

# Shopaholic Implementation Manual

## Item Class

Item
#int ID
#double price
#String image
#String description
#String name
#String size
#int quantity
#int percent
+boolean onSale
+boolean inCart
+addImage(String imagePath)
+getImage() : String
+discount()
+getPercent() : int
+getPrice() : double
+getName() : String
+getSize() : String
+setSize(String newSize)
+getDescription() : String
+getID() : int
+getQuantity() : int
+setQuantity(int newQuantity)

- Abstract class for all the items available in the Shopaholic app
- Holds information for the item including:
  - Item ID
  - Item Price
  - Path for image
  - Item description
  - Item name
  - Item size
  - Item quantity
  - Percent discounted
  - Whether item is on sale
  - Whether item is in a cart
- The item class also includes methods available for use for all item objects

- addImage(String imagePath): adds the inputted string as the image path for the item object
- getImage(): returns the image path of the item as a string
- discount(): this method generates a random discount on the item between 5% and 50% as well as indicates that the item is now on sale
- getPercent(): this returns the percent discount on the item as an integer
- getPrice(): this returns the price of one unit of the item as a double value
- getName(): returns the name of the item
- getSize(): returns the size of the item (S, M, L, XL)
- setSize(String newSize): sets the size of the item to the inputted string value
- getDescription(): gets the string description of the item
- getID(): gets the integer ID of the item
- getQuantity(): gets the integer quantity of the item
- setQuantity(int newQuantity): sets the quantity of the item to the inputted integer value

## Jewelry Class

Jewelry
#String metalType
+getMetalType() : String
+setMetalType(String metal)

- The Jewelry class is an abstract subclass of the Item class for Jewelry type items
- This class adds a field for the type of metal the Jewelry is made of
- The additional methods for this class include:
  - getMetalType(): returns the string value for the metal type assigned to the jewelry item
  - setMetalType(String metal): sets the type of metal of the jewelry to the passed parameter value

## Ring, Earrings, and Bracelet Classes

Ring
<u>-int number</u>
+Ring(String name, double price, String description)
-setID() : int

Bracelet
<u>-int number</u>
+Bracelet(String name, double price, String description) -setID() : int

Earrings
<u>-int number</u>
+Earrings(String name, double price, String description) -setID() : int

- These classes are subclasses of the Jewelry class
- Each of these classes have a constructor which sets the name, price, and description of that item type, as well as generates an ID for that instance
- These classes also have a field indicating the number, which is counting the amount of instances of each class made
- The method setID() will return a 3 digit ID for each item instance using the number as the lowest value
  - Ring items have IDs in the 100s
  - Bracelet items have IDs in the 200s
  - Earrings items have IDs in the 300s
- Example ID value: the first instance of a ring will have the ID 100, the second instance of a ring will have the ID 101, and so on.

## Shoes Class

Shoes
#String color
+getColor() : String +setColor(String newColor)

- The abstract class Shoes is a subclass of the Item class for Shoe type items
- It adds an additional field for the color of the shoe
- It also includes the additional methods:
  - getColor(): returns the color string of the shoe item
  - setColor(String newColor): sets the color value to the value of the new color parameter

## TennisShoes, DressShoes, and Slides Classes

<b>TennisShoes</b>
<u>-int number</u>
+TennisShoes(String name, double price, String description) -setID() : int

<b>DressShoes</b>
<u>-int number</u>
+DressShoes(String name, double price, String description) -setID() : int

<b>Slides</b>
<u>-int number</u>
+Slides(String name, double price, String description) -setID() : int

- Each of these classes are subclasses of the Shoes class
- The constructor for each of these sets the name, price, and description as well as creates an ID for the instance
- The ID is created the same way as in the Ring, Bracelet, and Earrings classes, and the number field serves the same function as described previously
  - TennisShoes item IDs are in the 400s
  - DressShoes item IDs are in the 500s
  - Slides item IDs are in the 600s

