

# Assignment 4

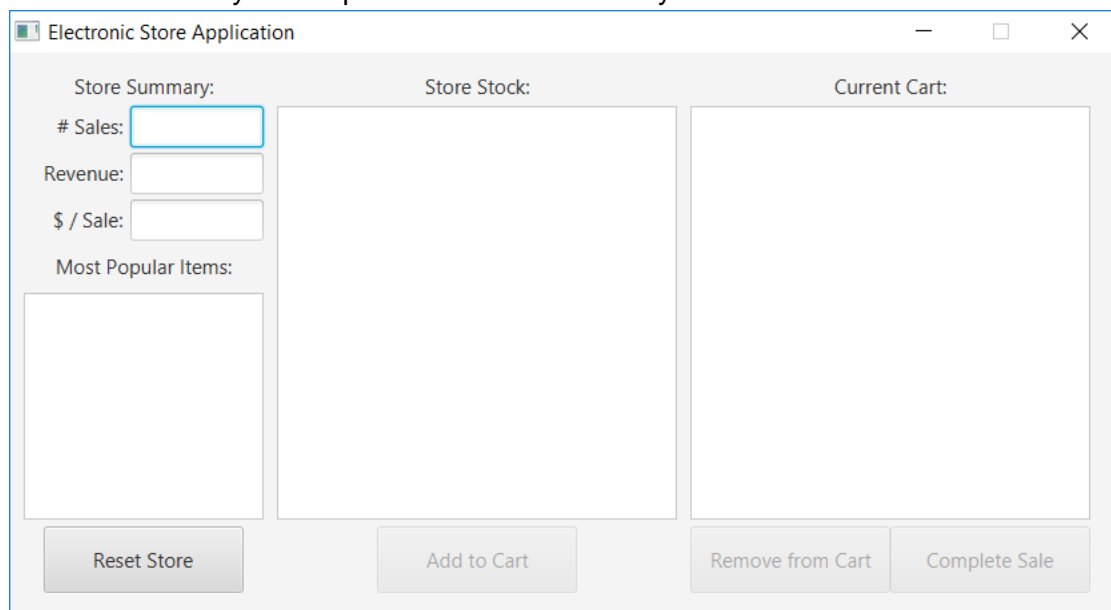
## Graphical user interfaces

Submit a single ZIP file called **assignment4.zip** containing your IntelliJ project. This assignment has 100 marks. See the marking scheme that is posted on the course webpage.

For this assignment, you will apply the MVC paradigm to build a graphical user interface to attach to the electronic store model that you have developed over the previous assignments. You can use your own model classes from the previous assignments, or you can download the model classes included within the zip file on the assignment page (a solution to A2) and incorporate those into your IntelliJ project as a starting point. You can make any changes to the existing code that are required to achieve the goals of this assignment (e.g., adding new attributes to existing classes). **You should ensure you understand the Model/View/Controller paradigm before beginning the assignment.** Your assignment must maintain a separation between the GUI and the underlying electronic store model. You must also continue to apply the principles of OOP, such as encapsulation. A recording showing a demonstration of how the GUI should work is included on the assignment page.

### 1) The GUI

An example of what the graphical user interface window should include is given in the picture below. The code for this view must be created in a class called `ElectronicStoreView`. You should make the GUI window non-resizable. The ratio of the window width/height should be approximately 2:1 (e.g., 800 wide, 400 high). It does not have to look exactly as the picture but should closely match.

