Topics

- Method Overloading [Supports Adhoc Polymorphism in Java]

What is Method Overloading?

- Two or More Methods are said to be overloaded if and only if the following two conditions are satisfied
  1. All the Methods have same name [Condition 1] and
  2. All the Methods have different signatures [Condition 2]
- Signature of a Method Constitutes (i) Method Name and (ii) Types and Order of its arguments
- Return type of a method does not form the part of method signatures
- Overloaded Methods may have same or different return types
- Call to an Overloaded Method is decided at Compile-Time and not at runtime.

Method Signature : Examples

innovate achieve lead

Method(s)	Signatures

Method Signature : Examples

Method(s)	Signatures

int doS (int a, float b, char c)
{
}
	doS(int, float, char)

Method Signature : Examples

Method(s)	Signatures

int doS (int a, float b, char c)
{
}
	doS(int, float, char)

float doS1 ()
{
}
	doS1()

Method Signature : Examples

Method(s)	Signatures

int doS (int a, float b, char c)
{
}
	doS(int, float, char)

float doS1 ()
{
}
	doS1()

void xyz(String x, double y)
{
}
	xyz(String , double)

Method Signature : Examples

innovate achieve lead

Method(s)	Signatures

int doS (int a, float b, char c)
{
}
	doS(int, float, char)

float doS1 ()
{
}
	doS1()

void xyz(String x, double y)
{
}
	xyz(String , double)

float xyz (double y, String x)
{
}
	xyz(double , String)

How Signatures of Two Overloaded Methods Can Be Different?

- Method 1 : Change the number of arguments to Overloaded Methods
  1. int sum (int x, int y) { … } → Two Arguments, Signature→ sum(int, int)
     int sum (int a, int b, int c) { … } → Three Arguments, Signature → sum(int, int, int)
  2. void doS(int x) { … } → One Argument, Signature → doS(int)
     int doS (int a, char c) { … } → Two Arguments, Signature →doS(int, char)
- Method 2 : Change or shuffle the type of any argument if number of arguments to Overloaded Methods are same
  1. int sum (int x, int y) { … } → Two Arguments, Signature → sum(int, int)
     int sum (float a, float b { … } → Two Arguments, Signature → sum(float, float)
  2. void doS(int x, float y) { … } → Two Arguments, Signature → doS(int, float)
     int doS (float a, int b) { … } → Two Arguments, Signature → doS(float, int)

Overloaded Methods : Example 1

innovate achieve lead

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

        // Constructor Method
        Box(double length, double width, double height)
        {
                this.length = length;
                this.width  = width;
                this.height = height;
        }// End of Constructor Method

        // Accessor Method for length
        public      double getLength()
        {
                return this.length;
        }// End of Method

        // Accessor Method for width
        public      double getWidth()
        {
                return this.width;
        }// End of Method
```

Overloaded Methods : Example 1 …..


```
// Accessor Method for height
public          double getHeight()
{
                return this.height;
}// End of Method

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

// OVERLOADED METHODS TO COMPARE TWO BOXES

// Object Method isEqual --> Overloaded 1
public          boolean         isEqual(Box other)
{
                /* This Method Compares 'this' box with 'other'. this' box is equal to other if they have same area otherwise they are unequal */
                        if(this.area() == other.area()) return true;
                        else            return false;
}// End of Method

// class Method isEqual --> Overloaded 2
public static boolean           isEqual(Box first, Box second)
{
                /* This Method Compares 'first' box with 'second'. 'first' box is equal to 'second'box if they have same area otherwise they are
                   unequal
                */
                        if(first.area() == second.area()) return true;
                        else            return false;
}// End of Method

}// End of class Box
```

Overloaded Methods : Example 1…..

innovate achieve lead

```
// Driver class
class Test
{
        public static void main(String args[])
        {
                Box b1 = new Box(10,6,8);
                Box b2 = new Box(20,4,8);
                Box b3 = new Box(10,6,8);

                // How to check b1 and b2 for equality

                // Method-1 --> Call Object Method by any  object refereces b1 or b2
                System.out.println(b1.isEqual(b2));

                // Method-2 --> Call class Method by passing both object references b1 and b2 as parameters
                System.out.println(Box.isEqual(b1,b2));

                System.out.println(b1.isEqual(b3));

                System.out.println(b1.isEqual(b2,b3));

        }// End of Method
}// End of class Test
```

<<Output>>

```
F:\>javac Overloading.java

F:\>java Test
false
false
true
false
```

Overloaded Methods : Example 2


```
class Test
{
        // Overloaded Method-1
        public static int sum(int a, int b)
        {
                return a + b;
        }// End of Method
        // Overloaded Method-2
        public static float sum(float a, float b)
        {
                return a + b;
        }// End of Method
        // Overloaded Method-3
        public static double sum(double a, double b)
        {
                return a + b;
        }// End of Method
        // Overloaded Method-4
        public static long sum(long a, long b)
        {
                return a + b;
        }// End of Method

        public static void main(String args[])
        {
                System.out.prinltn(sum(10.5, 10.6));
                System.out.prinltn(sum(10.5f, 10.6f));
                System.out.prinltn(sum(10, 10));
                System.out.prinltn(sum(10L, 10L));
        }// End of Method
}// End of class Test
```

<<Output>>

```
F:\>java Test
21.1
21.1
20
20
```

Thank You