United International University (UIU)

Department of CSE

Course Name: CSE 1116 | Object Oriented Programming Laboratory

Assignment - 2

Submission Guidelines:

Please Follow the guidelines below:

• Please solve the problems in the Java language and by creating appropriate project.

• For each question, create a separate project and Rename each project as Q1, Q2 etc….

• Zip all the INTELLIJ PROJECTs into a file. During zipping, you should name the folders the following way: **studentid name oop lab Section Assignment 2.zip**. If your ID is 12345 and your name is akil, the zip file should be "**12345_asif_oop lab_R_Assignment 2.zip**".

• **Don't submit .o or .exe file. Submit only the source file (.java)**

• **Deadline:** Check ELMS.
Please remember this is a strict deadline. Under no circumstances, this deadline will change. Failure to submit during the deadline will result in **penalty marks.**

• **DO NOT COPY from the internet, seniors, batchmates, or any other sources. Each of the assignments will be evaluated with a viva. You must be able to explain your code. Also, we will run a copy checker on the submissions. Any plagiarism will be severely penalized.**

• **DO NOT PUT the question in chatGPT and ask it to write the answer. If found out, there will be penalty.**

• For all the File Program, create .txt file as per requirement

Ques 1. Write a program for a large company to find the employees who are eligible for an increment. Do the following operations.

a) Take n number of employees' name, id, and last 6 months' performance scores as user input from keyboard.

b) Now calculate each employee's average score and write all the employee's information in a file named 'employee.txt'. Put each information in separate lines as follows-

Employee1 name
Employee1 id
Employee1 average score
Employee2 name
Employee2 id
Employee2 average score
……………..

c) Now read the average score of each employee from the 'employee.txt' file. Find out which employee scored more than 75 and write down their names on a separate file called 'increment.txt'.

Ques 2

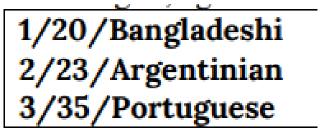
a) Suppose, you are given a string. Now create an ArrayList of characters containing each character of the string. Then change the third element(if it exists) of the ArrayList to 'z'. The program should then print out the contents of the list. **Write necessary codes.** [help: use set(index, element) function to set an element

b) You are given a class "Point" having two instance variables x and y. You are also given an ArrayList of type "Point". **Write necessary codes** to sort the ArrayList with respect to "x + y" in descending order using the "Collections.sort(<ArrayList>)" function.

Ques 3

You are given a file named “person.txt”. Each line of the file contains information about a person, containing id, age and nationality (divided by forward slash). The file looks like this: (**Create file named “person.txt” and insert the values for performing operation)**

person.txt

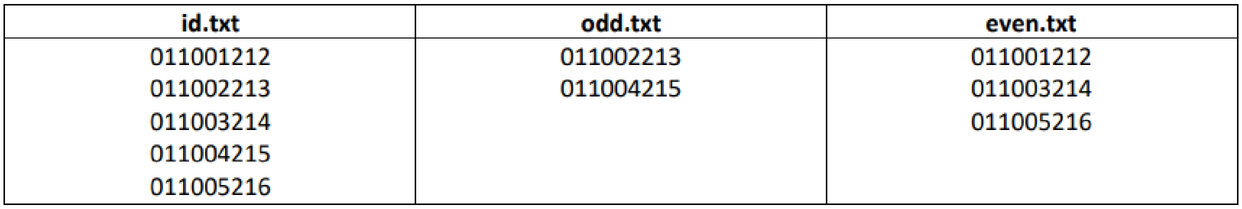


```
1/20/Bangladeshi
2/23/Argentinian
3/35/Portuguese
```

Write a function that will return the id of a person having maximum age (If there are multiple persons with maximum age, return any one of them).

