USN : 2GI19CS175 **Student Name** : Venkatesh G Dhongadi

Title of the Experiment(implementation of method overloading.**)**

Experiment No. 5

Date :_24/12/2020______

Problem Statement :

5.1) Create a Stack class having an integer array say elem and top_of_stack as instance variables. Define three overloaded methods having the following signatures:

a. initStack(int size) to create an array of specified size and initialize the top_of_stack
b. initStack(Stack another) to intialize the Stack object with state of the Stack object "another"
c. initStack(int [] a) to initialize contents of a[] to the instance variable elem.

Write following methods:

a. push(): Pushes the element onto the stack,
b. pop(): Returns the element on the top of the stack, removing it in the process, and
c. peek(): Returns the element on the top of the stack, but does not remove it.

Also write methods that check whether stack is full and stack is empty and return boolean value true or false appropriately.

Objectives of the Experiment :

1. Learn declaration and initialization of variables
2. Learn necessary of method overloading
3. Learn advantages of method overloading
4. Understand the overloaded methods in a real-life application
5. Learn the usage of Looping constructs and control statements
6. Learn to Display the result in a readable/proper format

Program Source code:

```
package termWork5a;
```

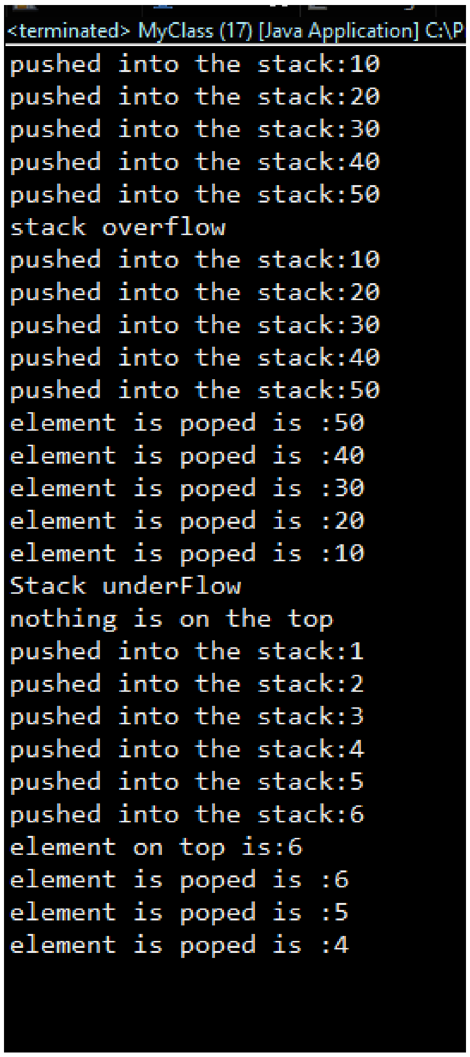
```
class MyStack{
        int []elem;
        int top;
        void initStack(int size) {
                elem=new int[size];
                top=-1;
        }
        void initStack(MyStack another) {
                elem=new int[another.elem.length];
                top=-1;
                for(int e:another.elem) {
                        push(e);

                }
        }
        void initStack(int []a) {
                elem=new int[a.length];
                top=-1;
                for(int e:a) {
                        push(e);

                }
        }
         void push(int e) {
                if(isfull()) {
                        System.out.println("stack overflow");
                }
                else
                {       elem[++top]=e;
                System.out.println("pushed into the stack:"+e);
                }
        }
         void pop() {
                 if(isEmpty())
                         System.out.println("Stack underFlow ");

                 else
                         System.out.println("element is poped is :"+ elem[top--]);




         }
         void peek()
         { if(isEmpty())
                 System.out.println("nothing is on the top ");
         else
                 System.out.println("element on top is:"+elem[top]);
```

```
		}
		boolean isfull()
		{
				return top==elem.length-1 ? true:false;



		}
		boolean isEmpty() {
				return top==-1 ?true:false;

		}
}
public class MyClass {

		public static void main(String[] args) {
				MyStack s1= new MyStack();
				s1.initStack(5);
s1.push(10);
s1.push(20);
s1.push(30);
s1.push(40);
s1.push(50);
s1.push(60);

MyStack s2=new MyStack();
s2.initStack(s1);
s2.pop();
s2.pop();
s2.pop();
s2.pop();
s2.pop();
s2.pop();
s2.peek();
int[]a = {1,2,3,4,5,6};
MyStack s3=new MyStack();
s3.initStack(a);
s3.peek();
s3.pop();
s3.pop();
s3.pop();




		}
```

