CS 102 - Project 02

Due Date: Sunday, May 3, 2020 at 23:59pm

For the term project you are going to implement a restaurant application for taking orders, calculating expenses and revenues. This is the first part of your project. In this part you will implement the model part of your project. In the second part of the project you are going to integrate it to a GUI.

You will be implementing several classes in this project. A UML diagram which displays the classes and their relations is presented in Figure 1. The description of the classes is also summarized in detail. An example run of the program is also provided in this document.

The Restaurant Application

The restaurant application you are implementing can be used by both the restaurant customers and the restaurant manager. Customers can use the application to give their orders. Furthermore, the restaurant manager can use the application to follow the restaurant's expenses and revenues. The manager can also display the employees or add more employees.

When the program starts, the following options will be displayed to users:

```
Welcome to OZU Restaurant!

-------------------------Main Menu-------------------------
1: Create Order
2: Manage Restaurant
3: Exit Program
-----------------------------------------------------------
>>
```

User can input 1 in order to create an order or 2 to manage the restaurant. If user enters a number that is not 1, 2 or 3, the program displays the menu again and wait for user input. In order to exit the program user needs to enter 3. An example snapshot of the program is as follows:

```
Welcome to OZU Restaurant!

-------------------------Main Menu-------------------------
1: Create Order
2: Manage Restaurant
3: Exit Program
-----------------------------------------------------------
>> 4
Try Again...
-------------------------Main Menu-------------------------
1: Create Order
2: Manage Restaurant
3: Exit Program
-----------------------------------------------------------
>> 3
Bye
```

