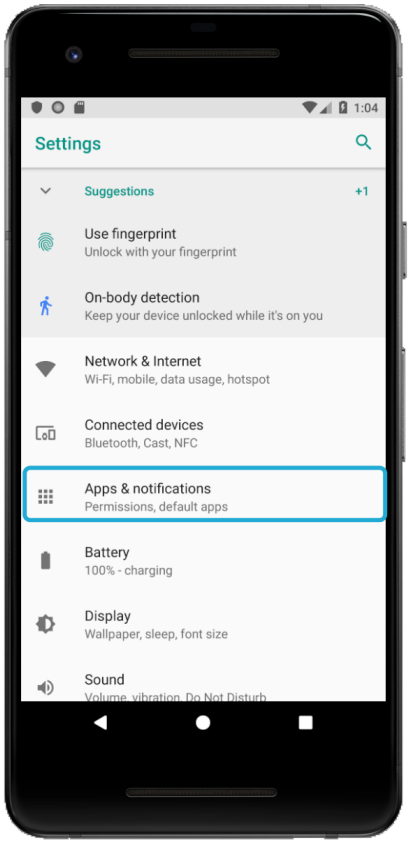
Be patient, the emulator may take a few minutes to load.
If the app launches and doesn't crash -- great! You are done **Step 1**. You can move onto **Step 2**.

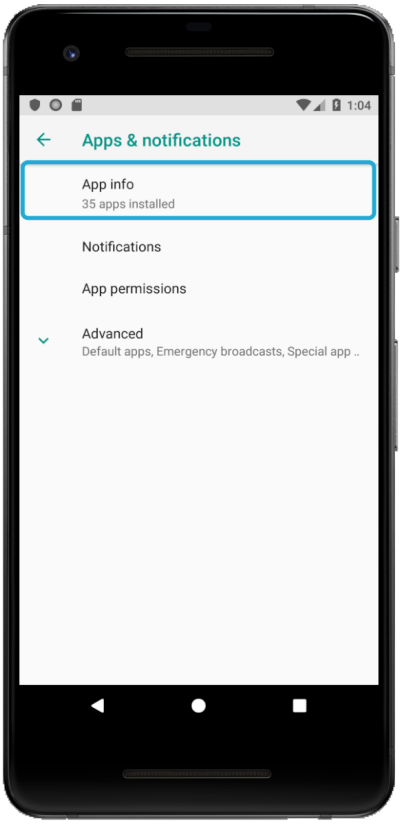
If it does crash -- don't worry. A message will appear to inform you that the app has crashed. Click **OK**. Then, click the button near the bottom of the screen that is made up of six circles.

Click and drag to scroll through the apps until you find the **Settings** app. Click **Settings**.

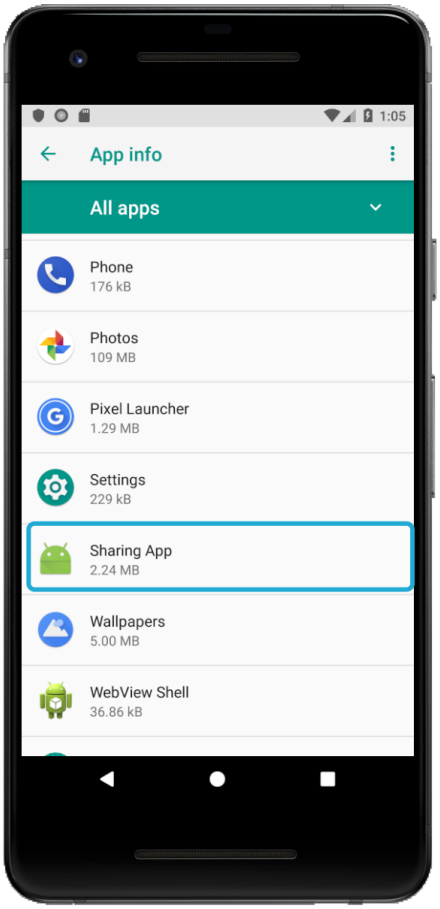


Then click **Apps & notifications**, then **App Info**.





This displays all apps on the emulator. Click and drag to scroll through the list. Near the bottom of the list you will find **Sharing App**. Click **Sharing App**.



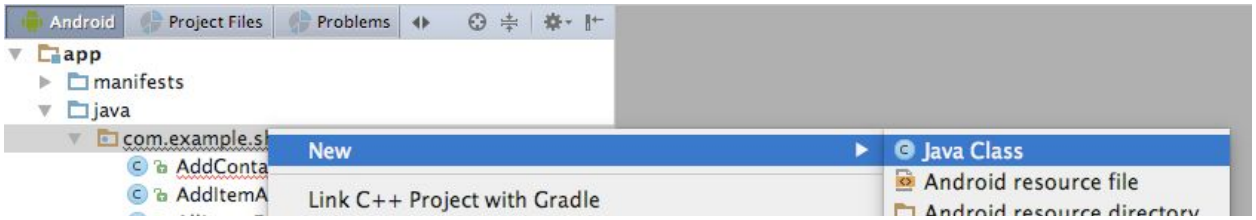
After clicking **Sharing App**, click **Storage**. Then click **CLEAR DATA**. A message will pop up asking you to confirm this action. Click **OK**. Now all the previously stored data has been erased. At this point you may minimize the emulator.

