JT23001

1.

```
import java.io.*;
import java.util.*;

public class main{

    public static void main (String[] args) throws IOException{

        Scanner sc = new Scanner(new File("input.txt"));
        String[] nums = sc.nextLine().split(", ");
        PrintWriter pw = new PrintWriter(new File("output.txt"));

        for (String num: nums) {
            int n = Integer.parseInt(num);
            int sum = n*(n+1)/2;
            pw.print(sum + ", ");
        }
        pw.close();
    }
}
```

IT23001

21

static is class-level, shared across instances
final is immutable and non-overridable.
static is accessed by classname and final can be accessed normally.

Accessing static with object instead of class works but not recommended.

Example –

```
class Example {
    static int staticVar = 10;
    final int finalVar = 20;

    static void staticMethod() { System.out.println("static method"); }
    final void finalMethod() { System.out.println("final method"); }
}
public
public class main {
    public static void main(String[] args) {
        Example obj = new Example();
        System.out.println(obj.staticVar);
    }
}
```

JT23001

3]

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter lower bound: ");
        int low = sc.nextInt();
        System.out.print("Enter upper bound: ");
        int high = sc.nextInt();
        for (int i = low; i <= high; i++) {
            if (isFactorion(i)) System.out.print(i + ", ");
        }
    }
    static boolean isFactorion(int n) {
        int sum = 0, temp = n;
        while (temp > 0) {
            sum += factorial(temp % 10);
            temp /= 10;
        }
        return sum == n;
    }
    static int factorial(int n) {
        int res = 1; for(int i = 1; i <= n; i++) res *= i; return res;
    }
}
```

I 123001

4]

Class var is shared by all instances

Instance var is unique to each instance

Local var is declared inside methods, not accessible outside.

this keyword refers to the current instance.

5]

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        System.out.println(sum(arr));
    }

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

IT23001

c]

Access modifier is a keyword used to control accessibility of class, method or variable.

Acess modifiers

Public: accessible anywhere

Private: Accessible only within class

Protected: Accessible within the same package or subclass.

2]

```
import java.util.*;

public class Main {

    public static void main (String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter co-efficients a, b and c: ");

        int a = sc.nextInt(), b = sc.nextInt(), 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*a);

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

1) 2300

```
if (root1 > 0 && root2 > 0)
    System.out.println("The smallest positive root is: " + Math.min(root1, root2));
else
    System.out.println("No real roots.");
}
}
```

8)

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a character: ");
        char ch = sc.next().charAt(0);
        if (Character.isLetter(ch)) System.out.println("Letter");
        else if (Character.isWhitespace(ch)) System.out.println("Whitespace");
        else if (Character.isDigit(ch)) System.out.println("Digit");
    }
}
```

IT2305

9]

Overriding: A subclass provides its specific implementation of a method defined in the superclass.

super is used to call the superclass methods.

Constructors cannot be overridden, and calling a superclass constructor may lead to unintended behaviour.

10]

Static: Belongs to the class, shared among instances.

Non static: Belongs to individual objects, requires an instance to access.

~~Example~~

Palindrome checker

```
import java.util.*;
public class Prem{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();
        System.out.println(s.equals(new StringBuilder(s).reverse().toString()) ?
        s + "is a palindrome." : s + "is not a palindrome.");
    }
}
```