Shopping Cart Graphical User Interface

Final Project - Part Three

Group Z

Monday, February 26, 2024

Table of Contents

Table 1. List of Preliminary Source Code Files

File Name	Type	Description
ShoppingCartMain.java	Java File	*Starts the program. Instantiates the Catalog with Items and passes them to the GUI to be displayed.*
ShoppingCartGUI.java	Java File	*Defines the content, layout, and behaviors of the GUI.*
ShoppingCart.java	Java File	*Handles the storage of shopping cart items, and calculation of total, including bulk pricing.*
Item.java	Java File	*Represents an item that a user may purchase. Items contain a 'name' and 'price' and may optionally contain a 'bulk_quantity' and 'bulk_price'.*
ItemOrder.java	Java File	*Represents a line item from an order (placed by the user). Contains an 'item' and 'quantity'.*
Catalog.java	Java File	*Represents a catalog that stores items. Contains a 'name' and 'size'.*

Note: See Appendix for Pseudocode details

Table 2. ShoppingCartMain (Main Method)

Methods				
Name	**Return Type**	**Parameters**	**Visibility**	**Description**
main	void	String[] args	Public	Begins execution of the program. Instantiates ‘Catalog’ and ‘ShoppingCartGUI’.

Table 3. ShoppingCartGUI

Attributes		
Name	**Type**	**Description**
spinners	HashMap<Item, JSpinner>	Maps catalog items to their corresponding quantity selectors (spinners).
frame	JFrame	The main window.
priceText	JLabel	Displays the total price of the order.
checkBox	JCheckbox	Checkbox for discount eligibility.
submitButton	JButton	Button to calculate the total price based on selected items and quantities.
clearButton	JButton	Button to clear all spinner values, resetting item quantities to zero.

Constructors			
Name	**Parameters**	**Visibility**	**Description**
ShoppingCartGUI	Catalog catalog	Public	Initializes the GUI components.

Methods				
Name	**Return Type**	**Parameters**	**Visibility**	**Description**
main	void	String[] args	public	Begins execution of the program. Instantiates ‘Catalog’ and ‘ShoppingCartGUI’.

Table 4. ShoppingCart

Attributes		
Name	**Type**	**Description**
size	integer	The maximum number of items allowed in the shopping cart (used primarily in the for loop).
discount	boolean	A boolean value to store whether or not the order will receive a discount.
cartList	arrayList<ItemOrder>	Serves as a container for the items added to the shopping cart.

Constructors			
Name	**Parameters**	**Visibility**	**Description**
ShoppingCart	int size	Public	Creates the shopping cart using the specified size limit.

Methods				
Name	**Return Type**	**Parameters**	**Visibility**	**Description**
add	void	ItemOrder order	Public	Adds an ItemOrder (line item) to the cartList.
setDiscount	void	boolean value	Public	Determines whether or not a discount is applied to the order.
total	double	None	Public	Calculates and returns the total price of all item orders in the cartList with the applicable discounts.

Table 5. Item

Attributes		
Name	**Type**	**Description**
name	String	The name of the item.
price	double	The regular price of the item.
bulk_quantity	integer	The minimum number of items that must be purchased in order to qualify for bulk pricing.
bulk_price	double	The price of the item when purchased in bulk.
format	DecimalFormat	A DecimalFormat object used to format the display of prices.

Constructors			
Name	**Parameters**	**Visibility**	**Description**
Item	String name, double price	Public	Initializes an item without bulk pricing.
Item	String name, double price, int bulk_quantity, double bulk_price	Public	Initializes an item without bulk pricing.

Methods				
Name	**Return Type**	**Parameters**	**Visibility**	**Description**
priceFor	double	int quantity	Public	Calculates the price for a specified quantity of the item, considering bulk pricing.
toString	String	None	Public	Returns a string representation of the item, including its name, price, and bulk pricing details if applicable.

Table 6. ItemOrder

Attributes		
Name	**Type**	**Description**
item	Item	Holds a reference to an 'Item' object (the item being ordered).
quantity	integer	The number of items being ordered.

Constructors			
Name	**Parameters**	**Visibility**	**Description**
ItemOrder	Item item, int quantity	Public	Instantiates an 'ItemOrder' to represent a 'line item' in the shopping cart.

Table 7. Catalog

Attributes		
Name	**Type**	**Description**
name	String	The name of the catalog (ie. “Food Catalog”).
size	integer	The maximum number of items the catalog can hold.
catalogList	ArrayList<Item>	A list to store the items in the catalog.

Constructors			
Name	**Parameters**	**Visibility**	**Description**
Catalog	String name, int size	Public	Creates a new instance of Catalog, with a name and max size.

Methods				
Name	**Return Type**	**Parameters**	**Visibility**	**Description**
add	boolean	Item item	Public	Adds an item to the catalogList (if there is room).
size	integer	None	Public	Returns the number of items in the catalog.
get	Item	int index	Public	Returns the item at the specified index in the catalogList ArrayList.
getName	String	None	Public	Returns the name of the catalog.

