Java Assigment-2

1. What are classes and objects in Java?

A class in Java is a blueprint used to create objects and defines their variables and methods.
It helps organize code and represent real-world concepts in a program.
An object is an instance of a class that contains actual data and can use the class methods.
Each object has its own memory and represents a real entity in the program.

2. What is the difference between primitive and reference data types?

Primitive data types store simple values directly in memory, while reference data types store the address of an object, not the actual data.
Primitive types are predefined in Java (int, char, boolean, etc.), whereas reference types are created by the user (classes, arrays, objects).
Primitive variables hold actual data, but reference variables point to the location where the object is stored.

3. What are access modifiers in Java?

Access modifiers in Java are keywords that control the visibility of classes, variables, methods, and constructors.

They help restrict access to data and protect the internal details of a program. Java provides four types: public, private, protected, and default.

4. Explain the concept of encapsulation

Encapsulation in Java is the concept of wrapping data and methods together into a single unit, called a class.
It is used to hide the internal details of how an object works from the outside world.
The main goal of encapsulation is to protect data from being accessed directly.
This is usually done by making variables private inside a class.
Public getter and setter methods are then used to read and update those variables.
Encapsulation helps in controlling how data is modified.
It increases security by preventing unauthorized access to important data.
It also improves code maintainability because changes inside the class do not affect other parts of the program.

5. What is inheritance and why is it used?

Inheritance in Java is the concept where one class (child) acquires the properties and methods of another class (parent).
It is used to **reuse existing code** instead of writing it again.
Inheritance creates a relationship between classes, allowing the child class to extend or modify parent behavior.
It supports **method overriding**, which enables dynamic polymorphism and flexibility in programs.
Overall, inheritance makes code easier to maintain, reduces duplication, and models real-world relationships in object-oriented programming.

1.Write a program to check if a number is positive, negative, or zero.

```
1.   import java.util.Scanner;

            public class PosNegZero {

                public static void main(String[] args){

                    Scanner sc=new Scanner(System.in);

                    System.out.println("Enter a Number: ");
                    int a=sc.nextInt();

                    if (a>0) {
                        System.out.println("No is positive !!");
                    }else if (a<0){
                        System.out.println("No is negative !!");
                    }else {
                        System.out.println("No is zero !!");
                    }

                }
            }
```

Output:- Enter a Number:

12

No is positive !!

Enter a Number:

-96

No is negative !!

Enter a Number:

0

No is zero !!

2.Write a program to swap two numbers without using a third variable.

```
import java.util.Scanner;

public class SwapTwoNo {

    public static void main(String[] args){

        Scanner sc=new Scanner(System.in);

        System.out.println("Enter 2 no: ");
        int a=sc.nextInt();
        int b=sc.nextInt();

        System.out.println("Before Swap: "+a+" "+b);
```

```
        a=a+b;
        b=a-b;
        a=a-b;

        System.out.println("After swap: "+a+" "+b);

    }
}
```

Output:-

Enter 2 no:

10 2

Before Swap: 10 2

After swap: 2 10

3.Write a program to find the factorial of a number using loop.

```
import java.util.Scanner;

public class Factorial{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter n value:");
        int n = sc.nextInt();

        int fact = 1;

        for (int i = n; i >= 1; i--) {
            fact = fact * i;
        }

        System.out.println(fact);
    }
}
```

Output:-

Enter n value:

5

120

4.Write a program to print Fibonacci series up to n terms.

```
import java.util.Scanner;

public class FibonacciSeries {

    public static void main(String[] args){
```

```
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter n value: ");
        int n =sc.nextInt();
        int a = 0, b = 1;

        for (int i = 1; i <= n; i++) {
            System.out.print(a + " ");
            int next = a + b;
            a = b;
            b = next;
        }
    }
}
```

Output:-

Enter n value:

20

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181

5.Write a program to reverse a number.

```
import java.util.Scanner;

class ReverseNo {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter num:");
        int num = sc.nextInt();
        int rev = 0;

        while (num > 0) {
            int digit = num % 10;
            rev = rev * 10 + digit;
            num = num / 10;
        }

        System.out.println("Reverse = " + rev);
    }
}
```

Output:-

Enter num:

123

Reverse = 321