Java Programming Day 8

By Kavuri Santosh Kumar

Object-Oriented Programming (OOP):

Object-Oriented Programming (OOP):

- It is a programming paradigm that models real-world scenarios using objects within the context of Java programming.
- Any object in existence can be identified by its characteristics (features) and behaviors (functions). These characteristics can be represented using variables (also known as attributes or fields), and behaviors can be implemented through methods (also called functions or operations).

Real-Time Scenarios of OOP:

Object-Oriented Programming (OOP) can be effectively used to model real-world scenarios. Here are some examples:

Car

- **Properties/Fields/Attributes/States/Specifications/Features (Global Variables):**
  - `price`
  - `brandName`
  - `topSpeed`
  - (Add more as required: `color`, `fuelType`, etc.)
- **Functions/Operations/Behavior/Actions (Methods):**
  - `travel()`
  - `race()`
  - `draft()`
  - (Add more as required: `start()`, `stop()`, etc.)

This is a clear and structured way to define the attributes and behaviors of the `Car` object.

1. **Banking System:**
    - Objects: Customers, Accounts, Transactions
    - Features: Customer Name, Account Number, Balance
    - Functions: Deposit, Withdraw, Check Balance
2. **E-Commerce Website:**
    - Objects: Users, Products, Orders
    - Features: Product Name, Price, Quantity
    - Functions: Add to Cart, Place Order, Make Payment
3. **Library Management System:**
    - Objects: Books, Members, Librarians
    - Features: Book Title, Author, ISBN
    - Functions: Issue Book, Return Book, Search Book
4. **School Management System:**
    - Objects: Students, Teachers, Classes
    - Features: Student Name, Grade, Teacher Subject
    - Functions: Assign Homework, Evaluate Exam, View Attendance
5. **Car Manufacturing:**
    - Objects: Cars, Engines, Employees
    - Features: Car Model, Engine Type, Employee Role
    - Functions: Assemble Parts, Test Drive, Quality Check

In each scenario, objects encapsulate their data (features) and behavior (functions), making OOP a natural fit for building solutions to real-world problems.

Class and Object in Java:

Class and Object

Class:

- A **class** is a keyword used to define a **non-primitive** or **user-defined** data type.
- A **class** represents the blueprint or template for creating objects. It defines the structure and behavior (properties and methods) that objects of that class will have.
- A class is treated as a **logical entity** in programming, as it does not have physical existence on its own until an object is created.

Object:

- An **object** is an instance of a class that occupies memory space. It is created during runtime when the class is instantiated.
- In Object-Oriented Programming (OOP), an object is considered as the **real-world entity** or a **mirror image** of a blueprint (class).

Important Notes:

1. A **blueprint class** (business class) should not contain the `main` method. Instead, the objects of the blueprint class must be instantiated in another class that has the `main` method. This second class is known as the **driver class** or **user-logic class**.
2. If a property (or member variable) of an object changes independently for each instance, it should be represented as a **non-static** property. If the property remains constant across all instances, it should be represented as **static**.
3. If an operation (method) depends on any non-static property, it must also be a **non-static** method.
4. Non-static properties should not be initialized inside the blueprint class directly. They should be initialized after the object is instantiated, typically in the **driver class**.
5. A **blueprint class** is also called a **business class**, as it encapsulates the logic and properties of a real-world entity.
6. A **user-logic class** or **driver class** is responsible for creating objects from the blueprint class and running the program's main logic.

This structure clarifies the distinction between class and object and their respective roles in Java.

Problem Statements:

Q1. Create a `Contact` class that stores information about an individual's contact details, such as their name, phone number, email, and address. Implement a method to save this contact information and print a message confirming the save operation.

Q2. Design a program to represent a real-world scenario of calculating the area of a circle. Create a `Circle` class that serves as a blueprint for the circle's properties and behavior.

Q3. Design a program to represent a real-world rectangle, calculating its area based on its properties. Create a `Rectangle` class that serves as a blueprint for the rectangle's properties and behavior.

Q4. Write a java program to create the blueprint to contact having the properties SID Code and phone number and the behavior directly that display the phone number.?

Q5. Write a java program to create the blueprint of a student with any four properties and a behavior and create their objects in the user logic (two objects) and call their behaviours.?

Q6. Write a java program to behavior shoot create a user logic PUBG and create two gun objects and display their shoot method.?

Q1. Create a `Contact` class that stores information about an individual's contact details, such as their name, phone number, email, and address. Implement a method to save this contact information and print a message confirming the save operation.

Then, create a `Directory` class where you can instantiate multiple `Contact` objects, set their properties (name, phone number, email, and address), and save them using the `save()` method.

