United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

***Final-term Exam* :: Trimester: Fall - 2019**

Course Code: CSE 1115 Course Title: Object Oriented Programming

Total Marks: **40** Duration: **2 hour**

NOTE: DO NOT BREAK THE SEQUENCE OF THE SUB-QUESTIONS.
For example, You can answer like this: 5.a, 5.b, 1.a, 1.b, 1.c…. but NOT like this: 1.a, 3.b, 4.a…

Question 1 [8 Marks]

Observe the code and answer the following questions :

```
class Player{
   int jerseyNo;
   double rating;

   public Player(int number, double rating){
       jerseyNo=number;
       this.rating=rating;
   }
}
```

```
public class finalExam {
   public static void main(String args[]){
       Player p1=new Player(7, 9.2);
       Player p2=new Player(9, 8.5);
       Player p3=new Player(10, 9.0);
       ArrayList<Player> list=new ArrayList<>();
       list.add(p1);
       list.add(p2);
       list.add(1,p3);
       list.add(2,new Player(5, 6.7));
       list.set(3, p3);
       //Create HashSet here for question (b)
   }
}
```

a) Write the **jerseyNo** and **rating** of each player in the given ArrayList list after executing the code. While writing, maintin the **order** of the players in the given ArrayList. [2]

b) Write a java code to create a **HashSet** naming **hSet** which contains the objects of the given list ArrayList after the given comment . Will the size of the created HashSet hSet be as same as the size of the given ArrayList list ? If no, explain why. [2]

c) Edit the Player class in the code so that **Collections.sort(list)** command will **sort** the 'list' ArrayList according to the **rating** of the player in **Descending** order. Only write the the updated Player class. [4]

Question 2 [8 Marks]

Suppose a file named "**employeeDirectory.txt**" contains the following lines :

```
100 Karim 10030
200 Rahim 50200
500 Mina 6500
1000 Sajib 11000
201 Rina 50000
```

You have to create a class named 'Employee' which has the following attributes:

- public int ID
- public String name
- public double salary

It also has a constructor which works as follows:

```
Employee(int a, String b, double c)
  { ID=a; name=b; salary= c;}
```

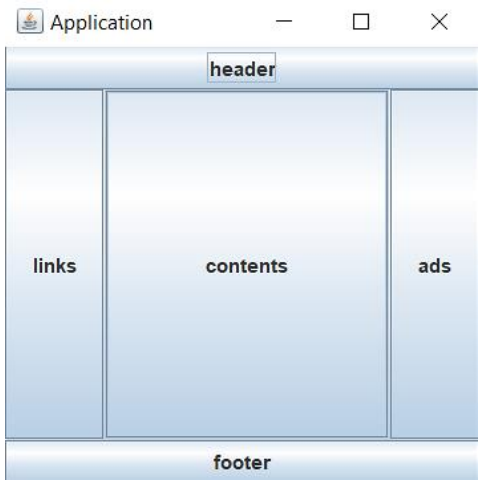
Each line of "**employeeDirectory.txt** " file contains all the information of an employee. For example, the employee whose ID is 1000, his name is Sajib and salary is 11000.

a) Now, you have to read all lines from the file and create an Arraylist of 5 **Employee** objects. [5]

b) You have to find the employee having the maximum salary and print his/her ID, name and salary to a file named, "**output.txt**". [Imagine both the files are in the current directory] [3]

Question 3 [8 Marks]

a) Write the code to create a Java GUI application like the given GUI, which organizes 5 buttons with **BorderLayout**. [4]



b) You are required to complete a Java GUI application like below which can **flip** or **rotate** an arrow-head. The "Flip" button flips the text horizontally. And the "Rotate" button rotates the arrow-head by 90 degrees clock-wise. You can call the "**flipMe**" and "**rotateMe**" functions to flip or rotate a string. [4]
(Do not need to write the whole code. Use the numbers in the comments and write that part of the code only.)

```
import javax.swing.*;
import java.awt.event.*;
class EditMe implements ActionListener{
    JButton flip, rotate;
    JTextField text;
    EditMe(){
        JFrame frame = new JFrame("EditMe");
        JPanel panel = new JPanel();
        frame.setContentPane(panel);
        frame.setSize(280, 150);
        frame.setLocation(300, 200);
        text = new JTextField(10);
        text.setText(">");
        flip = new JButton("Flip");
        rotate = new JButton("Rotate");
        panel.add(flip);
        panel.add(rotate);
        panel.add(text);
        /// 1. Add your code here

        frame.setVisible(true);
    }
    @Override
    public void actionPerformed(ActionEvent e)
    {
        /// 2. Add your code here
    }
    static String flipMe(String text) {
        if (text.equals(">")) return "<";
        if (text.equals("<")) return ">";
        return text;
    }

    static String rotateMe(String text){
        if(text.equals(">")) return "v";
        if(text.equals("v")) return "<";
        if(text.equals("<")) return "^";
        if(text.equals("^")) return ">";
        return text;
    }

    public static void main(String[] args) {
        new EditMe();
    }
}
```

