|  |
| --- |
| Name: |
| Student Reference Number: |



|  |  |  |
| --- | --- | --- |
| Module Code: SOFT255SL S | Module Name:  Software Engineering for the Internet using Java | |
| Coursework Title: Java GUI Based with OOPS Concepts | | |
| Deadline Date:01 Dec 2020 | | Member of staff responsible for coursework: |
| Programme:  Java SEE | | |
| Please note that University Academic Regulations are available under Rules and Regulations on the University website [www.plymouth.ac.uk/studenthandbook](http://www.plymouth.ac.uk/studenthandbook). | | |
| Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.   |  |  | | --- | --- | | Name | STUDENT REFERENCE | | BRA FERNANDO (Leader) | 10707197 | |  |  | |  |  | |  |  | |  |  |   ***We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.***  Signed on behalf of the group: | | |
| Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***  Signed : | | |
| Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.  I \*have used/not used translation software.  If used, please state name of software………………………………………………………………… | | |
| **Overall mark \_\_\_\_\_% Assessors Initials \_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_** | | |

\*Please delete as appropriateSci/ps/d:/students/cwkfrontcover/2013/14

**Self-service Food ordering System**

**Contents:**

1. **Introduction:** BRA FERNANDO
2. **UML Diagrams:**
3. **Use case:**
4. **Class Diagram:**
5. **Interfaces:**
6. **Home Page ---->**
7. **Selection Page --->**
8. **Meal Page --->** BRA Fernando
9. **Fast-food Page ---->**
10. **Beverages Page ---->**
11. **Snacks Page ---->**
12. **Order Details Page --->**
13. **Pay Page ---->**BRA FERNANDO
14. **Print Page ---->**
15. **Summary**

***Introduction***

The Self-Service Food Ordering System is developed with an aim for providing effectiveness to Canteen system at NSBM. Where the POS system consumes large amount of time of the clients (Students) to order their meals, the new system allows a better time management for the Students as well as the staff at the canteen. The Students (clients) can make orders at ease and add the food needed , or edit , or delete the goods needed at their ease without asking for help, this is more likely to increase the revenue that is generated to the canteen as well.

The Self-service food ordering System is built up with different pages to allows a much easier interaction for the customers, this includes 9 interfaces (pages) that’s guiding the Student through the process. Allowing a better customer interaction without any unease, making the system well known as possible.

The language that’s used for the development is Java SEE which is a cross platform language which is making the system to be run on each platform without any issues occurring .

**UML Diagrams**

UML Diagrams are based on a unified modelling language this allows the system to be visual presentable with its main actors , classes , roles and actions before the development process begins , thereby this visual representations can be used as aid for development by the developers for implementing the system successfully .

There were 2 main UML diagrams used as:

1. Use case:

Use case diagram can be described as a primary from of modelling that allows the system/software to be viewed on from the end user’s perspective allowing a visual presentation of system behaviour and users behaviour in the process. It’s a useful diagram especially to developers for an undeveloped system/software to understand what user requires and what’s not required.

