# Programming Assignment 1

**Submission Date :** 19.10.2023
**Due Date :** 02.11.2023 (23:00)
**Subject :** Online Book Marketing System
**Advisors :** R.A. Bahar GEZİCİ

# 1 Introduction

In this experiment, you are expected to gain knowledge of basic JAVA programming. The program you are going to develop will deal with variables, loops, string operations, and file read operations.

# 2 Problem

Create a Java program that simulates an Online Book Marketing System. The system should read a shopping list (purchaseOrder.txt) and a price list (priceCatalog.txt) from input files, consider product prices based on membership type (Premium, Standard, or Basic), shopping date, and validity dates, calculate the total cost of the items in the shopping list, and generate a bill reflecting the detailed pricing. For reading data from input files, you can use the code given in Appendix A. Your task is to calculate total cost of a shopping list based on the price list.

# 3 Requirements

## 3.1 Input Files:

1. **purchaseOrder.txt** : Contains the customer's name, surname, type of membership (Premium, Standard, or Basic), shopping date, and a list of book titles with quantities. Each line for the products is formatted as "Product Name [tab] Quantity," and it may contain one or more products in the shopping list of each customer. Shopping list file can contain maximum of 10 users. While *purchaseOrder.txt* file format is shown in Figure 1, you can find a sample input file for shopping lists for three customers in Figure 2.

> [customer name] *space* [surname] *tab* [type of membership] *tab* [shopping date] *tab* [product name] *tab* [quantity] *tab* [product name] *tab* [quantity] *newline* ...

Figure 1: The format of **purchaseOrder.txt** file

In this sample input file:

- John Doe is a Premium member who shopped on 26.01.2023.

- Jane Smith is a Standard member who shopped on 20.02.2023.

- Bob Johnson is a Basic member who shopped on 19.01.2023.

Figure 2: A sample for *purchaseOrder.txt* file

2. **priceCatalog.txt File**: Contains a list of book titles, type of membership, validity dates, and prices for each item. It also keeps the information about the expiry dates of the items. This file has the format as shown in Figure 3.



Figure 3: The format of *priceCatalog.txt* file

The price of books varies according to the membership type and shopping date. Even if the book is purchased by the same type of member, the price may differ when purchased at different times. Every line in the file has the product name, type of membership, start date, end date and price columns separated by a tab character. Same product can appear in the file multiple times. You should pay attention to the type of membership and validity dates of products. Same products can occur in the file several times with different prices. An example file is shown in Figure 4.



Figure 4: A sample for *priceCatalog.txt* file

This sample price list (Figure 4 file provides different prices for each book based on membership types and validity date ranges, as specified. The same book (e.g., Book A, Book B, Book C) is listed multiple times with the same membership type (e.g., Premium, Standard, Basic) but different prices for different date ranges. Each book's price varies based on the specified start and end dates. This allows you to test scenarios where the same book has different prices depending on when it is purchased by members with the same membership type.

### 3.2 Output

Once the shopping is completed and the right prices are calculated for every item in the shopping list, a bill is generated to reflect the total cost. The bills will be in the following format for each customer in the output screen as shown in Figure 5:

- Display the customer's name, type of membership, and shopping date.

- Display the shopping list, including product names, their corresponding prices, quantities, and the total cost of each item in the shopping list.

- Generate and display a detailed bill that includes the list of purchased items with quantities, detailed prices, and the total cost in the specified format.

**Notes**:

- The bill format includes each item's name, unit price, quantity, and the calculated amount.

- The total cost is displayed separately with the "Total Cost" label.

- Ensure that your program correctly extracts and displays the customer's name, type of membership (Premium, Standard, or Basic), and shopping date.

- Calculate the correct price for each item in the shopping list based on the provided detailed prices, considering membership type, shopping date, and validity dates.

## 4 Execution and Test

The input files are going to be given as **program arguments**. There will be text files (purchaseOrder.txt, priceCatalog.txt) in your working directory. The input files are going to be given as program arguments. "purchaseOrder" file is the **first argument** and the "priceCatalog" file is the **second argument**. In order to test your program, you should follow the following steps:

- Upload your java files to your server account (dev.cs.hacettepe.edu.tr)

- Compile your code (javac *.java)

- Run your program (**(java Main purchaseOrder.txt priceCatalog.txt)**)

- Control your output data (output.txt).

## 5 Submit Format

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

- ⟨*studentid*⟩.*zip*

     - src (Main.java, *.java)

## 6 Late Policy

You may use up to two extension days for the assignment. For each extension day, you will lose 10 points.

## 7 Grading Policy

| Task | Point |
|------|-------|
| Submitted | 1 |
| Compiled | 10 |
| Output | 89 |
| Total | 100 |

## 8 Notes

- The assignment must be original, individual work. Downloaded or modified source codes will be considered as cheating. Also the students who share their works will be punished in the same way.

- We will be using the Measure of Software Similarity (MOSS) to identify cases of possible plagiarism. Our department takes the act of plagiarism very seriously. Those caught plagiarizing (both originators and copiers) will be sanctioned.

- 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 .

- It is your duty to check the Piazza platform against any possible update about this assignment. If any instruction written by the TA violates any condition against this document, the new instruction(s) on Piazza is/are valid!

- Don't forget to write comments of your codes when necessary.

- The names of classes', attributes' and methods' should obey to Java naming convention.

- Save all work until the assignment is graded.

- Do not miss the deadline. Submission will be end at 02/11/2023, 23:00. The problem about submission after 23:00 will not be considered.

## 9 Appendix

```
------------Bill for Customer 1------------
Customer: John Doe
Membership Type: Premium
Date: 26.01.2023

Items Purchased:
Book A (Qty: 1) - 15.99 each
(Valid from 01.01.2023 to 31.01.2023)
Subtotal: 15.99
Book B (Qty: 2) - 13.39 each
(Valid from 01.01.2023 to 31.01.2023)
Subtotal: 26.78

Total Cost: 42.77

------------Bill for Customer 2------------
Customer: Jane Smith
Membership Type: Standard
Date: 20.02.2023

Items Purchased:
Book A (Qty: 2) - 17.99 each
(Valid from 01.02.2023 to 28.01.2023)
Subtotal: 35.98
Book B (Qty: 1) - 14.49 each
(Valid from 01.02.2023 to 28.02.2023)
Subtotal: 14.49

Total Cost: 50.47

------------Bill for Customer 3------------
Customer: Bob Johnson
Membership Type: Basic
Date: 19.01.2023

Items Purchased:
Book B (Qty: 2) - 15.10 each
(Valid from 01.01.2023 to 31.01.2023)
Subtotal: 30.20
Book A (Qty: 3) - 19.99 each
(Valid from 01.01.2023 to 31.01.2023)
Subtotal: 59.97

Total Cost: 90.17
```

Figure 5: A sample for *output.txt* file

```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class ReadFromFile {

    public static String[] readFile(String path) {
        try {
        int i =0;
        int length= Files.readAllLines(Paths.get(path)).size();
        String[] results = new String[length];
            for (String line : Files.readAllLines(Paths.get(path))) {
                results[i++]= line;
            }
        return results;
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

    public static void main(String[] args) {
        String[] lines= readFile("testfile.txt");
        for (String line : lines) {
            System.out.println(line);
            }
    }
}
```