F21AS: Advanced Software Engineering

Stage 1

Group Report

Jowita Knap	H00301565
Lorenzo James	H00309663
Lucía Parga Basanta	H00313273
Sabrina Chiesurin	H00314757

1. Team

The group consists of four members: Sabrina Chiesurin, Lorenzo James, Jowita Knap, Lucía Parga Basanta. Initially the work was divided into creation of the GUI and development of classes and methods.

Sabrina Chiesurin: responsible for creating Item Gui, in which the user can add and remove items from the basket, and display the details of the offers. Worked on the final arrangement of the report.

Lorenzo James: continued working on the core of the application, developing the methods for the items, basket, discounts and the collections. Worked on the import methods (text, read from text, and write, creation of the menu, orders and Lists).

Jowita Knap: started creating all the classes. Responsible for creating Order Gui, in which the user can select the category of items that their want to order and their can do the check out. JUnit Test for GUI.

Lucía Parga: continued working on the core of the application, developing the methods for the items, basket, discounts and the collections. Worked on the output part (implement methods, working on discounts, creation of lists). Organized the packages. Created the Junit Tests.

The group has decided together about type of Collections to use and for what. Within the group, members helped each other to develop the features of the program. Indeed It is not possible to define perfectly who dealt with what.

2. Our repository

https://github.com/lorenzo456/AdvancedSoftwareEngineeringCoursework

3. Coursework specificiation

This program meets the specification.

4. Data Stracture

Basket – LinkedList. The basket is going to store the elements the customer has selected to order. It will be stored as a linked list because we are going to be adding/removing elements frequently.

Item list –TreeSet. The item list is going to be storing the items the coffee shop has on sale. It will be stored in a tree set because the list will be searched through often, we want to display the items in order and we don´t want duplicated of the type items we have on sale.

Discounts – HashMap. The Discount list will store all the combinations of items which are on discount. It will be stored in a map, because each combination of elements need to have an unique key, we want to easily store combinations and don´t want duplicate discounts. Key will be the name of the discount and Value will be an array which contain the categories (as string) of the items included in that discount.

DiscountList – ArrayList. (Included in the Discount Class). This list will hold the values of the HashMap of the Discount class. We do not need extra specification with this List, so just and array is enough.

5. Group decisions – functionality of the program

Discounts

Originally HashMap, but we decided to simplify the complexity of the discounting methods and we change this to an enumerator.

DropDown

In GUI. We use it for amount of items. It was decided to use this to simplify the process for the user and some errors that typing a number will cause.

Display offers

With information inside each category: this was decided thinking of the end user (offer info available for the person to decide whether to buy an item or another).

GUI

Initially it was thought as four GUIs, one for each category (Hot drinks, Cold drinks, Meals, Desserts), then the code has been optimized to get a single GUI.

7. Testing

What did you test using JUnit? Give details of which JUnit tests relate to which methods. Which types of exception did you use, in which method, and for what purpose?

JUnit for Items.

Compares two items and assert if they are equal or not via its ID. ID on the List is described as a String, but this one has to be two letters (related with the categroty of item) and two numbers. Testing with assertEquals gives the best output for this test.

JUnit for Discounts.

Asserts true or false if and item has been discounted or not by creating a boolean isDiscounted.

JUnit for Add and Deleting items.

Asserts true or false when an item is added to the basket. Also, asserts true or false if all items (of Id "x") has been deleted. Both tests have created a boolean isAdded or is Deleted to make the method work.

Exceptions

Try and catch for the methods of writing and reading (OrderFile). Also try and catch has been used to optimize the method getItemByID (RunTimeException has been implemented).

OrderGUI JUnit tests:

Test – testInitGUI

Test of GUI creation. It would test constructor of OrderGUI class and initGUI method that initialize GUI. This test assert if GUI is visible and if window dimension is correct.

