BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
OBJECT-ORIENTED PROGRAMMING (CS F213)

Comprehensive Examination, Date: 09 Dec, 2023

PART-A: CLOSED BOOK: 40 M (90 Mins)   PART-B: OPEN BOOK: 40 M (90 Mins)

Timing: 9:00 AM - 12:00 Noon   First Semester, 2023-24   Max mark: 80

Instructions:

- Write Your **NAME** and **ID** clearly in each answer booklet. Write **Part-A** or **Part-B** at the top-right corner of answer booklet. Write END at the last page of answers in both booklets.
- Use Black/Blue Pen for writing or drawing Answers. Write the solutions of parts/sub-parts altogether of the same question number sequentially. Avoid pencils and other inks.
- Necessary assumptions should be made and mentioned in the answers in case of insufficient problem description, data, etc. provided in the questions. Maintain cleanliness of ans copy.
- Penalties would be as per AUGSD guidelines for violating examination rules.

Name: ID:

PART-A: Answer ALL The Questions

1. Write a method **removeDoubledLetters** that takes a string as argument and returns a new string with all doubled letters in the string replaced by a single letter. For example, if you call **removeDoubledLetters("tresidder")**, the method should return **"tresider"**. Similarly, if you call **removeDoubledLetters("bookkeeper")** your method should return **"bokeper"**. In writing your solution, keep in mind the following: (a) You need not to write a complete program. All you need is the definition of the method **removeDoubledLetters** that returns the desired result. (b) You may assume that all letters in the string are lower-case so that you don't have to worry about changes in capitalization. (c) You may assume that no letter appears more than twice in a row. **(8M)**

2. Precisely discuss the following: **(7M)**
(i) Comparable and Comparator Interface (Compare & Contrast).
(ii) A thread that sleeps by calling sleep and a thread that waits by calling wait (Compare & Contrast).
(iii) What if the main method is declared as private? – Compile Time OR Runtime Error and Why?
(iv) When sorting a collection of objects that implements the Comparable interface type, the sorting method compares and rearranges the objects. Explain the role of polymorphism in this situation. (v) Custom Exception; (vi) Serialization; (vii) Monitor (Multi-Thread).

3. In most word games, each letter in a word is scored according to its point value, which is inversely proportional to its frequency in English words. In Scrabble, the points are allocated as shown in Table 1. For example, the Scrabble word **FARM** is worth 9 points: 4 for the F, 1 each for the A and the R, and 3 for the M.

   Write a program that reads in words and prints out their score in Scrabble, not counting any of the other bonuses that occur in the game. You should ignore any characters other than uppercase letters in computing the score. Particularly, lowercase letters are assumed to represent blank tiles, which imply any letter, having a ZERO score.

Points	Letters
1	A, E, I, L, N, O, R, S, T, U
2	D, G
3	B, C, M, P
4	F, H, V, W, Y
5	K
8	J, X
10	Q, Z

Table 1: Scrabble Scoring Table

b) Write Short notes on Object Oriented Principles; and Method Overloading and Overriding. (**5+5+5=15M**)

4. *How do I love thee? Let me count the ways.*— **Elizabeth B. Browning, Sonnet 43, 1850**
Computers are very good at counting things. Write a method
private int countLove(String str); that takes a string as its argument and returns the number of times the word love appears in that string, ignoring differences in case, but making sure that love is not just part of a longer word like clover, glove, pullover, or slovenly. For example, if you were to call **countLove("Love in the clover.")**; your method should return 1. The word **Love** counts as a match because your method should ignore the fact that **Love** starts with an uppercase **L**. One possible use is illustrated by the following run method, which would count all the occurrences of love in an entire paragraph of text, ending with a blank line:

```java
public void run () {
    int count=0;
    while(true){
        String line=readLine();
        if(line.length()==0) break;
        count+=countLove(line);         }
        System.out.println("Love occurs: " +count + "times.");
}

/*  Input text: Shakespeares Sonnet 116

Let me not to the marriage of true minds
Admit impediments. Love is not love
Which alters when it alteration finds,
Or bends with the remover to remove.
O no! it is an ever-fixed mark
That looks on tempests and is never shaken;
It is the star to every wandering bark,
Whose worth's unknown, although his height be taken.
Love's not Time's fool, though rosy lips and cheeks
Within his bending sickle's compass come;
Love alters not with his brief hours and weeks,
But bears it out even to the edge of doom.
If this be error and upon me proved,
I never writ, nor no man ever loved.
*/
```

There are two matches in the second line, one in the ninth, and one in the eleventh. Note that Love's at the beginning of the ninth line counts because the apostrophe is not a letter, but that loved at the very end of the sonnet does not. The output is **Love occurs 4 times.** (**10M**)

All The Best

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

OBJECT-ORIENTED PROGRAMMING (CS F213)

Comprehensive Examination, Date: 09 Dec, 2023

PART-B: Answer ALL The Questions

The Instructions remain the same as specified in Part-A.

