COMP3275
Lab 5
SQLite Databases

Kyle De Freitas

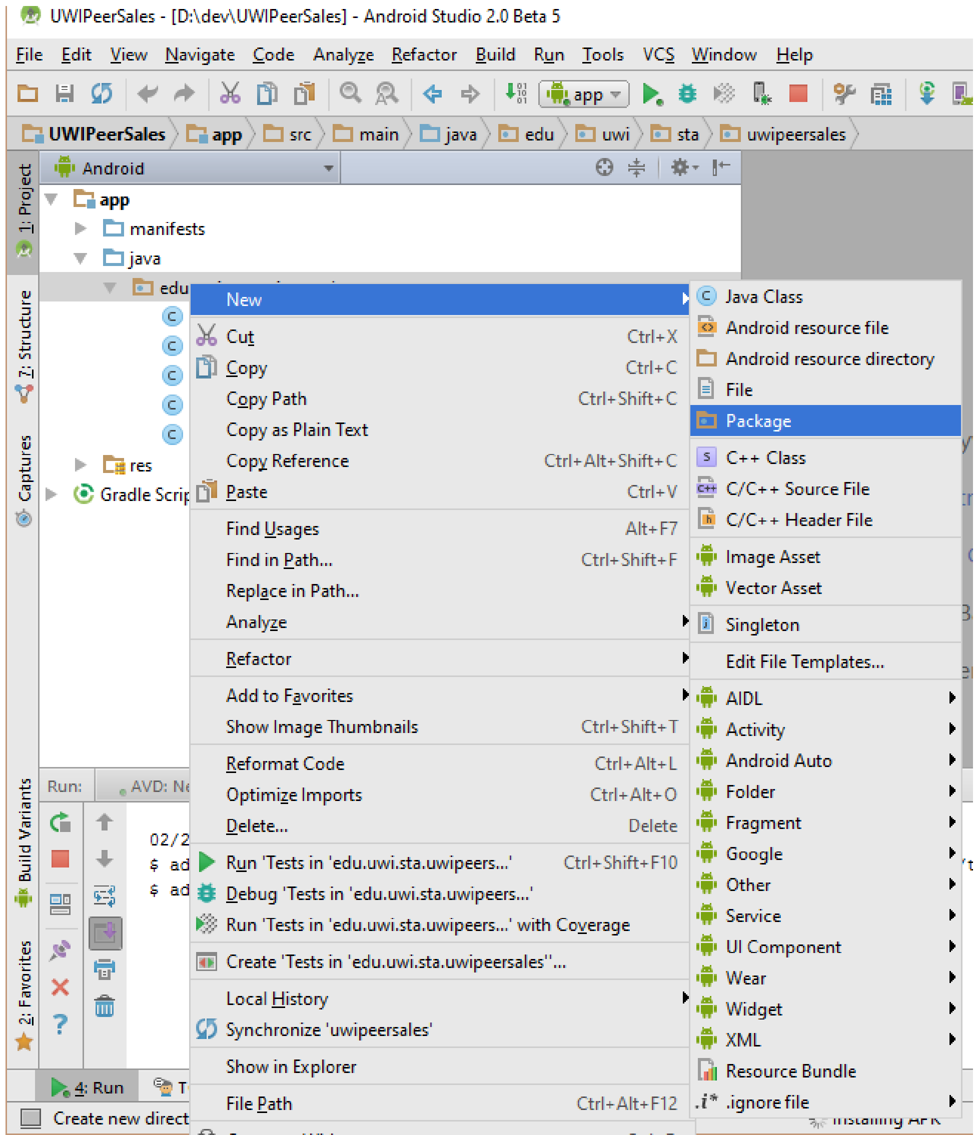
Department of Computing and Information Technology

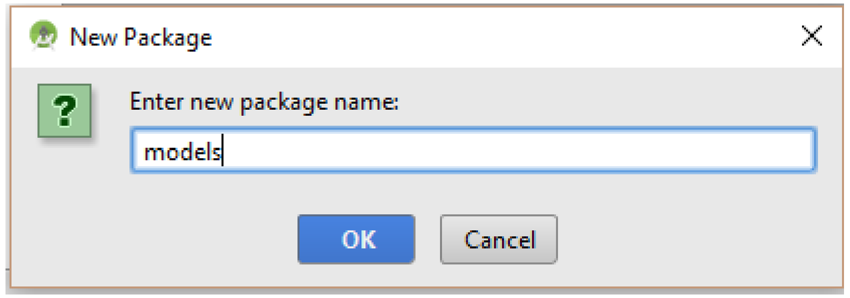
University of the West Indies

Introduction

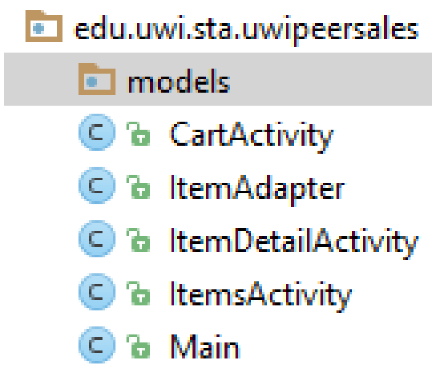
- This lab focuses on providing an introduction for using the SQLite Database to store, retrieve and delete data in an Android application.
- The manipulation of data is a fundamental operation in any application.
- We will be modifying the existing program from Lab 3. You can choose to use your own version or download the solution from the course website.
- Modification of http://developer.android.com/training/basics/data-storage/databases.html for the class

Create Models pacakage


UWIPeerSales - [D:\dev\UWIPeerSales] - Android Studio 2.0 Beta 5
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
UWIPeerSales app src main java edu uwi sta uwipeersales
Android
app
manifests
java
edu
res
Gradle Scrip
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy as Plain Text
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Find Usages Alt+F7
Find in Path... Ctrl+Shift+F
Replace in Path...
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Run 'Tests in 'edu.uwi.sta.uwipeers...' Ctrl+Shift+F10
Debug 'Tests in 'edu.uwi.sta.uwipeers...'
Run 'Tests in 'edu.uwi.sta.uwipeers...' with Coverage
Create 'Tests in 'edu.uwi.sta.uwipeersales''...
Local History
Synchronize 'uwipeersales'
Show in Explorer
File Path Ctrl+Alt+F12
Java Class
Android resource file
Android resource directory
File
Package
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Singleton
Edit File Templates...
AIDL
Activity
Android Auto
Folder
Fragment
Google
Other
Service
UI Component
Wear
Widget
XML
Resource Bundle
.ignore file
Run:
AVD: Ne
4: Run
Create new direct