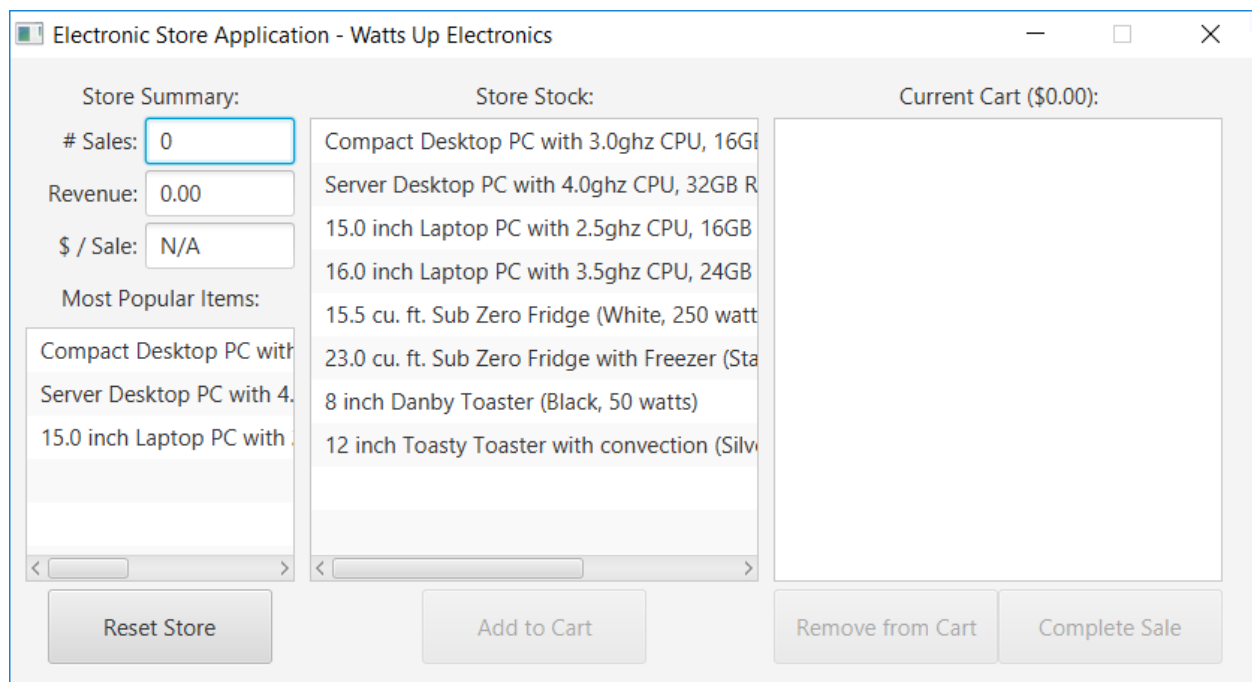


The screenshot shows a window titled "Electronic Store Application". The window is divided into three main sections: "Store Summary:", "Store Stock:", and "Current Cart:". The "Store Summary:" section contains four input fields: "# Sales:", "Revenue:", "\$ / Sale:", and "Most Popular Items:". The "Store Stock:" and "Current Cart:" sections are empty. At the bottom of the window, there are four buttons: "Reset Store", "Add to Cart", "Remove from Cart", and "Complete Sale".

## 2) When the Application Starts

Create a class called `ElectronicStoreApp`, which will represent the controller component of your program. This class should extend from the JavaFX `Application` class. When the application starts, your controller should create an instance of `ElectronicStore`, as well as an instance of your view defined in part 1. To create the `ElectronicStore` instance, you must use the static `createStore()` method from the `ElectronicStore.java` file included on the assignment page and save the returned `ElectronicStore` instance in a variable within the `ElectronicStoreApp`. If you are using your own code as a base, you can copy/paste the `createStore()` method from the example. The `ElectronicStore` instance will represent the model that the GUI application will interact with.

When the application starts, the window title must include the name of the store. The 'Add to Cart', 'Remove from Cart', and 'Complete Sale' buttons should initially be disabled. The '# Sales', 'Revenue', and '\$ / Sale' TextFields should be initialized to default values representing the starting state of an electronic store. The store stock ListView should display all the items currently in stock within the electronic store model. The most popular items ListView should show 3 products that exist in the store's stock. At this point, all products will have an equal popularity since no sales have taken place. The image below shows an example of what the initial window should look like:



### 3) Event Handling

You must add event handling to the ElectronicStoreApp class to meet the requirements outlined below.

#### Store Stock:

- 1) If an item has been selected inside of the Store Stock ListView, the 'Add to Cart' button should be enabled. If no item has been selected, the button should be disabled.
- 2) When an item in the Store Stock ListView has been selected and the 'Add to Cart' button is clicked, one unit of the selected item should be added to the current cart (see requirements for Current Cart below).
- 3) If a product does not have any items left in stock (i.e., they have all been sold or added to the current cart), it should not be displayed in the Store Stock ListView. Note that when a product is added to the cart, the amount of that product available in stock should decrease.

#### Current Cart:

- 1) The Current Cart ListView should show the quantity and product information for each product that has been added to the cart (e.g., 3 x Server Desktop PC with 4.0ghz CPU, 32GB RAM, 500GB SSD drive). Each added product should only show up once in this list. The number at the start of the entry will indicate how many times the product has been added to the list.
- 2) If no products have been added into the cart, the 'Complete Sale' button should be disabled. If products are in the cart, the button should be enabled.
- 3) The total dollar amount in brackets beside 'Current Cart', representing the total value of all products in the current cart, should be updated each time a product is added to the cart or removed from the cart.
- 4) If an item in the Current Cart ListView has been selected, the 'Remove from Cart' button should be enabled. If no item in the Current Cart ListView is selected, the 'Remove from Cart' button should be disabled.
- 5) If an item in the Current Cart ListView is selected and the 'Remove from Cart' button is clicked, the quantity of that product in the cart should be reduced by 1 and the quantity in stock should be increased by 1.

#### Complete Sale:

- 1) When there is at least one product in the current cart and the 'Complete Sale' button is clicked, those items should be sold. You must update the '# Sales' (increase by 1), the 'Revenue' (increase by the total value of the cart), and the '\$ / Sale' fields on the GUI window and the related variables in the underlying model.
- 2) The total dollar amount listed in brackets beside 'Current Cart' should be reset to \$0.00 when a sale is completed.

**Most Popular Items:**

- 1) The most popular items ListView should show the 3 products that have sold the most units. If two or more items have sold an equal number of units, those items can be listed in any order.
- 2) The most popular items list view should only account for products that have officially been sold (i.e., it should not account for items that are added to the cart).

**Reset Store:**

- 1) When the 'Reset Store' button is clicked, the model should be reset to its initial state (i.e., the electronic store model should be recreated). The GUI should also update to show the reset state of the store.