Test – testCreateBasketPanel

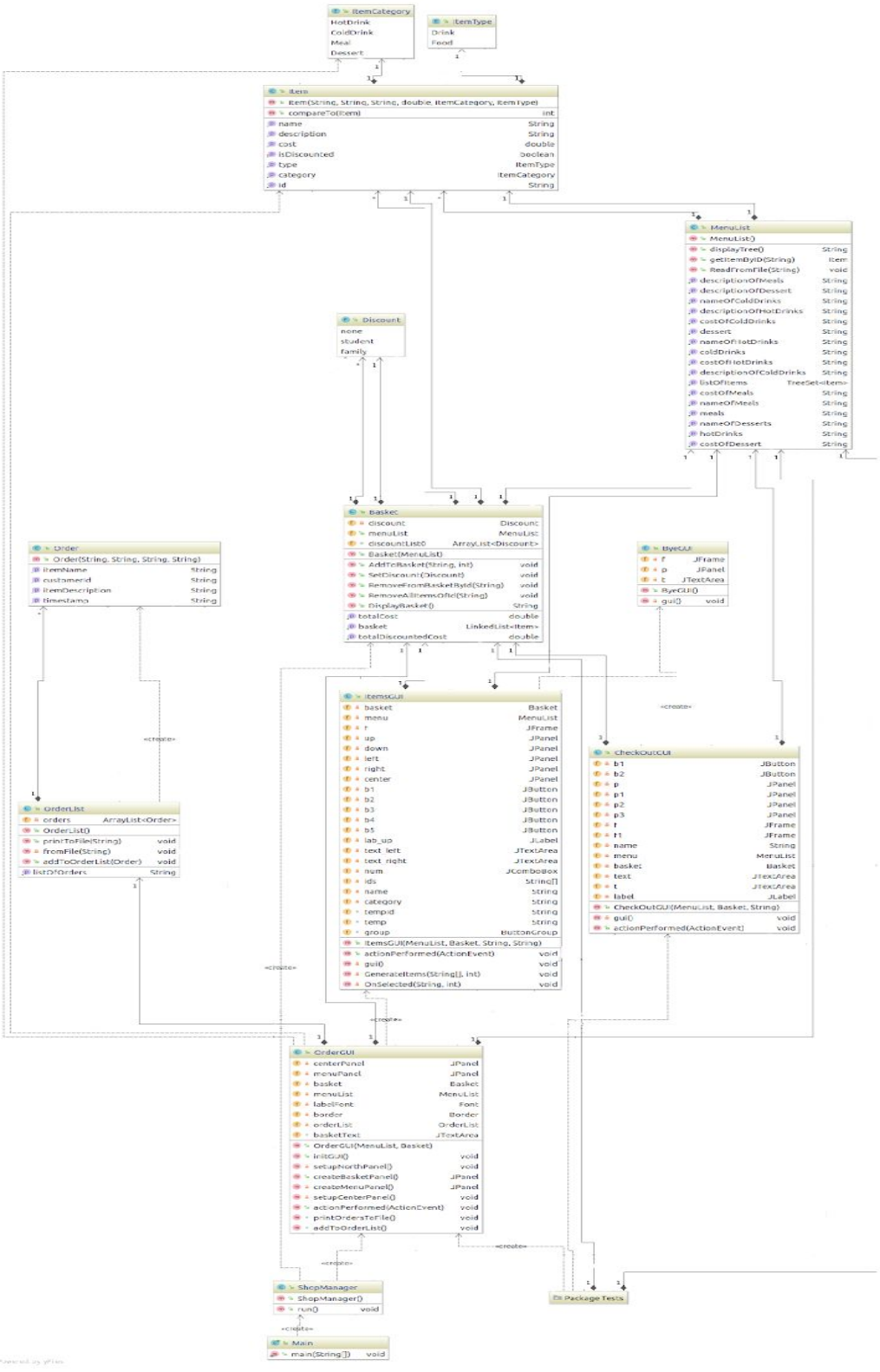
Test of basket panel creation. It would test if above method return panel with components that are instances of JLabel and JButton.

Test – testActionPerformed

Test action handling. It would first create an Fake ActionEvent and test whatever Jframe is disposed after opening ItemsGUI.

