Final Exam Preperation

(Handwritten Assignment)

IT24039

Ishtiak Ahmed Ayon

IT24039

Q1 Write a Java program that reads a series of numbers from a file (input.txt), determines ~~the highest~~ each number in the series and calculates the sum of natural numbers up to that number and writes the result to another file (out.put.txt) Use Scanner to read the file and Printwriter to write to the file.

~~sample~~ input. txt:

10, 55, 1000, ... 100

Output. txt

55, 1540, 500500, ... 5050

Answer:

Java code:

```
import java.io.File;
import java.io.Printwriter;
import java.io.scanner;

public class SumOfNaturalnum{
```

(P.t.o)

IT24039

```
public static void main(String[] args){
    try{
        Scanner filescanner = new scanner(
                    new File("input.txt"));
        String content = filescanner.nextline
        Filescanner.close();
        String[] numbers = content.split(",");
        Printwriter writer = new Printwriter(
                    "output.txt");

        for(int i=0 ; i<numbers.length; i++)
        {
            int n=Integer.parseInt(numbers[i]
                        .trim());
            long sum=(long)(n*(n+1))/2;
            writer.print(sum);
            if (i< numbers.length-1){
                writer.print(",");
            }
        }
```

IT24039

```
    Writerr.close;
    System.out.println ("Proocessing done");

 } catch (Exception e){

        System.out.println ("Errror:" + e.getMessage());

   }
  }
}
```

IT24039.

Q2|

Answer:

Differences between static and final keyword in Java.

1. static:

- ☐ Belongs to the class, not to objects
- ☐ Only one copy shared by all objects
- ☐ Can be accessed using class name
- ☐ static methods can not access non-static members directly

2. final:

- ☐ Used to restrict modification
- ☐ Final field: value can not be changed after initialization
- ☐ Final method: can not be overridden by subclasses
- ☐ Final class: can not be inherited

IT24039

Static vs Final :

Aspect	static	final
Purpose	Shared at class level	Prevents change
Field	One copy for all objects	Constant value
Method	Belongs to class.	Cannot be overridden
Inheritance	Allowed	Restricted

Accessing static method using object:

```
class   Test {
        static int  x = 10;
        static  void  show(){
            System.out.println("static method");
        }
}
```

IT24039

```
public class Main {

    public static void main(String[] args){

        Test t = new Test();

        System.outprintln (t.x);
         t.show();

        System.out.println (Test.x);

        Test.show();

    }

}
```

IT24103?

Question - 3

All factorian numbers

FactorianRange.Java

```
import java.util.scanner;

public class FactorianRange{

    static int factorial (int n){
        int fact =1;
        for (int i=1; i<=n; i++){
            fact *= i;
        }
        Return fact;
    }

    static boolean isFactorian (int num){
        int temp = num;
        int sum = 0;

        while(temp>0){
            int digit = temp%10;
            sum += factorial (digit);
            temp /= 10;
        }
        return sum== num;
    }
```

IT24039

```
public static void main(String[] args){
        Scanner sc = new Scanner(system.in);

        System.out.print("Enter lower bound");
        int lower = sc.nextInt();

        System.out.println("Factorian numbers
        in the given range : ");

        for(int i=lower; i<=upper; i++){
            if (isFactorian(i)){
                System.out.println(i);
            }
        }

        sc.close();
    }

}
```

IT24039

Q-4

Differences among Class, Instance and Local variables:

1) Class variable (static variable):

- ☐ Declared with static keyword inside a class
- ☐ Shared by all objects of class
- ☐ Memory allocated once at class loading time

2) Instance variable:

- ☐ Declared inside a class but outside methods without static
- ☐ Each object has its own copy
- ☐ Memory Allocated when an object is created
- ☐ Accessed using object reference

3) Local variable:

- ☐ Declared inside a method, constructor, or block
- ☐ scope limited to that block only

□ no default value

□ memory allocated during method execution

Comparison Table:

Feature	Class Variable	Instance	Local
Keyword	static	None	None
Scope	Whole class	Object specific	Block/method
Memory	Once	Per object	During Execution
Default value	Yes	Yes	No

Significance of 'this' keyword:

□ refers to current object

□ resolves name conflict between instance variables and parameters

□ Helps pass current object as parameter

IT24039

Q-5| Sum of elements in an Integer Array

ArraySum.Java

```
public class ArraySum {

    static int calculateSum(int[] arr){
        int sum = 0;
        for (int i : arr) {
            sum += i;
        }
        return sum;
    }

    public static void main (String[] args){
        int[] numbers = {10, 20, 30, 40, 50}

        int result = calculateSum(numbers);

        System.out.println(" Sum of array element
                       = " + result);

    }

}
```

Q-6

Access Modifier:

An access modifier defines the visibility and accessibility of classes, methods, variables and constructors in Java. It controls where a member can be accessed from.

Accessibity comparison of Access Modifiers

Modifier / Feature	public	protected	private
Same class	yes	yes	yess
Same package	Yes	Yes	No
Subclass (Different package)	Yes	Yes	No
Other classes	Yes	No	No

IT24039

Different types of variables in Java (with example):

1. Instance Variable:

□ Declared inside a class but outside methods

□ each object has its own copy

Example:

```
class Student {
    int age;
}
```

2. Class variable (Static variable):

□ Declared using static

□ Shared by all objects

Example:

```
class Student {
    static String college = "MBSTU";
}
```

3. Local variable:

□ Declared inside a method or block

□ exists only within scope

Example:

```
void show() {
    int x = 10;
}
```

IT24035

Q-7 | Smallest positive root of a quadratic equation : $ax^2 + bx + c = 0$

Smallest Positive Root.java :

```
import   java.util.Scanner;

public  class  SmallestPositiveRoot {

    public static void main (String[] args) {
        Scanner sc = new Scanner (System.in);
        System.out.print ("Enter coefficients a, b,
              and c : ");
        int a = sc.nextInt ();
        int b = sc.nextInt();
        int c = sc. nextInt ();
        double discriminant = b* b - 4* a * c;

        if (discriminant <0) {
           system.out.println ("No real roots");
        } else {
           double root1 = (-b + Math.sqrt (
                  discriminant)) / (2.0 * a);
```

IT24039

```
double root2 =(-b - Math.sqrt (discriminant)) /
                    (2.0 * a);

double smallestroot = Double.MAX_VALUE;

if (root1>0)
    smallestroot = root1;
if (root2 >0)
    smallestroot = Math.min (smallestroot,
                               root2);

if (smallest root == Double.MAX_VALUE){
    System.out.println ("No positive value");
}else{
    System.out.println ("The smallest positive
        root is : " + smallestroot );

}
}

sc.close();
}

}
```

Q-8

CharacterCheck.java

```
public class CharacterCheck{
    static void checkCharacter(char ch){
        if(Character.isLetter(ch)){
            System.out.println("It is a letter");
        }else if(Character.isDigit(ch)){
            System.out.println("It is a digit");
        }else if(Character.isWhitespace(ch)){
            System.out.println("It is a whitespace");
        }else {
            System.out.println("Other character");
        }

    public static void main(String[] args){
        char ch = 'A';
        checkcharacter(ch);
    }
}
```

IT 24039

Passing array ~~to~~ a function in Java)

In ~~j~~ Java, an array is passed to a function by reference. This allows the method to access and modify the original array.

Example:

```
public class ArrayPassIng {
        static void displayArray (int[] arr){
           for (int x : arr){
               System.out.print (x + "  ");
           }
        }

        public static void main (String[] args){
            int[] numbers= {10, 20, 30, 40, 50};
            displayArray (numbers;

        }

~~}~~ }
```