Matthew Regan

Object-Oriented Programming

**CONTENTS**

[Task 1 3](#_Toc377285400)

[P1 3](#_Toc377285401)

[Task 2 3](#_Toc377285402)

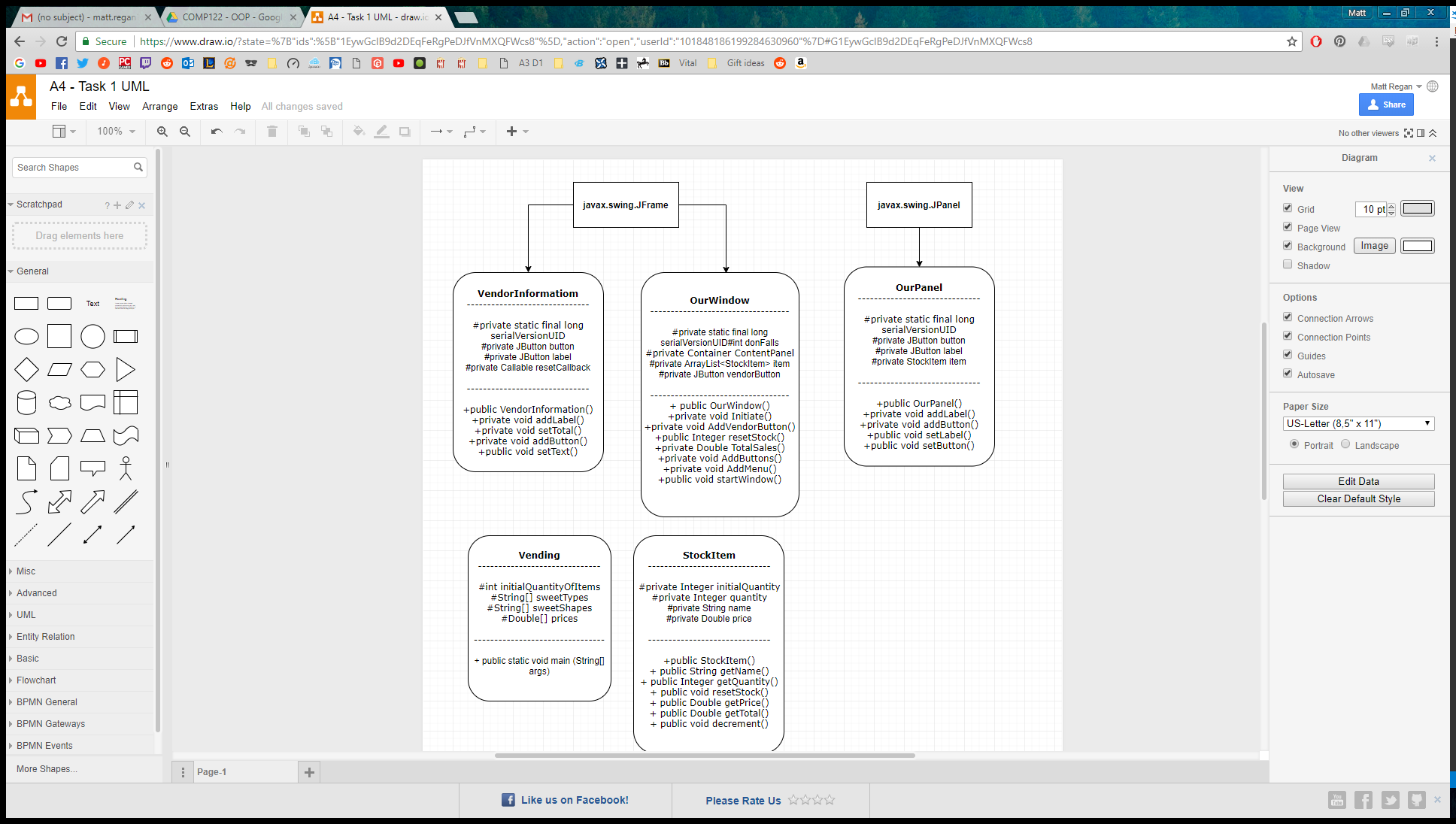
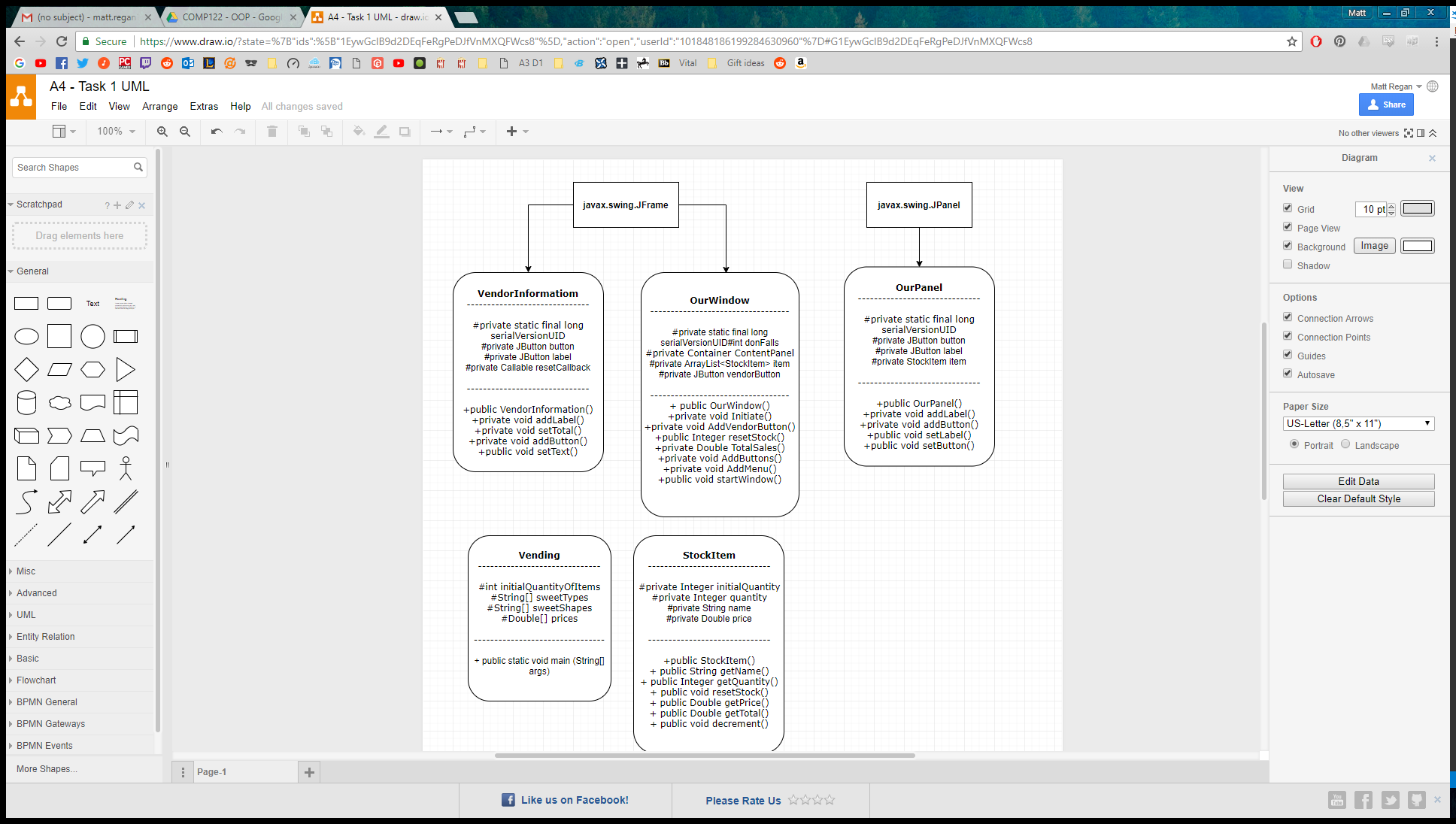
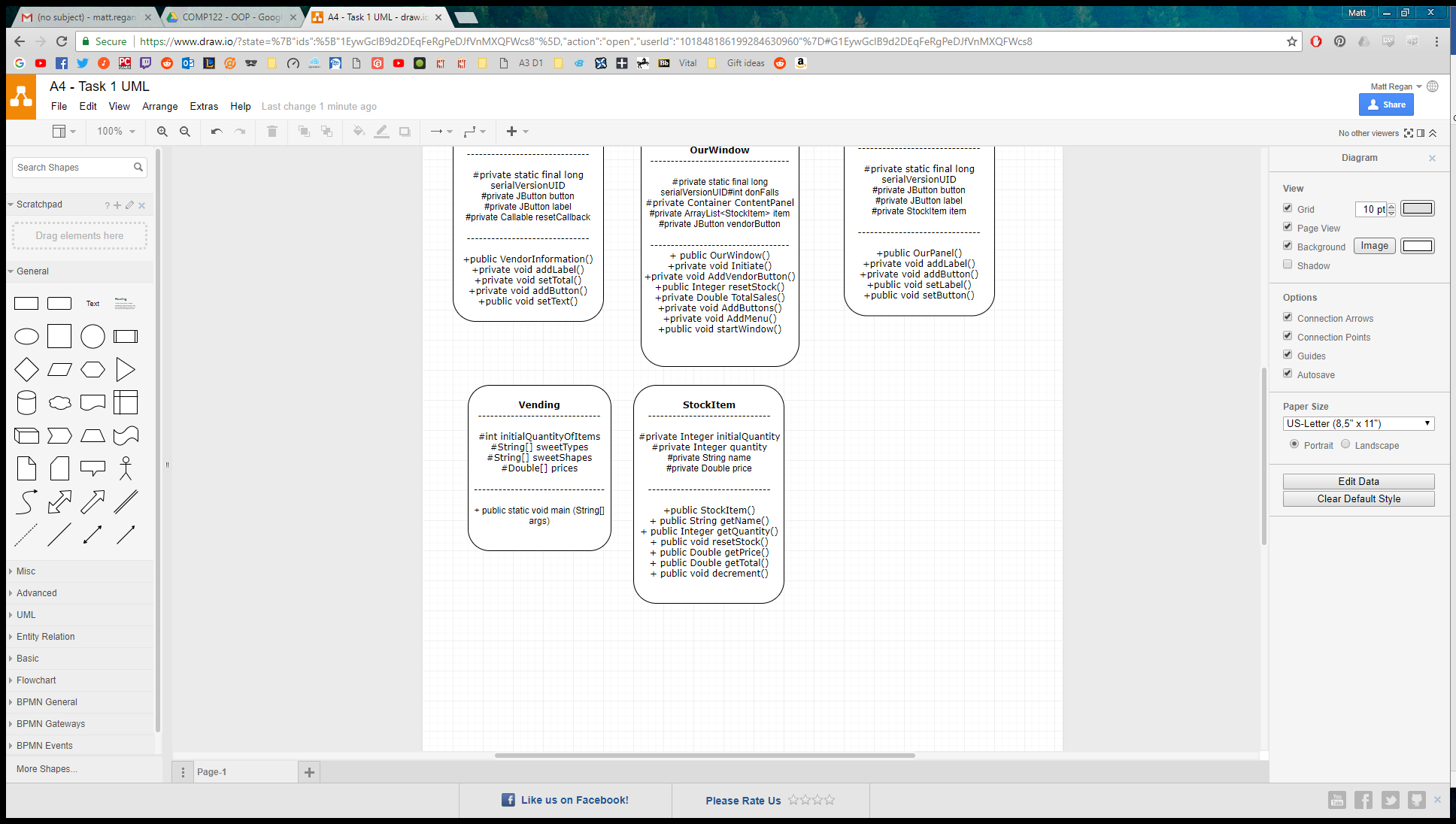
[P2 3](#_Toc377285403)

