Final Exam Preparation

Ans. 01

```
import java.io.File;
import java.io.PrintWriter;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class SumOfNaturalNumber {
    public static void main(String[] args) {
try {
    Scanner sc = new Scanner(new File("input.txt"));
        String line = sc.nextLine();
    String[] numbers = line.split(";");
        int maxNumber = Integer.MIN_VALUE;
for (String number : numbers) {
```

```
        if (currentNumber > maxNumber){
            maxNumber = currentNumber;
        }
    }

long sum = (long) maxNumber * (maxNumber+1) /2;

PrintWriter writer = new PrintWriter("output.txt");
    writer.println(sum);
    writer.close();

}   catch (FileNotFoundException){
        system.out.println("File not found" + e.getMessage);
    }
}

}
```

Ans. 02

Key Differences:

1. Static:

→ Shared by all instances of the class

→ Accessed via class name or object

2. Final:

→ Cannot be modified or overriden.

→ Must be initialized once

Example:

```
Class MyClass {
    static int staticField = 10;
  Static int staticField = 10;
static void staticMethod () {
        system.out.println ("Static Method");
```

```
    }
}

public class Main {
    public static void main(String[] args) {
        MyClass obj = new MyClass();
        obj.staticMethod();
        System.out.println(obj.staticField);
    }
}
```

Ans. 05

```
public class ArraySum {

    public static int sumOfArray(int[] arr){

        int sum = 0;

        for (int num: arr) {

            sum += num;

        }

        return sum;

    }

    public static void main(String[] args){

        int[] numbers = {1,2,3,4,5};

        int result = sumOfArray(numbers);

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

    }

}
```

Ans. 06

⊞ Access Modifier: It can control the visibility of classes, methods and variables in java. They define where these elements can be accessed from in the program.

⊞ Different types of Access Modifiers:

1. Public - Accessible from anywhere
   Ex: public class MyClass {};
2. Private - Accessible only where it declared
   Ex: private int age;
3. Protected - Accessible within the same package and by the sub-class.
   Ex: protected int salary
4. Default - with only the same package.
   Ex: Int id;

Ans. 07 :

```
import java.util.Scanner;

public class QE Solver{

    public static void main(String[] args){

Scanner sc = new Scanner(System.in);

  int a = sc.nextInt();

  int b = sc.nextInt();

  int c = sc.nextInt();

double discriminant = b*b - 4*a*c;

    if(discriminant < 0){

        System.out.print("No real roots");

    }

    else{

      double root1 = (-b + Math.sqrt(discriminant))/(2*a);

      double root2 = (-b - Math.sqrt(discriminant))/(2*a);
```

```
double smallestPositiveRoot = Double.MAX_VALUE;

if(root1 > 0 && root1 < smallestPositiveRoot)
    smallestPositiveRoot = root1;

if(root2 > 0 && root2 < smallestPositiveRoot)
    smallestPositiveRoot = root2;

if(smallestPositiveRoot >= Double.MAX_VALUE) {
    System.out.println("No real roots");
}

else {
    System.out.println("Smallest positive root" + );
}

}

}
```

Ans. 09

Method Overriding: Method overriding is a feature in java that allows a subclass to provide a new implementation for a method that is already define in its superclass.

What happens when a sub-class over-rides a method:-

1. Subclass method executes, replacing the superclass method.
2. Runtime polymorphism determines method execution at runtime.
3. Superclass method is hidden unless called using `super`.
4. Overriding rules apply.
5. `super` can call the overridden superclass method.

⊛ Potential issue when overriding methods:

1. Visibility restriction
2. Exception Limitation
3. final & static methods.

⊛ Issues with constructors:

1. Construction can not be overriden because they are not inherites.
2. `super()` must be used for super class initialization.

Ans. 10

Static members	Non-Static members
1. Belong to the class and are shared among all obj.	1. Belong to the individual obj. each instance has its own copy.
2. Accessed using class name of instance	2. Access only through an obj of the class
3. Stored in the method area	3. Stored in the heap memory
4. Used for constants utility methods and shared properties.	4. Used for object specific behaviour and instance data.

Ans. 12

```
import java.util.Scanner;

class BaseClass {
    void printResult(String msg, Object result) {
        System.out.println(msg + result);
    }
}

class SumClass extends BaseClass {
    double sumSeries() {
        double sum = 0.0;
        for (double i = 1.0; i >= 0.1; i -= 0.1) {
            sum += i;
        }
        return sum;
    }
}
```

```
class DivisonMultipleClass extends BaseClass {

    int gcd(int a, int b) {

        while (b!=0) {

    int temp = b;

        b = a%b;
        a = temp;
    }

    return a;
}

class NumberConversion extends BaseClass {

    string toBinary(int num) {

        return IntegerToBinaryString(num);

    }
    string toHex(int num) {

        return Integer.toHexString(num);

    }
    string toOctal(int num) {

        return IntegerToOctalString(num);

    }
}
```