## ClothingItem Class

<b>ClothingItem</b>
#String color
+getColor() : String +setColor(String newColor)

- The ClothingItem class is an abstract subclass of the Item class

- It includes an additional field storing the color of the Clothing item
- The additional methods included are:
  - getColor(): gets the string value of the color assigned to the Clothing item
  - setColor(String newColor): Sets the color field to the value passed as the new color parameter

### TShirt, Sweatshirt, and Sweatpants Classes

TShirt
<u>-int number</u>
+TShirt(String name, double price, String description) -setID() : int

Sweatshirt
<u>-int number</u>
+Sweatshirt(String name, double price, String description) -setID() : int

Sweatpants
<u>-int number</u>
+Sweatpants(String name, double price, String description) -setID() : int

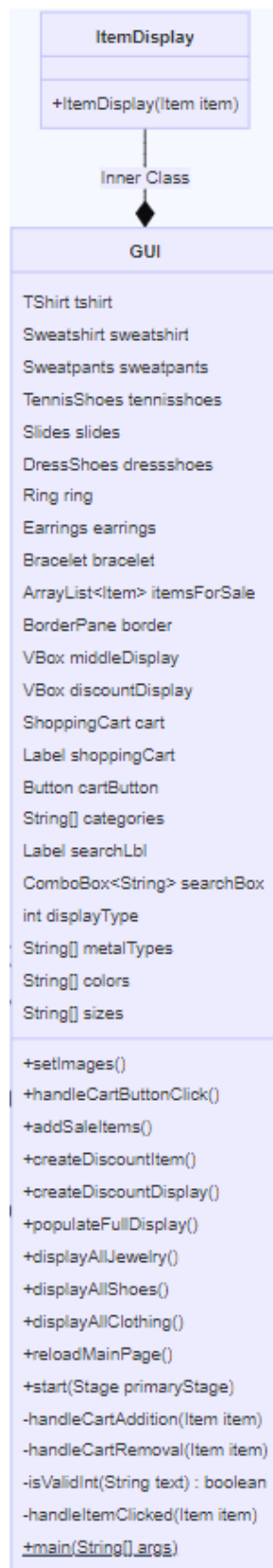
- The TShirt, Sweatshirt, and Sweatpants classes are subclasses of the ClothingItem class
- The constructors assign the name, price, and description to the instance and sets the instance ID
- The ID is set in the same way as the Ring, Bracelet, and Earrings class, and the number field serves the same purpose as described there
  - The IDs of TShirts are in the 700s
  - The IDs of Sweatshirts are in the 800s
  - The IDs of Sweatpants are in the 900s

## ShoppingCart Class

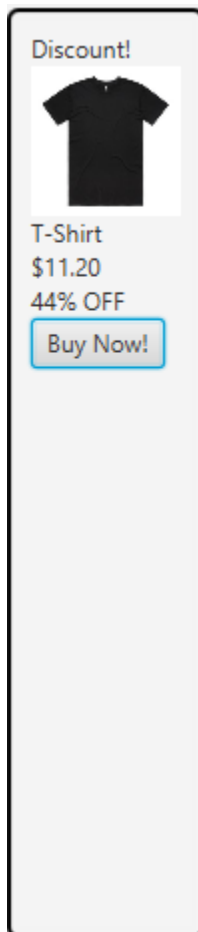
ShoppingCart
-List<Item> items
+ShoppingCart() +addItem(Item item) +getNumItems() : int +getCart() : List<Item> +getTotalCost() : double +removeItem(Item item) +clearAll()

- The ShoppingCart class contains all methods and fields related to a shopping cart
- The ShoppingCart contains a List of Items which hold each item within the shopping cart
- The ShoppingCart constructor creates an empty ArrayList assigned to the items field
- Shopping cart related methods include:
  - addItem(Item item): adds the item passed to the method into the items list
  - getNumItems(): returns the integer value corresponding to the number of items in the list items
  - getCart(): returns the list items of the shopping cart instance
  - getTotalCost(): returns the double value corresponding to the cost of all the items in the cart by adding up the unit price multiplied by the quantity associated with each item in the items list
  - removeItem(Item item): removes the item passed to the method from the items list
  - clearAll(): removes all items from the items list until it is empty

