CSS-105 Fundamentals of Programming
Project-2 (Individual, 5 points)
Deadline: December 9 (Wednesday) 23:59

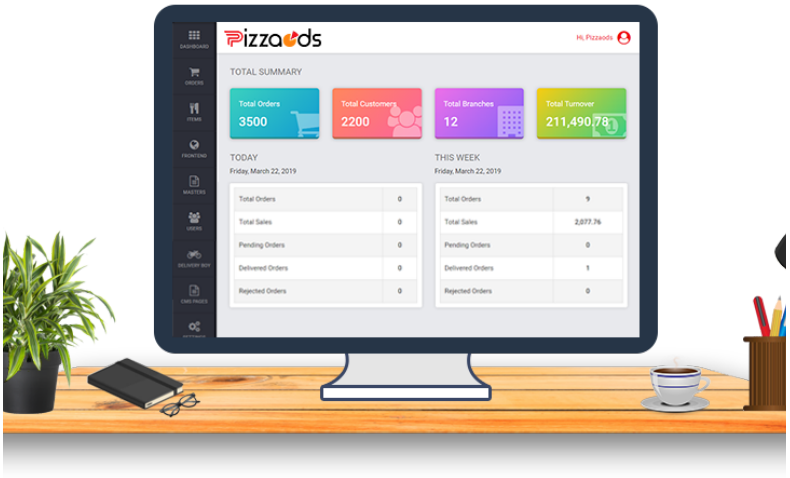
Introduction

The purpose of this project is to provide you with the opportunity to apply the latest topics that were covered in this course, including strings, arrays, sorting, and other fundamental principles of programming.

You should be able to complete the assignment within a few days. Please start early so that you have time to ask questions for clarification.

Project Requirements Overview

You will be continuing your work on pizza orders, but this time you will be implementing a program for the pizza cafe itself, that is, an application that will be processing the orders that are placed. The program shall again operate in the command-line, imitating a bot that will assist the pizza store manage the orders. The aim is to automate the process, save time, and ease the job of managers.



This time you will be writing your own program from scratch. It shall allow the user to do some useful operations on the data provided as input. The full list of operations and input details are given below, and sample input/output test cases attached at the end of this document.

Detailed Requirements

You should implement the following methods in your program:

1) **getTotalPrice**: calculates the total cost of all pizza orders;
2) **searchByID**: filters the given list by showing only the orders with a specific ID. If such an ID does not exist, displays “No result”;
3) **searchByDate**: filters the given list by showing only the orders that were placed at a specific date. If such a date does not exist, displays “No result”;
4) **sortByID**: sorts the orders by ID value in ascending order (smallest to biggest). The orders with the same ID should be kept in the same order as in the original list that is provided;
5) **sortByDateAndTime**: sorts the orders by Date value in ascending order (oldest to newest). If the dates are the same, then it should be sorted by Time. The orders with the same Time should be kept in the same order as in the original list that is provided;
6) **mostPopularSize**: determines the most frequent size based on the input list of orders. The value shall be one of: 20, 30, or 40. In case of multiple results, each should be displayed on a new line;
7) **mostPopularPizzaType**: determines the most popular pizza type based on the topping combinations. In case of multiple results, each should be displayed on a new line. See sample output for the correct format.

The input consists of a list of all orders placed with full information and the operation to be executed on this list (search, sort, etc.).

INPUT FORMAT
OP: operation to be executed, a number between 1-7, each corresponding to a method described above (Ex: 1-getTotalPrice, 2-searchByID, etc.). In case of search operations, an extra input field will be provided to indicate the search criteria (ID number or Date);
N: total number of orders
The next **N** lines each consisting of full information about order in the following format:
ID Date Time Price isBirthday Size Topping1 Topping2 Topping3 Topping4

ID: a four digit number in the range 0000 - 9999;
Date: order placement date in DD.MM.YYYY format;
Time: order placement time in HH:MM format;
Price: cost of pizza as an integer (Note: no constraints on number of digits);
isBirthday: indicates whether a client qualifies for the birthday discount. The value shall be either "Yes" or "No";
Size: size of pizza, either 20, 30, or 40;
Topping1: Cheese (Yes/No)
Topping2: Meet (Yes/No)
Topping3: Sausage (Yes/No)
Topping4: Vegetables (Yes/No)

Program Design
You must work on a single file called *PizzaOrdersManagement.java*. The program shall be written completely by you from scratch. You should implement the methods described above, using the topics discussed within our course (except classes). If you need help or just some clarification, you should ask in Moodle forum page, during office hours, or by email, at least.

