

Introduction to Object Oriented Programming

-

Project - Sports Shops

-

Nathan Dunne - K00211819

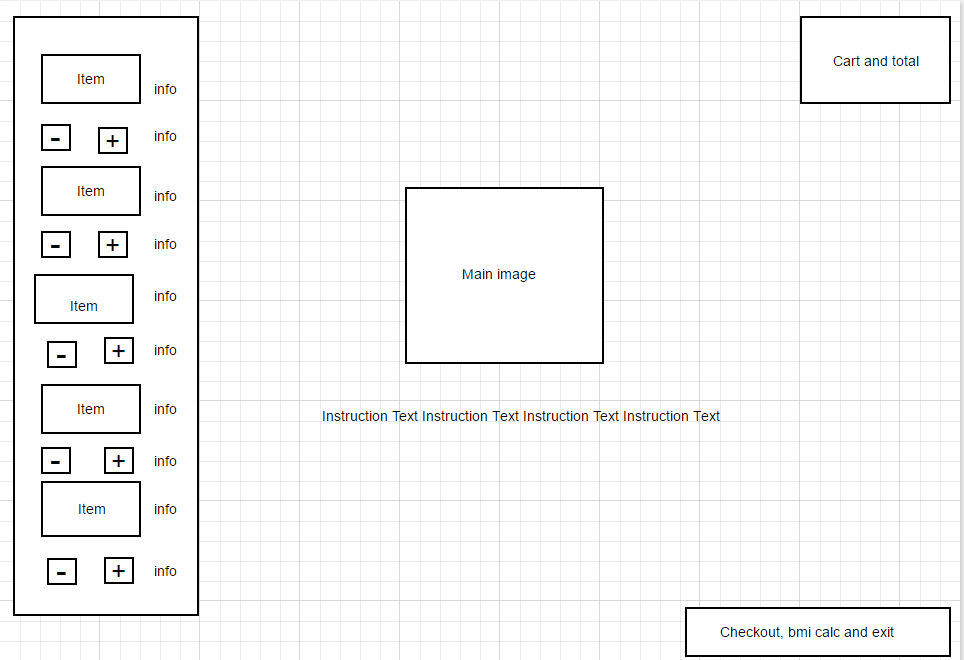
Lecturer – Natasha Kiely

**authorship statement**

**(Declaration of compliance and ownership)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Declaration** | | | | **tick 🗸** |
| I wish to confirm that I have read and understand all policies and procedures around project and assignment submission, plagiarism and malpractice as described in the **institute policy on plagiarism** | | | |  |
| I wish to confirm that I have retained a copy of this project/assignment for my own personal use and in the event of being requested for an additional copy by my teacher. | | | |  |
| I hereby declare that all work described in this project/assignment, except where referenced, is entirely my own. All sources of information have been acknowledged by means of reference as required. | | | |  |
| **signed:** |  | **date:** |  | |

Prototype Demonstrating the GUI



Description of how I built the GUI

I built the GUI using many of the components available in the swing library and I mainly utilized the GridBagLayout Manager for positioning and weighting.

From scratch I had a main frame “ShopFrame” that I built my components into. These include 4 main panels throughout the Frame.

StockPanel:

Situated on the left is the stock panel. This is actually anchored to the northwest using GridBagConstraints anchor variable. The anchor variable allowed me to easily position panels in compass directions of my choosing.

Within the StockPanel I have labels that represent the values of each item’s respective instance variables. I also used images to try allow the user to better visualise what they were buying. To increment and decrement the amount of particular items stock and cart array lists I used simple buttons with images on them.

CartPanel:

Situated on the top right is the cart panel. This panel contains the list of items that are present in the cart and the total amount of the sum of each item’s price that has been added to the cart.

I attempted to make the list’s appearance on the panel dynamic by incrementing a counter that would represent the GridBagContraints y value based on when they are first introduced to the cart. Apart from some minor issues I thought this worked well.

Instructions Panel:

In this panel I wanted to have something that would “fill” the middle space that had be created. I decided on using an image that represents the logo of the shop along with a brief introduction. I also decided to add labels that showed instructions to this panel.

Options Panel:

This panel contained buttons that once clicked either bring up new windows or allow you to exit the program.

For both the checkout and BMI calculator buttons I decided that I would create a new JFrame in the mouseclick event. I would then have separate panels that allow me to customise what was inside of these frames.

For the checkout I decided on keeping it simple by having some labels detailing instructions and showing a pre-generated int that would represent the voucher number. I also included an input field for this voucher number and a button to initiate the discount calculation. Once initiated the previous buttons would be removed and a final checkout button would appear. Upon clicking it a message dialog box would pop up detailing the final total, it also thanks the user.

For the BMI calculator I had pretty much the same except this time there were 2 input boxes that the user had to fill in. Upon hitting enter the system would calculate the BMI based on the input and overwrite a JLabel that would show what that calculation output was. I used some error handling to make sure that the input would be numbers that I could use in the calculation.

GridBagLayout:

This layout manager was recommended to me while looking at tutorials and guides. While powerful it did become very tedious to use as I had to constantly declare new values for the x and y positions of buttons, labels and so forth.

I also had to constantly overwrite inset values to control the padding of each item.

Overall it was the most time consuming aspect of the project and I still don’t feel like I have used it well enough plus I feel like I have almost nothing to show from it relative to the time spent tinkering with it.

Explanation of Events

Incrementing and decrementing items within the stock and cart:

For these two functions I made separate classes rather than anonymous or inner classes. I felt like these needed to be pulled away from the stock panel that they were in. Within my constructor for both classes I pass in the item I want to increment/decrement, the associated stockText of that item and both my stock and cart objects. I required all 4 of these to perform the function I wanted.

The event that is performed is then made pretty simple. It first updates the stockText of the item. This is a label that is placed next to each item in the StockPanel, detailing how many of the item remains in stock. To be more human readable I used +/- 1 to not have the stock or items in the cart be at a 0 value.

It then calls add/remove item to/from the cart/stock method depending on whether I am adding to the cart and taking from the stock or vice versa.

Checkout and BMI frame creators:

I wanted to be able to create the instance of the frame on the push of a button so I made separate classes for these event handlers.

They both pass in their respective checkout and bmi panels, create new JFrame objects and set some default values.

In the checkout frame creator I decided to implement a condition statement that checks if there is actually any items in the cart to begin with by examining the value of the cart total. If the total was zero, I implemented a small dialog box informing them that they cannot check out with no items in the cart.

Below the instantiating and setting of the default values of the JFrame I pass in a string input into the cartpanel. This String input is the value of the total from the cart.

I done this here so it would immediately show the value when the frame is shown to the user.

Frame and system closers:

To create these, I used inner classes as their function was quite straightforward. For the CloseShop action listener I used System.exit(0). For the BMI Frame I used dispose inside an inner class within the BMI Panel.

BMI Calculator listener:

For this I used an anonymous class that takes in the values from input fields, does a calculation and returns that output to the user through a label.

CheckoutPanel listener:

For this I also used an anonymous class that first checks if the voucher number entered is correct and then applies a discount. It then displays the new value, removed several buttons and labels from the frame and adds a final checkout frame.

UML Class Diagram