## GUI Class




- The class GUI contains all necessary functions and objects that allow for the user interface to be created and extends the JavaFX class Application
- The GUI class creates instances of each type of item to be used in the app as well as an arraylist itemsForSale, which is populated with each item instance in the method addSaleItems()
- An instance of the ShoppingCart class called cart is also made to hold items in the cart
- The setImages() method adds corresponding image paths to each of the item instances
- The createDiscountItem() method chooses a random item ID and discounts the item associated with it
- The createDiscountDisplay() method lays out a visual display of the discounted item in the VBox discountDisplay, an example of which can be seen here:



- The populateFullDisplay() method creates a spread of 5 random item displays using the inner class ItemDisplay. This full display is saved in the VBox middleDisplay. The display never includes the discount item. An example of a full display can be seen here:






Sweatpants

\$30.00

Casual pants

Add to Cart




Clasp Bracelet

\$69.99

Gold bracelet with an easy use clasp

Add to Cart




Diamond Ring

\$3500.00

1 carat diamond ring

Add to Cart




Dress Shoes

\$110.00

Real leather dress shoes

Add to Cart




Slides

\$33.99

Casual slides

Add to Cart

- The `displayAllJewelry()` method displays `ItemDisplays` of all Jewelry items in the main display. If a Jewelry type is discounted, it is displayed using the `VBox discountDisplay`, while the rest of the Jewelry is displayed in the `VBox middleDisplay`. An example of a Jewelry display can be seen here:




