QUIZ2

TAs : Alpay TEKİN
Due Date : 01.12.2024 - Sunday (23:00:00)
Programming Language: Java 8

1 Introduction

Object-Oriented Programming (OOP) is a key paradigm in software development that emphasizes modularity, reusability, and abstraction. By leveraging inheritance, encapsulation, and abstraction, OOP allows us to design robust and maintainable systems. In this quiz, you will implement a simple system for managing a restaurant menu. The system processes menu items from a file and calculates the total price and calorie count of customer orders. The goal is to demonstrate your understanding of OOP principles, such as class hierarchies, method overriding, and data encapsulation.

2 Restaurant Menu System

In this quiz, you will implement a program to manage and display orders for a restaurant using Object-Oriented Programming (OOP) principles. The goal is to create a modular, reusable, and readable system that handles multiple tables and their respective orders. This will demonstrate your understanding of OOP concepts like inheritance, method overriding, and data encapsulation. Your program will process two input files:

- A menu file containing details about the available menu items.
- A table file that defines the orders placed by different tables.

The program will then output a detailed summary of each table's orders, including the total price and total calorie count. The system is structured into the following components:

2.1 Menu Items

The menu is divided into the following categories:

- Soup
- Beverage
- Main Course
- Salad
- Dessert

Each menu item has a name:

- A tyoe of the menu item(Soup,Salad,MainCourse,Dessert,Beverage)
- A unique code (e.g., ID001)
- A name (e.g., Vegetable Soup)
- A price (e.g., 5.30$)
- A calorie count (e.g., 72.0 cal)

2.2 Tables

The system manages orders for multiple tables. Each table has:

- A name (e.g., Table 1)
- A list of orders (menu items)

Your program should display all orders of the table and calculate and display the total price and total calories for the table.

3 Specifications

- Prices should be displayed with exactly two decimal places.
- For single-digit decimal values, pad with a leading zero (e.g., "10.05" instead of "10.5").
- Account for decimal parts exceeding 100 (carry over to integer part).
- Display detailed information for each menu item.
- Display menu item information, total price and calories of the each table at the end.
- You must also **use appropriate Collections** to store and manage your menu items and tables.
- You must use **Object-Oriented Programming (OOP) principles (encapsulation, inheritance, abstraction, and polymorphism** wherever it is logical.

4 Input and Output

4.1 Input Format

You will work with two input files:

```
<type>,<code>,<name>,<price>,<calorie>
```

Figure 1: Content of the menu.txt

```
<name><tab><code_1>,<code_2>,<code_3>,...<code_n>
```

Figure 2: Content of the orders.txt

- menu.txt contains the list of menu items with unique codes as shown in 1.
- orders.txt contains the list of tables and their corresponding ordered menu items by unique code as shown in 2.

```
Soup,ID001,Vegetable Soup,5.30,72
Soup,ID002,Tomato Soup,6.00,102
Salad,ID003,Chicken Cesar Salad,5.00,469
Salad,ID004,Corn Salad,3.75,228
Salad,ID005,Greek Salad,4.50,72
MainCourse,ID006,Chicken Noodle,7.00,114
MainCourse,ID007,Fajita,9.50,234
MainCourse,ID008,Samosa,7.30,112
Desert,ID009,Ice Cream,3.00,267
Desert,ID010,Pudding,2.50,288
Desert,ID011,Apple Pie,3.50,411
Beverage,ID012,Coke,1.50,189
Beverage,ID013,Apple Juice,2.00,170
Beverage,ID014,Orange Juice,2.00,170
Soup,ID015,Vegetable Soup,5.30,72
Beverage,ID016,Coke,1.50,189.0
MainCourse,ID017,Chicken Noodle,7.00,114
Dessert,ID018,Ice Cream,3.00,267
Salad,ID019,Caesar Salad,4.25,180
```

Figure 3: Example input

4.2 Output Format

Your program should:

- Print information about each menu item in a formatted manner (e.g., Soup: Vegetable Soup Price: 5.30$ Calorie: 72.0 cal).
- Calculate and display the total price and total calories of all items.

```
Table 1 ID001,ID004,ID006,ID009,ID014
Table 2 ID002,ID005,ID008,ID010,ID016
Table 3 ID015,ID005,ID008,ID010,ID016
```