}

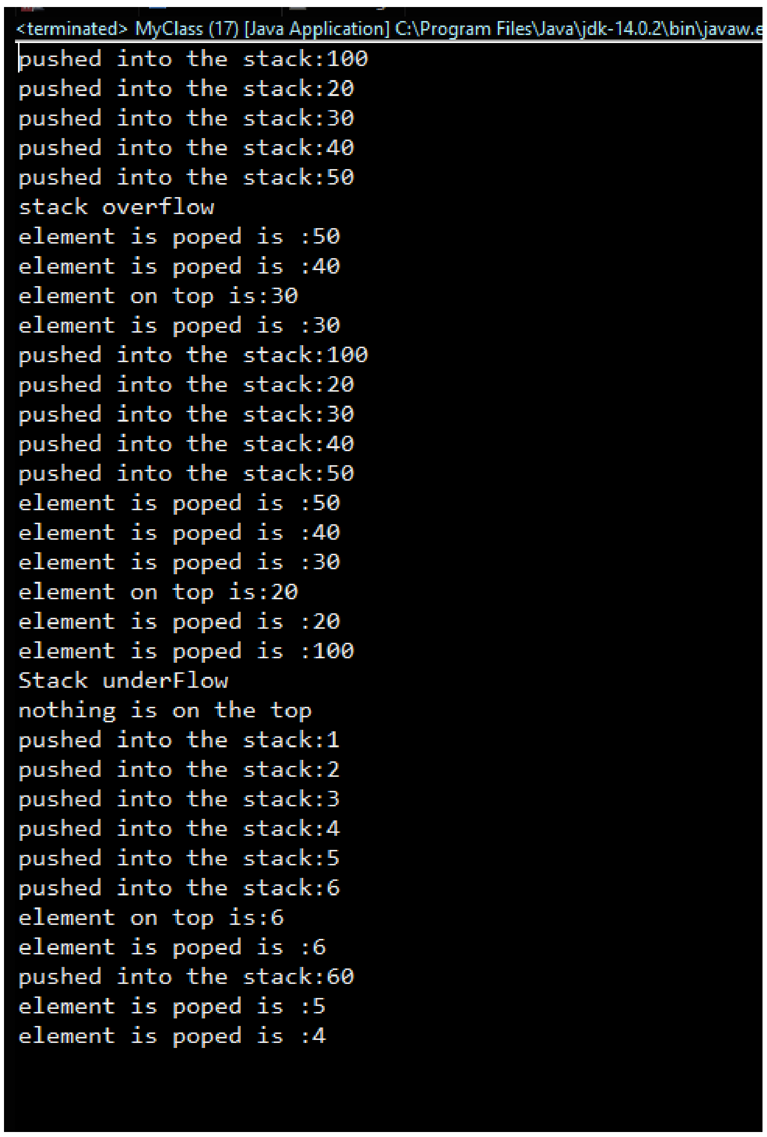
Input & Output:

Case1:



```
<terminated> MyClass (17) [Java Application] C:\P
pushed into the stack:10
pushed into the stack:20
pushed into the stack:30
pushed into the stack:40
pushed into the stack:50
stack overflow
pushed into the stack:10
pushed into the stack:20
pushed into the stack:30
pushed into the stack:40
pushed into the stack:50
element is poped is :50
element is poped is :40
element is poped is :30
element is poped is :20
element is poped is :10
Stack underFlow
nothing is on the top
pushed into the stack:1
pushed into the stack:2
pushed into the stack:3
pushed into the stack:4
pushed into the stack:5
pushed into the stack:6
element on top is:6
element is poped is :6
element is poped is :5
element is poped is :4
```

Case 2:



```
<terminated> MyClass (17) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.e
pushed into the stack:100
pushed into the stack:20
pushed into the stack:30
pushed into the stack:40
pushed into the stack:50
stack overflow
element is poped is :50
element is poped is :40
element on top is:30
element is poped is :30
pushed into the stack:100
pushed into the stack:20
pushed into the stack:30
pushed into the stack:40
pushed into the stack:50
element is poped is :50
element is poped is :40
element is poped is :30
element on top is:20
element is poped is :20
element is poped is :100
Stack underFlow
nothing is on the top
pushed into the stack:1
pushed into the stack:2
pushed into the stack:3
pushed into the stack:4
pushed into the stack:5
pushed into the stack:6
element on top is:6
element is poped is :6
pushed into the stack:60
element is poped is :5
element is poped is :4
```

Outcomes of the Experiment :