Clasp Bracelet

\$69.99

Gold bracelet with an easy use clasp

Add to Cart




Diamond Ring

\$3500.00

1 carat diamond ring

Add to Cart



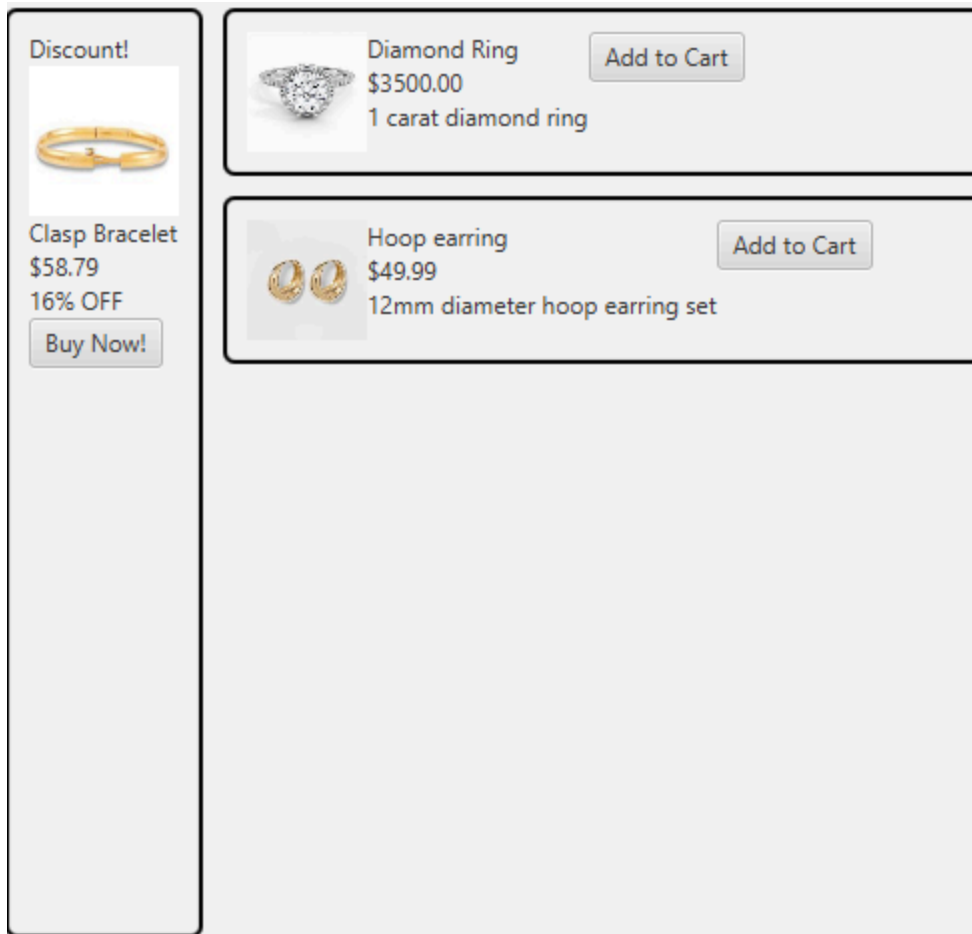
Hoop earring

\$49.99

12mm diameter hoop earring set

Add to Cart

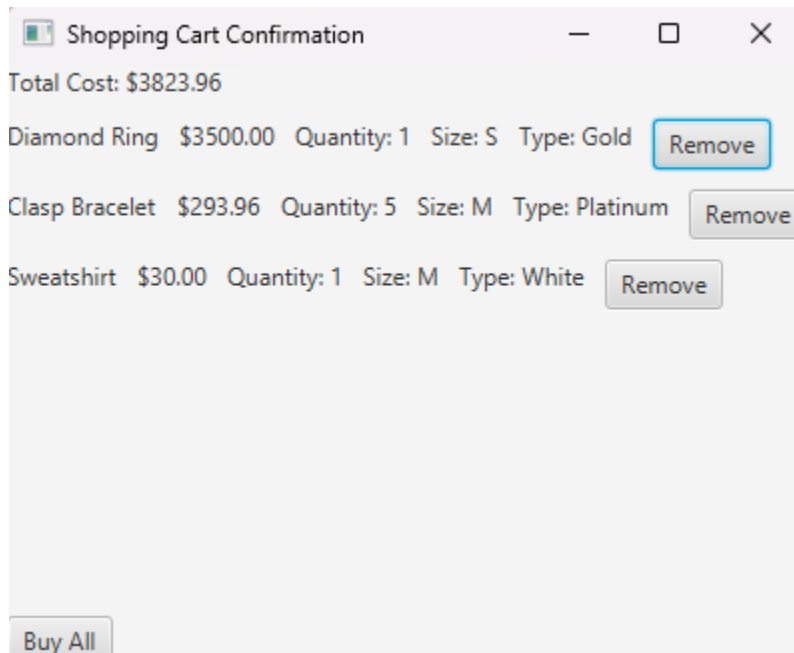
Example of Jewelry display with discounted item:



- The `displayAllShoes()` and `displayAllClothing()` methods have the same functionality as the `displayAllJewelry()` method, except with shoes and clothing items, respectively.
- The `reloadMainPage()` method will recreate the `middleDisplay` and `discountDisplay` `VBoxes` according to which `displayType` is currently being used.
  - Type 0: full display and discount display
  - Type 1: Jewelry display
  - Type 2: Shoes display
  - Type 3: Clothing display
- The `start(Stage primaryStage)` method sets up the initial state of the program
  - Sets up the main display of Shopaholic using these JavaFX objects:
    - `BorderPane` border
    - `VBox` `middleDisplay`
    - `VBox` `discountDisplay`
    - `Label` `shoppingCart`
    - `Button` `cartButton`
    - `Label` `searchLbl`
    - `ComboBox` `searchBox`: this object uses the `String[]` `categories` array which displays the categories that can be searched with
  - Sets the on action events for the `searchBox` and the `cartButton`

- `searchBox`: when a different category is chosen, the display changes to only display items of that chosen category. It saves the type of display in the variable `displayType`
  - `cartButton`: sets the cart button to act on the `handleCartButtonClick()` method when it is clicked
- The method `handleCartButtonClick()` creates the cart window which displays all items in the cart. This window includes the total cost of the cart and a display of each item's name, price, quantity, size, and color/metal type. Each item can be removed from the cart by clicking the remove button next to the item, at which point the main display will be reloaded. A Buy All button will clear all items from the shopping cart and recreate the main display with the full display spread.