---

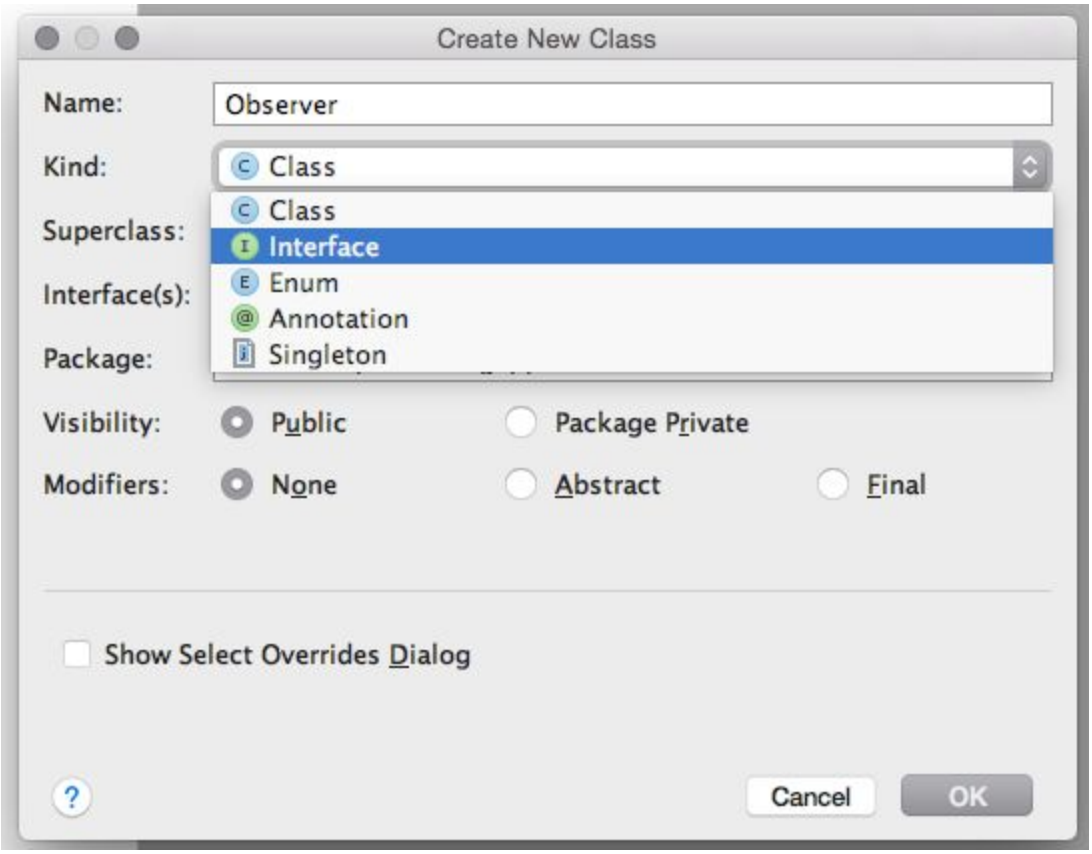
2. Create and implement the Observer Interface

The **Observer** Interface is an essential part of the MVC pattern.

Create a new interface by right-clicking on the **com.example.sharingapp** folder, then click **New → Java Class**.



Name it **Observer**. From the **Kind** dropdown select the option **Interface**, then click **OK**.



This creates an empty **Observer** interface.

Replace the contents of the **Observer** interface with:

```
package com.example.sharingapp;

/**
* Observer Interface
*/
public interface Observer {
  public void update();
}
```

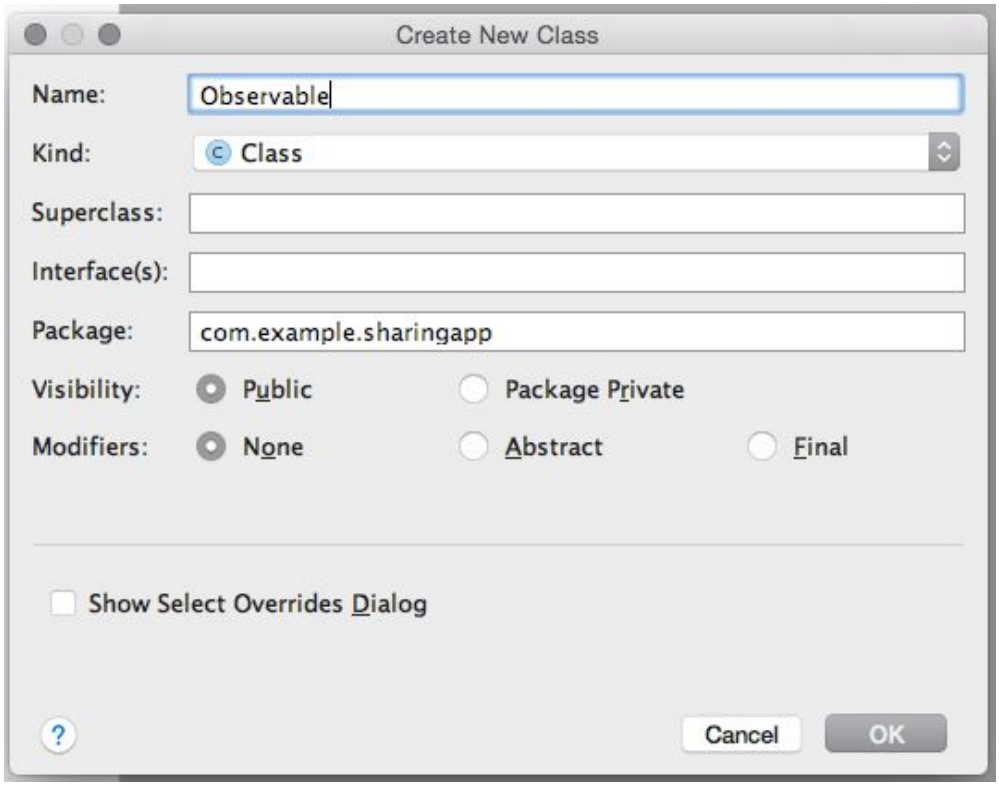
---

3. Create and implement the Observable class

The **Observable** class is also an essential part of the MVC pattern.