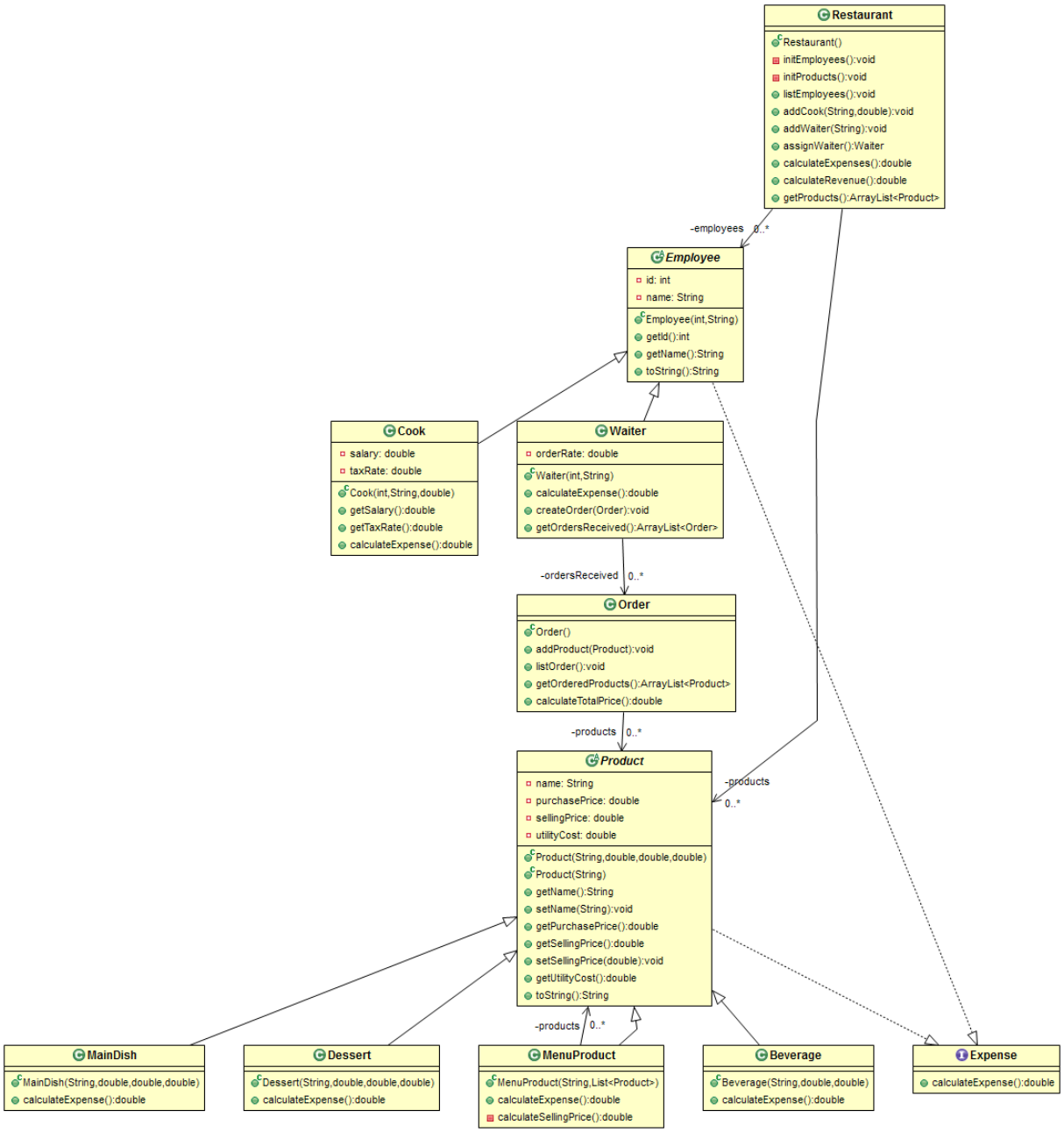


Figure 1. UML Diagram of the Restaurant Application

You will be given the Main.java which contains the main method for displaying all these options. This Main.java will be using other classes that you will be implementing.

Manage Restaurant

If user enters 2, manage restaurant menu will be displayed. In this menu, the manager can either list the employees, add a new employee, calculate expenses of the restaurant or calculate the revenue. Manager can also enter 5 in order to get back to Main menu. An example run of the manage restaurant is as following:

```
>> 2
--------------------Manage Restaurant--------------------
1: List Employees
2: Add Employee
3: Calculate Expenses
4: Calculate Revenue
5: Go back to Main Menu
---------------------------------------------------------
>> 1
Employee 1: Monica
Employee 2: Ross
Employee 3: Phobe
Employee 4: Rachel
--------------------Manage Restaurant--------------------
1: List Employees
2: Add Employee
3: Calculate Expenses
4: Calculate Revenue
5: Go back to Main Menu
---------------------------------------------------------
>> 2
---------------------Add Employee--------------------
1: Add Cook
2: Add Waiter
3: Go back to Manage Restaurant Menu
-----------------------------------------------------
>> 1
Name of the Cook:
>> Chandler
Salary of the Cook:
>> 100
---------------------Add Employee--------------------
1: Add Cook
2: Add Waiter
3: Go back to Manage Restaurant Menu
-----------------------------------------------------------
>> 2
Name of the Waiter:
>> Joey
Returning to Main Menu
---------------------Add Employee--------------------
1: Add Cook
2: Add Waiter
3: Go back to Manage Restaurant Menu
-----------------------------------------------------------
>> 3
Returning to Manage Restaurant Menu
--------------------Manage Restaurant--------------------
1: List Employees
2: Add Employee
3: Calculate Expenses
4: Calculate Revenue
5: Go back to Main Menu
-----------------------------------------------------------
>> 1
Employee 1: Monica
Employee 2: Ross
Employee 3: Phobe
Employee 4: Rachel
Employee 5: Chandler
Employee 6: Joey
```

As one can notice, there are two types of employees in this restaurant: (1) cook and (2) waiter. When manager wants to list all employees, all waiters are displayed in the following format:

Employee *employeeID*: *employeeName*

employeeID starts from 1 and incremented whenever a new employee is added. If a new cook is added to the restaurant, then both the name and salary of the cook needs to be entered. In case of adding a waiter, only the name of the waiter is inserted. The waiters do not have a base salary but instead get some percentage of the order they got from customers. More details will be explained later in the document.

```
----------------------Manage Restaurant----------------------
1: List Employees
2: Add Employee
3: Calculate Expenses
4: Calculate Revenue
5: Go back to Main Menu
-------------------------------------------------------------
>> 3
Employee expenses: 236.0
Order expenses: 0.0
Total Expense: 236.0
----------------------Manage Restaurant----------------------
1: List Employees
2: Add Employee
3: Calculate Expenses
4: Calculate Revenue
5: Go back to Main Menu
-------------------------------------------------------------
>> 4
Total Revenue: 0.0
----------------------Manage Restaurant----------------------
1: List Employees
2: Add Employee
3: Calculate Expenses
4: Calculate Revenue
5: Go back to Main Menu
-------------------------------------------------------------
```

In a restaurant, there are two types of expenses:

(1) Employees' salaries and tax if any
(2) Purchase costs of ingredients and utility cost of cooking the main dish or desserts (only the expenses of the restaurant are the sum of these)

At the above example, there are no waiter costs since there is not any order given. There are two cooks, both with salary 100. There is a 18% cost for the tax. The total expense for the employees is therefore 2*(100 + 100*18%) = 236.