Ques 4

Suppose you have a file “**id.txt”** that contains the ids of multiple UIU students. Write a java code to write the odd ids in the id.txt file to another file called **“odd.txt”** and the even ids in the id.txt file to another file called “**even.txt**”. Check the following example for clarification:



id.txt	odd.txt	even.txt
011001212 011002213 011003214 011004215 011005216	011002213 011004215	011001212 011003214 011005216

Ques 5

Create a file named "**e.txt.**" Take 5 strings as input from the user and write them to the "e.txt" file using **BufferedWriter**. Read the data from the "e.txt" file using **BufferedReader** and form a single sentence with words separated by a single space. Write this sentence to a single line in a new file named "**f.txt**" using **BufferedWriter.**

Ques 6

Write a program to calculate the profit for each product. The product list and corresponding information can be found in the given input file "**AnnualSell.txt**". Write output to "**Profit.txt**" file in the following format:

Item   Profit
...   .
...   .
Total Profit:
Max Profit Item

Exception Handling:

- Handle All file IO Exception
- If any of the price is less than zero then throw InvalidPriceException.
- If TotalUnitSold is less than Zero then throw TotalUnitSoldMnimumBoundException.
- If TotalUnitSold is greater than 1000 then throw TotalUnitSoldMaximumBoundException
- Create ncecessary custom Exception classes
- Close all file after the completion of operation

……………………………………………

AnnualSell.txt

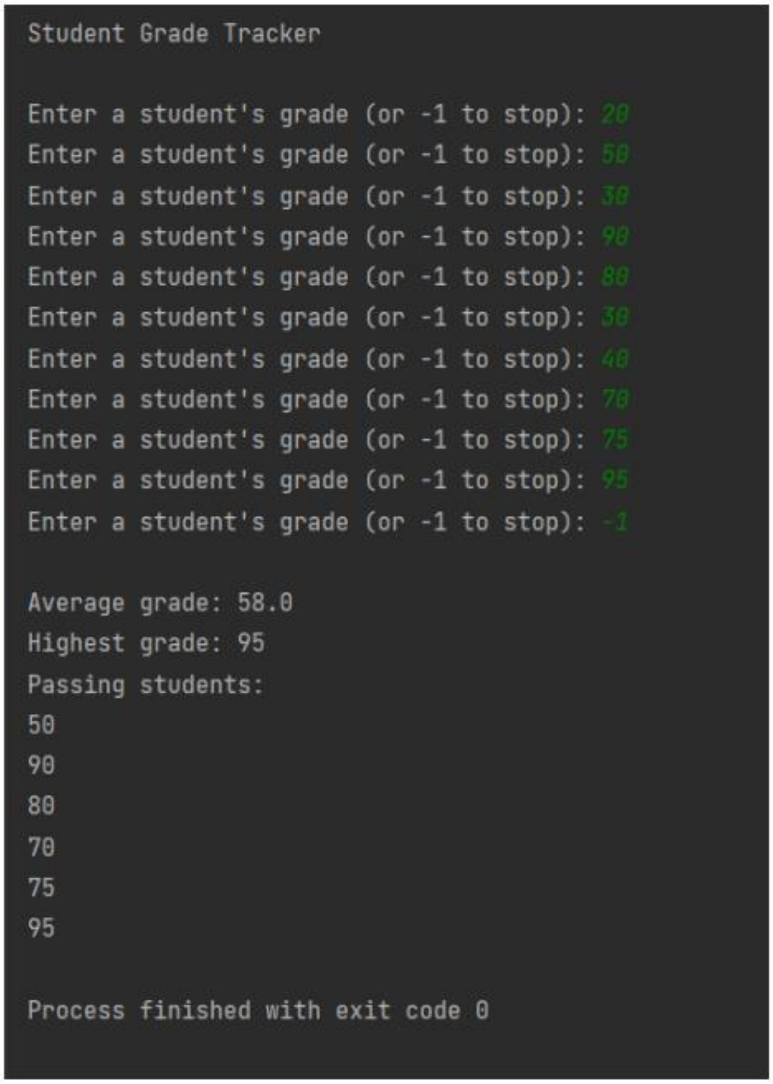
Item	UnitCost	UnitSellingPrice	TotalUnitSold
Pen	8	10	500
Paper	8.5	10	450
File	50	55	100
Java Book	250	-260	75
Mouse	300	298	65
Ruler	5.5	6	600
Pencil	7	7.5	45

Ques 7

You are tasked with creating a program to track the grades of students in a class. Implement a Java program that does the following:

- Define an **ArrayList** to store the grades of students (integer values).
- Write a method called **addGrade** that takes an integer (student's grade) as input and adds it to the ArrayList.
- Implement a method named **calculateAverage** that calculates and returns the average grade of all the students.
- Create a method called **highestGrade** that finds and returns the highest grade in the ArrayList.
- Design a function named **listPassingStudents** that prints the grades of students who have passed (grades >= 50).
- Write a main method to demonstrate the functionality of your program. Prompt the user to input grades until -1 is entered by the user, then display the average grade, highest grade, and list of passing students.