CODE:

Blue print class

```
class Contact {
    String name;
    Long phno;
    String email;
    String address;
    public void save() {
        // Saving logic goes here
```

```
        System.out.println("Contact saved: " + name);
    }
}
```

UserLogic class

```
public class Directory {
    public static void main(String[] args) {
        Contact obj = new Contact();
        obj.name = "Santosh Kumar";
        obj.phno = 9133239363L;
        obj.email = "kavuri@gmail.com";
        obj.address = "Hyderabad";
        obj.save();
        Contact obj1 = new Contact();
        obj1.name = "Naveen";
        obj1.phno = 9133239363L;
        obj1.email = "kavuri123@gmail.com";
        obj1.address = "Hyderabad";
        obj1.save();
    }
}
```

Output:

```
Contact saved: Santosh Kumar
```

Q2. Design a program to represent a real-world scenario of calculating the area of a circle. Create a `Circle` class that serves as a blueprint for the circle's properties and behavior. The class should have the following:

CODE:-

Blue print class:-

```
class Circle {
    static double pi = 3.14;
    double radius;
    public void areaOfCircle() {
        System.out.println("Area of circle is: " + (3.14 * Circle.radius * this.radius));
    }
}
```

User Logic class:-

```
public class AreaOfCircle {
    public static void main(String[] args) {
        Circle c1 = new Circle();
        c1.radius = 5.0;
        c1.areaOfCircle();
        Circle c2 = new Circle();
        c2.radius = 10.5;
        c2.areaOfCircle();
    }
}
```

Output:-

Area of circle is: 78.5

Area of circle is: 346.185

Q3. Design a program to represent a real-world rectangle, calculating its area based on its properties. Create a `Rectangle` class that serves as a blueprint for the rectangle's properties and behavior. The class should include the following:

CODE:-

Blue Print class:-

```
class Rectangle {
    double length;
    double breadth;
    public void areaOfRectangle() {
        System.out.println("Area of rectangle is: " + (this.length * this.breadth));
    }
}
```

User Logic class:-

```
public class AreaOfRectangle {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle();
        r1.length = 6.4;
        r1.breadth = 2.4;
        r1.areaOfRectangle();
        Rectangle r2 = new Rectangle();
        r2.length = 12.8;
        r2.breadth = 5.6;
```

```
        r2.areaOfRectangle();
    }
}
```

Output:-

Area of rectangle is: 15.36

Area of rectangle is: 71.68

Q4. Write a java program to create the blueprint to contact having the properties SID Code and phone number and the behavior directly that display the phone number.?

CODE:-

Blue Print Class:-

```
class Contact {
    String stdcode;
    Long phno;
    public void displayPhoneNumber() {
        System.out.println("Phone number is: " + this.phno);
    }
}
```

User Logic:-

```
public class DisplayPhoneNumber {
    public static void main(String[] args) {
        Contact c1 = new Contact();
        c1.stdcode = "+91";
        c1.phno = 9133239363L;
        c1.displayPhoneNumber();
        Contact c2 = new Contact();
        c2.stdcode = "+20";
        c2.phno = 9866883720L;
        c2.displayPhoneNumber();
        Contact c3 = new Contact();
        c3.stdcode = "+120";
        c3.phno = 13452334123L;
        c3.displayPhoneNumber();
    }
}
```

Output:-

```
Phone number is: 9133239363
Phone number is: 9866883720
Phone number is: 13452334123
```

Q5. Write a java program to create the blueprint of a student with any four properties and a behavior and create their objects in the user logic (two objects) and call their behaviours.?

Code:-

```
class Student {
    String name;
    int idno;
    long phno;
    String qualification;
    // Method should be declared as instance method and corrected the syntax
    public void detailsOfStudent() {
        System.out.println("Student name: " + this.name);
        System.out.println("Qualification: " + this.qualification);
    }
}
class DetailsOfStudent {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.name = "Santosh kumar";
        s1.qualification = "B.Tech";
        s1.detailsOfStudent();
        Student s2 = new Student();
        s2.name = "Naveen";
        s2.qualification = "B.Tech(CSE)";
        s2.detailsOfStudent();
    }
```

```
}
```

Output:-

```
Student name: Santosh kumar
Qualification: B.Tech
Student name: Naveen
Qualification: B.Tech(CSE)
```

Q6. Write a java program to behavior shoot create a user logic PUBG and create two gun objects and display their shoot method.?

Code:-

```
class Gun {
    String name;
    String shootType;
    public void displayShoot() {
        System.out.println("Name of the gun: " + this.name);
        System.out.println("Type of shooting method: " + this.shootType);
    }
}
class PUBG {
    public static void main(String[] args) {
        Gun g1 = new Gun();
        g1.name = "M400 rifle";
        g1.shootType = "Single and automatic modes of fire";
        g1.displayShoot();
        Gun g2 = new Gun();
        g2.name = "AK-47";
```

```
        g2.shootType = "Semi-auto rifle";
        g2.displayShoot();
    }
}
```

Output:-

Name of the gun: **M400 rifle**

Type of shooting method: **Single and automatic modes of fire**

Name of the gun: **AK-47**

Type of shooting method: **Semi-auto rifle**