JAVA OOP

CONTRUCTOR

2 INTRODUCTION

A constructor in Java is similar to a method that is invoked when an object of the class is created.

Unlike Java methods, a constructor has the same name as that of the class and does not have any return type. For example,

```
class Test {

  Test() {

    // constructor body

  }

}
```

3 INTRODUCTION

```
class Main {
  private String name;
  // constructor
  Main() {
    System.out.println("Constructor Called:");
    name = "Programiz";
  }
}
```

4 INTRODUCTION

```
public static void main(String[] args) {
    // constructor is invoked while
    // creating an object of the Main class
    Main obj = new Main();
    System.out.println("The name is " + obj.name);
  }
```

5 TYPES OF CONSTRUCTOR

In Java, constructors can be divided into 3 types:

No-Arg Constructor

Parameterized Constructor

Default Constructor

1. JAVA NO-ARG CONSTRUCTORS

Similar to methods, a Java constructor may or may not have any parameters (arguments).

If a constructor does not accept any parameters, it is known as a no-argument constructor. For example,

```
private Constructor() {

   // body of the constructor

}
```

1. JAVA NO-ARG CONSTRUCTORS

```
class Main {
  int i;
  // constructor with no parameter
  private Main() {
    i = 5;
    System.out.println("Constructor is called");
  }
}
```

2. JAVA PARAMETERIZED CONSTRUCTOR

A Java constructor can also accept one or more parameters. Such constructors are known as parameterized constructors (constructor with parameters).

```
class Main {

  String languages;

  // constructor accepting single value

  Main(String lang) {

    languages = lang;

    System.out.println(languages + " Programming Language");

  }
```

3. JAVA DEFAULT CONSTRUCTOR

If we do not create any constructor, the Java compiler automatically create a no-arg constructor during the execution of the program. This constructor is called default constructor.

```
class Main {

  int a;

  boolean b;

}
```

3. JAVA DEFAULT CONSTRUCTOR

```
public static void main(String[] args) {
    // A default constructor is called
    Main obj = new Main();
    System.out.println("Default Value:");
    System.out.println("a = " + obj.a);
    System.out.println("b = " + obj.b);
  }
```

IMPORTANT NOTES ON JAVA CONSTRUCTORS

Constructors are invoked implicitly when you instantiate objects.

The two rules for creating a constructor are:

The name of the constructor should be the same as the class.

A Java constructor must not have a return type.

If a class doesn't have a constructor, the Java compiler automatically creates a default constructor during run-time. The default constructor initializes instance variables with default values. For example, the int variable will be initialized to 0

12

IMPORTANT NOTES ON JAVA CONSTRUCTORS

Constructor types:

No-Arg Constructor - a constructor that does not accept any arguments

Parameterized constructor - a constructor that accepts arguments

Default Constructor - a constructor that is automatically created by the Java compiler if it is not explicitly defined.

A constructor cannot be abstract or static or final.

A constructor can be overloaded but can not be overridden.

13

CONSTRUCTORS OVERLOADING IN JAVA

Similar to Java method overloading, we can also create two or more constructors with different parameters. This is called constructors overloading.

```
class Main {

  String language;

  // constructor with no parameter

  Main() {

    this.language = "Java";

  }
```

CONSTRUCTORS OVERLOADING IN JAVA

```
// constructor with a single parameter
  Main(String language) {
    this.language = language;
  }
  public void getName() {
    System.out.println("Programming Langauage: " + this.language);
  }
```

15

CONSTRUCTORS OVERLOADING IN JAVA

```
public static void main(String[] args) {
    // call constructor with no parameter
    Main obj1 = new Main();
    // call constructor with a single parameter
    Main obj2 = new Main("Python");
    obj1.getName();
    obj2.getName();
  }
```