Idris Ratlamwala

C31 - 2003145

Experiment 8: Program on method and constructor overloading

**Theory :**

**Method Overloading** is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to constructor overloading in Java, that allows a class to have more than one constructor having different argument lists.

For example the argument list of a method add(int a, int b) having two parameters is different from the argument list of the method add(int a, int b, int c) having three parameters.

## **Three ways to overload a method**

In order to overload a method, the argument lists of the methods must differ in either of these:

**1.** Number of parameters.  
For example: This is a valid case of overloading

add(int, int)

add(int, int, int)

**2.** Data type of parameters.  
For example:

add(int, int)

add(int, float)

**3.** Sequence of Data type of parameters.  
For example:

add(int, float)

add(float, int)

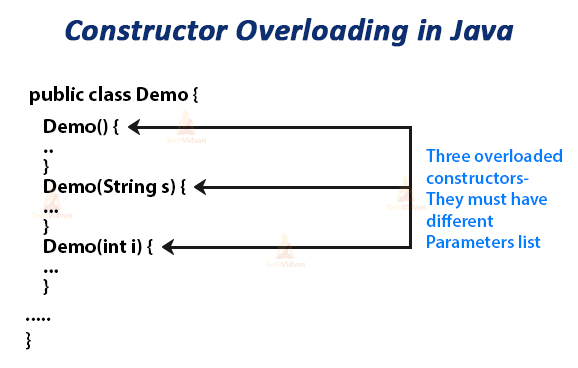
**Invalid case of method overloading:**  
When I say argument list, I am not talking about return type of the method, for example if two methods have same name, same parameters and have different return type, then this is not a valid method overloading example. This will throw compilation error.

int add(int, int)

float add(int, int)

Like methods, constructors can also be overloaded

**Constructor overloading** is a concept of having more than one constructor with different parameters list, in such a way so that each constructor performs a different task. For e.g. Vector class has 4 types of constructors. If you do not want to specify the initial capacity and capacity increment then you can simply use default constructor of Vector class like this Vector v = new Vector(); however if we need to specify the capacity and increment then we call the parameterized constructor of Vector class with two int arguments like this: Vector v= new Vector(10, 5);



**A.**

**Aim :** Calculate area of different shapes ( Square, Rectangle, Circle) using method overloading and multiple class concept.

**Program :**

    class Area{

        int calculateArea(int l){

            return l\*l;

        }

        int calculateArea(int l,int b){

            return l\*b;

        }

        double calculateArea(float r){

            return 3.142\*r\*r;

        }

    }

    public class AreaMethodOL {

        public static void main(String[] args) {

            Area a1= new Area();

            System.out.println("Area of square with sides 12 is "+a1.calculateArea(12));

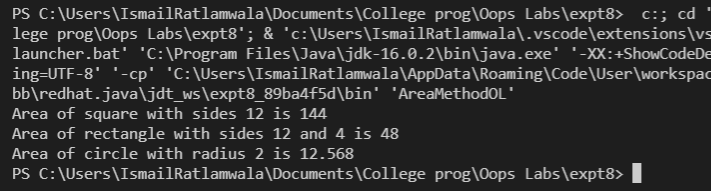
            System.out.println("Area of rectangle with sides 12 and 4 is "+a1.calculateArea(12,4));

            System.out.println("Area of circle with radius 2 is "+a1.calculateArea((float)2));

        }

    }

**Output :**



**B.**

**Aim :** Calculate area of different shapes ( Square, Rectangle , Circle) using

constructor overloading and multiple class concept.

**Program :**

        class Areas{

        Areas(int l){

            System.out.println("Area of square is "+l\*l);

        }

        Areas(int l,int b){

            System.out.println("Area of square is "+l\*b);

        }

        Areas(float r){

            System.out.println("Area of square is "+3.142\*r\*r);

        }

    }

    public class AreaConstOL {

        public static void main(String[] args) {

            new Areas(12); //Square

            new Areas(12\*4); //Rectangle

            new Areas((float)5); //Circle

        }

    }

**Output** :

