**Session: Constructors in Java**

**Introduction**

* A **constructor** is a special method in Java used to initialize objects.
* It is invoked when an object is created using the new keyword.
* **Key Features**:
  + Has the same name as the class.
  + Does not have a return type (not even void).

**Topics Covered**

1. What is a Constructor?
2. Types of Constructors:
   * Default Constructor
   * Parameterized Constructor
   * Constructor Overloading
3. Does a Constructor Return a Value?
4. Copying the Values of One Object into Another
5. Can a Constructor Perform Tasks Other Than Initialization?
6. Best Practices

**1. What is a Constructor?**

* A constructor is used to assign initial values to the object's fields or execute startup logic.

**Example**:

java

class Student {

String name;

int age;

// Constructor

Student() {

name = "Unknown";

age = 0;

System.out.println("Constructor called!");

}

}

public class Main {

public static void main(String[] args) {

Student s = new Student(); // Constructor is called here

System.out.println(s.name + " " + s.age);

}

}

**Output**:

sql

Constructor called!

Unknown 0

**2. Types of Constructors**

**(i) Default Constructor**

* A constructor that takes no arguments.
* If no constructor is explicitly defined in the class, the compiler provides a default constructor.

**Example**:

java

class Car {

String brand;

// Default Constructor

Car() {

brand = "Toyota";

}

}

public class Main {

public static void main(String[] args) {

Car c = new Car();

System.out.println("Brand: " + c.brand);

}

}

**Output**:

Brand: Toyota

**(ii) Parameterized Constructor**

* A constructor that accepts arguments to initialize fields.

**Example**:

java

class Car {

String brand;

// Parameterized Constructor

Car(String b) {

brand = b;

}

}

public class Main {

public static void main(String[] args) {

Car c = new Car("Honda");

System.out.println("Brand: " + c.brand);

}

}

**Output**:

Brand: Honda

**(iii) Constructor Overloading**

* Defining multiple constructors in the same class with different parameter lists.

**Example**:

java

class Person {

String name;

int age;

// Default Constructor

Person() {

name = "Unknown";

age = 0;

}

// Parameterized Constructor

Person(String n, int a) {

name = n;

age = a;

}

}

public class Main {

public static void main(String[] args) {

Person p1 = new Person();

Person p2 = new Person("Alice", 25);

System.out.println(p1.name + " " + p1.age);

System.out.println(p2.name + " " + p2.age);

}

}

**Output**:

sql

Unknown 0

Alice 25

**3. Does a Constructor Return a Value?**

* **Direct Answer**: No, a constructor does not return any value, not even void.
* However, it implicitly returns the current instance of the class.

**Misconception Example**:

java

class Example {

Example() {

System.out.println("Constructor called!");

}

// This is NOT a constructor because of the return type

void Example() {

System.out.println("This is a method, not a constructor!");

}

}

**5. Can a Constructor Perform Tasks Other Than Initialization?**

* Yes, constructors can perform other tasks, but **it is not a best practice** to overuse constructors for purposes beyond initialization.

**Example**:

java

class Logger {

Logger() {

System.out.println("Log: Object Created!");

}

}

**Practical Usage**:

* Open a connection.
* Load configuration files.
* Initialize static resources.

**6. Best Practices**

1. **Always Define a Constructor Explicitly**:
   * Avoid relying on the default constructor unless it's intentional.
2. **Use Parameterized Constructors for Custom Initialization**.
3. **Encapsulate Initialization Logic in Helper Methods**:
   * If the logic grows too complex, move it out of the constructor.
4. **Avoid Calling Non-Final Methods in Constructors**:
   * It can lead to unexpected behavior during inheritance.

**Hands-On Exercise**

1. Create a Book class with the following:
   * Fields: title, author, price.
   * Constructors:
     + Default constructor with no parameters.
     + Parameterized constructor with all fields.
     + Copy constructor.
   * Test the constructors in the main method.

**Conclusion**

* Constructors are essential for object initialization in Java.
* They can be overloaded, parameterized, or default.
* While constructors can do more than initialization, they should focus on it to maintain code clarity and maintainability.

Encourage questions related to:

* Constructor chaining.
* The difference between constructors and methods.
* Practical uses of constructors.