Atma Ram Sanatan Dharma College
University of Delhi

Programming In Java

Submitted By

KHUSHAL SACHDEVA

20/88044

BSc. (Hons.) Computer Science (SEM - 2)

Submitted To

Ms Parul Jain

Department of Computer Science

Q1.What is the difference between function overloading and constructor overloading in Java?

Answer:

Function Overloading

1. Writing more than one function within a class with a unique set of arguments is called as method overloading
2. All function must share the same name
3. An overloaded method if not static can only be called after instantiating the object as per the requirement
4. Overloaded function can be static, and it can be accessed without creating an object
5. An overloaded function can be final to prevent subclasses to override the method
6. An overloaded function can be private to prevent access to call that method outside of the class

Constructor Overloading

1. Writing more than 1 constructor in a class with a unique set of arguments is called as Constructor Overloading
2. All constructors will have the name of the class
3. Overloaded constructor will be executed at the time of instantiating an object
4. An overloaded constructor cannot be static as a constructor relates to the creation of an object
5. An overloaded constructor cannot be final as constructor is not derived by subclasses it won’t make sense
6. An overloaded constructor can be private to prevent using it for instantiating from outside of the class

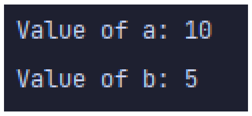
Write a suitable program to illustrate the following:

a) Default constructor

CODE:

```
class Q1Constructor {
   // a. Default constructor
   int a = 10;
   int b = 5;
}
public class Main {
   public static void main(String[] args) {
       //defines a default constructor if not defined any by the user
       Q1Constructor obj = new Q1Constructor();
       System.out.println("Value of a: " + obj.a);
       System.out.println("Value of b: " + obj.b);
   }
}
```

Output:



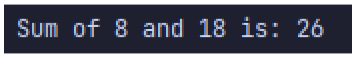
```
Value of a: 10
Value of b: 5
```

b) Parameterized constructor

CODE:

```
class Parameter{
   // b. Parameterized constructor
   public Parameter(int a, int b) {
       int sum = a + b;
       System.out.println("Sum of " + a + " and " + b + " is: " + sum );
   }
}
public class Main {
   public static void main(String[] args) {
       //defines a parameterized constructor
       new Parameter(8,18);
   }
}
```

Output:



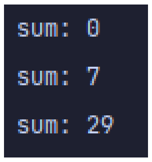
```
Sum of 8 and 18 is: 26
```

c) Method overloading with different number of parameters

CODE:

```
class Overload{
   // Method overloading with different number of parameters
   public String sum(){
       int Sum = 0;
       return "sum: "+Sum;
   }
   public String sum(int a, int b){
       int Sum = a + b;
       return "sum: "+Sum;
   }
   public String sum(int a, int b,int c){
       int Sum = a + b + c;
       return "sum: "+Sum;
   }
}
public class Main {
   public static void main(String[] args) {
       Overload obj = new Overload();
       System.out.println(obj.sum());
       System.out.println(obj.sum(5,2));
       System.out.println(obj.sum(6,18,5));
   }
}
```

Output:



```
sum: 0
sum: 7
sum: 29
```

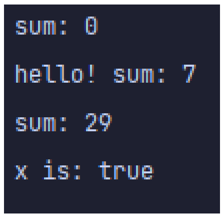
d) Method overloading with different type of parameters

CODE:

```
class Overload{
   // Method overloading with different number of parameters
   public String sum(){
       int Sum = 0;
       return "sum: "+Sum;
   }
   public String sum(int a, int b, String hello){
       int Sum = a + b;
```

```
        return hello + " sum: "+Sum;
    }
    public String sum(int a, int b,int c, boolean x){
        int Sum = a + b + c;
        return "sum: "+Sum +"\nx is: " + x;
    }
}
public class Main {
    public static void main(String[] args) {
        Overload obj = new Overload();
        System.out.println(obj.sum());
        System.out.println(obj.sum(5,2,"hello!"));
        System.out.println(obj.sum(6,18,5, true));
    }
}
```

Output:



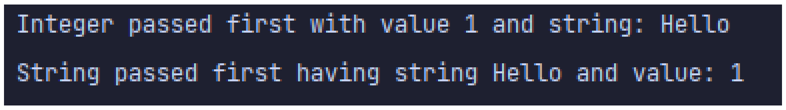
```
sum: 0
hello! sum: 7
sum: 29
x is: true
```

e) Method overloading with different sequence of parameters

CODE:

```
class Overload {
    public String Par(int a, String b) { //Method when integer is passed first
        return "Integer passed first with value " + a +" and string: " + b;
    }

    public String Par(String a, int b) { //same method when string is passed first
        return "String passed first having string " + a +" and value: " + b;
    }
}
public class Main{
public static void main(String[] args){
        Overload o=new Overload();
        System.out.println(o.Par(1,"Hello")); //calls the 1st method
        System.out.println(o.Par("Hello",1)); //calls the 2nd method
        }
}
```

Output:



```
Integer passed first with value 1 and string: Hello
String passed first having string Hello and value: 1
```

Q2. Write the advantages of using packages in java. Write a suitable program that illustrates different levels of protection in classes/subclasses belonging to same package or different packages.

Answer:

A package is a namespace that organizes a set of related classes, subclasses, subpackages and interfaces. They can be categorized into two types namely-: built in packages and user defined packages. Some Examples of built in packages are-: java, lang, awt, javax, io etc.

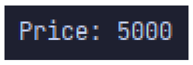
Some advantages of using packages are:-

- It is used to categorize by the classes and interfaces.
- It is easy to maintain.
- It provides access protection.
- It removes naming collisions.
- It provides reusability of code.
- We can create our own package or extend an already available package.
- It is also easier to locate the related classes

Code:

```java
// Inside Package differentPackage
package differentPackage;

class class2{
    int a = 5;
    String s;
}

// Public class
public class parentClass {
    public int price;
    public void setPrice(int price,boolean isAdmin) {
        if(!isAdmin){
            System.out.println("You cannot set the price");
        }
        else{
            this.price = price;
        }
    }
    public static void main(String[] args) {
        // We can access default class in the same package
        class2 obj = new class2();
        System.out.println("a = " + obj.a);
    }
}

// Inside Package samePackage
package samePackage;

import differentPackage.parentClass; // we can access public class from
different package
//import differentPackage.class2;
/*
   Can't access the parent class as it is default class so we can't
   access it in different package we can only access default classes
   in the same package only.
*/

public class class1 {
   // we can access the public class in different package
   //we can easily call the class in the same package as it is public
   public static void main(String[] args) {
        parentClass bb = new parentClass();
        bb.setPrice(5000, true);
        System.out.println("Price: " + bb.price);
   }
}
```

Output:



```
Price: 5000
```