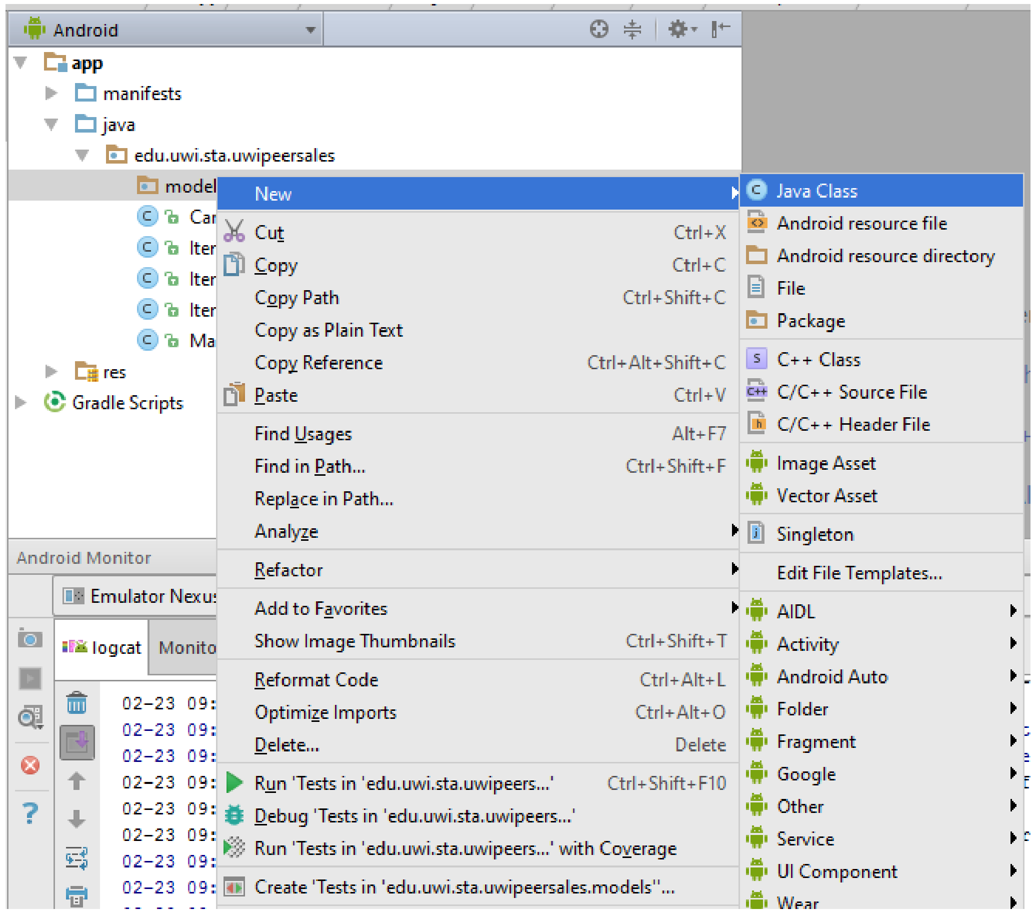
Enter new package name: models

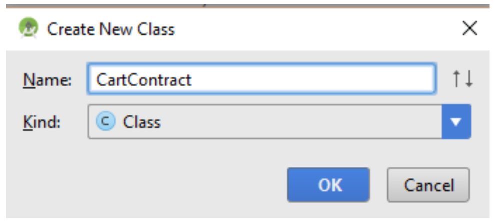


Create data schema for models

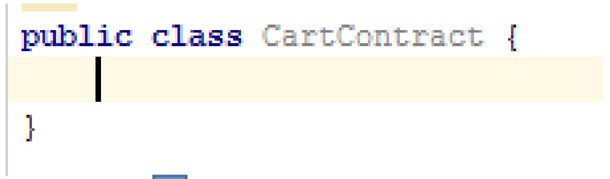
- We will create two models:
  - Cart
  - Product (previously called item)
- We will start with the cart and allow the user to add items to the cart which will be stored in the database for the cart.