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

Name the class **Observable**. Click **OK**.



This creates an empty **Observable** class.
Replace the contents of the **Observable** class with:

```
package com.example.sharingapp;
import java.util.ArrayList;

/**
* Superclass of Item, ItemList, Contact, ContactList
*/
public class Observable {

  private ArrayList<Observer> observers = null;

  public Observable(){
     observers = new ArrayList<Observer>();
  }

  // Notify observers when need to update any changes made to model
  public void notifyObservers() {
     for (Observer observer : observers) {
        observer.update();
     }
  }

  public void addObserver(Observer observer) {
     observers.add(observer);
  }

  public void removeObserver(Observer observer) {
     if (observers.contains(observer)) {
        observers.remove(observer);
     }
  }
}
```

4. Update the Item class

Double click on the **Item** class to open it.

We need to update the **Item** class so that:

- it inherits from the **Observable** class, and
- all methods that make a change to the model call the **notifyObservers()** method.

Replace the contents of **Item** with:

```
package com.example.sharingapp;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.util.Base64;

import java.io.ByteArrayOutputStream;
import java.util.UUID;

/**
* Item class
*/
public class Item extends Observable {

  private String title;
  private String maker;
  private String description;
  private Dimensions dimensions;
  private String status;
  private Contact borrower;
  protected transient Bitmap image;
  protected String image_base64;
  private String id;

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

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

  public String getId(){
     return this.id;
  }

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

```
}

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

public void setTitle(String title) {
    this.title = title;
    notifyObservers();
}

public String getTitle() {
    return title;
}

public void setMaker(String maker) {
    this.maker = maker;
    notifyObservers();
}

public String getMaker() {
    return maker;
}

public void setDescription(String description) {
    this.description = description;
    notifyObservers();
}

public String getDescription() {
    return description;
}

public void setDimensions(String length, String width, String height) {
    this.dimensions = new Dimensions(length, width, height);
    notifyObservers();
}

public String getLength(){
    return dimensions.getLength();
}

public String getWidth(){
    return dimensions.getWidth();
}

public String getHeight(){
    return dimensions.getHeight();
}

public void setStatus(String status) {
    this.status = status;
    notifyObservers();
}

public String getStatus() {
    return status;
}

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

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

  public void addImage(Bitmap new_image){
     if (new_image != null) {
        image = new_image;
        ByteArrayOutputStream byteArrayBitmapStream = new ByteArrayOutputStream();
        new_image.compress(Bitmap.CompressFormat.PNG, 100, byteArrayBitmapStream);

        byte[] b = byteArrayBitmapStream.toByteArray();
        image_base64 = Base64.encodeToString(b, Base64.DEFAULT);
     }
     notifyObservers();
  }

  public Bitmap getImage(){
     if (image == null && image_base64 != null) {
        byte[] decodeString = Base64.decode(image_base64, Base64.DEFAULT);
        image = BitmapFactory.decodeByteArray(decodeString, 0, decodeString.length);
        notifyObservers();
     }
     return image;
  }
}
```

---

5. Update the ItemList class

Double click on the **ItemList** class to open it.

We need to update the **ItemList** class so that:

- it inherits from the **Observable** class, and
- all methods that make a change to the model call the **notifyObservers()** method.

Replace the contents of **ItemList** with:

```
package com.example.sharingapp;

import android.content.Context;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.lang.reflect.Type;
import java.util.ArrayList;

/**
* ItemList class
*/
public class ItemList extends Observable{
```

```
private static ArrayList<Item> items;
private String FILENAME = "items.sav";

public ItemList() {
    items = new ArrayList<Item>();
}

public void setItems(ArrayList<Item> item_list) {
    items = item_list;
    notifyObservers();
}

public ArrayList<Item> getItems() {
    return items;
}

public void addItem(Item item) {
    items.add(item);
    notifyObservers();
}

public void deleteItem(Item item) {
    items.remove(item);
    notifyObservers();
}

public Item getItem(int index) {
    return items.get(index);
}

public int getIndex(Item item) {
    int pos = 0;
    for (Item i : items) {
        if (item.getId().equals(i.getId())) {
            return pos;
        }
        pos = pos + 1;
    }
    return -1;
}

public int getSize() {
    return items.size();
}

public void loadItems(Context context) {

    try {
        FileInputStream fis = context.openFileInput(FILENAME);
        InputStreamReader isr = new InputStreamReader(fis);
        Gson gson = new Gson();
        Type listType = new TypeToken<ArrayList<Item>>() {
        }.getType();
        items = gson.fromJson(isr, listType); // temporary
        fis.close();
    } catch (FileNotFoundException e) {
        items = new ArrayList<Item>();
    } catch (IOException e) {
        items = new ArrayList<Item>();
    }
    notifyObservers();
}
```

