2.5 Method Overloading

In Java there are two type of overloading:-
1) Method Overloading
2) Constructor Overloading

2.5.1 Overloading Method:

The Java programming language supports ***overloading* methods**, and Java can distinguish between methods with different ***method signatures***.

This means that methods within a class can have the same name if they have different parameter lists (there are some qualifications to this that will be discussed in the lesson titled "Interfaces and Inheritance").

- Method Overloading means to define different methods with the same name but different parameters lists and different definitions.
- It is used when objects are required to perform similar task but using different input parameters that may vary either in number or type of arguments.
- Overloaded methods may have different return types. It is a way of achieving polymorphism in java.

```
int add( int a, int b) // prototype 1
int add( int a , int b , int c) // prototype 2
double add( double a, double b) // prototype
```

Example :

```
class Sample{
   int addition(int i, int j){
       return i + j ;
    }
String addition(String s1, String s2){
```

```
    return s1+s2;
}
  double addition(double d1, double d2){
      return d1+d2;
      }
}
class AddOperation
{
  public static void main(String args[])
  {
     Sample S = new Sample();
     System.out.println(S.addition(1,2));
     System.out.println(S.addition("Hello","World"));
     System.out.println(S.addition(1.5,2.2));
   }
}
```

2.5.2 Overloading Constructor:

- Like method overloading, Java also supports constructor overloading.
- **Writing multiple constructors, with different parameters, in the same class is known as constructor overloading.**

- Depending upon the parameter list, the appropriate constructor is called when an object is created.

Let us see the Constructor Overloading concept programmatically.

```
public class Student
{
public Student()     // I ,  default constructor
{
System.out.println("Hello 1");
}
public Student(String name)
 // II,  parameterized constructor with single parameter
{
System.out.println("Student name is " + name);
}

public Student(String name, int marks)
// III,  parameterized constructor with two parameters
{
System.out.println("Student name is " + name + " and marks are " +
marks);
}
public static void main(String args[])
{
Student std1 = new Student();        // calls I
Student std2 = new Student("Mr.Reddy");       // calls II
Student std3 = new Student("Mr.Raju", 56);    // calls III
}
}
```