Create data schema for models

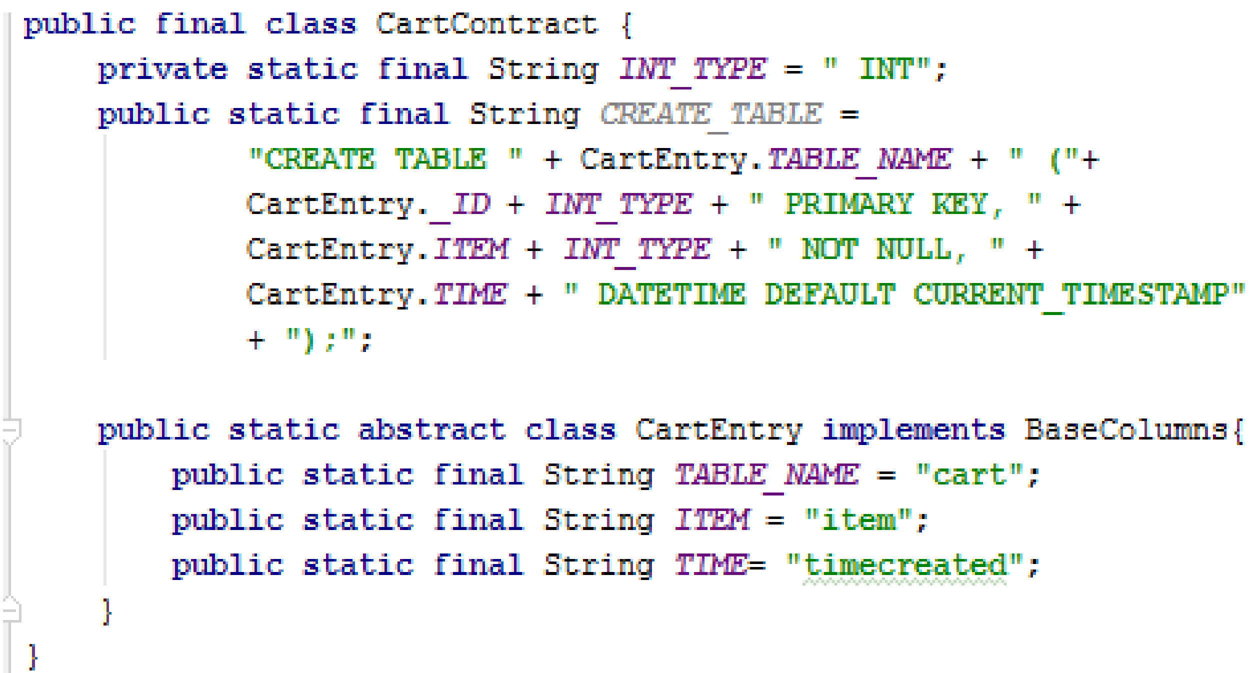




Name: CartContract
Kind: Class



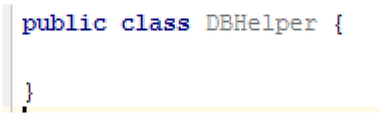
```java
public class CartContract {
    
}
```



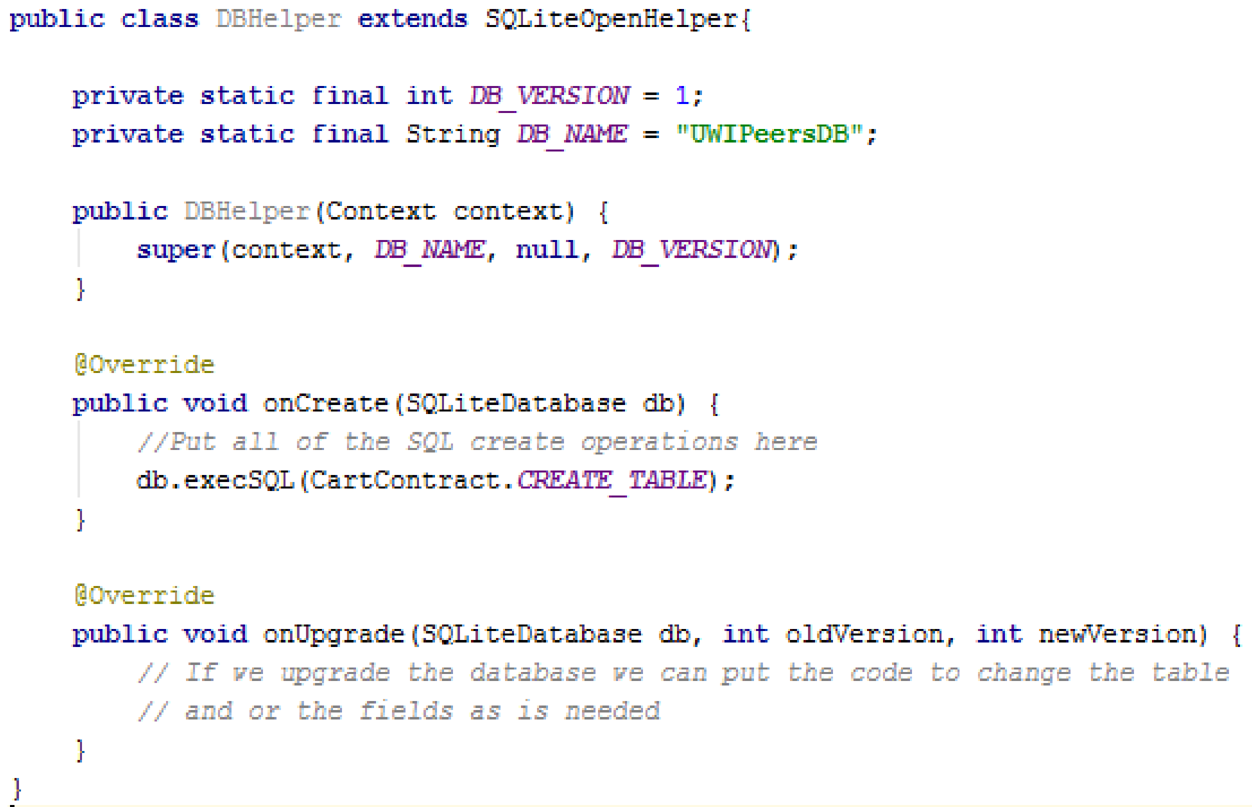
```java
public final class CartContract {
    private static final String INT_TYPE = " INT";
    public static final String CREATE_TABLE =
            "CREATE TABLE " + CartEntry.TABLE_NAME + " ("+
            CartEntry._ID + INT_TYPE + " PRIMARY KEY, " +
            CartEntry.ITEM + INT_TYPE + " NOT NULL, " +
            CartEntry.TIME + " DATETIME DEFAULT CURRENT_TIMESTAMP"
            + ");";

    public static abstract class CartEntry implements BaseColumns{
        public static final String TABLE_NAME = "cart";
        public static final String ITEM = "item";
        public static final String TIME= "timecreated";
    }
}
```

Database helpers

- We need to utilize a database helper class to manage the database creation and version management.
- We do not access the raw SQLite Database that is available but the Android system provides an object that we can perform common SQL CRUD operations on.
- The Database Helper will use our Contracts (Data model schema) to create the table when the database is created.
- The creation of the database is not explicit, but will occur the first time that a request for the database is made.
- In this example we will create our database helper called DBHelper which will extend Android SQLiteOpenHelper.
- While it is possible for one application to have multiple database, we will in this course limit it to only one, as this satisfies the majority of situations that will be encountered.



