OOP in Java

Lecture 03

Agenda Points

- Introduction to Constructors in Java
- What is a Constructor?
- Types of Constructors in Java
- Default Constructor Explained
- Parameterized Constructor Explained
- Constructor Overloading Concepts
- Constructor Chaining in Java
- Best Practices for Using Contractors
- Conclusion and Summary of Key Points

Introduction to Constructors in Java

- A type of member function that is automatically executed when an object of that class is created is known as **constructor**.
- The constructor has no return type and has <u>same name that of class name</u>
- The constructor can work as a normal function but it cannot return any value.
- It is normally defined in classes to **initialize** data member
- Automatic initialization is carried out using a special member function called a constructor without requiring a separate call to a member function
- Constructors are invoked automatically when an object of a class is created.

What is a Constructor?

- A constructor is a block of code similar to a method that's called when an instance of an object is created.
- It initializes the newly created object.

```
class ClassName {
    // Constructor
    ClassName() {
        // Initialization code
    }
}
```

Types of Constructors in Java

- Default Constructor
- Parameterized Constructor

Default Constructor

- A constructor that has no parameters is called a default constructor.

- Java automatically provides a default constructor if no other constructors are defined.

```
class MyClass {
    // Default constructor
    MyClass() {
        System.out.println("Default Constructor called.");
    }
}
public class Main {
    public static void main(String[] args) {
        MyClass obj = new MyClass(); // Calls the default constructor
    }
}
```

Parameterized Constructor

- A constructor that accepts arguments to **initialize an object** is called a parameterized constructor.
- Allows setting initial values for object attributes.

```
class MyClass {
    int x;

    // Parameterized constructor
    MyClass(int val) {
        x = val;
        System.out.println("Parameterized Constructor called with value: " + x);
    }
}

public class Main {
    public static void main(String[] args) {
        MyClass obj = new MyClass(10); // Calls the parameterized constructor
    }
}
```

Constructor Overloading

- Overloading is the ability to define multiple constructors with different parameters.
- Each constructor performs different tasks.

```
class MyClass {
    int x;
    String y;

    // Default constructor
    MyClass() {
        x = 0;
        y = "Default";
    }

    // Parameterized constructor with one parameter
    MyClass(int val) {
        x = val;
    }

    // Parameterized constructor with two parameters
    MyClass(int val, String str) {
        x = val;
        y = str;
    }
}
```

Constructor Overloading

- Overloading is the ability to define multiple constructors with different parameters.
- Each constructor performs different tasks.

```
public class Main {

    public static void main(String[] args) {

        MyClass obj1 = new MyClass();              // Default constructor

        MyClass obj2 = new MyClass(10);        // Constructor with one parameter

        MyClass obj3 = new MyClass(20, "Hi");  // Constructor with two parameters

    }

}
```

Constructor Chaining in Java

- Constructor chaining occurs when a constructor calls another constructor of the same class or a parent class.
- Achieved using this() or super() keywords.

```
class MyClass {
    int x;

    // Default constructor
    MyClass() {
        this(5); // Calls parameterized constructor
        System.out.println("Default constructor");
    }

    // Parameterized constructor
    MyClass(int val) {
        x = val;
        System.out.println("Parameterized constructor with value: " + x);
    }
}

public class Main {
    public static void main(String[] args) {
        MyClass obj = new MyClass(); // Calls default constructor
    }
}
```

Best Practices for Using Constructors

- Keep constructors simple.
- Avoid complex logic inside constructors.
- Use constructor overloading wisely.
- Prefer factory methods over constructors when dealing with complex object creation.

Conclusion

- Constructors initialize objects in Java.
- There are default and parameterized constructors.
- Constructor overloading allows multiple constructors.
- Constructor chaining provides a clean and efficient way to initialize objects.

Final Thought: Constructors are essential for object-oriented programming in Java, ensuring that objects are properly initialized.

Thank you for your attention

Any question please…