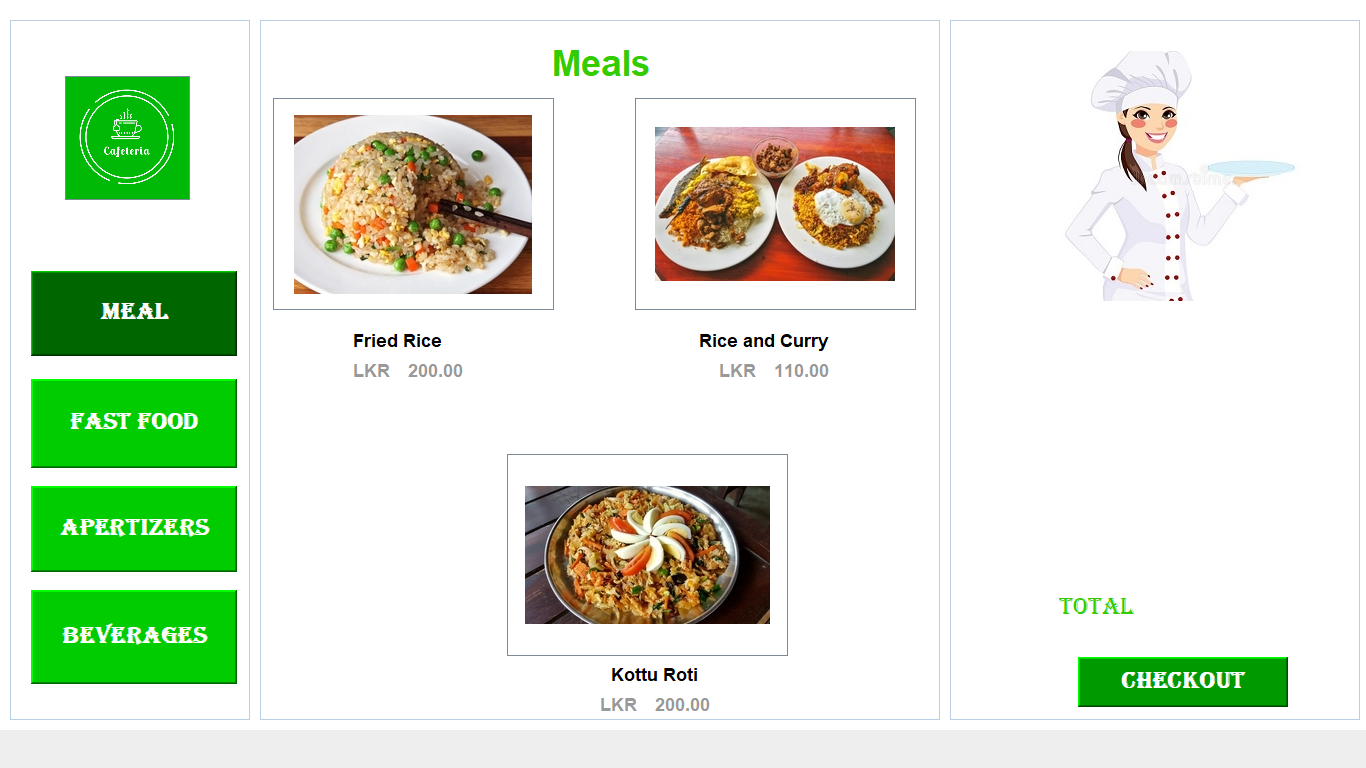
The Use case for Self-service Food ordering System:

1. Class Diagram:

***Interfaces of Self-service Fast-food System***

1. Home Page:
2. Selection Page:
3. ***Meal Page:***

The left Panel has menu pages for the clients to Order from if Clicked they will be redirect to the Clicked (JFrame) page ,the ordering process can be carried out at much easier and pleasant to the eye.



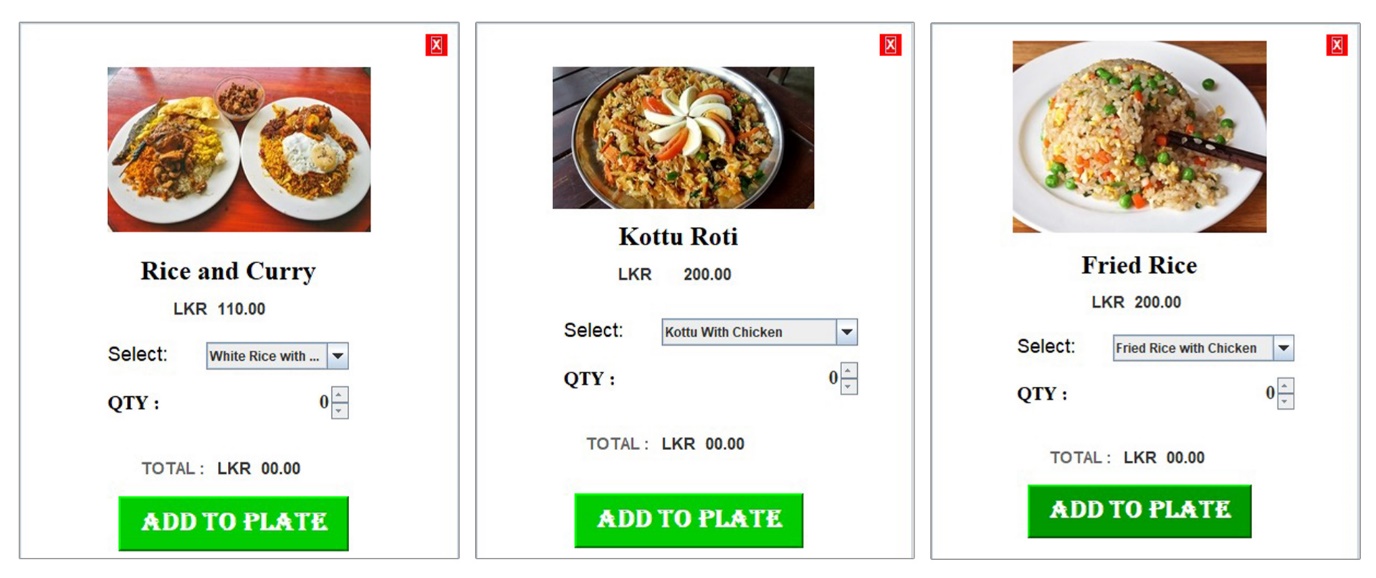
On mouse move the colour will be Dark green & on mouse exit the Colour will be reset to default light green