6. UML class diagram(s)

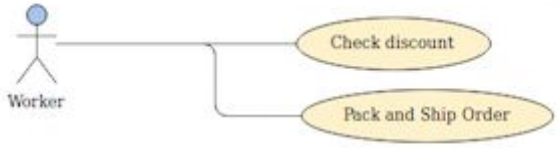
Class diagram showing the associations between the classes, and the contents of each class.


ItemCategory
HotDrink
ColdDrink
Meal
Dessert
ItemType
Drink
Food
Item
Item(String, String, String, double, ItemCategory, ItemType)
compareTo(Item) int
name String
description String
cost double
isDiscounted boolean
type ItemType
category ItemCategory
id String
MenuList
MenuList()
displayTree() String
getItemByID(String) Item
ReadFromFile(String) void
descriptionOfMeals String
descriptionOfDessert String
nameOfColdDrinks String
descriptionOfHotDrinks String
costOfColdDrinks String
dessert String
nameOfHotDrinks String
coldDrinks String
costOfHotDrinks String
descriptionOfColdDrinks String
listOfItems TreeSet<Item>
costOfMeals String
nameOfMeals String
meals String
nameOfDesserts String
hotDrinks String
costOfDessert String
Discount
none
student
family
Basket
discount Discount
menuList MenuList
discountList0 ArrayList<Discount>
Basket(MenuList)
AddToBasket(String, int) void
SetDiscount(Discount) void
RemoveFromBasketById(String) void
RemoveAllItemsOfId(String) void
DisplayBasket() String
totalCost double
basket LinkedList<Item>
totalDiscountedCost double
Order
Order(String, String, String, String)
ItemName String
customerId String
ItemDescription String
timestamp String
byeGUI
f JFrame
p JPanel
t JTextArea
ByeGUI()
gui() void
itemsGUI
basket Basket
menu MenuList
f JFrame
up JPanel
down JPanel
left JPanel
right JPanel
center JPanel
b1 JButton
b2 JButton
b3 JButton
b4 JButton
b5 JButton
lab_up JLabel
text_left JTextArea
text_right JTextArea
num JComboBox
ids String[]
name String
category String
tempId String
temp String
group ButtonGroup
ItemsGUI(MenuList, Basket, String, String)
actionPerformed(ActionEvent) void
gui() void
GenerateItems(String[], int) void
OnSelected(String, int) void
checkOutGui
b1 JButton
b2 JButton
p JPanel
p1 JPanel
p2 JPanel
p3 JPanel
f JFrame
f1 JFrame
name String
menu MenuList
basket Basket
text JTextArea
t JTextArea
label JLabel
CheckOutGUI(MenuList, Basket, String)
gui() void
actionPerformed(ActionEvent) void
OrderList
orders ArrayList<Order>
OrderList()
printToFile(String) void
fromFile(String) void
addToOrderList(Order) void
listOfOrders String
OrderGUI
centerPanel JPanel
menuPanel JPanel
basket Basket
menuList MenuList
labelFont Font
border Border
orderList OrderList
basketText JTextArea
OrderGUI(MenuList, Basket)
initGUI() void
setupNorthPanel() void
createBasketPanel() JPanel
createMenuPanel() JPanel
setupCenterPanel() void
actionPerformed(ActionEvent) void
printOrdersToFile() void
addToOrderList() void
ShopManager
ShopManager()
run() void
Main
main(String[]) void
Package Tests
«create»