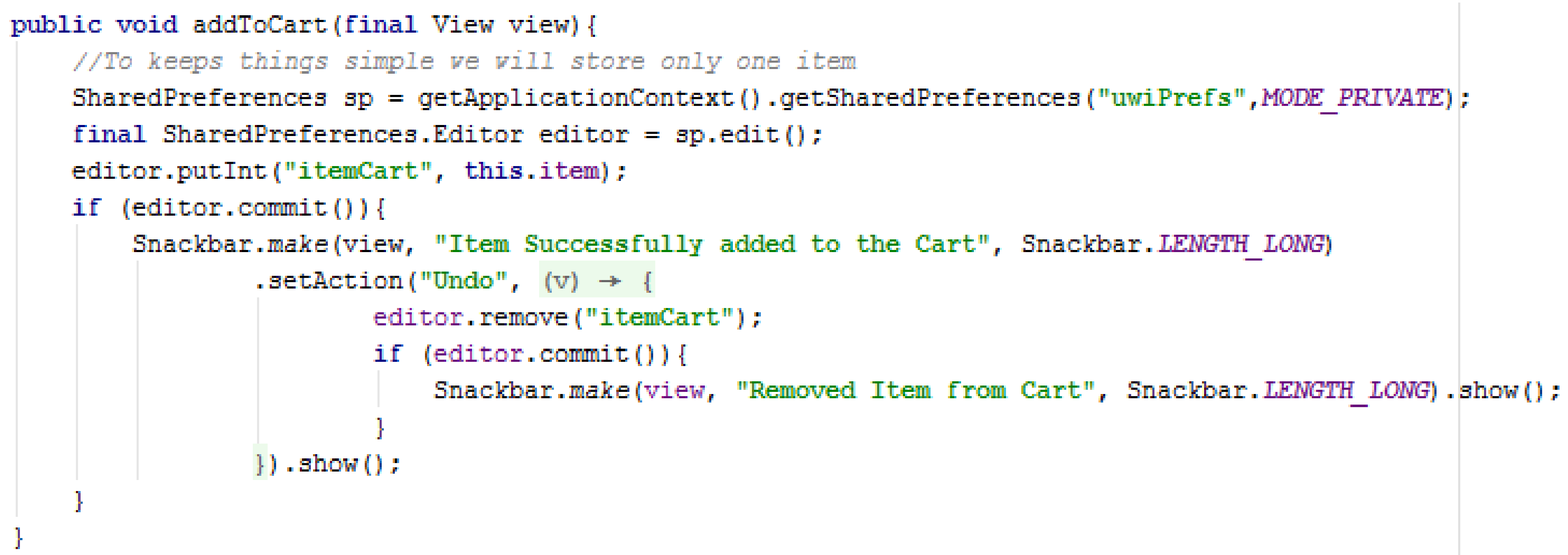
```
public class DBHelper {

}
```



```
public class DBHelper extends SQLiteOpenHelper{

    private static final int DB_VERSION = 1;
    private static final String DB_NAME = "UWIPeersDB";

    public DBHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //Put all of the SQL create operations here
        db.execSQL(CartContract.CREATE_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // If we upgrade the database we can put the code to change the table
        // and or the fields as is needed
    }
}
```

Add Data to Database (Cart Table)

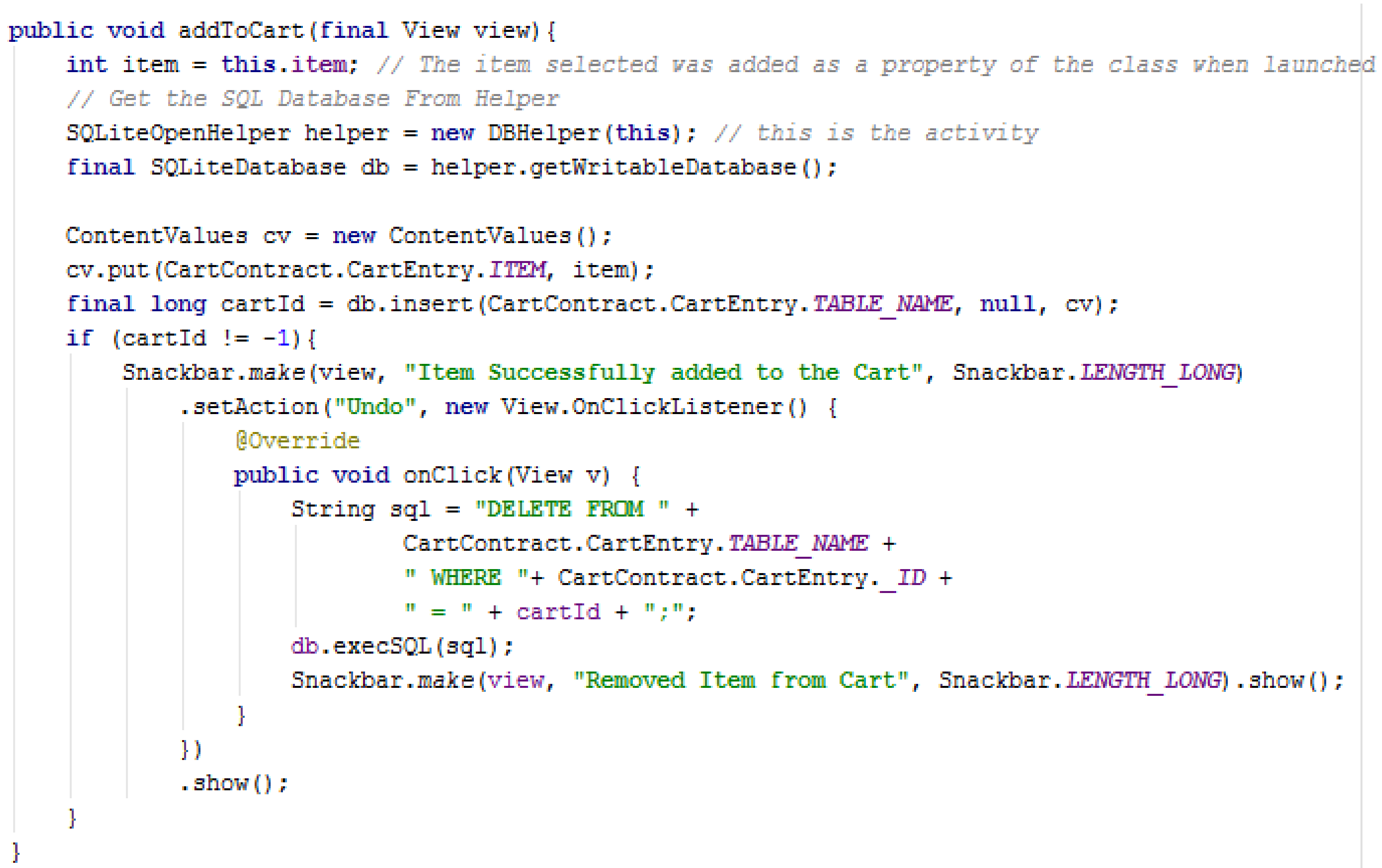
- Here we can extend the example from the add to cart functionality defined in the previous lab. Rather than utilizing the shared preferences, we can utilize the database to store the values.
- We can adjust the code as follows:
- From:



```
public void addToCart(final View view){
    //To keeps things simple we will store only one item
    SharedPreferences sp = getApplicationContext().getSharedPreferences("uwiPrefs",MODE_PRIVATE);
    final SharedPreferences.Editor editor = sp.edit();
    editor.putInt("itemCart", this.item);
    if (editor.commit()){
        Snackbar.make(view, "Item Successfully added to the Cart", Snackbar.LENGTH_LONG)
                .setAction("Undo", (v) → {
                        editor.remove("itemCart");
                        if (editor.commit()){
                            Snackbar.make(view, "Removed Item from Cart", Snackbar.LENGTH_LONG).show();
                        }
                }).show();
    }
}
```