Since the restaurant did not get any order, the revenue is 0.

Create Order

When user wants to create an order, the system randomly assigns a waiter as shown below.

```
------------------------Main Menu-------------------------
1: Create Order
2: Manage Restaurant
3: Exit Program
-----------------------------------------------------------
>> 1
Hi, I am Phobe.
I will be our waiter today.
What would you like to get today?
-----------------------Create Order----------------------
1: List Order
2: Add Product
3: Complete Order
4: Go back to Main Menu
-----------------------------------------------------------
>> 1
You have not ordered anything yet!
-----------------------Create Order----------------------
1: List Order
2: Add Product
3: Complete Order
4: Go back to Main Menu
-----------------------------------------------------------
>> 2
-----------------------Add Product-----------------------
1: Pizza
2: Burger
3: Coke
4: Lemonade
5: Tiramusu
6: Cake
7: Ice Cream
8: Hunger Games Menu
9: Kids Menu
>> 9
Kids Menu: 7.9
-----------------------Create Order----------------------
1: List Order
2: Add Product
3: Complete Order
4: Go back to Main Menu
-----------------------------------------------------------
>> 2
-----------------------Add Product-----------------------
1: Pizza
2: Burger
3: Coke
4: Lemonade
5: Tiramusu
6: Cake
7: Ice Cream
8: Hunger Games Menu
9: Kids Menu
-----------------------------------------------------------
>> 1
Kids Menu: 7.9
Pizza: 6.0
```

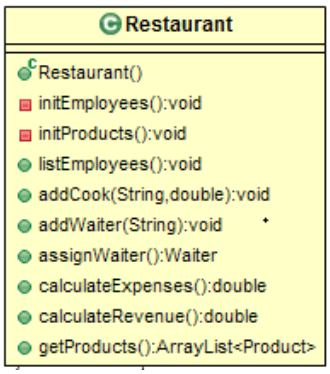
```
----------------------Create Order----------------------
1: List Order
2: Add Product
3: Complete Order
4: Go back to Main Menu
---------------------------------------------------------
>> 3
Your order is complete!
Returning to Main Menu
-------------------------Main Menu-------------------------
1: Create Order
2: Manage Restaurant
3: Exit Program
---------------------------------------------------------
>> 2
--------------------Manage Restaurant--------------------
1: List Employees
2: Add Employee
3: Calculate Expenses
4: Calculate Revenue
5: Go back to Main Menu
---------------------------------------------------------
>> 3
Employee expenses: 237.39
Order expenses: 8.8
Total Expense: 246.19
--------------------Manage Restaurant--------------------
1: List Employees
2: Add Employee
3: Calculate Expenses
4: Calculate Revenue
5: Go back to Main Menu
---------------------------------------------------------
>> 4
Total Revenue: 13.9
```

If user does not enter 3, the order will not be completed.
After the order is given, the expenses and revenue has been changed as you can see above.

Implementation Details

Restaurant.java:

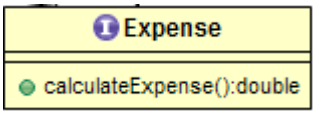
Manage restaurant option uses the Restaurant.java class. Partial implementation of the Restaurant.java will be provided to you. You need to implement the rest of this class yourself.



The constructor, initEmployees and initProducts methods are implemented for your convenience. You need to implement the other methods yourself.