At the end of the laboratory sessions the students should be able to

1. Demonstrate the use of Classes and their member methods to solve real life problems.
2. Identify appropriate variables and their types
3. Learn how to create class, methods and object of class
4. Learn how to pass parameter to the class
5. Learn method overloading
6. Learn how data validation is important

Conclusions :

From the given problem statement, we could identify the necessary variables of appropriate type, and conditional statements and the necessary program logic. The program was written in Eclipse IDE by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We also understood the use of built-in class System and its method println to display the result. The program was executed for two-three sets of input and result obtained were verified to be correct and recorded

2)(practice) Implement a linear search function by using method overloading concept for an array of integers, double and character elements

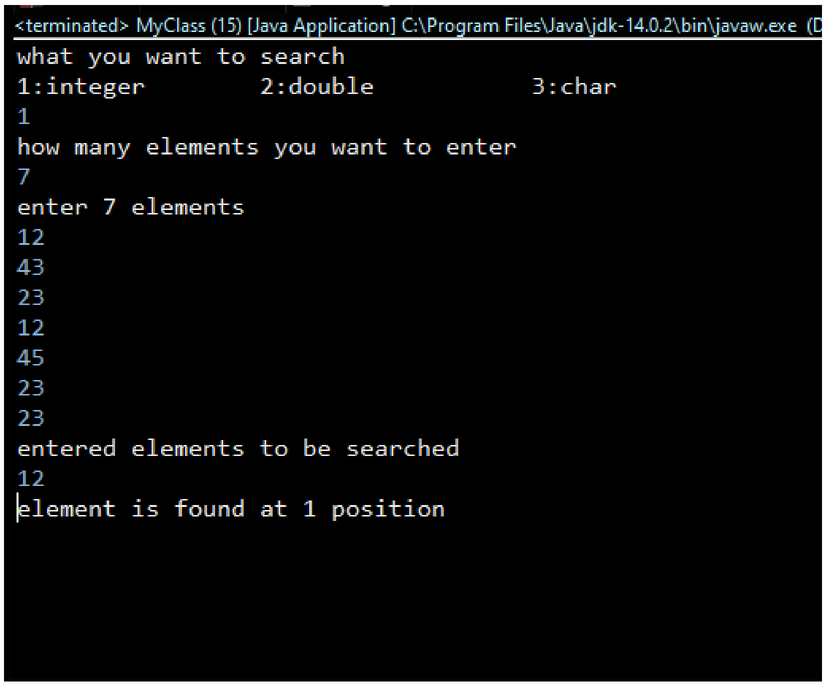
Program Source code:

```
package termWork5;
import java.util.Scanner;


class MyLinearSearch{

        int linearSearch(int[] c,int key2) {
                for(int i=0;i<c.length;i++) {
        if(c[i]==key2)
        {
        return i;
        }

                }

                        return -1;


        }
        double linearSearch(double []a,double key) {

                for(int i=0;i<a.length;i++)
                        if(a[i]==key)
                                return i;

                                return -1;

        }
        int linearSearch(char []a,char key) {

                for(int i=0;i<a.length;i++)
                        if(a[i]==key)
                                return i;

                                return -1;

        }


}

public class MyClass {

        public static void main(String[] args) {

        MyLinearSearch m=new MyLinearSearch();
                Scanner in=new Scanner(System.in);
                System.out.println("what you want to search");
        System.out.println("1:integer \t 2:double  \t    3:char");
                int choice=in.nextInt();
                System.out.println("how many elements you want to enter");
                int n=in.nextInt();
                System.out.println("enter "+ n+" elements ");
                switch(choice) {

                        case 1: int[]a=new int[n];
                                for(int i=0;i<n;i++)
                                        a[i]=in.nextInt();
                System.out.println("entered elements to be searched");
                int key=in.nextInt();
                int position=m.linearSearch(a, key);
                check(position);
```

```
				break;
			case 2:
				double[]a1=new double[n];
				for(int i=0;i<n;i++)
					a1[i]=in.nextDouble();
				System.out.println("entered elements to be searched");
				double key1=in.nextInt();
				int position1=(int)m.linearSearch(a1, key1);
				check(position1);
				break;
			case 3:
				char[]c=new char[n];
				for(int i=0;i<n;i++)
					c[i]=in.next().charAt(0);
				System.out.println("entered elements to be searched");
				char key2= in.next().charAt(0);
				int position2=(int)m.linearSearch(c, key2);
				check(position2);
				break;
				}



		}

		private static void check(int position1) {
			// TODO Auto-generated method stub
			if(position1>=0)
				System.out.println("element is found at "+(position1+1)+" position");
			else
				System.out.println("element is not found\n");
		}


	}
```