Add Data to Database (Cart Table)

- TO:

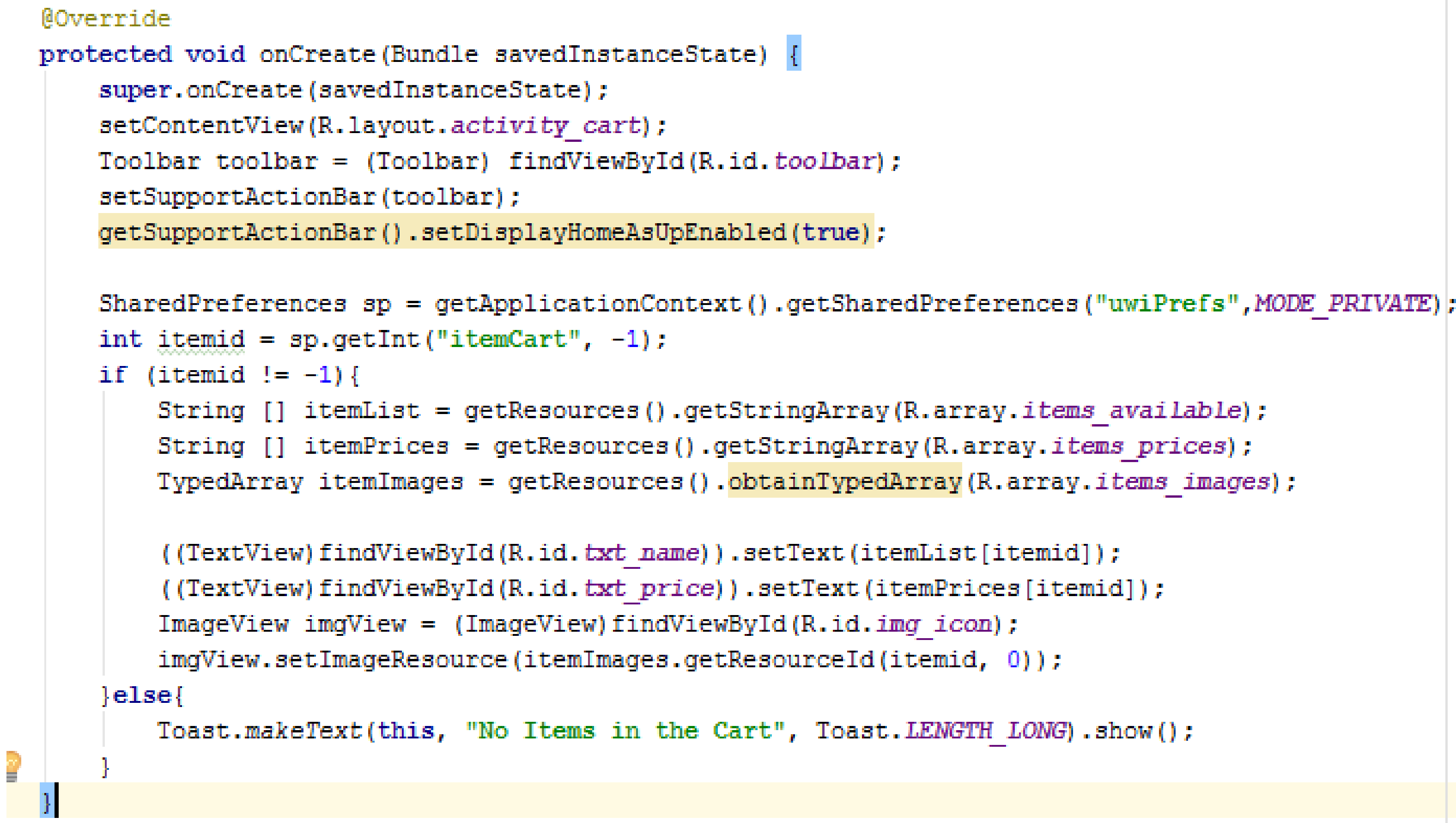


```java
public void addToCart(final View view){
    int item = this.item; // The item selected was added as a property of the class when launched
    // Get the SQL Database From Helper
    SQLiteOpenHelper helper = new DBHelper(this); // this is the activity
    final SQLiteDatabase db = helper.getWritableDatabase();

    ContentValues cv = new ContentValues();
    cv.put(CartContract.CartEntry.ITEM, item);
    final long cartId = db.insert(CartContract.CartEntry.TABLE_NAME, null, cv);
    if (cartId != -1){
        Snackbar.make(view, "Item Successfully added to the Cart", Snackbar.LENGTH_LONG)
            .setAction("Undo", new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    String sql = "DELETE FROM " +
                            CartContract.CartEntry.TABLE_NAME +
                            " WHERE "+ CartContract.CartEntry._ID +
                            " = " + cartId + ";";
                    db.execSQL(sql);
                    Snackbar.make(view, "Removed Item from Cart", Snackbar.LENGTH_LONG).show();
                }
            })
            .show();
    }
}
```