Question 4 [8 Marks]

a) Complete the code in such a way that it produces the given output using anonymous inner class. [2]

```
class Student {
    public void study(){
        System.out.println("Student is studying");
    }
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.study();
        Student adnan = /* your code here */
        adnan.study();
    }
}
```

Output:

```
Student is studying
Adnan is studying
```

b) Observe the following code carefully. The server is sending some data to the client. **Write** the **constructor** of the **Client class** in such a way that it produces the given output. [2]

```
public class Server {
  public Server(int port){
    try {
      ServerSocket server = new ServerSocket(port);
      Socket socket = server.accept();
      DataOutputStream outputStream =
          new DataOutputStream(socket.getOutputStream());
      outputStream.writeUTF("Hello!");
      outputStream.writeUTF("This is Server");

      socket.close();
      outputStream.close();
    } catch (IOException e) {}
  }
  public static void main(String[] args) {
    Server server = new Server(4000);
  }
}
```

```
public class Client {
  public Client(String address, int port){
    try {
      // Socket socket =
      // DataInputStream inputStream =
      // Rest of the code
      socket.close();
      inputStream.close();
    } catch (IOException e) {}
  }
  public static void main(String[] args) {
    Client client = new Client("127.0.0.1",
      4000);
  }
}
```

Output of Client in the CONSOLE:

```
Hello!
This is Server
```

OR

Provide an alternative implementation with ReentrantLock to fulfill the purpose of the "synchronized" keyword in the following code segment. [2]

```
class Library {
    synchronized void accessDocuments() {
        //critical region starts
        authorizedAccess();
        //critical region ends
    }
///rest of the body.........
```

c) The **main method** of the following code contains some errors. **Fix the errors** and **rewrite** the **main method** (without comments). You can not **remove** any statement, you can only **update** or **add** statements. **Underline** the fixes that you have made. [4]

```
public class ThreadTest implements Runnable {
   public void run() {
     for (int i = 0; i < 20; i++) {
        System.out.println(
          Thread.currentThread().getName() + " " + i);
        try {
          Thread.sleep(30);
        } catch (InterruptedException e) {
           e.printStackTrace();
        }
     }
  }
```

```
  public static void main(String[] args) {
    // Create the threads
    Thread t1 = new ThreadTest("Thread 1");
    Thread t2 = new ThreadTest("Thread 2");

    // Start the threads
    t1.run();   t2.run();

    t1.combine();  t2.combine();

    // Following statement should execute
    // after all the threads are finished
    System.out.println("MAIN END");
  }
}
```

Question 5 [8 Marks]

a) Modify the "**StudentDemo**" **constructor**, so that it follows the following rules:

I. If the **age** value passed to StudentDemo constructor is **negative** or **greater than 100**, then a custom exception of class "**ValidationException**" will be thrown with the message "Invalid age value".
II. If the **name** value passed to StudentDemo constructor is empty, then a custom exception of class "**ValidationException**" will be thrown with the message "Name can not be empty".
III. If there is no error, the constructor should set the **age** and **name** instance variables.

```
class ValidationException extends Exception{
    ValidationException(String customExceptionName){
        super(customExceptionName);
    }
}
class StudentDemo{
  int age;
  String name;
  StudentDemo(String name , int age) throws ValidationException{
     // Write your code here
  }
}
```

Example :

```
1) new StudentDemo("", 23);
```
Throws **ValidationException** with message "Name can not be empty".

```
2) new StudentDemo("Rahim",-17);
```
Throws **ValidationException** with message "Invalid age value".

```
3) new StudentDemo("Karim",27);
```
No exception is thrown

b) Write the output of the following programs: [2+2]

i.
```
public class ExceptionOutput_1 {

  static int p_method(int x , int y){
    int div = 0 ;
    try{
      div = x / y ;
    }catch(NumberFormatException e){
      System.out.println("Catch inside p method");
    }
    return div;
  }

  static int q_method(int x , int y){
    int z = 0;
    try{
      z = p_method(x,y);
    }catch(NumberFormatException e){
      System.out.println("Catch inside q method");
    }
    return z;
  }

  public static void main(String[] args) {
    int a = 10 , b = 0;
    try{
      System.out.println("Exception created..");
      int i = q_method(a,b);
    }catch(ArithmeticException e){
      System.out.println("Exception: " +
          "Divide by 0");
    }
  }
}
```

ii.
```
public class ExceptionOutput_2 {
  public static void main(String[] args) {
    int[] arr = new int[10];

    try{
      System.out.println("Start Change");
      arr[10] = 11 / 0;
      System.out.println("Exception might be created");

    }catch(ArithmeticException e){
      System.out.println("Inner Catch 1");
    }catch(ArrayIndexOutOfBoundsException e){
      System.out.println("Inner Catch 2");
    }
    finally {
      System.out.println("Inside finally block");
    }
    System.out.println("End Change");
  }
}
```