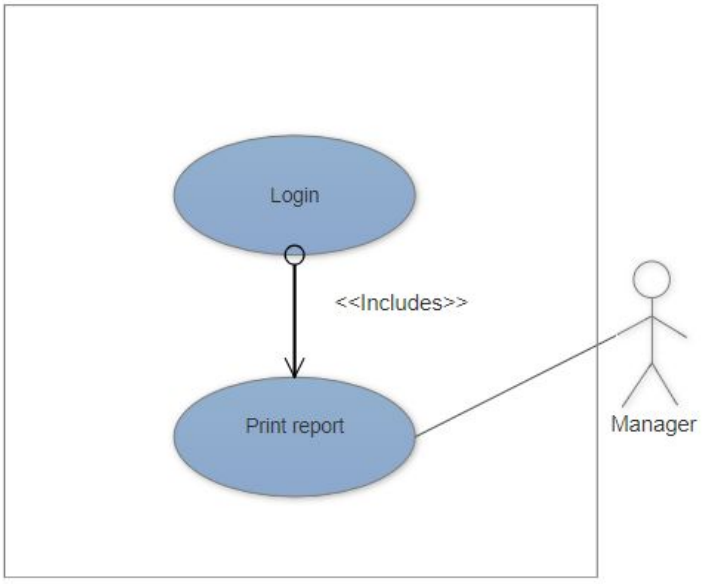
Use Case Diagrams

Description of the possible interaction with the system that the different actuators will have in our Programme.

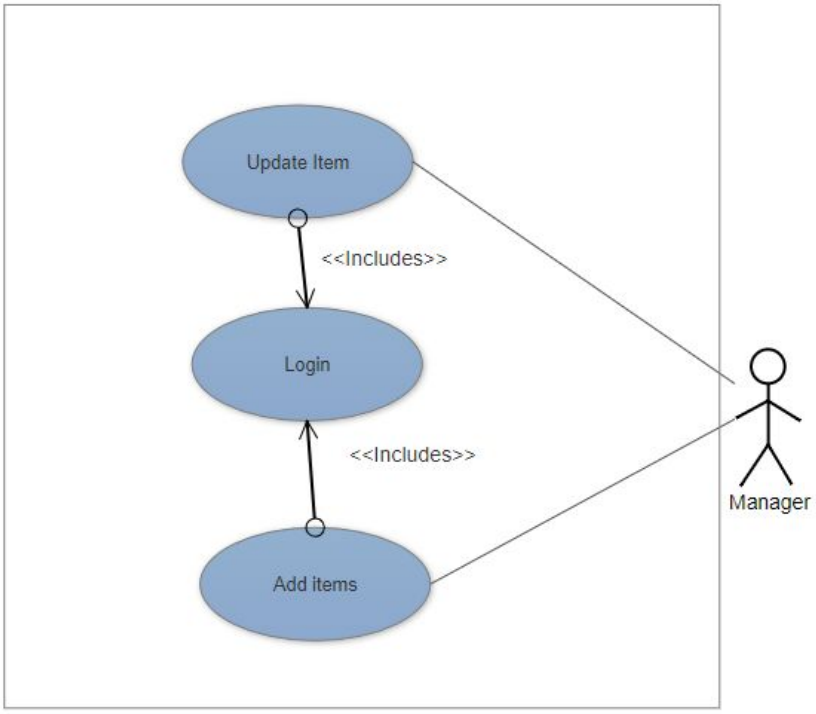
Use Case Diagram for: check discounts (Worker Use Case)