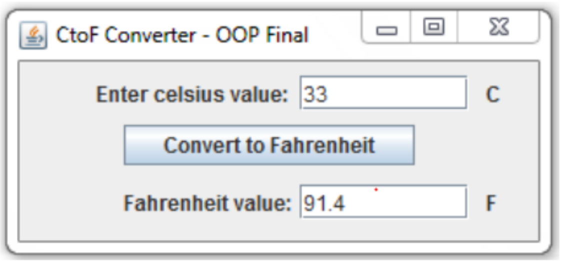
Sample input and output:



```
Student Grade Tracker

Enter a student's grade (or -1 to stop): 20
Enter a student's grade (or -1 to stop): 50
Enter a student's grade (or -1 to stop): 30
Enter a student's grade (or -1 to stop): 90
Enter a student's grade (or -1 to stop): 80
Enter a student's grade (or -1 to stop): 30
Enter a student's grade (or -1 to stop): 40
Enter a student's grade (or -1 to stop): 70
Enter a student's grade (or -1 to stop): 75
Enter a student's grade (or -1 to stop): 95
Enter a student's grade (or -1 to stop): -1

Average grade: 58.0
Highest grade: 95
Passing students:
50
90
80
70
75
95

Process finished with exit code 0
```

Ques 8

You are required to complete a Java GUI application that has the functionality of converting temperature from Celsius value to Fahrenheit value. The application takes Celsius value from one JTextField (namely textFieldCelsius), converts this value into Fahrenheit value on click of a JButton (namely btnConvert) and displays the Fahrenheit value into another JTextField(namely textFieldFahrenheit). Suppose necessary code for GUI creation is already provided. Formula for converting C to F: F= (C x (9/5)) + 32 .



Enter celsius value: 33 C

Fahrenheit value: 91.4 F

Ques 9

Create a Java Application named ***LibraryManager*** . Now write a class named ***Book*** which has the following **private** attributes – (title : String), (ISBN : String), (Author:String), (dateOfPublish:String) (dd-mm-yyyy).

Write another a class named ***MyLibrary*** that consists a **main** method. ***MyLibrary*** class also has the following members – (collection: Book[]) which is private instance variable.

There are two methods as well –

- ***<u>Public void addBook(Book b):</u>*** This method is called to add a new book to the collection array. If there is already a book with the same title and ISBN number this method throws **DuplicateItemException** with the message- "This book is already included in the collection." **DuplicateItemException** is a user defined exception that you need to create.
- ***<u>Public Book searchBook(String title):</u>***This method is used to search for a book in the collection with given title. If the title matches any book the whole Book object is returned. If no book is found this method throws a user defined **ItemNotFoundException** with the message – "The item being searched does not exist in the collection".

Inside the main method in ***MyLibrary*** write code to add these books using addBook().

1. To Kill a Mockingbird, 8123, Harper Lee, 05-11-1960
2. The Great Gatsby, 1212, F. Scott Fitzgerald, 26-04-1925
3. One Hundred Years of Solitude, 9280, Gabriel Márquez, 13-11-1967
4. The Great Gatsby, 1212, F. Scott Fitzgerald, 26-04-1925

Search for these books-

1. One Hundred Years of Solitude 2. Brave New World

** Use try-catch block inside main method to handle exceptions.

<u>***Write appropriate constructor and getter-setters for all classes</u>.

Ques 10.

Write a Java program to merge the information contents of two text files. The two text Files are “students.txt” and “marks.txt”. The contents of the “students.txt” are as follows(ID and name of one student in each line) and “ marks.txt”(Student’s name and their exam marks in each line)-

student.txt

```
1079  Sharif
1086  Labib
2078  Amir
……..
```

result.txt

```
Sharif  97
Labib   67
Amir    85
…..
```

merged.txt

```
1079    Sharif  97
1086    Labib   67
2078    Amir    85
```

Your application will read the two files and create the “merged.txt” file that contains all the information as shown in the figure. (Hint: Read one line from the two files at a time and combine them to create the corresponding line in merged.txt and write it down).