Name: ID:

---

1. You want to evaluate the arithmetic value of a given *Postfix* expression. A brief introduction is given about infix, and postfix. Infix notation is the common arithmetic formula/notation, in which operators are written infix-style between the operands e.g. 3 + 4 . In postfix, every operator follows all of its operands e.g. 3 4 + . Another example of a postfix is given whose arithmetic results can be derived using a Stack's **push**() and **pop**() operation which works as Last In First Out (LIFO) manner. Postfix: 3 6 2 + * 8 4 / - which can be converted into Infix: 3 * (6 + 2) - (8/4). Parenthesis are given for understanding unambiguous operator precedence. Now, the code snippets are given below as skeleton for writing your code in the specified blocks only, without altering others.

```java
public interface Expression  {
  public double compute(); }
```

```java
public class Number implements Expression{
    private final double n;
    public Number(double n){
        this.n = n;     }
    public double compute() {
        return n;     }
}
```

```java
public class Add implements Expression{
    private final Expression leftExpression, rightExpression;
    public Add(Expression leftExpression,Expression rightExpression ){
        this.leftExpression = leftExpression;
        this.rightExpression = rightExpression;     }
    public double compute() {
        return leftExpression.compute() + rightExpression.compute(); }
 }
```

```java
public class Substract implements Expression{
 //WRITE YOUR CODE IN THE ANSWER SCRIPT //
}
```

```java
public class Product implements Expression{
 //WRITE YOUR CODE IN THE ANSWER SCRIPT //
}
```

```java
public class Division implements Expression {
 //WRITE YOUR CODE IN THE ANSWER SCRIPT //
}
```

```java
public class ExpressionUtils {
    public static boolean isOperator(String s) {
        if (s.equals("+") || s.equals("-") || s.equals("*") || s.equals("
   /"))
            return true;
        else
```

```java
            return false;
    }

    public static Expression getOperator(String s, Expression left,
   Expression right) {
        switch (s) {
        case "+":
            return new Add(left, right);
        //WRITE YOUR CODE FOR OTHER CASES IN THE ANSWER SCRIPT //
        }
        return null;
    }
}
```

```java
import java.util.Stack;
public class ComputePattern {
    public static void main(String[] args) {
        String tokenString = "5 3 + 7 2 - /"; // POSTFIX EXPRESSION //
        Stack<Expression> stack = new Stack<>();
        String[] tokenArray = tokenString.split(" ");
        for (String s : tokenArray) {
            if (ExpressionUtils.isOperator(s)) {
                Expression rightExpression = stack.pop();
                Expression leftExpression = stack.pop();
                Expression operator = ExpressionUtils.getOperator(s,
   leftExpression,rightExpression);
                double result = operator.compute();
                stack.push(new Number(result));
            } else {
                Expression i = new Number(Integer.parseInt(s));
                stack.push(i);   }
        }
        System.out.println(//WRITE YOUR CODE IN THE ANSWER SCRIPT TO
   DISPLAY THE RESULT //);
     }
}
```

Fill the missing code sections with correct code so that the overall program executes perfectly (8M). Draw detailed class diagrams for implementing the design pattern (6M). Which design pattern have you used to solve the problem (0.5M) ? What will be the expected output (0.5M)? (**15M**)

2.
```java
//Assume required Libraies are imported //

class EvenOddThread extends Thread{
Integer start;
Integer N;
Printer print;
Boolean isEven;
EvenOddThread(Integer start, Integer N, Printer print , Boolean isEven){
//WRITE YOUR CODE IN THE ANSWER SCRIPT //
}

public void run(){
while(start <= N){
if(isEven)
print.evenPrinter(start);
else
print.oddPrinter(start);
start = //WRITE YOUR CODE IN THE ANSWER SCRIPT //;
}
}
}

class Printer {
public volatile boolean isOdd = false;
public synchronized void evenPrinter(int num){
if(!isOdd){
try {
           wait();
          } catch (InterruptedException e) {Thread.currentThread().
   interrupt(); }
}
System.out.println("Even " + num);
isOdd = false ;
notify();
}

public synchronized void oddPrinter(int num){
//WRITE YOUR CODE IN THE ANSWER SCRIPT //
}

public class CompreTest {
public static void main (String[] args) throws java.lang.Exception {
Printer printerObj = new Printer();
EvenOddThread t1 = new EvenOddThread(41,60,printerObj,false); // Odd Td
EvenOddThread t2 = new EvenOddThread(42,60,printerObj,true);  // Even Td
System.out.println("Results.....");
//WRITE YOUR CODE IN THE ANSWER SCRIPT //
}
}
```

This coding problem pertaining to display Even-Odd values only one time within a given range using Multi-Threading. Fill the code snippets as instructed (8M). Can you suggest an alternative implementation using a suitable design pattern? If yes, draw a detailed class diagram of your answer (6M). What is the expected Output? (1M). **(15M)**

3. Identify a suitable design pattern and write a class-level implementation of the Fig. 1. Analyze your answer with SOLID principles and/or anti-patterns, if any. **(1+7+2=10M)**

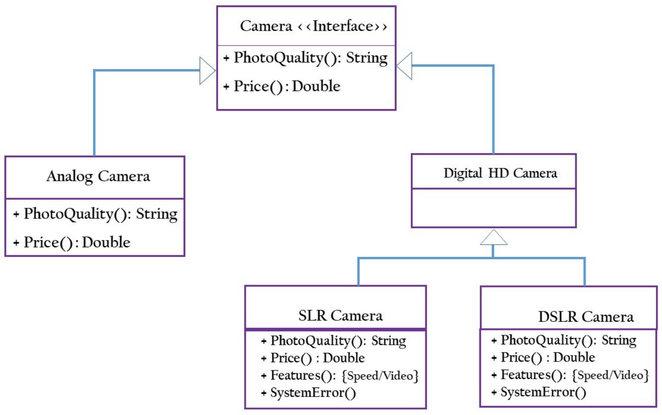

Figure 1: Implement the given problem in Q.3 using Java

All The Best