```
    public boolean saveItems(Context context) {
        try {
            FileOutputStream fos = context.openFileOutput(FILENAME, 0);
            OutputStreamWriter osw = new OutputStreamWriter(fos);
            Gson gson = new Gson();
            gson.toJson(items, osw);
            osw.flush();
            fos.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
            return false;
        } catch (IOException e) {
            e.printStackTrace();
            return false;
        }
        return true;
    }

    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;
    }

    public ArrayList<Item> filterItemsByStatus(String status){
        ArrayList<Item> selected_items = new ArrayList<>();
        for (Item i: items) {
            if (i.getStatus().equals(status)) {
                selected_items.add(i);
            }
        }
        return selected_items;
    }
}
```

At this point have updated the **Item** related model to inherit from the **Observable** class and to make calls to the notifyObservers() method when the model is changed. We will have to do this for the **Contact** related model as well, but we will revisit that later in the tutorial.

Next, we will implement the **Item** related controllers. Recall that controllers are used by views (Activities/Fragments) to interact with the model. In MVC views do not make direct contact with the model, instead controllers act as the buffer between them.

---

6. Create and implement the ItemController class

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

Name the class **ItemController**. Click **OK**. This creates an empty **ItemController** class.

Replace the contents of **ItemController** with:

```
package com.example.sharingapp;

import android.graphics.Bitmap;

/**
* ItemController is responsible for all communication between views and Item object
*/
public class ItemController {

  private Item item;

  public ItemController(Item item){
     this.item = item;
  }

  public String getId(){
     return item.getId();
  }

  public void setId() {
     item.setId();
  }

  public void setTitle(String title) {
     item.setTitle(title);
  }

  public String getTitle() {
     return item.getTitle();
  }

  public void setMaker(String maker) {
     item.setMaker(maker);
  }

  public String getMaker() {
     return item.getMaker();
  }

  public void setDescription(String description) {
     item.setDescription(description);
  }

  public String getDescription() {
     return item.getDescription();
  }

  public void setDimensions(String length, String width, String height) {
     item.setDimensions(length, width, height);
  }

  public String getLength() {
     return item.getLength();
  }

  public String getWidth(){
     return item.getWidth();
  }
```

```
  public String getHeight(){
     return item.getHeight();
  }

  public void setStatus(String status) {
     item.setStatus(status);
  }

  public String getStatus() {
     return item.getStatus();
  }

  public void setBorrower(Contact borrower) {
     item.setBorrower(borrower);
  }

  public Contact getBorrower() {
     return item.getBorrower();
  }

  public void addImage(Bitmap new_image){
     item.addImage(new_image);
  }

  public Bitmap getImage(){
     return item.getImage();
  }

  public Item getItem() { return this.item; }

  public void addObserver(Observer observer) {
     item.addObserver(observer);
  }

  public void removeObserver(Observer observer) {
     item.removeObserver(observer);
  }
}
```

---

7. Create and implement the ItemListController class

Create a new class by right-clicking on the **com.example.sharingapp** folder, then click **New** → **Java Class**.
Name the class **ItemListController**. Click **OK**. This creates an empty **ItemListController** class.

Replace the contents of **ItemListController** with:

```
package com.example.sharingapp;

import android.content.Context;

import java.util.ArrayList;

/**
* ItemListController is responsible for all communication between views and ItemList object
*/
public class ItemListController {

  private ItemList item_list;
```

```
  public ItemListController(ItemList item_list){
    this.item_list = item_list;
  }

  public void setItems(ArrayList<Item> item_list) {
    this.item_list.setItems(item_list);
  }

  public ArrayList<Item> getItems() {
    return item_list.getItems();
  }

  public boolean addItem(Item item, Context context){
    AddItemCommand add_item_command = new AddItemCommand(item_list, item, context);
    add_item_command.execute();
    return add_item_command.isExecuted();
  }

  public boolean deleteItem(Item item, Context context) {
    DeleteItemCommand delete_item_command = new DeleteItemCommand(item_list, item, context);
    delete_item_command.execute();
    return delete_item_command.isExecuted();
  }

  public boolean editItem(Item item, Item updated_item, Context context){
    EditItemCommand edit_item_command = new EditItemCommand(item_list, item, updated_item, context);
    edit_item_command.execute();
    return edit_item_command.isExecuted();
  }

  public Item getItem(int index) {
    return item_list.getItem(index);
  }

  public int getIndex(Item item) {
    return item_list.getIndex(item);
  }

  public int getSize() {
    return item_list.getSize();
  }

  public void loadItems(Context context) {
    item_list.loadItems(context);
  }

  public ArrayList<Contact> getActiveBorrowers() {
    return item_list.getActiveBorrowers();
  }

  public ArrayList<Item> filterItemsByStatus(String status){
    return item_list.filterItemsByStatus(status);
  }

  public void addObserver(Observer observer) {
    item_list.addObserver(observer);
  }

  public void removeObserver(Observer observer) {
    item_list.removeObserver(observer);
  }
}
```

Now the item related commands are called from the **ItemListController** class. That is, the **addItem()** method uses the **addItemCommand**, the **deleteItem()** method uses the **deleteItemCommand**, and the **editItem()** method uses the **editItemCommand**.

At this point we have made all **Item** related controllers. We will have to make controllers for the **Contact** related model as well, but we will revisit that later in the tutorial.

Next, we will update our views (Activities/Fragments) to make use of **ItemController** and **ItemListController**.

---

8. Update AddItemActivity

Double click on the **AddItemActivity** class to open it.

We need to:

- Create instances of the **ItemController** and **ItemListController** classes, and
- Replace calls to **Item** with calls to **ItemController**.
- Replace calls to **ItemList**, and **AddItemCommand** with calls to **ItemListController**.

Replace the current contents of **AddItemActivity** with:

```
package com.example.sharingapp;

import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.provider.MediaStore;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;

/**
* Add a new item
*/
public class AddItemActivity extends AppCompatActivity {

  private EditText title;
  private EditText maker;
  private EditText description;
  private EditText length;
  private EditText width;
  private EditText height;

  private ImageView photo;
  private Bitmap image;
  private int REQUEST_CODE = 1;

  private ItemList item_list = new ItemList();
  private ItemListController item_list_controller = new ItemListController(item_list);

  private Context context;
```

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

    setContentView(R.layout.activity_add_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);
    photo = (ImageView) findViewById(R.id.image_view);

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

    context = getApplicationContext();
    item_list_controller.loadItems(context);
}

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();

    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;
    }

    Item item = new Item(title_str, maker_str, description_str, image, null);
    ItemController item_controller = new ItemController(item);
    item_controller.setDimensions(length_str, width_str, height_str);

    // Add item
```

```
        boolean success = item_list_controller.addItem(item, context);
        if (!success) {
            return;
        }

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

    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);
        }
    }
}
```

---

9. Update EditItemActivity

Double click on the **EditItemActivity** class to open it.

We need to update **EditItemActivity** to:

- Implement the **Observer** interface
- Create instances of the **ItemController and ItemListController** classes, and
- Replace calls to **Item** with calls to **ItemController**.
- Replace calls to **ItemList, EditItemCommand,** and **DeleteItemCommand** with calls to **ItemListController**.

Replace the current contents of **EditItemActivity** with:

```
package com.example.sharingapp;

import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.provider.MediaStore;
import android.support.v7.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 implements Observer {

  private ItemList item_list = new ItemList();
  private ItemListController item_list_controller = new ItemListController(item_list);

  private Item item;
  private ItemController item_controller;

  private Context context;

  private ContactList contact_list = new ContactList();
  private ContactListController contact_list_controller = new ContactListController(contact_list);

  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;

  private ArrayAdapter<String> adapter;
  private boolean on_create_update = false;
  private int pos;

  @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);

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

```
    context = getApplicationContext();

    item_list_controller.addObserver(this);
    item_list_controller.loadItems(context);

    on_create_update = true;

    contact_list_controller.addObserver(this);
    contact_list_controller.loadContacts(context);

    on_create_update = false;
}

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) {

    // Delete item
    boolean success = item_list_controller.deleteItem(item, context);
    if (!success) {
        return;
    }

    // End EditItemActivity
    item_list_controller.removeObserver(this);

    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_controller.getContactByUsername(borrower_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_controller.getId(); // Reuse the item id
    Item updated_item = new Item(title_str, maker_str, description_str, image, id);
    ItemController updated_item_controller = new ItemController(updated_item);
    updated_item_controller.setDimensions(length_str, width_str, height_str);

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

    // Edit item
    boolean success = item_list_controller.editItem(item, updated_item, context);
    if (!success) {
        return;
    }

    // End EditItemActivity
    item_list_controller.removeObserver(this);

    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_controller.setBorrower(null);
            item_controller.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);
            }
        }
    }

    /**
     * Only need to update the view from the onCreate method
     */
    public void update() {
        if (on_create_update){
            adapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_dropdown_item,
                    contact_list_controller.getAllUsernames());
            borrower_spinner.setAdapter(adapter);

            item = item_list_controller.getItem(pos);
            item_controller = new ItemController(item);

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

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

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

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

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

Notice that everything related to the **ContactListController** is shown in red. This makes sense because we have not yet created the **ContactListController** class. For now, this is fine -- just ignore the errors. Later on it the tutorial you will have a chance to create and implement the **ContactListController** class.

---

10. Update ItemAdapter to use ItemController and ItemController

Double click on the **ItemAdapter** class to open it.

We need to:

- Create an instance of the **ItemController** class, and
- Replace calls to the **Item** class with calls to the **ItemController**

```
package com.example.sharingapp;

import android.content.Context;
import android.graphics.Bitmap;
import android.support.v4.app.Fragment;
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;

/**
* ItemAdapter is responsible for what information is displayed in ListView entries.
*/
public class ItemAdapter extends ArrayAdapter<Item> {

  private LayoutInflater inflater;
  private Fragment fragment;
  private Context context;

  public ItemAdapter(Context context, ArrayList<Item> items, Fragment fragment) {
     super(context, 0, items);
     this.context = context;
     this.inflater = LayoutInflater.from(context);
     this.fragment = fragment;
  }

  @Override
  public View getView(int position, View convertView, ViewGroup parent) {

     // getItem(position) gets the "item" at "position" in the "items" ArrayList
     // (where "items" is a parameter in the ItemAdapter creator as seen above ^^)
     Item item = getItem(position);
     ItemController item_controller = new ItemController(item);

     String title = "Title: " + item_controller.getTitle();
     String description = "Description: " + item_controller.getDescription();
     Bitmap thumbnail = item_controller.getImage();
     String status = "Status: " + item_controller.getStatus();

     // Check if an existing view is being reused, otherwise inflate the view.
```

```
    if (convertView == null) {
        convertView = inflater.from(context).inflate(R.layout.itemlist_item, parent, false);
    }

    TextView title_tv = (TextView) convertView.findViewById(R.id.title_tv);
    TextView status_tv = (TextView) convertView.findViewById(R.id.status_tv);
    TextView description_tv = (TextView) convertView.findViewById(R.id.description_tv);
    ImageView photo = (ImageView) convertView.findViewById(R.id.image_view);

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

    title_tv.setText(title);
    description_tv.setText(description);

    // AllItemFragments: itemlist item shows title, description and status
    if (fragment instanceof AllItemsFragment ) {
        status_tv.setText(status);
    }

    // BorrowedItemsFragment/AvailableItemsFragment: itemlist item shows title and description only
    if (fragment instanceof BorrowedItemsFragment || fragment instanceof AvailableItemsFragment) {
        status_tv.setVisibility(View.GONE);
    }

    return convertView;
  }
}
```