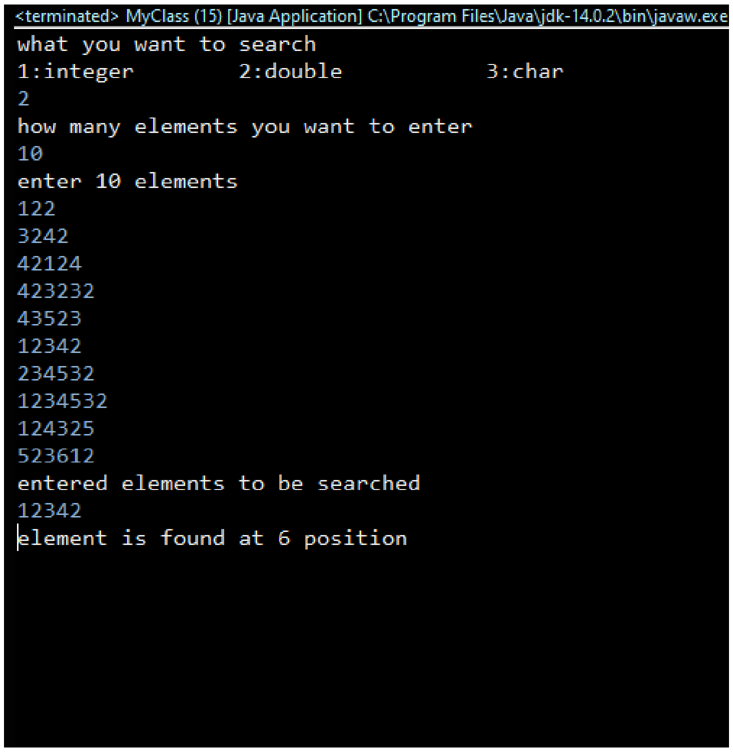
Input&Output:

Case1:



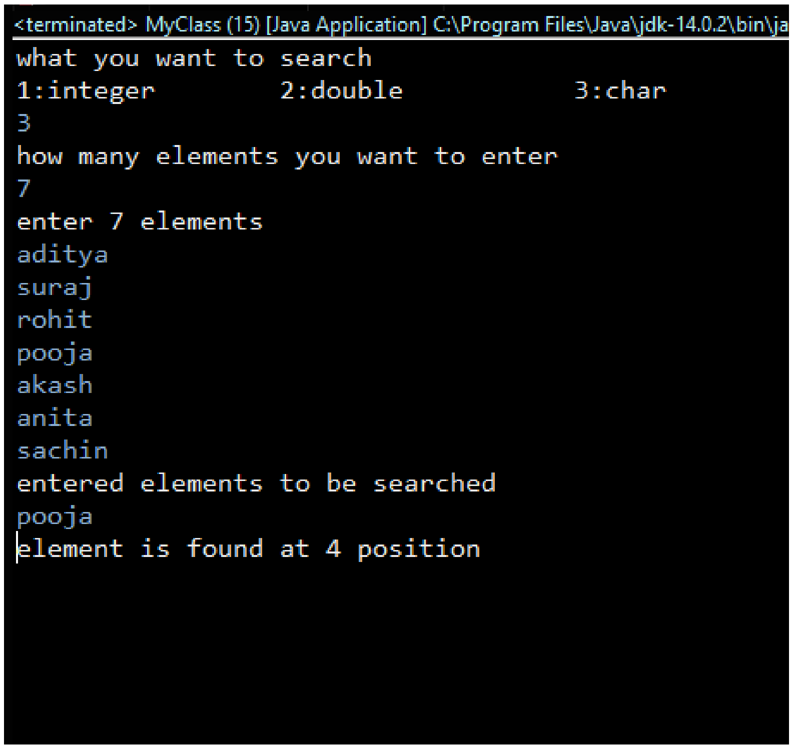
```
<terminated> MyClass (15) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe  (D
what you want to search
1:integer        2:double           3:char
1
how many elements you want to enter
7
enter 7 elements
12
43
23
12
45
23
23
entered elements to be searched
12
element is found at 1 position
```