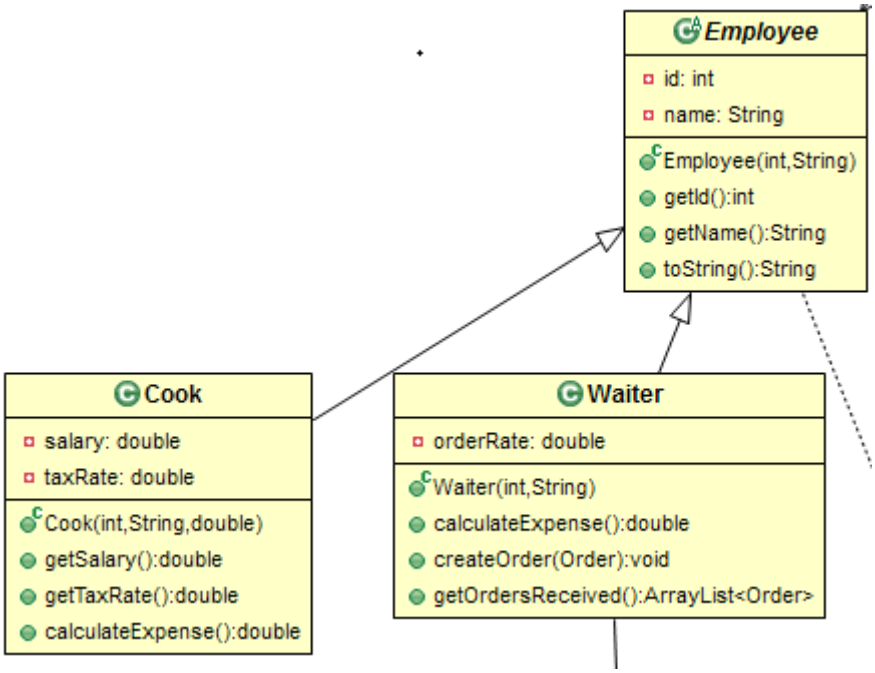
- **listEmployees** method prints out all employees.
- **addCook** method adds a cook to the restaurant. It takes the name and salary of the cook.
- **addWaiter** method adds a waiter to the restaurant. It takes the name of the waiter.
- **assignWaiter** method randomly selects a waiter and returns it.
- **calculateExpenses** method calculates the employee expenses and order expenses and returns the sum.
- **calculateRevenue** method calculates the revenue (earned money) from orders. This is not the profit. Profit is the Revenue – Expenses.
- **getProducts** returns the products.

Expense.java:



This class is an interface. A restaurant can have several type of expenses like the salary of the employees, utility costs and ingredient expenses. Create an interface with the calculateExpense() method which will be implemented by employee and product classes. This method returns the calculated expense.

Employee Classes: Employee.java, Cook.java and Waiter.java

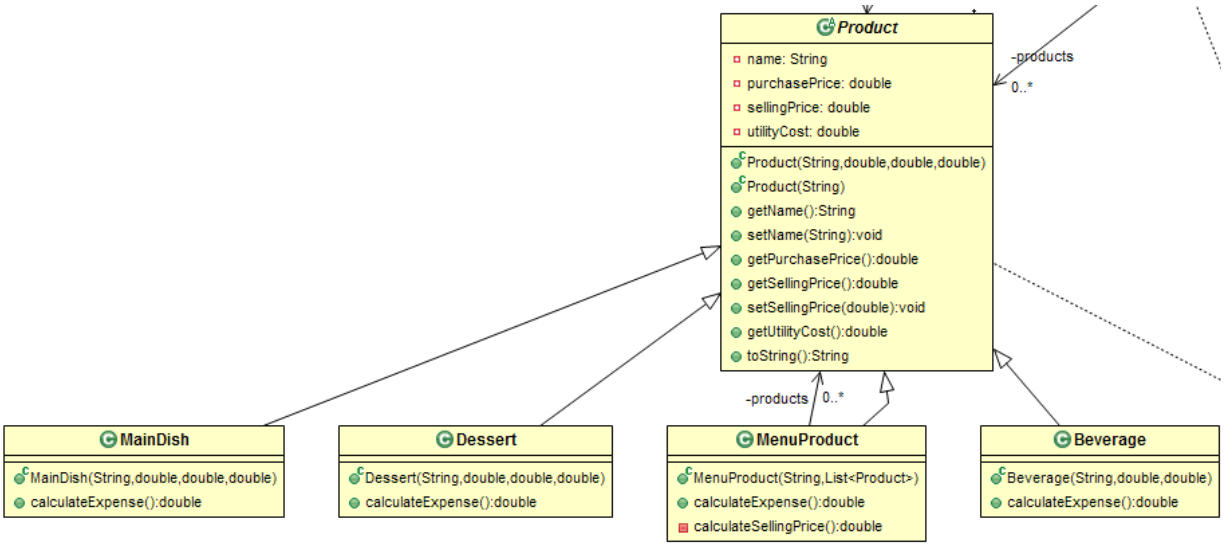


In this restaurant you will be implementing two types of employees: cook and waiter. All employees have some common attributes like their employee identification number (id) and name. Implement the necessary getMethods to access these attributes. Employee class implements the Expense Interface but there is not a default implementation of the calculateExpense() method. We should not allow the instantiation of an Employee. It needs to be one of the types either a cook or a waiter.