Pop up Pages (JFrames)

The Checkout button that will be redirecting to Order details Page

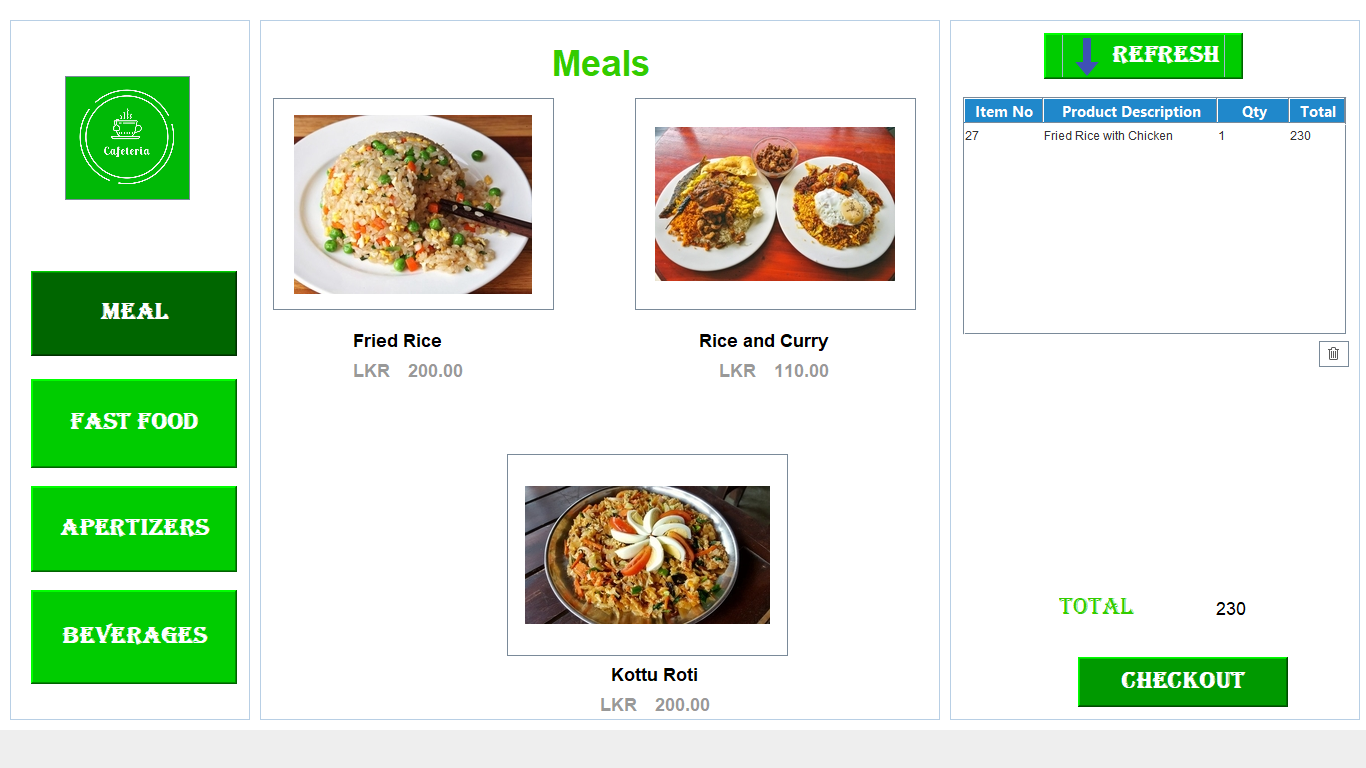
The Meal page is designed with three main meals available in the canteen, that is Fried rice, Kotta Roti and Rice and curry.