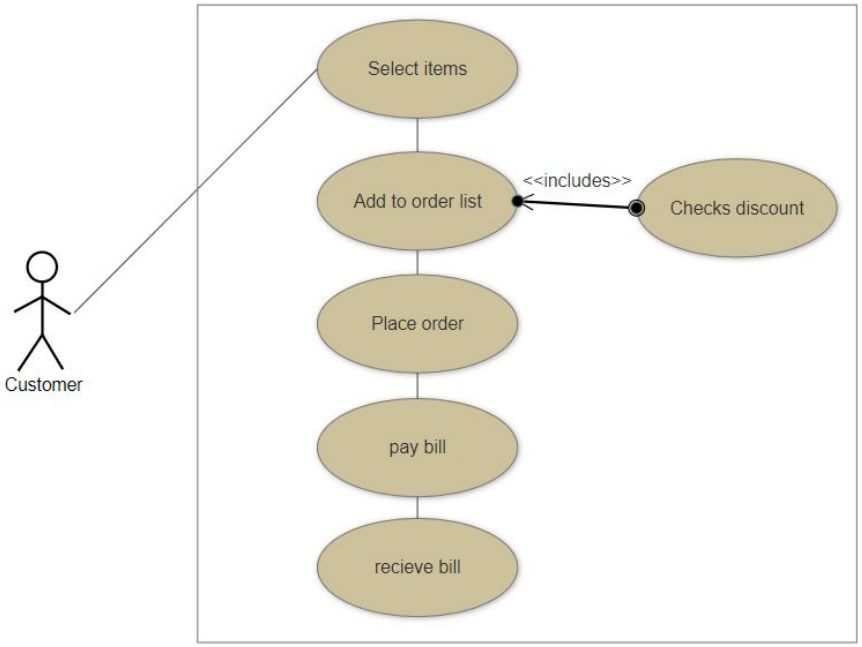
USE CASE DIAGRAM: Print report



USE CASE DIAGRAM: Add items

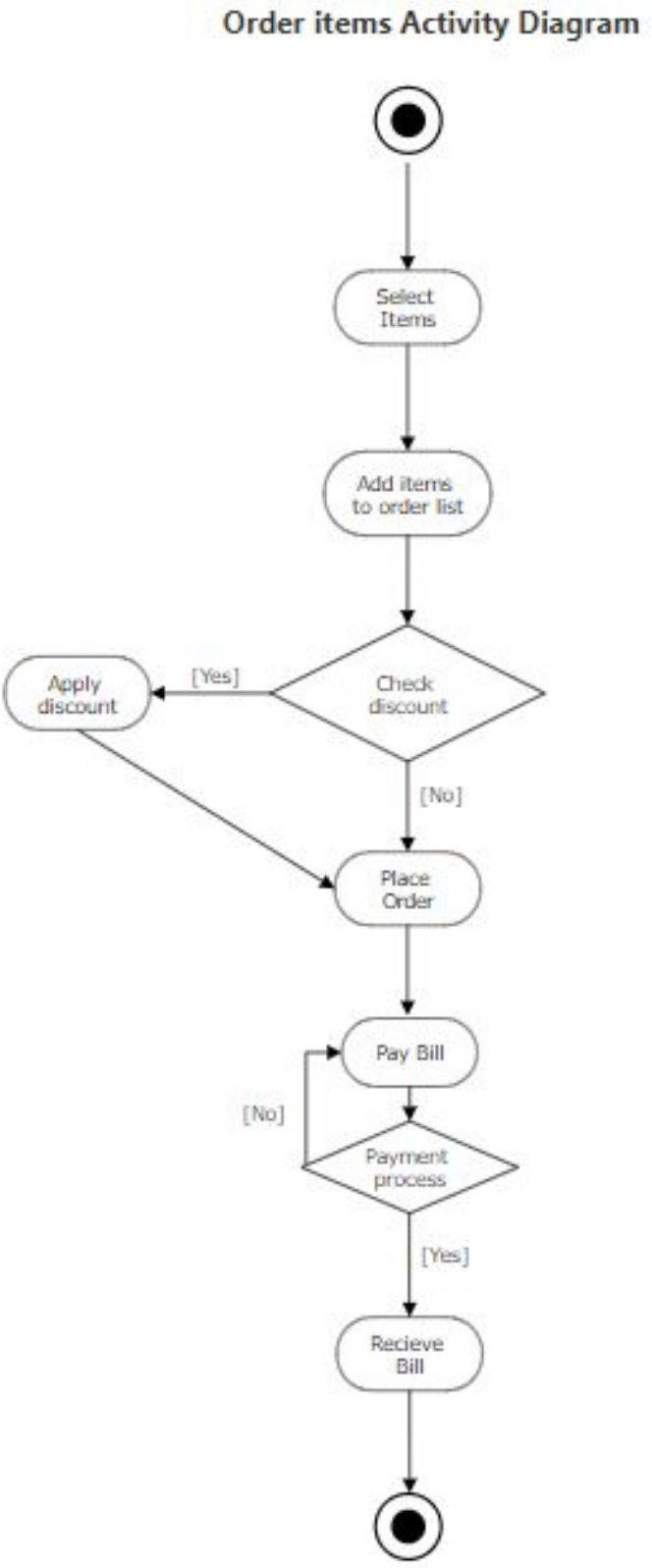


Use Case Diagram: Apply discount



Activity diagrams

Showing the workflow and how software and human activities interact.



Add items Activity Diagram