DAY 12

1. What is a Constructor?

A **constructor** is a special method used to initialize an object immediately after its creation. Its primary purpose is to set the initial state of the object, ensuring it's ready to be used. When you use the new keyword to create an object, the constructor is automatically invoked.

- **Real-Time Use Case:** Imagine a **Student** class for a school's system. When you create a new student object, the constructor can be used to assign them a unique ID and set their initial grade level. This prevents you from having to manually set these values after the object is created.

---

2. Thumb Rules for Creating a Constructor

To create a constructor properly, you must follow these key rules:

- It **must have the same name as the class**.
- It **cannot have a return type**, not even void.
- It can have any **access modifier** (e.g., public, private).
- The new keyword always calls a constructor when an object is instantiated.

---

3. Default vs. Explicit Constructors

Default (Implicit) Constructor

If you don't write any constructors in your class, the Java compiler automatically provides a **default constructor**. This is a public, no-argument constructor with an empty body. It's a safety net to ensure objects can always be created. The compiler only provides this if you haven't written any constructors yourself.

Explicit Constructors

An **explicit constructor** is one that you, the programmer, write yourself. Once you create **any** explicit constructor, the compiler will **not** provide the default one. This gives you full control over how your objects are initialized.

---

4. Types of Explicit Constructors

There are two primary types of explicit constructors you will use.

a) No-Argument Constructor

This is a constructor that takes no parameters. It's often used to create an object with default values.

- **Code Example:**

Java

```
public class Bike {
   String color;
   // No-argument constructor
   public Bike() {
      this.color = "Black"; // Sets a default color
   }
}
```

b) Parameterized Constructor

This constructor takes one or more parameters to initialize the object's instance variables with specific values. This is the most common approach for creating objects with meaningful data.

- **Code Example:**

Java

```
public class Car {
   String model;
   int year;
   // Parameterized constructor
   public Car(String model, int year) {
      this.model = model;
      this.year = year;
   }
}
```

With this constructor, you can create a Car object like this: Car myCar = new Car("Toyota", 2023);.

DAY 12

Constructor Overloading

Like regular methods, constructors can be **overloaded**. This means you can have multiple constructors in a single class, as long as each one has a different number or type of parameters. This provides flexibility in how objects can be created.

- **Code Example:**

Java

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

   // 1. No-argument constructor (for default values)
   public Person() {
      this.name = "Unknown";
      this.age = 0;
   }

   // 2. Parameterized constructor (to set both fields)
   public Person(String name, int age) {
      this.name = name;
      this.age = age;
   }
}
```

DAY 12

5. Interview Questions on Constructors

1. **What is a constructor, and what is its primary purpose?**
    - **Answer:** A constructor is a special method used to initialize an object. Its main purpose is to set the initial state of the object's instance variables.
2. **What is the difference between a default constructor and a no-argument constructor?**
    - **Answer:** A **default constructor** is one that the compiler automatically provides if no other constructors exist. A **no-argument constructor** is one that you explicitly write yourself, even if it has no parameters.
3. **What are the rules for creating a constructor?**
    - **Answer:** It must have the same name as the class, it cannot have a return type, and it can have any access modifier.
4. **Can a class have more than one constructor? If so, how?**
    - **Answer:** Yes, through **constructor overloading**. A class can have multiple constructors as long as each has a different parameter list (different number or types of parameters).
5. **What happens if you don't define a constructor in your class?**
    - **Answer:** The Java compiler will automatically provide a public, no-argument default constructor.