# Part 1 Requirements

For the final assignment of this module I have been challenged to create a graphical application, something I haven’t had experience in before now. More specifically I have been asked to create a graphical application to simulate the actions and calculations a vending machine typically performs. The core aspects of my vending machine application need to include buttons for each item, their associated costs and their remaining stock. Additionally, I need to have a button that creates a new window showcasing “vendor information”. This information is the total sales of items that have been dispensed from the vending machine as well as a button to reset the stock and total sales of all items.

# Part 1 – Analysis and Design

The principles of a vending machine are there are rows of items, each item has a cost and upon pressing a button an item is dispensed and the stock level of that item decrements. All of these are core ideas I have implemented in previous programs however I now have to do it within a graphical space. Going over the lectures as well as some online research I understood that the creation, manipulation and utilisation of JFrames would be essential in creating an efficient graphical application in Java. My initial thoughts and drafts of this vending machine application took some experimentation before I discovered the most efficient method for creating and calling multiple JFrames. My initial designs had blocks of code for creating each individual button and label. My final version uses method calls to create instances of the same object without repeating code.

# Part 1 – Testing

For the following tests I haven’t included screenshots of my outputs as they could not reasonably fit in the table and not squish the other columns. My output column will describe what the given screenshot is illustrating.

|  |  |  |  |
| --- | --- | --- | --- |
| Test No. | Input | Output | Expectations/Comments |
| 1. | User purchases each of the chocolate sweets | The stock of each “Chocolate” item reduces by 1 and the total sales now equal £3.10. | As expected. |
| 2. | User purchases 4 Caramel Bombs | Stock of Caramel Bombs equals 0 and total sales equals £4.00. | As expected |
| 3. | User purchases 5 Caramel Bombs | Stock of Caramel Bombs equals 0 and total sales equals £4.00. A dialogue box appears upon the fifth press of the button telling the user that given product is out of stock. The button then becomes disabled. | As expected |
| 4. | After purchasing 5 Caramel Bombs the user resets the vending machine. | The total sales returns to 0 and the stock of every item returns to 4. All items buttons can be pressed another 4 times before needing to be reset again. | Almost as expected. The number representing the stock level for each item doesn’t reset to 4 and instead sticks at 0. However, When the user purchases an item that was previously out of stock the stock information “catches up” and shows 3 left in stock. |
| 5. | The user purchases everything in the vending machine | All items have a stock of 0 and total sales equals £37.20. | As expected |