Appendix - Pseudocode

ShoppingCartMain

```
// Begin execution of program

// Define each 'Item' with a name, price, bulk_quantity (optional), and
bulk_price (optional)
Define Item pizzaSlice with name "Pizza Slice", price $1.99
Define Item hotDog with name "Hot Dog", price $1.50
Define Item pasta with name "Pasta", price $9.99
Define Item bread with name "Bread", price $0.99, bulk quantity 10, bulk
price $0.79
Define Item chocolate with name "Chocolate", price $2.99, bulk quantity 5,
bulk price $2.49

// Add each 'Item' to the 'foodCatalog' object
Add pizzaSlice to foodCatalog
Add hotDog to foodCatalog
Add pasta to foodCatalog
Add bread to foodCatalog
Add chocolate to foodCatalog

// Instantiate 'ShoppingCartGUI', passing 'foodCatalog' to it for
initialization
Instantiate ShoppingCartGUI with foodCatalog
```

ShoppingCartGUI

```
// Define a HashMap to link catalog items to their respective JSpinner
controls
Define spinners as HashMap<Item, JSpinner>

// Initialize JFrame with specified behaviors
Create JFrame named frame
Set frame's default close operation to EXIT_ON_CLOSE
Set frame's resizable property to false
Set frame size to 350 width and 400 height
Center the frame on the screen

// Add a JPanel at the top for displaying total order price
Create JPanel named topPanel

// Initialize with $0.00 or appropriate starting value
Create JLabel named totalPriceLabel with text "Total Order Price: $0.00"
Add totalPriceLabel to topPanel
Add topPanel to the top of frame

// Panel to hold all item JPanels
Create JPanel named mainPanel with a vertical BoxLayout

// Loop through each item in the catalog
For each item in catalog
    Create JPanel named itemPanel
    Define SpinnerModel named quantityModel with range from 0 to 100
    Create JSpinner named quantitySpinner with quantityModel
    Create JLabel named itemLabel with text showing item's name and price
    If item has bulk pricing
        Add a tooltip to itemLabel with bulk pricing details
    Add item and quantitySpinner to itemSpinnerMap
    Add itemLabel and quantitySpinner to itemPanel
    Add itemPanel to mainPanel

// Make the mainPanel scrollable
Create JScrollPane named scrollPane with mainPanel as the viewport view
Add scrollPane to the center of frame

// Bottom panel for checkout button
Create JPanel named bottomPanel
Create JButton named checkoutButton with text "Checkout"
Add an action listener to checkoutButton to handle the checkout process
```

```
Add checkoutButton to bottomPanel
Add bottomPanel to the bottom of frame

// Make frame visible
Set frame visibility to true
```

ShoppingCart

```
// Initialize an ArrayList to hold ItemOrder objects
ArrayList<ItemOrder> itemOrder

// Boolean flag for discount application
boolean hasDiscount

// Constructor for ShoppingCart with a specified size
Create ShoppingCart with size parameter
    Allocate space for itemOrders array with specified size
    Set hasDiscount to false

// Method to add an ItemOrder to the cart
Add ItemOrder to cart
    Loop through itemOrders array to find an empty slot
        If empty slot found, insert ItemOrder into this slot
        Else, notify that the cart is full or handle overflow differently

// Method to enable or disable a discount
Toggle discount status
    Set hasDiscount to the passed boolean value

// Method to calculate the total price of the cart
Calculate total price
    Initialize total price to 0
    For each itemOrder in itemOrders
        If itemOrder is not null, add its price to total
    If hasDiscount is true, apply discount to total price
    Return the calculated total price
```

Item

```
// Declare private member variables
Declare String name            // Name of the item
Declare Double price           // Price of the item
Declare Integer bulkQuantity   // Bulk quantity threshold for discount
(optional)
Declare Double bulkPrice       // Discounted price for bulk purchases
(optional)

// Constructor with name and price
Constructor Item(String aName, Double aPrice)
    name = aName
    price = aPrice

// Overloaded constructor with bulk pricing details
Constructor Item(String aName, Double aPrice, Integer aBulkQuantity, Double
aBulkPrice)
    name = aName
    price = aPrice
    bulkQuantity = aBulkQuantity
    bulkPrice = aBulkPrice

// Method to calculate price for a given quantity, considering bulk discounts
Method priceFor(Integer quantity)
    If bulkQuantity > 0 and quantity is divisible by bulkQuantity
        Return (quantity / bulkQuantity) * bulkPrice
    Else if bulkQuantity > 0
        Integer remainder = quantity % bulkQuantity
        Return ((quantity - remainder) / bulkQuantity) * bulkPrice +
(remainder * price)
    Else
        Return quantity * price

// Method to generate a string representation of the item
Method toString()
    If bulkPrice > 0 and bulkQuantity > 0
        Return name + ": $" + price + " (Bulk price: $" + bulkPrice + " for "
+ bulkQuantity + " or more)"
    Else
        Return name + ": $" + price
```

ItemOrder

```
// Declare the ItemOrder structure
Declare Class ItemOrder
    Declare Item item           // The item being ordered
    Declare Integer quantity    // The quantity of the item ordered

    // Constructor for ItemOrder
    Constructor ItemOrder(Item item, Integer quantity)
        this.item = item
        this.quantity = quantity
```

Catalog

```
// Declare the Catalog class
Declare Class Catalog
    Declare String name            // Name of the catalog
    Declare Integer maxSize        // Maximum size of the catalog
    Declare Item[] items           // Array to store items
    Declare Integer itemCount      // Current count of items in the catalog

    // Constructor for Catalog
    Constructor Catalog(String name, Integer size)
        this.name = name
        this.maxSize = size
        this.items = new Item[size]
        this.itemCount = 0

    // Add method to add an item to the catalog
    Method Add(Item item)
        If itemCount < maxSize
            items[itemCount] = item
            itemCount++
            Return true
        Else
            Return false

    // Size method to return the number of items in the catalog
    Method Size()
        Return itemCount

    // Get method to return an item by index
    Method Get(Integer index)
        If index >= 0 and index < itemCount
            Return items[index]
        Else
            Return null // Or throw an exception based on preference

    // GetName method to return the name of the catalog
    Method GetName()
        Return name
```