Object Oriented Programming using Java

Prepared By:
Suyel, PhD
Assistant Professor
Dept. of CSE, NIT Patna

Outline

1. Constructor
2. Default Constructor
3. No Argument Contractor
4. Parameterized Constructor
5. Difference between Constructor and Method

Constructor

- In Java, a constructor is a block of codes similar to the method.
- Every class has a constructor.
- If we don't define a constructor in a class, compiler builds a default constructor.
- Constructor is used to initialize any object.
- At the time of calling constructor, memory for the object is allocated in the memory.
- Every time an object is created using the **new** operator, at least one constructor is called.
- A class can have many constructors.

Constructor (Cont…)

- Syntax:

```
public class Constructor1
{
   Constructor1()
   {
      ...
   }
   ...
}
```

- We use a constructor to give initial values to the instance variables defined by the class or to perform any other start-up procedures required to create a fully formed object.
- There are three rules defined for the constructor:
  - Constructor name must be the same as its class name.
  - A constructor must have no explicit return type.
  - A Java constructor cannot be abstract, static, final and synchronized.

Constructor (Cont…)

- A constructor can be overloaded, but cannot be overridden.
- There are three types of constructor:
  - **Default:** If we do not implement any constructor in a class, compiler uses a default constructor.
  - **No argument:** It is a constructor with no arguments.
  - **Parameterized:** This type of constructor has parameters or arguments.

Default Constructor

- If we do not implement any constructor in the class, Java compiler automatically inserts a default constructor into the code, which is known as **default constructor**.
- We would not find it in the source code (java file) as it would be inserted into the code during compilation and exists in the .class file (class file).
- If we implement any constructor, then compiler does not insert the default constructor.
- This type of constructor is used to provide the default values to the object like 0, null, etc., depending on the type.

Default Constructor (Cont…)

- **Process by the compiler**

```
class Constructor1
{
    public static void main (String args[])
   {
      Constructor1 obj=new Constructor1()
    }
}
```

```
class Constructor1
{
   Constructor1()
    {
    }
   public static void main (String args[])
   {
       Constructor1 obj=new Constructor1()
    }
}
```

Default Constructor (Cont…)

```
class Const1
{
   int x;
   String str;

   public static void main(String[] args)
   {
      Const1 obj = new Const1();

      System.out.println("Default values are: ");
      System.out.println("x: " + obj.x);
      System.out.println("str: " + obj.str);
   }
}
```

Default Constructor (Cont…)

- Output

  Default values are:

  x: 0

  str: null

No Argument Constructor

- Similar to method, a Java constructor may or may not have any parameters.
- Constructor with no arguments is known as **no argument constructor**.
- The signature is same as default constructor, however body can have any code unlike default constructor, where the body of the constructor is empty.

No Argument Constructor (Cont…)

```
class Const2
{
   public Const2()
   {
      System.out.println("This is what we call no argument constructor");
   }

   public static void main(String args[])
   {
      Const2 obj= new Const2();
   }
}
```

No Argument Constructor (Cont…)

- Output

```
This is what we call no argument constructor
```

No Argument Constructor (Cont…)

```
class Const3
{
   int x;

   Const3()
   {
      x = 100;
      System.out.println("Constructor is called");
   }

   public static void main (String args[])
   {
      Const3 obj = new Const3();
      System.out.println("Value of x: " + obj.x);
   }
}
```

No Argument Constructor (Cont…)

- Output

  Constructor is called

  Value of x: 100

No Argument Constructor (Cont…)

Const4.java

```
class Const4
{
   public static void main (String args[])
   {
      Test obj = new Test();

      System.out.println(obj.x);
      System.out.println(obj.str);
   }
}
```

Test.java

```
class Test
{
   int x;
   String str;

   private Test()
   {
      System.out.println("Constructor called");
   }
}
```

No Argument Constructor (Cont…)

- Output

```
Const4.java:6: error: Test() has private access in Test
      Test obj = new Test();
                 ^
1 error
```

Parameterized Constructor

- Constructor with arguments is known as **parameterized constructor**.
- The parameterized constructor is used to provide different values to distinct objects. However, we can provide the same values also.

Parameterized Constructor (Cont…)

```
class Const5
{
   int ID;
   String name;

   Const5(int x, String str)
   {
      ID = x;
      name = str;
   }

   public static void main(String args[])
   {
      Const5 obj = new Const5(1010, "Anubhav");
      System.out.println("Student ID: " + obj.ID + " and Student Name: " + obj.name);
   }
}
```

Parameterized Constructor (Cont…)

- Output

  Student ID: 1010 and Student Name: Anubhav

Parameterized Constructor (Cont…)

```
class Const6
{
   int ID;
   String name;

   Const6(int x, String str)
   {
      ID = x;
      name = str;
   }

   void print()
   {
      System.out.println("Student ID: " + ID + " and Student Name: " + name);
   }

   public static void main(String args[])
   {
      Const6 obj1 = new Const6(1010, "Anubhav");
      Const6 obj2 = new Const6(2020, "Smita");
      obj1.print();
      obj2.print();
   }
}
```

Parameterized Constructor (Cont…)

- Output

  Student ID: 1010 and Student Name: Anubhav

  Student ID: 2020 and Student Name: Smita

Difference between Constructor and Method

Constructor	Method
Constructor is used to initialize an object.	Method is used to define functionality of an object.
Constructors are invoked implicitly.	Methods are invoked explicitly.
If constructor is not present, a default constructor is invoked by the compiler.	Here, no default method is invoked.
Constructor does not return any value.	Method may/may not return a value.
Constructor must have the same name as the class.	Method must not have the same name as the class.
Here, we cannot use static keyword.	In method, we can use static keyword.


thank
you . . .

Slides are prepared from various sources, such as Book, Internet Links and many more.