---

11. Update ItemsFragment

Double click on the **ItemsFragment** class to open it.

We need to update **ItemsFragment** to:

- Implement the **update()** method from the **Observer** interface
- Create an instance of the **ItemListController** class, and
- Replace calls to the **ItemList** class with calls to the **ItemListController**

Replace the current contents of **ItemsFragment** with:

```
package com.example.sharingapp;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
```

```
import java.util.ArrayList;

/**
* Superclass of AvailableItemsFragment, BorrowedItemsFragment and AllItemsFragment
*/
public abstract class ItemsFragment extends Fragment implements Observer {

  private ItemList item_list = new ItemList();
  ItemListController item_list_controller = new ItemListController(item_list);

  View rootView;
  private ListView list_view;
  private ArrayAdapter<Item> adapter;
  private ArrayList<Item> selected_items;
  private LayoutInflater inflater;
  private ViewGroup container;
  private Context context;
  private Fragment fragment;
  private boolean update = false;

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
     context = getContext();

     // Don't update view yet. Wait until after items have been filtered.
     item_list_controller.loadItems(context);
     update = true;

     this.inflater = inflater;
     this.container = container;

     return rootView;
  }

  public void setVariables(int resource, int id ) {
     rootView = inflater.inflate(resource, container, false);
     list_view = (ListView) rootView.findViewById(id);
     selected_items = filterItems();
  }

  public void loadItems(Fragment fragment){
     this.fragment = fragment;
     item_list_controller.addObserver(this);
     item_list_controller.loadItems(context);
  }

  public void setFragmentOnItemLongClickListener(){

     // When item is long clicked, this starts EditItemActivity
     list_view.setOnItemLongClickListener(new android.widget.AdapterView.OnItemLongClickListener() {

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

           Item item = adapter.getItem(pos);

           int meta_pos = item_list_controller.getIndex(item);
           if (meta_pos >= 0) {

              Intent edit = new Intent(context, EditItemActivity.class);
              edit.putExtra("position", meta_pos);
              startActivity(edit);
           }
           return true;
```

```
        }
    });
  }

  /**
   * filterItems is implemented independently by AvailableItemsFragment, BorrowedItemsFragment and AllItemsFragment
   * @return selected_items
   */
  public abstract ArrayList<Item> filterItems();

  /**
   * Called when the activity is destroyed, thus we remove this fragment as an observer
   */
  @Override
  public void onDestroy() {
    super.onDestroy();
    item_list_controller.removeObserver(this);
  }

  /**
   * Update the view
   */
  public void update(){
    if (update) {
      adapter = new ItemAdapter(context, selected_items, fragment);
      list_view.setAdapter(adapter);
      adapter.notifyDataSetChanged();
    }
  }
}
```

---

12. Update AllItemsFragment

Double click on **AllItemsFragment** to open it.

We need to:

- Replace the call to the **ItemList** class with a call to the **ItemListController**.instance instead.
- Make some small changes to the **OnCreateView()** method due to the addition of the **update()** method in **ItemsFragment**

Replace the contents of **AllItemsFragment** with:

```
package com.example.sharingapp;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import java.util.ArrayList;

/**
* Displays a list of all items
*/
public class AllItemsFragment extends ItemsFragment {

  @Override
```

```
  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

      super.onCreateView(inflater, container, savedInstanceState);
      super.setVariables(R.layout.all_items_fragment, R.id.my_items);
      super.loadItems(AllItemsFragment.this);
      super.setFragmentOnItemLongClickListener();

      return rootView;
  }

  public ArrayList<Item> filterItems() {
      return item_list_controller.getItems();
  }
}
```

---

13. Update AvailableItemsFragment

Double click on **AvailableItemsFragment** to open it.

We need to:

- Replace the call to the **ItemList** class with a call to the **ItemListController**.instance instead.
- Make some small changes to the **OnCreateView()** method due to the addition of the **update()** method in **ItemsFragment**

Replace the contents of **AvailableItemsFragment** with:

```
package com.example.sharingapp;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import java.util.ArrayList;

/**
* Displays a list of all "Available" items
*/
public class AvailableItemsFragment extends ItemsFragment{

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

      super.onCreateView(inflater,container, savedInstanceState);
      super.setVariables(R.layout.available_items_fragment, R.id.my_available_items);
      super.loadItems(AvailableItemsFragment.this);
      super.setFragmentOnItemLongClickListener();

      return rootView;
  }

  public ArrayList<Item> filterItems() {
      String status = "Available";
      return item_list_controller.filterItemsByStatus(status);
  }
}
```

14. Update BorrowedItemsFragment

Double click on **BorrowedtemsFragment** to open it.