Cooks get a base salary. The restaurant pays their base salary as well as their tax which is calculated by multiplying the base salary with the tax rate (18%). The cook class holds these additional two attributes: the salary and tax rate.

Waiters are paid over the orders they serve. From each order they earn 10% of total price of the order. Even though it is not visible at the UML diagram Waiter class also holds an ArrayList which for the received orders (more details of this order class is explained in the rest of this document). You need to iterate over this orders list to calculate the salary of a waiter. The restaurant does not pay tax for the waiters. When manager wants to see the orders, they need to call **getOrdersReceived** method from each waiter. In order to add an order to this list of orders **createOrder** method needs to be called with the order sent as an argument.

Product Classes:
Product.java, MainDish.java, Dessert.java, Beverage.java and MenuProduct.java



Implement a product class which holds the name of the product, its selling price, its purchase price (the cost of ingredients), and its utility cost (cost of cooking). The necessary get and set methods needs to be implemented. This product class also implements the Expense interface but leaves the implementation of the **calculateExpense** method to its subclasses.

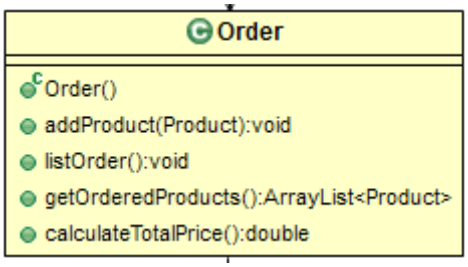
The restaurant sells both individual products and menus which are a combination of individual products. Restaurant has three types of individual products. These are main dish, dessert and beverage.

The expense of main dish and desserts are calculated as the sum of the purchase price as well as the utility cost (like the gas used while cooking). However, the expense of beverages is just its purchase price (beverages do not need a processing therefore they don't have utility costs).

The menu product class contains an ArrayList of individual products. The **calculateExpense** method of the menu product calls the **calculateExpense** method from all these products and sum these total expenses to find the total expense of the menu product. Similarly, the selling price of the menu product is calculated by calling the **getSellingPrice** method of these products. The selling price of the menu is calculated and set by calling this method. The total price of menu is lower than buying those

individual products alone. When calculating the selling price of the menu a 10% discount is applied to main dishes, 20% discount is applied to desserts and 50% discount to beverages.

Order.java:



An order is created by waiters for each group of customers. Order class holds an ArrayList of ordered products (either individual or menu).

- **addProduct** method is used to add a product to the order.
- **listOrder** method displays the ordered products.
- **getOrderedProducts** method returns the list of ordered products.
- **calculateTotalPrice** method calculates the total selling price of the order by iterating over products.

Submission (Due Date: Sunday, May 3, 2020 at 23:59 pm)

- The GitLab side of the project submission will be as follows:
  - Please create a GitLab account using your OzU e-mail address.
  - Create a private repository on your account and clone it to your computer.
  - Don't work on the master branch.
  - Create a branch called "project02" in your repository.
  - We expect you to have at least 20 commits in this project. Please frequently so that your project evaluation can be tracked easily.
  - Once you're finished with the project create a merge/pull request and assign Nitel Muhtaroglu as the reviwer.
  - You should follow the file and method naming conventions. Point will be deducted if you don't do so. Submit the following java files via your pull-request:
    - Beverage.java
    - Cook.java
    - Dessert.java
    - Employee.java
    - Expense.java
    - Main.java
    - MainDish.java
    - MenuProduct.java
    - Order.java
    - Product.java

        - Restaurant.java
        - Waiter.java
- On LMS side:
    - We expect you to give us a link of your merge/pull request.
    - An approximately 10-page long presentation showing your design, explanations of your decisions and your memory diagrams.
    - An approximately ~10 min video of your presentation. You simply need to talk on your submitted presentation and record your voice and your computer's screen similar to our online classes.

Do all the implementations yourself. In case of a plagiarism, you will receive -100.