Unit 3

Java as Object Oriented Programming Language Overview

Lecture 1

The expert in anything was once a beginner. - Helen Hayes

Classes and Objects in Java

Lecture 1

Agenda

- What is Class
- Examples
- What is Object
- Examples
- Demonstration
- Practice Programs

Class in Java: Introduction

- A **class** in Java is a blueprint or template for creating objects. It defines the **data (variables)** and **behavior (methods)** that the objects will have.
- A **blueprint** is a **plan or template** that is used to create something.
- In **Java**, a **class** is called a blueprint because it defines how objects should be structured and behave. Just like an **architect's blueprint** guides the construction of a building, a **Java class** guides the creation of objects.

Example

```
class Car {
    String brand;  // Data (Variable)
    void drive() { // Behavior (Method)
        System.out.println("The car is driving");
    }
}
```


Here, Car is a class with a brand variable and a drive() method.

Blueprint (Class): A car design that defines what parts a car should have.

Object (Instance): A real car made from that blueprint, like a Toyota or Honda.

Example 2:

```
class Student {
    String name;  // Attribute (Variable)
    int age;

    void study() {  // Method (Behavior)
        System.out.println(name + " is studying.");
    }
}
```

Object in Java: Introduction

- An object is a real-world entity created from a class. It represents something with state (data) and behavior (actions).
- Think of a class as a blueprint and an object as the actual thing built from that blueprint.
- Syntax to create object:


```
ClassName objectName = new ClassName();
```

Example 1

```
class Car {
    String brand;  // Attribute (State)

    void drive() {  // Method (Behavior)
        System.out.println(brand + " is driving.");
    }

    public static void main(String[] args) {
        Car myCar = new Car(); // Creating an object inside main()
        myCar.brand = "Toyota"; // Assigning a value
        myCar.drive();  // Calling method
    }
}
```

Example 2

```
class Car {
    String brand;

    void drive() {
        System.out.println(brand + " is driving.");
    }
}

public class Main {
    public static void main(String[] args) {
        Car myCar = new Car();  // Creating an object
        myCar.brand = "Toyota";
        myCar.drive();  // Calling a method
    }
}
```

Demonstration

Practice Programs

1. **Create a Student class with name and age, then display the values.**
2. **Create a Rectangle class length and width, then calculate the area.**

Unit 3

Java as Object Oriented

Programming Language Overview

Constructor in Java

Lecture 2

"Education is the most powerful weapon which you can use to change the world." – Nelson Mandela

Constructor in Java

Lecture 2

Agenda

- Constructor
- Examples
- Types of Constructor
- Examples
- Constructor Overloading
- Practice Programs

Constructor in Java: Introduction

- A constructor in Java is a special method used to initialize objects.
- It has the same name as the class and does not have a return type.
- Constructors are automatically called when an object of the class is created.
- They help in setting up initial values for object attributes.

Example of Constructor

```
public class Student {
    String name;
    int age;

    // Constructor
    Student(String n, int a) {
        name = n;
        age = a;
    }

    void display() {
        System.out.println("Name: " + name + ",
Age: " + age);
    }

    public static void main(String[] args) {
        Student s1 = new Student("John", 20);
        s1.display();
    }
}
```

Total No. of Questions : 8]

SEAT No. :

P1535

[Total No. of Pages : 3

[6002] - 164

S.E. (Computer Engineering)

PRINCIPLES OF PROGRAMMING LANGUAGES

(2019 Pattern) (Semester - IV) (210255)

Time : 2½ Hours] *[Max. Marks : 70*

Instructions to the candidates:

1) *Answer Q.1 or Q.2, Q.3 or Q.4, Q.5 or Q.6, Q.7 or Q.8.*
2) *Figures to the right indicate full marks.*
3) *Neat diagrams must be drawn whenever necessary.*
4) *Make suitable assumptions whenever necessary.*

Q1) a) Explain following features of java in detail **[6]**

i) Security
ii) Platform Independence
iii) Object - oriented

b) Write short note on **[6]**

i) Garbage collector
ii) this

c) Define constructor. Which are the types of Constructor used in Java? Explain with example. **[6]**

Why constructors are important

- Constructors help in setting up initial values for object attributes.
- Unlike regular methods, constructors are **invoked automatically** upon object creation.

Types of Constructors

1. Default Constructor
2. Parameterized Constructor
3. Copy Constructor

1. Default Constructor

- A constructor that takes **no parameters**.
- It is provided by Java **automatically** if no constructor is explicitly defined.
- Used to initialize instance variables with **default values**.

Example

```
class Animal {
    String type;

    // Default Constructor
    Animal() {
        type = "Unknown";
    }

    void display() {
        System.out.println("Animal Type: " + type);
    }

    public static void main(String[] args) {
        Animal a = new Animal();
        a.display();
    }
}
```


Output:

```
Animal Type: Unknown
```

2. Parameterized Constructor

- A constructor that **accepts parameters** to initialize an object with specific values.
- Helps in setting different values for different objects.

Example:

```
class Employee {
    String name;
    int id;

    // Parameterized Constructor
    Employee(String n, int i) {
        name = n;
        id = i;
    }

    void display() {
        System.out.println("Employee Name: " + name + ", ID: " + id);
    }

    public static void main(String[] args) {
        Employee e1 = new Employee("Alice", 101);
        e1.display();
    }
}
```


Output:

```
Employee Name: Alice, ID: 101
```

3. Copy Constructor

- A constructor that **copies the values** of one object into another object.
- It is used when we want to create a new object with the same values as an existing object.

```java
class Book {
    String title;

    // Parameterized Constructor
    Book(String t) {
        title = t;
    }

    // Copy Constructor
    Book(Book b) {
        title = b.title;
    }

    void display() {
        System.out.println("Book Title: " + title);
    }

    public static void main(String[] args) {
        Book b1 = new Book("Java Programming");
        Book b2 = new Book(b1);
        b1.display();
        b2.display();
    }
}
```


Output:

```
Book Title: Java Programming
Book Title: Java Programming
```

Constructor Overloading

- Just like methods, constructors can be **overloaded** (multiple constructors in the same class with different parameters).

Example

```
class Person {
    String name;
    int age;

    // Default Constructor
    Person() {
        name = "Unknown";
        age = 0;
    }

    // Parameterized Constructor
    Person(String n, int a) {
        name = n;
        age = a;
    }

    void display() {
        System.out.println("Name: " + name + ", Age: " + age);
    }

    public static void main(String[] args) {
        Person p1 = new Person();
        Person p2 = new Person("John", 25);

        p1.display();
        p2.display();
    }
}
```


Output:

```
Name: Unknown, Age: 0
Name: John, Age: 25
```