We need to:

- Replace the call to the **ItemList** class with a call to the **ItemListController**.instance instead.
- Make some small changes to the **OnCreateView()** method due to the addition of the **update()** method in **ItemsFragment**

Replace the contents of **BorrowedtemsFragment** with:

```
package com.example.sharingapp;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import java.util.ArrayList;

/**
* Displays a list of all "Borrowed" Items
*/
public class BorrowedItemsFragment extends ItemsFragment {

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

     super.onCreateView(inflater,container, savedInstanceState);
     super.setVariables(R.layout.borrowed_items_fragment, R.id.my_borrowed_items);
     super.loadItems(BorrowedItemsFragment.this);
     super.setFragmentOnItemLongClickListener();

     return rootView;
  }

  public ArrayList<Item> filterItems() {
     String status = "Borrowed";
     return item_list_controller.filterItemsByStatus(status);
  }
}
```

15. Update ContactsActivity

Double click on **ContactsActivity** to open it.

We need to:

- Create an instance of the **ItemListController** class, and
- Replace the call to the **ItemList** class with a call to the **ItemListController**

Replace the contents of **ContactsActivity** with:

```
package com.example.sharingapp;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

/**
* Displays a list of all contacts
* Note: You will not be able edit/delete contacts which are "active" borrowers
*/
public class ContactsActivity extends AppCompatActivity implements Observer {

  private ContactList contact_list = new ContactList();
  private ContactListController contact_list_controller = new ContactListController(contact_list);

  private ContactList active_borrowers_list = new ContactList();
  private ContactListController active_borrowers_list_controller = new ContactListController(active_borrowers_list);

  private ItemList item_list = new ItemList();
  private ItemListController item_list_controller = new ItemListController(item_list);

  private ListView my_contacts;
  private ArrayAdapter<Contact> adapter;
  private Context context;

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

     context = getApplicationContext();

     contact_list_controller.addObserver(this);
     contact_list_controller.loadContacts(context);
     item_list_controller.loadItems(context);

     // 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);

           // Do not allow an "active" borrower to be edited
           active_borrowers_list_controller.setContacts(item_list_controller.getActiveBorrowers());
           if (active_borrowers_list_controller != null) {
              if (active_borrowers_list_controller.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_controller.loadContacts(context); // must load contacts again here
            int meta_pos = contact_list_controller.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_controller.loadContacts(context);
}

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

/**
 * Called when the activity is destroyed, thus we remove this activity as a listener
 */
@Override
protected void onDestroy() {
    super.onDestroy();
    contact_list_controller.removeObserver(this);
}

/**
 * Update the view
 */
public void update(){
    my_contacts = (ListView) findViewById(R.id.my_contacts);
    adapter = new ContactAdapter(ContactsActivity.this, contact_list_controller.getContacts());
    my_contacts.setAdapter(adapter);
    adapter.notifyDataSetChanged();
}
}
```

Notice that everything related to **ContactController** and **ContactListController** are shown in red. Ignore this for now.

---

16. Update the Contact class

Double click on the **Contact** class to open it.

Update **Contact** so that:

- It inherits from the **Observable** class, and
- All methods that make a change to the model call the **notifyObservers()** method.

Hint: this step is analogous to **Step 4**.

---

17. Update the ContactList class to extend the Observable class

Double click on the **ContactList** class to open it.

Update **ContactList** so that:
- It inherits from the **Observable** class, and
- All methods that make a change to the model call the **notifyObservers()** method.

Hint: this step is analogous to **Step 5**.

---

18. Create and implement the ContactController class

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

Name the class **ContactController**. Click **OK**. This creates an empty **ContactController** class.
Now it's your turn to implement it.

Hint: this step is analogous to **Step 6**.

---

19. Create and implement the ContactListController class

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

Name the class **ContactListController**. Click **OK**. This creates an empty **ContactListController** class.
Now it's your turn to implement it.

Hint: this step is analogous to **Step 7**.

Remember that contact related commands should be called from the **ContactListController** class. Don't forget to implement the following methods in **ContactListController**:
- **addContact()** uses the **AddContactCommand**,
- **deleteContact()** uses the **DeleteContactCommand**, and
- **editContact()** uses the **EditContactCommand**.

Additionally, add the following method to the **ContactListController**

```
public Contact getContactByUsername(String username) {
  return contact_list.getContactByUsername(username);
}
```

You'll notice that we get an error when we call the **getContactByUsername()** method of the **ContactList** class because the method doesn't exist get. We'll add that in a moment.

```
public Contact getContactByUsername(String username) {
    return contact_list.getContactByUsername(username);
}
```

---

20. Update the ContactList class

Double-click on the **ContactList** class.

Add the following method to the **ContactList** class:

```
public Contact getContactByUsername(String username){
  for (Contact c : contacts){
    if (c.getUsername().equals(username)){
      return c;
    }
  }
  return null;
}
```

---

21. Update AddContactActivity

Double click on the **AddContactActivity** class to open it.

Update **AddContactActivity** to:

- Create an instance of the **ContactListController** class, and
- Replace calls to **ContactList** and **AddContactCommand** with calls to the **ContactListController**.