Notes on Grading
There are 7 TASKS (methods) to be done in total, and each task has a different weight depending on its complexity. Make sure that your program works exactly as described in the project requirements. Do not attempt to submit someone else's work. It shall be considered as plagiarism, and get ZERO points. Partial completion of the tasks will be worth some points even if you do not finish the whole assignment.

Do NOT share your code with anyone. It would be considered academic dishonesty, and strictly penalized. All the works submitted shall be inspected by a special program and reviewed by the instructors. Any kind of similarity, or not being able to answer questions on the project gives the instructor full right to penalize the work, and even cancel the results that have already been graded because of cheating issues. In simple words, the fact that you did not cheat yourself does not matter. All the sides included in cheating (which is a crime) will be penalized. Again, do not share your code under any conditions.

Although you are strongly encouraged to ask questions and have discussions on Moodle, make sure you do not share your code there as well, please.

Turning In
1. Include all your work in a single **JAVA** file;
2. Turn in via **HackerRank** (invitations will be later sent to your SDU email addresses);
3. Your lab instructor may ask you to demonstrate your work and answer some questions.

Works that are not turned in correctly or are late will NOT be accepted.

Sample Run of the Program:

Input-1:

```
1
6
9999 09.12.2020 23:57 1600 No 30 Yes Yes Yes Yes
9999 09.12.2020 23:59 2200 No 40 Yes No Yes Yes
1234 01.01.2021 00:01 2000 Yes 40 Yes No Yes Yes
7245 01.12.2020 01:28 1400 No 30 Yes No Yes Yes
7777 09.12.2020 23:59 990 Yes 20 No No No No
7245 01.12.2020 01:27 2400 No 40 Yes Yes Yes Yes
```

Output-1:

```
Total Price:
10590
```

Input-2:

```
2 7245
6
9999 09.12.2020 23:57 1600 No 30 Yes Yes Yes Yes
9999 09.12.2020 23:59 2200 No 40 Yes No Yes Yes
1234 01.01.2021 00:01 2000 Yes 40 Yes No Yes Yes
7245 01.12.2020 01:28 1400 No 30 Yes No Yes Yes
7777 09.12.2020 23:59 990 Yes 20 No No No No
7245 01.12.2020 01:27 2400 No 40 Yes Yes Yes Yes
```

Output-2:

```
Search by ID: 7245
7245 01.12.2020 01:28 1400 No 30 Yes No Yes Yes
7245 01.12.2020 01:27 2400 No 40 Yes Yes Yes Yes
```

Input-3:

```
5
6
9999 09.12.2020 23:57 1600 No 30 Yes Yes Yes Yes
9999 09.12.2020 23:59 2200 No 40 Yes No Yes Yes
1234 01.01.2021 00:01 2000 Yes 40 Yes No Yes Yes
7245 01.12.2020 01:28 1400 No 30 Yes No Yes Yes
7777 09.12.2020 23:59 990 Yes 20 No No No No
7245 01.12.2020 01:27 2400 No 40 Yes Yes Yes Yes
```

Output-3:

```
Sort by date and time:
7245 01.12.2020 01:27 2400 No 40 Yes Yes Yes Yes
7245 01.12.2020 01:28 1400 No 30 Yes No Yes Yes
9999 09.12.2020 23:57 1600 No 30 Yes Yes Yes Yes
9999 09.12.2020 23:59 2200 No 40 Yes No Yes Yes
7777 09.12.2020 23:59 990 Yes 20 No No No No
1234 01.01.2021 00:01 2000 Yes 40 Yes No Yes Yes
```

Input-4:

```
7
6
9999 09.12.2020 23:57 1600 No 30 Yes Yes Yes Yes
9999 09.12.2020 23:59 2200 No 40 Yes No Yes Yes
1234 01.01.2021 00:01 2000 Yes 40 Yes No Yes Yes
7245 01.12.2020 01:28 1400 No 30 Yes No Yes Yes
7777 09.12.2020 23:59 990 Yes 20 No No No No
7245 01.12.2020 01:27 2400 No 40 Yes Yes Yes Yes
```

Output-4:

```
Most popular pizza type(s):
Cheese+Sausage+Vegetables
```