Extract From Database (Cart)

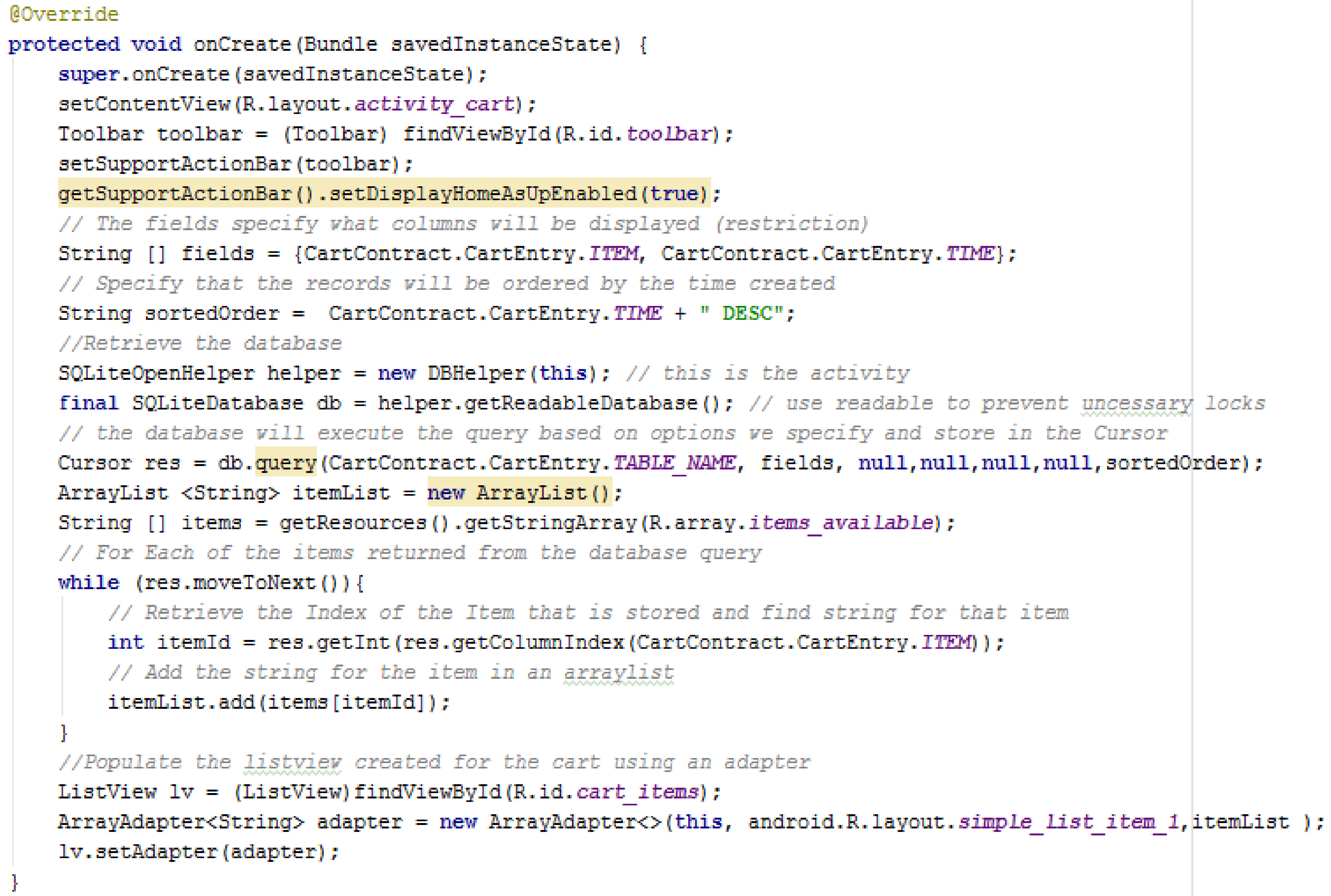
- We want to do more than simply add to the cart, we also want to display the information that is stored to the cart. On the activity responsible for displaying the content of the cart we can add code that will display the information from the cart within the page:
- We will adjust the CartActivity as follows:

From:

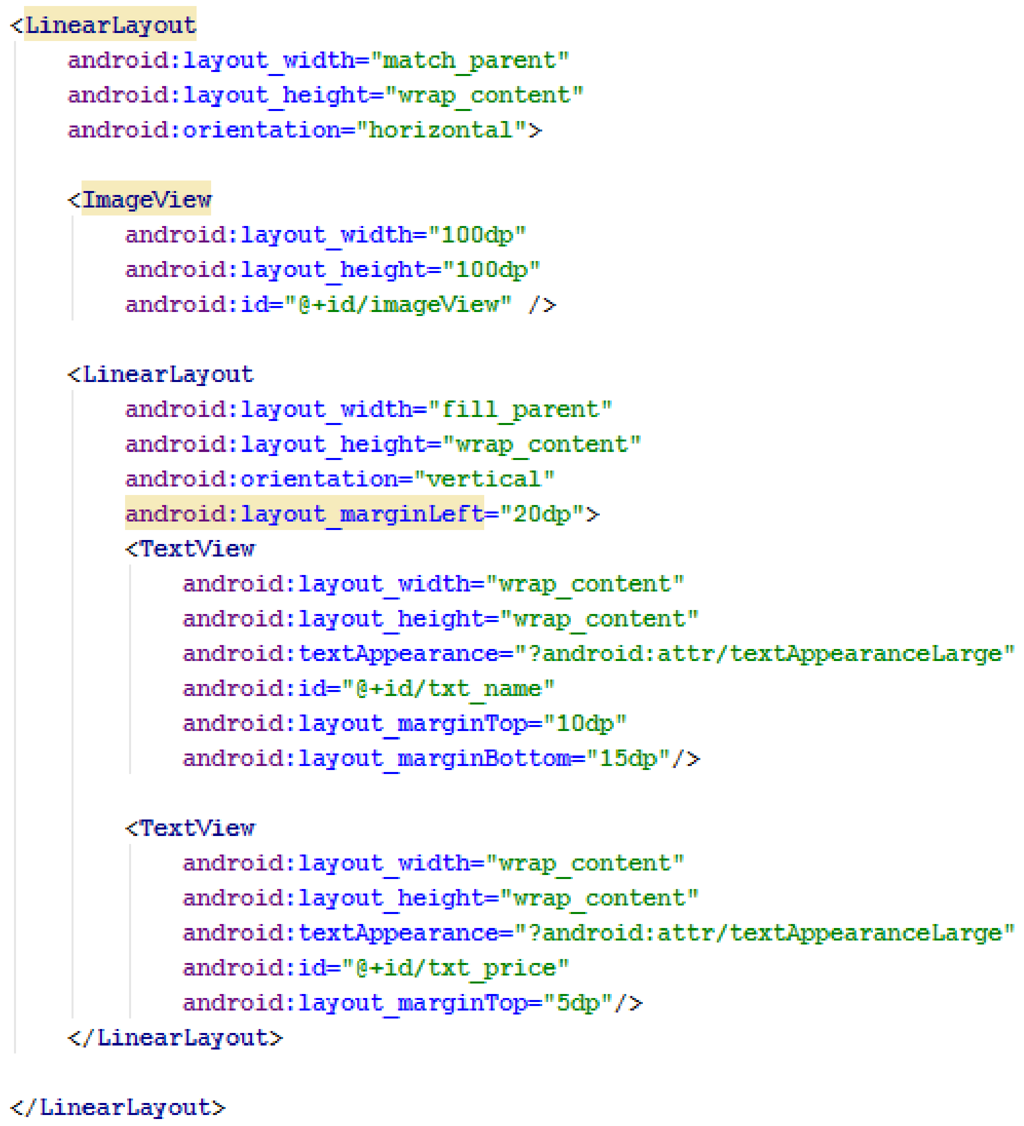


```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cart);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);

    SharedPreferences sp = getApplicationContext().getSharedPreferences("uwiPrefs",MODE_PRIVATE);
    int itemid = sp.getInt("itemCart", -1);
    if (itemid != -1){
        String [] itemList = getResources().getStringArray(R.array.items_available);
        String [] itemPrices = getResources().getStringArray(R.array.items_prices);
        TypedArray itemImages = getResources().obtainTypedArray(R.array.items_images);

        ((TextView)findViewById(R.id.txt_name)).setText(itemList[itemid]);
        ((TextView)findViewById(R.id.txt_price)).setText(itemPrices[itemid]);
        ImageView imgView = (ImageView)findViewById(R.id.img_icon);
        imgView.setImageResource(itemImages.getResourceId(itemid, 0));
    }else{
        Toast.makeText(this, "No Items in the Cart", Toast.LENGTH_LONG).show();
    }
}
```

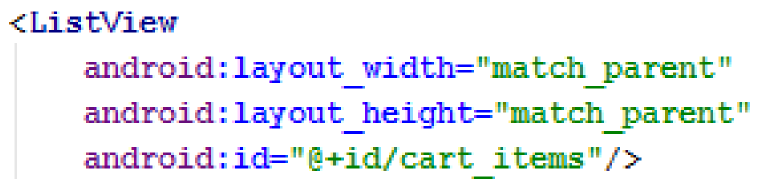
- To



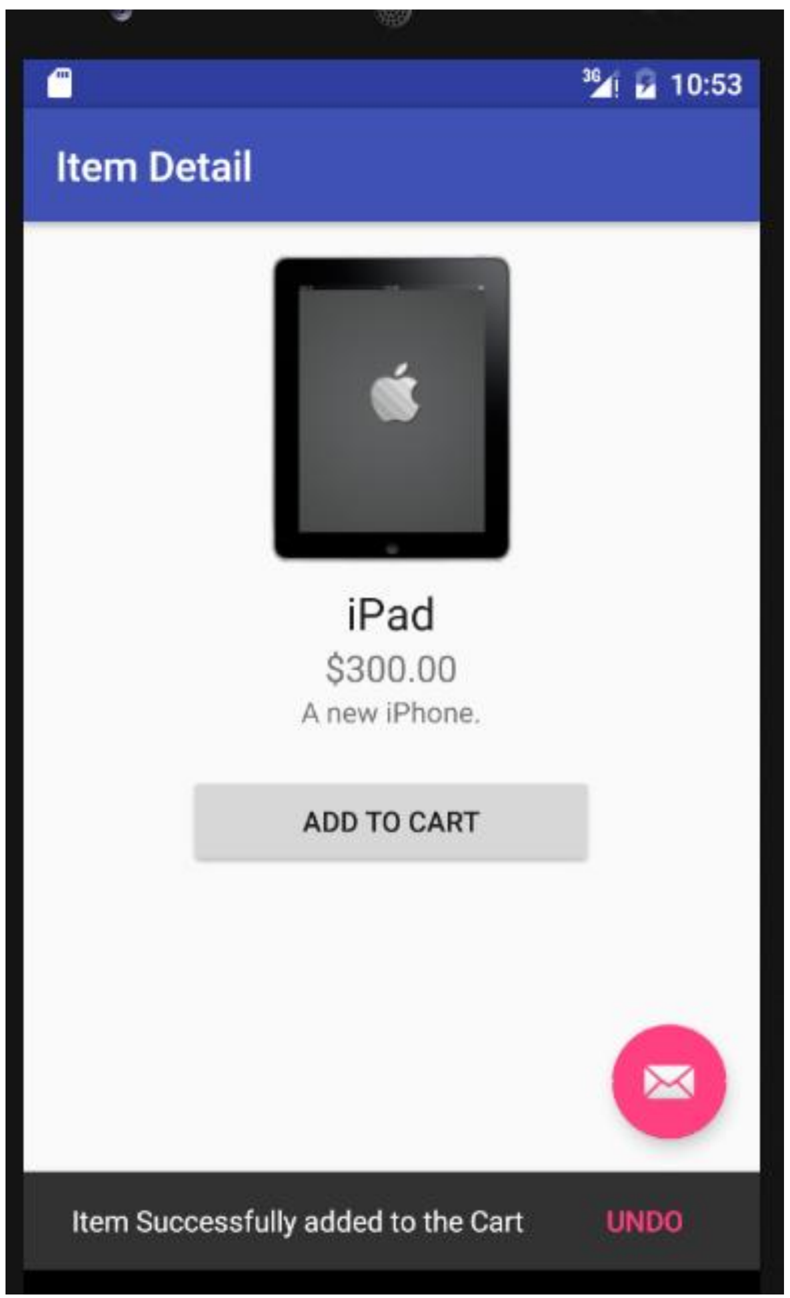
```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cart);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    // The fields specify what columns will be displayed (restriction)
    String [] fields = {CartContract.CartEntry.ITEM, CartContract.CartEntry.TIME};
    // Specify that the records will be ordered by the time created
    String sortedOrder =  CartContract.CartEntry.TIME + " DESC";
    //Retrieve the database
    SQLiteOpenHelper helper = new DBHelper(this); // this is the activity
    final SQLiteDatabase db = helper.getReadableDatabase(); // use readable to prevent uncessary locks
    // the database will execute the query based on options we specify and store in the Cursor
    Cursor res = db.query(CartContract.CartEntry.TABLE_NAME, fields, null,null,null,null,sortedOrder);
    ArrayList <String> itemList = new ArrayList();
    String [] items = getResources().getStringArray(R.array.items_available);
    // For Each of the items returned from the database query
    while (res.moveToNext()){
        // Retrieve the Index of the Item that is stored and find string for that item
        int itemId = res.getInt(res.getColumnIndex(CartContract.CartEntry.ITEM));
        // Add the string for the item in an arraylist
        itemList.add(items[itemId]);
    }
    //Populate the listview created for the cart using an adapter
    ListView lv = (ListView)findViewById(R.id.cart_items);
    ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1,itemList );
    lv.setAdapter(adapter);
}
```