With Each Having their own popups pages (Frames) where the client can select the food from them. This can be visually seen as:



When the order is added from the popups by the customer the Order will be seen in the order details table such as :

The customer can refresh the order to check recently added



The Order Details

The concepts that are applied to Building the Meal JFrame is:

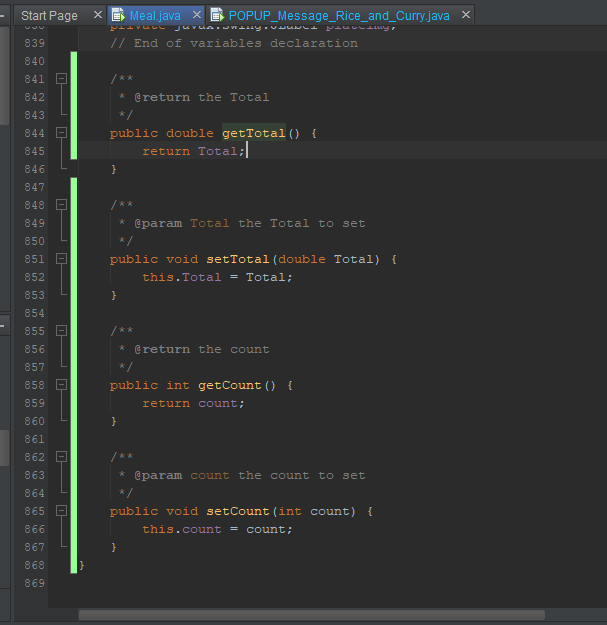
1. OOP Concepts
2. Inheritance:

Inheritance is Where the GUI JFrame (Meal) inherits all properties from javax.swing.JFrame which establishes a path to create GUI interface.

This process is establish because javax.swing.JFrame class inherits all the public properties(methods and fields) from java.awt.Frame class . And java.awt.Frame class inherits all the public properties from the java.awt.Windows, therefore javax.swing.JFrame class also inherits all those methods . This process can be seen in all the JFrames.

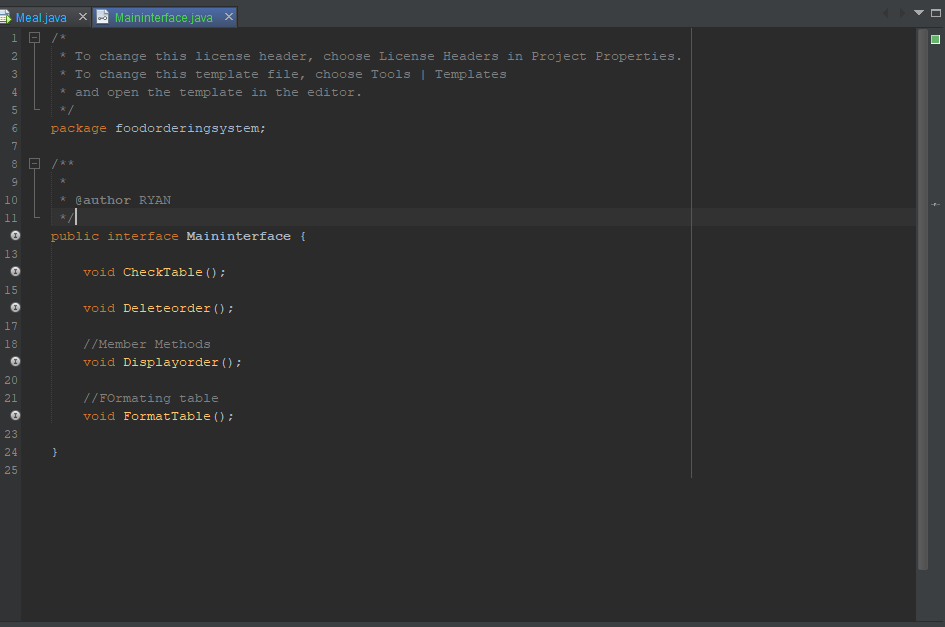
1. Encapsulation:

Encapsulation is a process of hiding sensitive data by wrapping fields or methods inside classes. This is implemented in system Meal frame as :



1. Interfaces:

Interfaces are one way for which the methods reusability can be implemented since java does not support more than one inheritance to take place. So, this makes implementation much easier for other JFrames to use the code, increase code cohesiveness and reusability .



