17-Constructor

Introduction

In Java, a **constructor** is a special method used to initialize objects. The constructor is automatically called when an object of a class is created. It can be used to set initial values for object attributes, ensuring that an object is in a consistent state right from the moment it is created.

Understanding Constructors

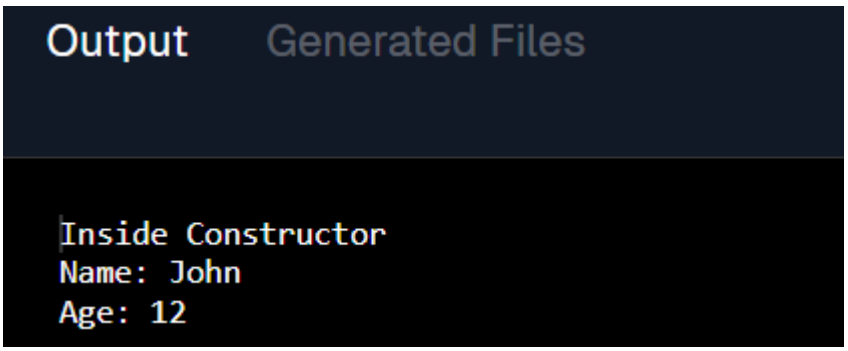
- **Constructor vs. Method**: Although a constructor is similar to a method, it differs in several ways:
    - A constructor **must** have the same name as the class.
    - Constructors **do not** have a return type, not even void.
    - A constructor is called **only once** when an object is created, whereas methods can be called multiple times.

How Constructors Work

When you create an object in Java using the new keyword, the constructor is called to initialize the object. If no constructor is explicitly defined in a class, Java provides a **default constructor** that initializes the object with default values.

```java
class Human {
    private int age;
    private String name;

    public Human() {  // Constructor
        System.out.println("Inside Constructor");
        age = 12;
        name = "John";
    }

    public int getAge() {
        return age;
    }

    public String getName() {
        return name;
    }
}

public class Demo {
    public static void main(String[] args) {
        Human obj = new Human();  // Constructor is called
        System.out.println("Name: " + obj.getName() + "\nAge: " + obj.getAge());
    }
}
```

In this example, when the Human object is created, the constructor is called, and the instance variables age and name are initialized with default values.



Why Use Constructors?

- **Avoid Default Values**: Without a constructor, instance variables might take default values (null for strings, 0 for integers), which may not be desired. Using a constructor, you can ensure that your object is initialized with meaningful values.
- **Custom Initialization**: Constructors allow you to set up an object exactly how you want it right when it's created.

Types of Constructors in Java

Java supports two main types of constructors:

1. **Default Constructor**: Provided automatically if no constructor is defined. Initializes object with default values.
2. **Parameterized Constructor**: Allows passing parameters to assign custom values during object creation.

3. Constructor vs. Method - A Quick Comparis

Feature	Constructor	Method
Purpose	Initializes the state of an object	Exposes the behavior of an object
Return Type	None	Must have a return type or void
Invocation	Implicitly invoked during object creation	Explicitly invoked using an object
Provided by Compiler	Yes, if no constructor is defined	No
Name	Same as the class name	Can be any valid identifier

FAQs on Java Constructors

1. **What is a constructor in Java?**
    - A constructor in Java is a special method used to initialize objects.

2. **Can a Java constructor be private?**
    - Yes, a constructor can be declared private, usually to restrict object creation from outside the class.
3. **Can a constructor be static, final, or abstract?**
    - No, a constructor cannot be static, final, or abstract because constructors are meant to initialize instances, and these modifiers conflict with the purpose of a constructor.
4. **Why can't we override constructors in Java?**
    - Constructors are not inherited, so they cannot be overridden. Overriding applies to methods that are inherited by subclasses.
5. **Is it mandatory to define a constructor in a class?**
    - No, it’s not mandatory. If you don’t define any constructor, the Java compiler provides a default constructor.