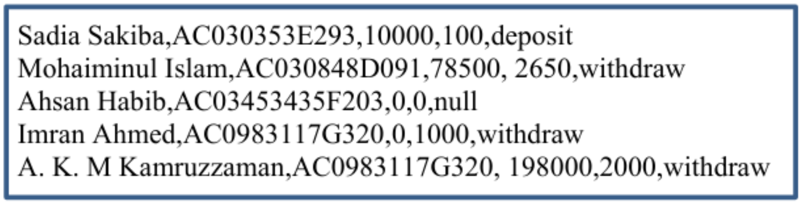
Ques 11. Create a Bank account management portal. You have to complete three tasks.

Task 1: The BankAccount class consists of the following attributes-

- (Account Name : String) , private
- (Account Number :String) , private
- (Balance: double), private
- (Last Transaction: double), private
- (Last transaction type: String), private

**** Use constructor, getter setters, toString() etc. where necessary.**

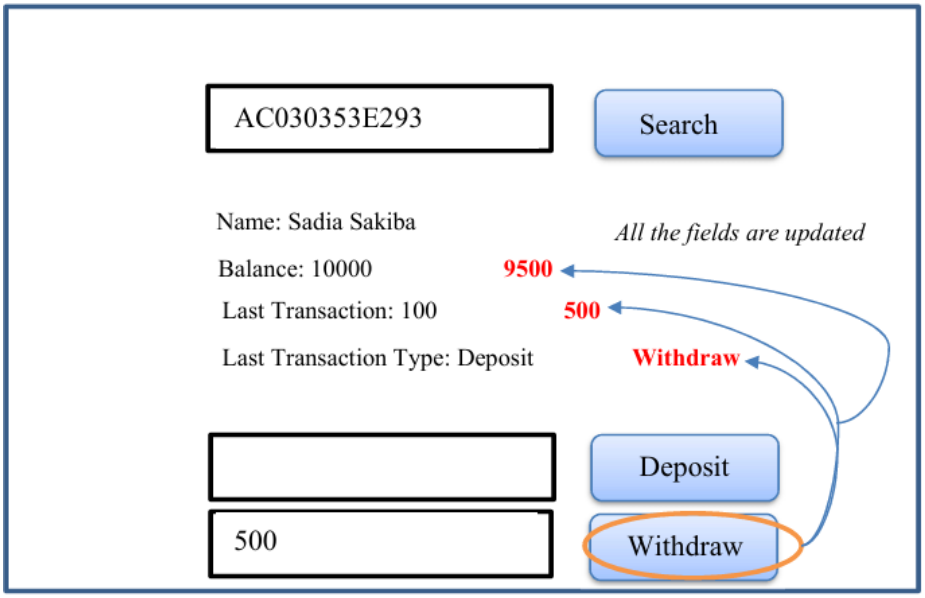
Task 2: Now from the BankAccountApp class create a main function and read from the file named **accounts.txt.** This file contains the information on the following bank accounts.



```
Sadia Sakiba,AC030353E293,10000,100,deposit
Mohaiminul Islam,AC030848D091,78500, 2650,withdraw
Ahsan Habib,AC03453435F203,0,0,null
Imran Ahmed,AC0983117G320,0,1000,withdraw
A. K. M Kamruzzaman,AC0983117G320, 198000,2000,withdraw
```

Create a **BankAccount** object with the information you get from each line and store them in an **ArrayList** of **BankAccount** objects name accounts. This is a private variable inside **BankAccountApp** class.

Task 3: Now **design a GUI portal** like the one below in which when we enter a person's Account Id it displays all of their information(by searching from the ArrayList). Also we can deposit and withdraw money from the account by putting some amount in the textfields and clicking the corresponding button. Update the balance, last transaction and last transaction type accordingly.



AC030353E293

Name: Sadia Sakiba

Balance: 10000 9500

Last Transaction: 100 500

Last Transaction Type: Deposit Withdraw

500

All the fields are updated