1. Constructors:

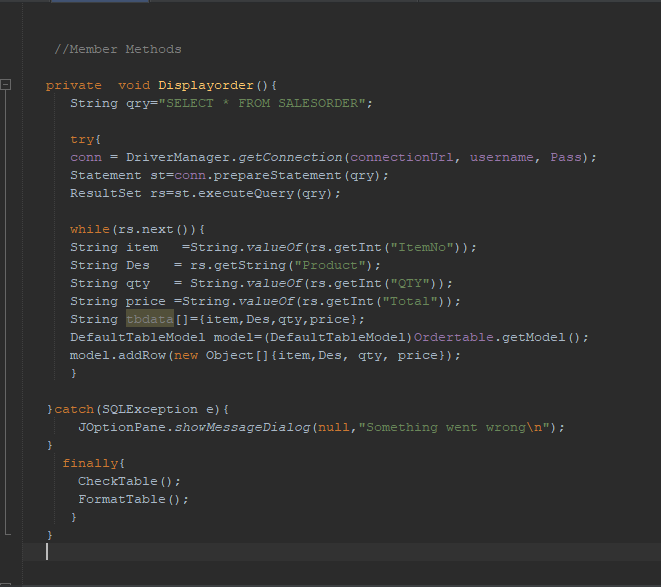
Are used to initiate the pages (JFrames) and set components inside the JFrame. Both Default and Parameterized Constructors are used. Utilized in Each JFrame.

Encapsulation is used to calculate the total price of the meals and set the total to be displayed in the interface, this allows hiding sensitive data.

1. Java Exceptions :

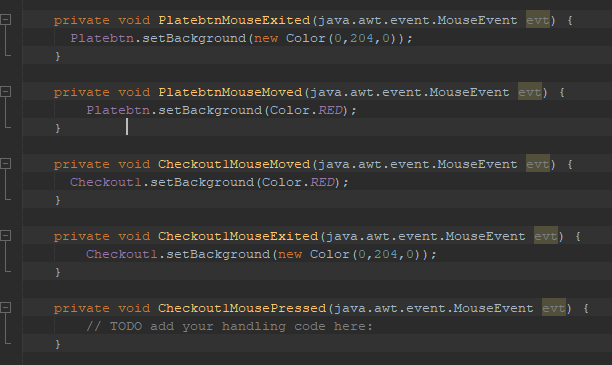
When executing the java programs several errors could occur, especially when executing SQL queries, if the errors ever occur the Whole system would crash for prevention of that the System is using try and catch in all methods that is implemented.

EX:



1. Event Listeners:

An Event listener are used to register a particular event and take action based on it to achieve a particular intended result.

For ex:

**POPUP Frames**

The Popup Pages (JFrames) also follows the same concepts used in the Meal Page (JFrame) , such as :

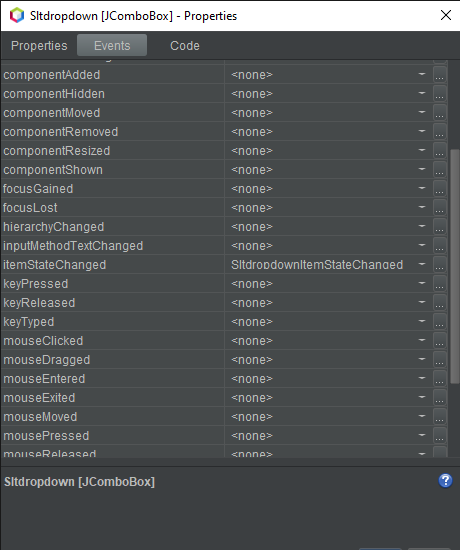
1. Inheritance
2. Encapsulation.
3. Exception Handling
4. Interfaces
5. Constructors

Apart from the OOPs there is additionally:

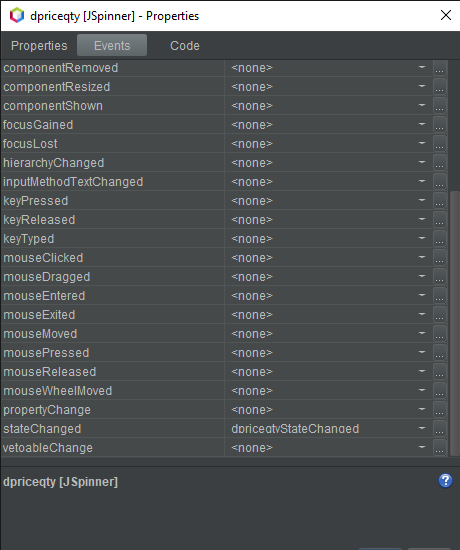
1. Event Listeners for each POPUP JFrame:

Each Popup JFrame will be allowing the customers (students) with options to choose the order from and in real time the Calculations will be done using proprieties in the JComboBox and JSpinner in java Swing.

1. JComboBox: ItemPropertychanged:



1. JSpinner: StateChange



On order the details will be added into the Order table where each customer is provided with facilities to delete the order or edit the order that is displayed in the table.

**Summary**

The functionalities that’s being implemented in code though using methods are aimed for implemented with the idea of reusability and code cohesiveness for each calculation , insertion , deletion , edition or any other operations implemented .