

Topics



- Constructors in Java
- Creating/Instantiating Objects
- Types of Constructors



Role of Constructor Method

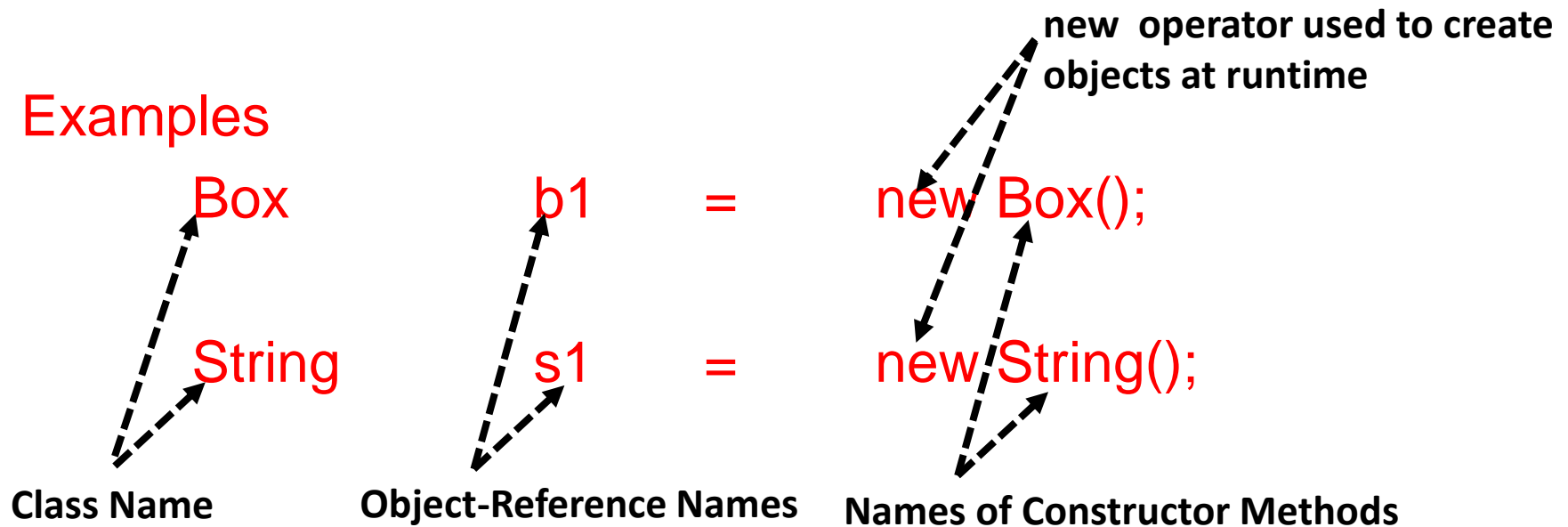
- Constructor is a Method of a class having a name similar to the class itself
- Helps to Create and Initialize the Objects
- If no constructor is defined for the class then a default un-parameterized constructor will be provided by Java Runtime Environment.
- Constructor Method has no return type not even void.
- Code written inside constructor method is automatically executed for every object creation of that class.
- If a class has its own constructor then default constructor will not be available
- Constructor Method can have its own Access Modifier : public, protected, package-private and private

Object Creation Syntax

- Creating/Instantiating Objects

<Class-Name> object-reference = new Constructor();

Examples



Note : Above Examples Use Unparametrized Constructor

Object Creation Example



```
/* File Name : Box.java */
class Box
{
    private    double    length;    // length of a Box
    private    double    width;     // width of a Box
    private    double    height;    // height of a Box

    /* Method to Compute the Area of Box */
    public     double     area()
    {
        return 2* (length * width + width * height + height * length);
    } // End of Method

    /* Method to Compute the Volume of Box */
    public     double     volume()
    {
        return length * width * height;
    } // End of Method
} // End of class Box
```

Class Box Has No Constructor

```
class BoxTest
{
    public     static     void     main (String [ ] args)
    {
        Box    b1         =     new
        Box    b2         =     new

    } // End of Method
} // End of class BoxTest
```

```
Box();
Box();
```

**Constructor
Provided by
JRE**

Types of Constructors



1. Un-parameterized Constructor
2. Parameterized Constructor
3. Overloaded Constructors

Class With Unparametrized Constructor



```
// File Name : Demo.java
```

```
class A  
{
```

```
    private int a;  
    private int b;
```

```
    A()  
    {  
        System.out.println("Hello Welcome");  
    }
```

Un-parameterized
Constructor

```
} // End of class A
```

```
class ConstructorDemo
```

```
{  
    public static void main(String args[])  
    {
```

```
        A a1 = new A();
```

```
        A a2 = new A();
```

```
        A a3 = a1;
```

```
        A a4 = a2;
```

```
    } // End of Method
```

```
} // End of class ConstructorDemo
```

Only Two Objects are Created in This Example

:A



:A



Class With Unparametrized Constructor



```
// File Name : Demo.java
```

```
class A
```

```
{
```

```
    private int a;
```

```
    private int b;
```

```
    A()
```

```
    {
```

```
        System.out.println("Hello Welcome");
```

```
    }
```

```
}// End of class A
```

```
class ConstructorDemo
```

```
{
```

```
    public static void    main(String args[])
```

```
    {
```

```
        A a1 = new A();
```

```
        A a2 = new A();
```

```
        A a3 = a1;
```

```
        A a4 = a2;
```

```
    } // End of Method
```

```
}// End of class ConstructorDemo
```

Predict the O/P of This Code

Output

```
F:\>javac Demo.java
```

```
F:\>java ConstructorDemo
```

```
Hello Welcome
```

```
Hello Welcome
```