Hint: this step is analogous to **Step 8**.

---

22. Update EditContactActivity

Double click on the **EditContactActivity** class to open it.

We need to update **EditContactActivity** to:

- Implement the **Observer** interface

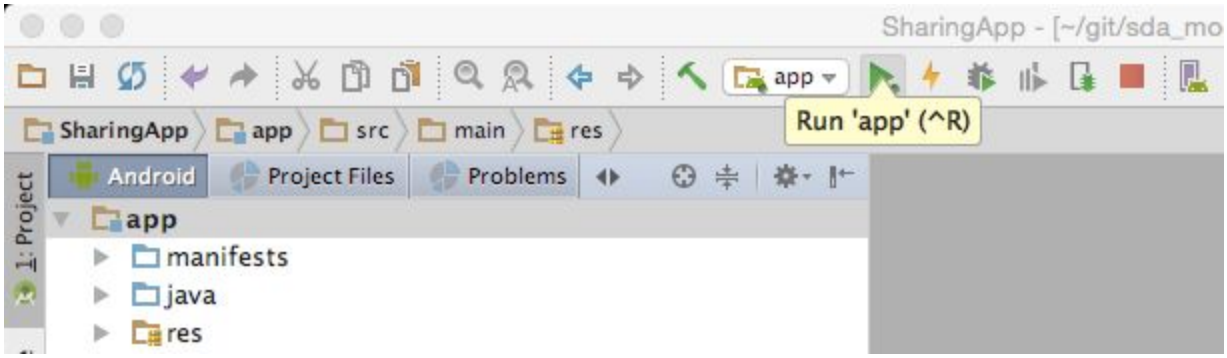
- Create instances of the **ContactController and ContactListController** classes,
- Replace calls to **Contact** with calls to **ContactController**.
- Replace calls to **ContactList, EditContactCommand,** and **DeleteContactCommand** with calls to **ContactListController**.

Hint: this step is analogous to **Step 9**.

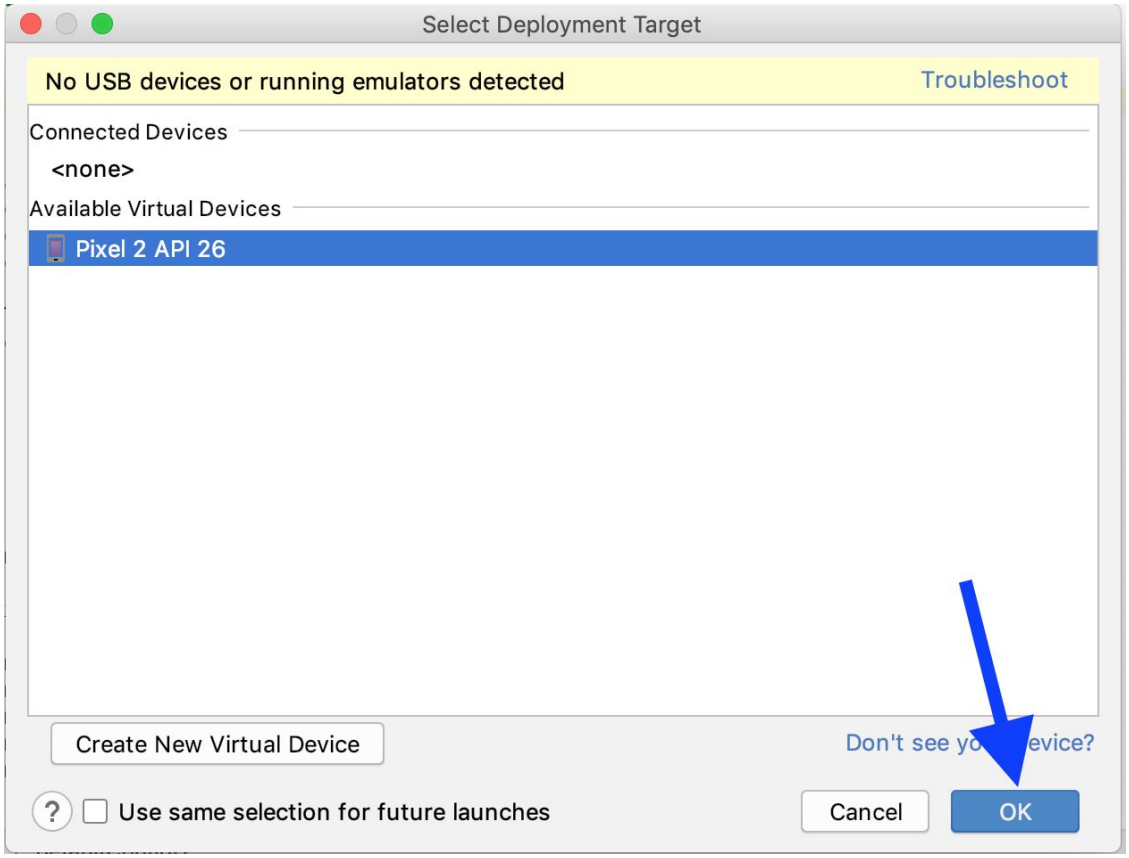
---

23. Run the app

Assuming you have correctly implemented the MVC Design Pattern then at this point you should be able to run the app by clicking the **play** button.



If the emulator is not already launched you will be prompted to select it, then press **OK**.



Be patient! It make take a few minutes to open and launch SharingApp.