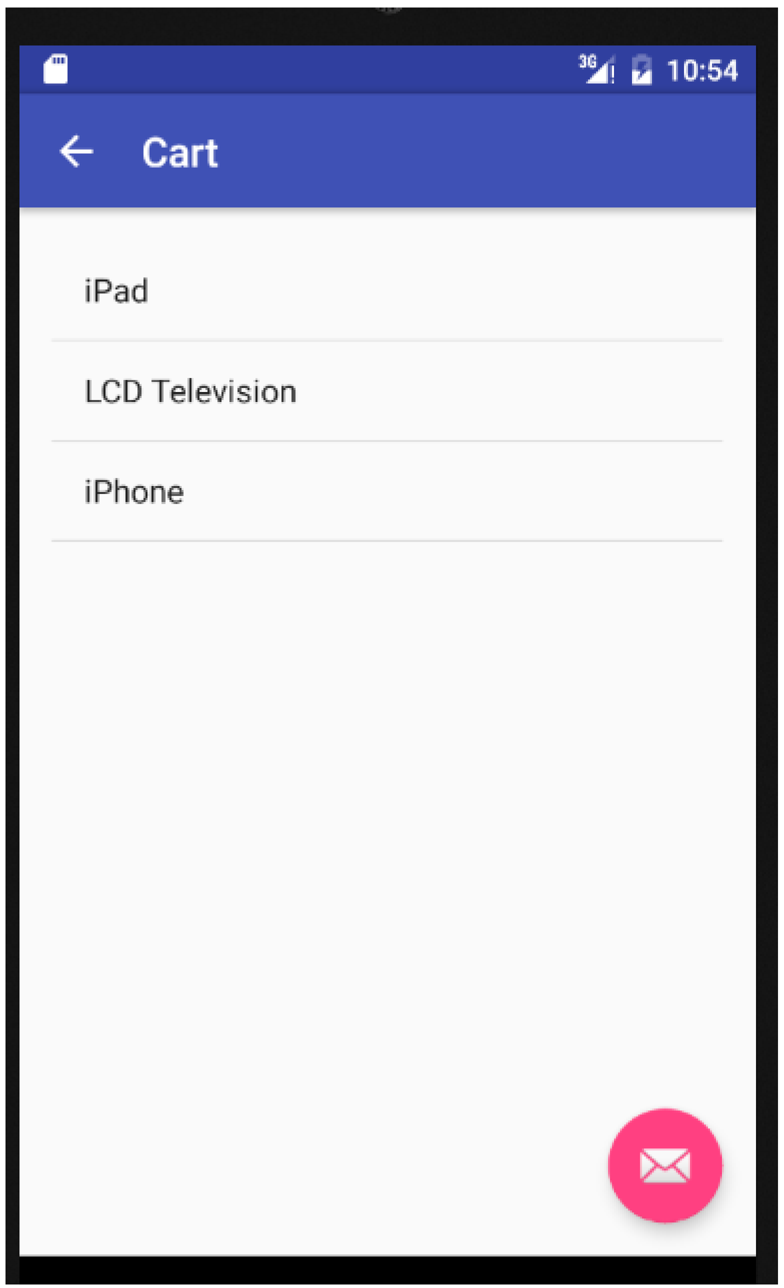


```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <ImageView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:id="@+id/imageView" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginLeft="20dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:id="@+id/txt_name"
            android:layout_marginTop="10dp"
            android:layout_marginBottom="15dp"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:id="@+id/txt_price"
            android:layout_marginTop="5dp"/>
    </LinearLayout>

</LinearLayout>
```



```
<ListView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/cart_items"/>
```


10:53
Item Detail
iPad
$300.00
A new iPhone.
ADD TO CART
Item Successfully added to the Cart
UNDO


10:54
Cart
iPad
LCD Television
iPhone

Extra

- Adjust the cart to display the image and the date when the item was added to the cart using a custom adapter.
- Also add a button to remove from cart. (use the code from the unto action of the action bar in the “ItemDetailActivity” as an example.)

HW

- Recommend that everyone attempt the following tasks following the lab:
  - In Peer groups complete the following:
    - Create a schema for the items (listing of products)
      - The Items should have (name, price, image (NB take note of the date type for the image))
    - Add the default items to the database table
    - Adjust the application so that the item id is drawn from table instead of the XML.
- Solution for the HW will b made available subsequently.