Figure 4: Example input

```
==============================
Table Name: Table 1
------------------------------
Orders:
Soup: Vegetable Soup Price: 5.30$ Calorie: 72.0 cal
Salad: Corn Salad Price: 3.75$ Calorie: 228.0 cal
Main Course: Chicken Noodle Price: 7.00$ Calorie: 114.0 cal
Dessert: Ice Cream Price: 3.00$ Calorie: 267.0 cal
Beverage: Orange Juice Price: 2.00$ Calorie: 170.0 cal
------------------------------
Total Price: 21.5$
Calories: 851.0 cal
==============================
==============================
Table Name: Table 2
------------------------------
Orders:
Soup: Tomato Soup Price: 6.00$ Calorie: 102.0 cal
Salad: Greek Salad Price: 4.50$ Calorie: 72.0 cal
Main Course: Samosa Price: 7.30$ Calorie: 112.0 cal
Dessert: Pudding Price: 2.50$ Calorie: 288.0 cal
Beverage: Coke Price: 1.50$ Calorie: 189.0 cal
------------------------------
Total Price: 21.80$
Calories: 763.0 cal
==============================
==============================
Table Name: Table 3
------------------------------
Orders:
Soup: Vegetable Soup Price: 5.30$ Calorie: 72.0 cal
Salad: Greek Salad Price: 4.50$ Calorie: 72.0 cal
Main Course: Samosa Price: 7.30$ Calorie: 112.0 cal
Dessert: Pudding Price: 2.50$ Calorie: 288.0 cal
Beverage: Coke Price: 1.50$ Calorie: 189.0 cal
------------------------------
Total Price: 21.10$
Calories: 733.0 cal
==============================
```

Figure 5: Example output

5 Test and Execution

Your code must be compiled and executed under Java 8 and dev.cs.hacettepe.edu.tr. If your code does not compile and execute under developer server, then you will be graded as 0 for code part even if it works on your own machine. Your program must take input file, name of the spacecraft and total energy level of the spacecraft as an argument. Sample run command is as follows:

```
javac Main.java
java Main menu.txt orders.txt
```

6 Grading Policy

Task	Grade
Class Implementations using OOP Principles	50(7 per each, 7 full implementation, +1 bonus)
Usage of Collections	20
Correct Output	20*
Comments in JavaDoc Style	10**
Total	100

***Even though producing correct output and commenting seems like enough to get full credit, you must obey to the given rules in the PDF (for example using concept of OOP and four pillars of OOP etc.) otherwise you may face with some point deductions which may result with a grade that is as low as zero. There will be two overall multipliers about quality of your OOP and clean code separately which will vary in between 0 and 1! Note that there may be any other multipliers or point deductions in case of violation of the rules.**

**** The score of the clean code comment part will be multiplied by your overall score (excluding clean code comment part) and divided by the maximum score that can be taken from these parts. Say that you got 60 from all parts excluding clean code comment part and 10 from clean code comment part, your score for clean code comment part is going to be 10*(60/80) which is 7.5 and your overall score will be 60+7.5=67.5.**

7 Submit Format

File hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system).

```
-b<studentID>.zip
  -Main.java
  -⋆.java (optional)
```

Late Policy

You have two days for late submission. You will lose 10 points from maximum evaluation score for each day (your submitted study will be evaluated over 90 and 80 for each late submission day). You must submit your solution in at the most two days later than submission date, otherwise it will not be evaluated. Please do not e-mail to me even if you miss the deadline for a few seconds due to your own fault as it would be unfair for your friends, e-mail submissions will not be considered if you do not have a valid issue.

Notes and Restrictions

- Your code must be able to execute on our department's developer server (dev.cs.hacettepe.edu.tr).
- You must use JavaDoc commenting style for this project, and you must give brief information about the challenging parts of your code, do not over comment as it is against clean code approach. Design your comments so that if someone wants to read your code they should be able to easily understand what is going on. You can check here to access Oracle's own guide about JavaDoc Sytle.
- **Use inheritance, abstraction, and encapsulation where it makes logical and practical sense. Otherwise, you risk losing points for not properly applying OOP principles.**
- You must obey given submit hierarchy and get score (1 point) from the submit system.
- Do not miss the submission deadline.
- You can benefit from Internet sources for inspiration but do not use any code that does not belong to you.
- You can discuss high-level (design) problems with your friends but do not share any code or implementation with anybody.
- Source code readability is a great of importance. Thus, write READABLE SOURCE CODE, comments, and clear MAIN function. This expectation will be graded as "clean code".
- Use UNDERSTANDABLE names for your variables, classes, and functions regardless of the length. The names of classes, attributes and methods must obey to the Java naming convention. This expectation will be graded as "coding standards".
- You can ask your questions through course's Piazza group, and you are supposed to be aware of everything discussed in the Piazza group. General discussion of the problem is allowed, but **DO NOT SHARE** answers, algorithms, source codes and reports.
- All assignments must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.