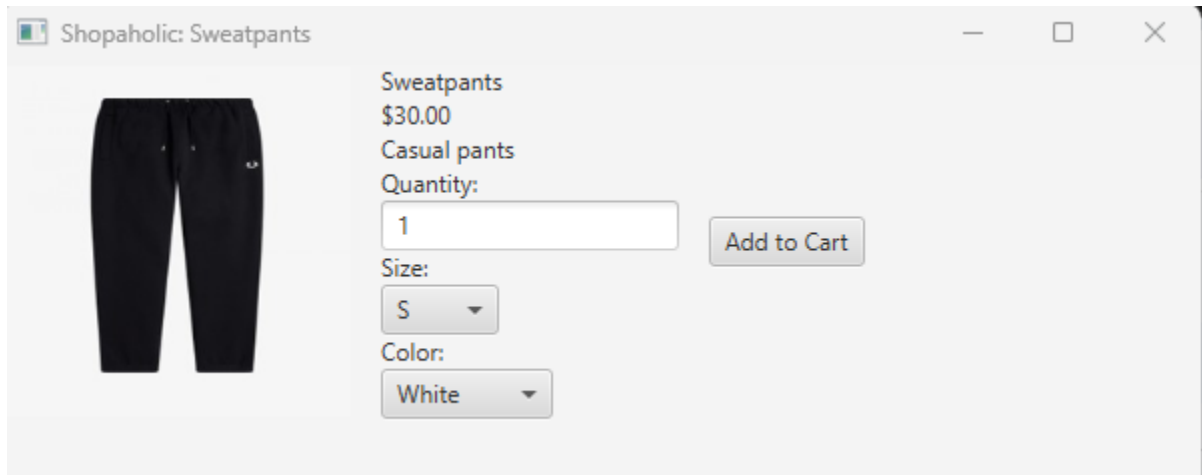
Shopping cart display example:



- The `handleCartAddition(Item item)` method is used when an Add to Cart button is clicked. This method adds the item passed through the method to the cart and updates the number of items displayed on the `cartButton`
- The `handleCartRemoval(Item item)` is used when a Remove button is clicked. This method removes the item that was passed through the method from the cart and updates the number of items displayed on the `cartButton`
- The method `isValidInt(String text)` returns a boolean value corresponding to whether the text input to the method is an integer or not
- The `handleItemClick(Item item)` method is used whenever an `ItemDisplay` or `discountDisplay` is clicked. It pulls up a separate window with information on the item which was passed to the method.
  - The item's image is shown, as well as the name, price, and description
  - In this window, `ComboBoxes` with the available sizes(`String[] sizes`) and the available colors(`String[] colors`)/metal types(`String[] metalTypes`) are available to choose these attributes of the item

- The quantity can be chosen by typing an integer into the TextField (Only integers are allowed to be typed in through use of the isValidInt() method)
- The entered attributes are saved upon clicking the Add to Cart button, which adds the item into the cart with these attributes
- If the item is already in the cart, a Remove button will appear instead. The item will be removed from the cart when the button is clicked

An example of the item window can be seen here:



- The inner class ItemDisplay is an extension of the HBox class. It creates a display of an item.
  - The constructor of the item display takes the item passed to it and displays its image, its name, price, and description, and a button.
    - If the item is not in the cart, the button is the Add to Cart button
    - If the item is in the cart, the button is a Remove button

An example of what is made when an ItemDisplay instance is created can be seen here:

