Problem Solving using Java

Statement: Check whether the given number is even or odd.

Code:

```
import java.util.Scanner

Class Test {
    public static void main(String args[]){
        System.out.println("Enter a number:");
        Scanner sc = new Scanner(System.in);
        int num = Integer.parseInt(sc.nextLine());
        int x = num % 2 ;
        if (x==0){
                SOPl ("The number is even");
        }
        else{
                SOPl ("The number is odd");
        }
    }
}
```

N.B: System.out.println = SOPl

But, When you will write code that time must use this

```
public class Test{
    public static void main(String args[]){
        int x = Integer.parseInt("9");
        double c = Double.parseDouble("5");
        int b = Integer.parseInt("444",16);
        SOP(x)
        SOP(c)
        SOP(b)

        Scanner sc= new Scanner(System.in);
        int a = sc.nextInt

        double rz = Float.parseFloat(sc.nextline());
    }
}
```

Output

9
5.0
1092

Input - output

For char

```
Scanner sc = new Scanner(System.in);
char c = sc.next().charAt(0);
SOP.out.println(c);
```

For String

```
String st = sc.nextLine();
SOP(st);
```

For Numerical Input

```
int num = sc.nextInt();
double salary = sc.nextDouble();
float num2 = sc.nextFloat();
```

Classes & Objects in Java

Class

A Class in java is a set of objects which shares Common characteristics/behavior and common properties. It is user-defined blueprint or prototype from which objects are created.

For example, Student is a class while a particular Student named Adnan is an object.

* Class Declaration in Java

```
access_modifier Class <Class name>
{
        data members ;
        method ;
        Constructor ;
        nested Class ;
        interface ;
}
```

Example:

```
public class Sample"
{   int id = 9 ;
    String batch = "59 batch" ;
}
```

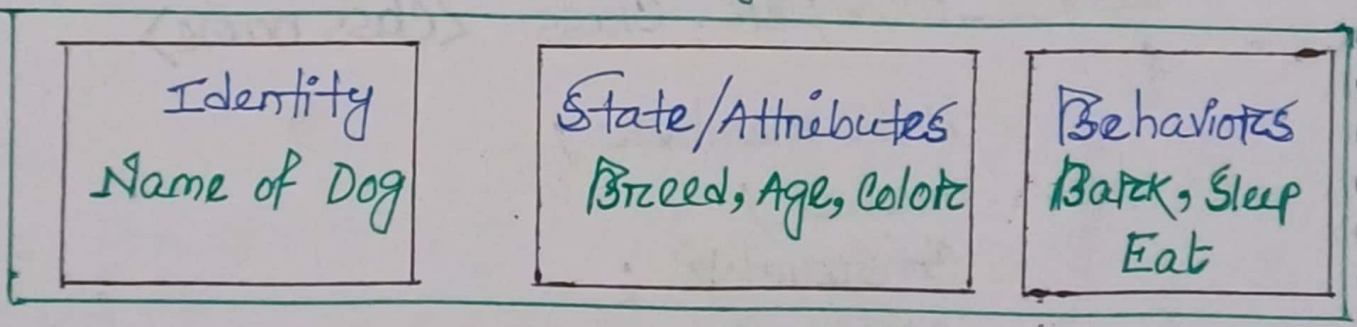
Objects

Objects are the instances of a class that are created to use the attributes and methods of a class.

A object consists of:

- State
- Behavior
- Identity

Example of an object : dog



⇒ 3 Ways to initialize object.

1. By reference Variable
2. By Method
3. By Constructor

1. Object and Class Example: Initialization through reference.

Initializing an object means storing data into the object.

```
class Student {
    int id;
    String name;
}
public class Test {
    public static void main(String args[]){
        Student s1 = new Student();
        s1.id = 9;
        s1.name = "Rakib";
        SOPl(s1.id + " " + s1.name);
    }
}
```

2. Initialization through Method

```
public class Student {
    int id;
    String name;
    void insertRecord (int i, String n){
        id = i;
        name = n;
    }
}
public class Test {
    public static void main (String args[]){
        Student S1 = new Student();
        Student S2 = new Student();
        S1.insertRecord (138, "Rakib");
        S2.insertRecord (137, "Sakib");
    }
}
```

3. Initialization through a Constructor

```
public class Employee {
    int id;
    String name;
    String companyName;
    void insert (int i, String n, String c){
        id
```

Anonymous object

An object which has no reference is known as an anonymous object. It can be used at the time of object creation only.

```
public class calculation{
    void fact(int n){
        int fact = 1;
        for(int i=1; i<=n; i++){
            fact = fact*i;
        }
        system.out.println(fact);
    }
    public static void main(String args[]){
        new calculation().fact(5);
        // Calling method with anonymous object;
    }
}
```

Anonymous objects are typically used when you need an object to perform specific action or method call and you don't need to reuse the object later

Constructor in OOP

Constructor in Java is used to create the instance of the class. Constructors are almost similar to methods except for two things - its name is the same as the class name and it has no return type.

Types of Java Constructors

There are two types of Constructors in Java

1. Default Constructor (no-arg Constructor)
2. Parameterized Constructor.

1. Default Constructor

A Constructor is called "Default Constructor" when it doesn't have any Constructor.

Example:

```
public class Bike {
    // Creating a default Constructor
    Bike(){
        System.out.println("Bike is Created")
    }
    public static void main(String args[]){
        Bike b = new Bike();
    }
}
```

2. Parameterized Constructor

A Constructor which has specific number of parameters is called a parameterized Constructor.

* Why use the parameterized Constructor?

The parameterized Constructor is used to provide different values to distinct objects. However, we can provide the same value also.

Example :

```
public class Student {
    int id;
    String name;
    // Creating a parameterized Constructor
    Student (int n, String m){
        id = n;
        name = m;
    }
    void display (){
        System.out.println (id + " " + name);
    }
    public static void main (String args[]){
        Student S1 = new Student (1, "Rafi");
        Student S2 = new Student (2, "Riya");
        S1.display();
        S2.display();
    }
}
```

Method

A method in Java is a block of code that, when called, performs specific actions mentioned in it.

Example:

```
public class Test {
    public int addNumbers(int a, int b){
        int sum = a+b;
        return sum;
    }
    public static void main(String args[]){
        int num1 = 7, num2 = 9;
        Test ob = new Test();
        int result = ob.addNumbers(num1,num2);
        System.out.println(result);
    }
}
```

Difference between Constructor and Method.

Constructor	Method
A Constructor helps in initialising an object.	A method is grouping of instructions that returns a value upon its execution
Constructor must not have return type.	Method must have return type.
The name of the Constructor and class will always be the same	For the method, we can use any name.
→ Cannot be inherited by subclasses.	→ Can be inherited by subclasses.