Class With Unparametrized Constructor



```
// File Name : Demo.java
class A
{
    private int a;
    private int b;
    A()
    {
        for(int i =0; i<10; i++)
            System.out.println ("Hello");
    }
}
} // End of class A
class ConstructorDemo
{
    public static void    main(String args[])
    {
        A a1 = new A();
        A a2 = new A();
        A a3 = new A();
        A a4 = new A();
    } // End of Method
} // End of class ConstructorDemo
```

Predict The Output

Class With Parameterized Constructor



// File Name : Demo.java

class A

{

private int a;

private int b;

A(int a, int b)

{

this.a = a;

this.b = b;

displayValues();

} // End of Constructor Method

/* Method to Display the Attribute Values */

public void displayValues()

{

System.out.println(" a = " + a);

System.out.println(" b = " + b);

} // End of Method

} // End of class A

class ConstructorDemo

{

public static void main(String args[])

{

A a1 = new A();

} // End of Method

} // End of class ConstructorDemo

**Parameterized
Constructor**

Compile-Time Error

F:\>javac Demo.java

Demo.java:27: cannot find symbol
symbol : constructor A()

location: class A

A a1 = new A();

^

1 error

Class With Parameterized Constructor



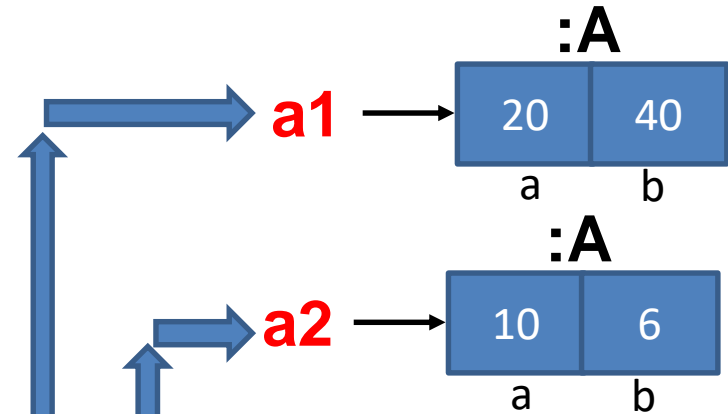
// File Name : Demo.java

```
class A
{
    private int a;
    private int b;

    A(int a, int b)
    {
        this.a = a;
        this.b = b;
        displayValues();
    } // End of Constructor Method

    /* Method to Display the Attribute Values */
    public void displayValues()
    {
        System.out.println(" a = " + a);
        System.out.println(" b = " + b);
    } // End of Method
} // End of class A

class ConstructorDemo
{
    public static void main(String args[])
    {
        A a1 = new A(20,40);
        A a2 = new A(10,6);
    } // End of Method
} // End of class ConstructorDemo
```



Output

```
F:\>javac Demo.java
F:\>java ConstructorDemo
a = 20
b = 40
a = 10
b = 6
```

Class with Overloaded Constructors



- Constructor Methods are Overloaded if they have same name but different signatures**

```
// File Name : Demo.java
class Triangle
{
    private double      a;          // Side 1 of Triangle
    private double      b;          // Side 2 of Triangle
    private double      c;          // Side 2 of Triangle

    // Constructor Method - 1 Used to Create Equilateral Triangle
    Triangle(double side)
    {
        a = b = c = side;
        displaySides();
    } // End of Constructor

    // Constructor Method - 1 Used to Create Isosceles Triangle
    Triangle(double side1, double side2)
    {
        a = b = side1;  c = side2;
        displaySides();
    } // End of Constructor

    // Constructor Method - 3 Used to Create Scalene Triangle
    Triangle(double side1, double side2, double side3)
    {
        a = side1;  b = side2;  c = side3;
        displaySides();
    } // End of Constructor
}
```

```
// Method to print the sides of Triangle
public      void      displaySides()
{
    System.out.println("Side 1:" + a);
    System.out.println("Side 2:" + b);
    System.out.println("Side 3:" + c);
} // End of Method
} // End of Class

/* Driver Class */
class ConstructorDemo
{
    public static void main(String args[])
    {

        Triangle T1 = new Triangle(10);
        Triangle T2 = new Triangle(20, 30);
        Triangle T3 = new Triangle(10,6,8);

    } // End of Method
} // End of class ConstructorDemo
```

Class with Overloaded Constructors



- Output of the Java Program of Previous Slide

```
F:\>javac Demo.java
```

```
F:\>java ConstructorDemo
```

```
Side 1:10.0
```

```
Side 2:10.0
```

```
Side 3:10.0
```

```
Side 1:20.0
```

```
Side 2:20.0
```

```
Side 3:30.0
```

```
Side 1:10.0
```

```
Side 2:6.0
```

```
Side 3:8.0
```

Constructors With Access Modifiers



// File Name : Demo.java

class XYZ

{

private int x, y, z;

// Constructor with 'private' Modifier

private XYZ (int a)

{

x = a;

}

// Constructor with public Modifier

public XYZ (int y, int z)

{

this.y = y;

this.z = z;

}

// End of class XYZ

// Driver Class

class Test

{

// Driver Method

public static void main(String args[])

{

A a1 =

new A(10);

A a2 =

new A(10,40);

} // End of Method

} // End of class Test

Output

F:\>javac Demo.java

Demo.java:25: cannot find symbol

symbol : constructor A(int)

location: class A

A a1 = new A(10);
^

1 error

Compile-Time
Error



Constructor Summary

- Method of a class having same name as that of class
- No return type (not even void) for Constructor Methods
- Can have access Specifiers (Modifiers)

Thank You