Case2



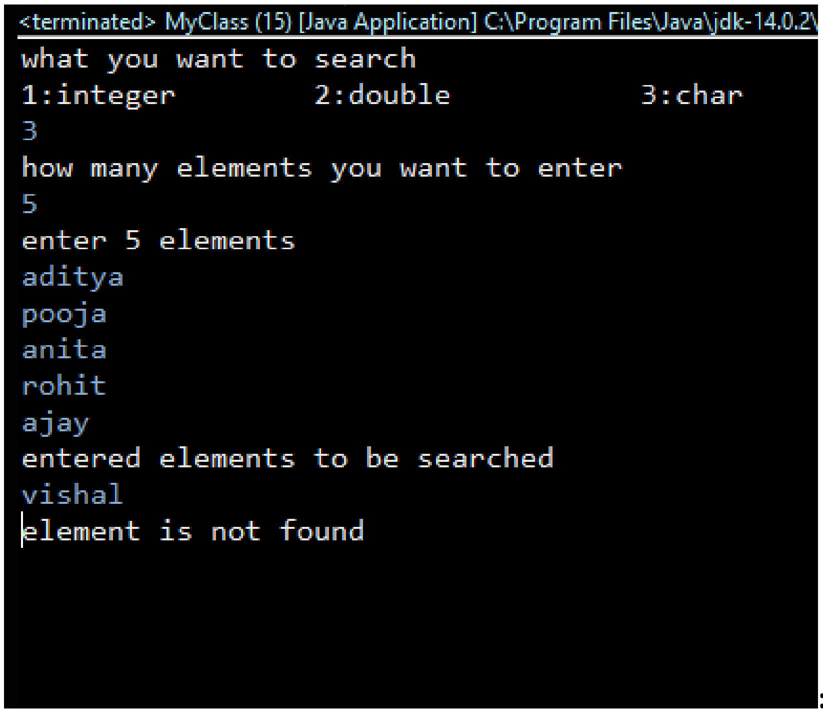
```
<terminated> MyClass (15) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe
what you want to search
1:integer        2:double          3:char
2
how many elements you want to enter
10
enter 10 elements
122
3242
42124
423232
43523
12342
234532
1234532
124325
523612
entered elements to be searched
12342
element is found at 6 position
```

Case3:



```
<terminated> MyClass (15) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\ja
what you want to search
1:integer        2:double          3:char
3
how many elements you want to enter
7
enter 7 elements
aditya
suraj
rohit
pooja
akash
anita
sachin
entered elements to be searched
pooja
element is found at 4 position
```

Case4



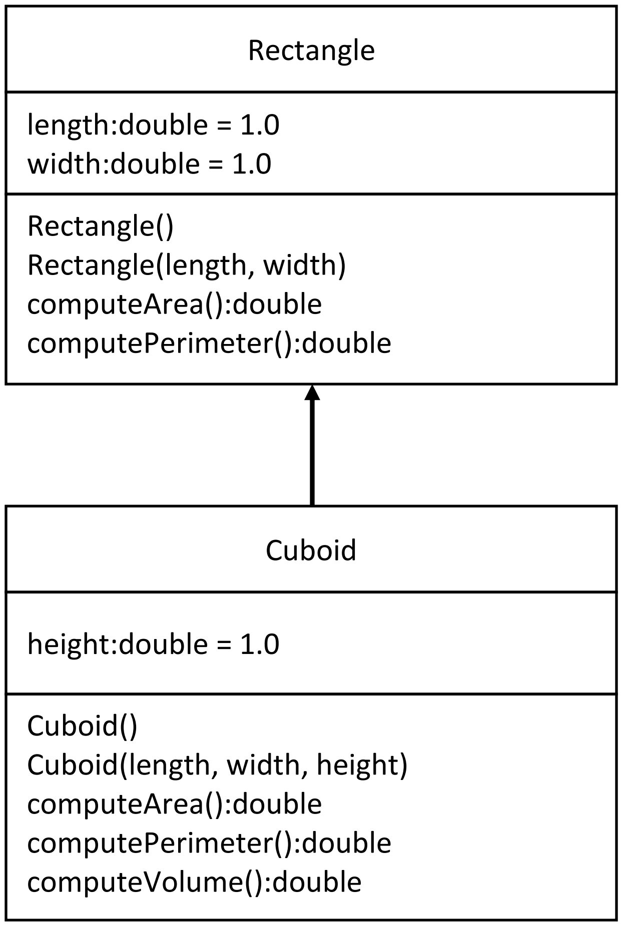
```
<terminated> MyClass (15) [Java Application] C:\Program Files\Java\jdk-14.0.2\
what you want to search
1:integer        2:double           3:char
3
how many elements you want to enter
5
enter 5 elements
aditya
pooja
anita
rohit
ajay
entered elements to be searched
vishal
element is not found
```

:

5 b) Overriding.

5 b.1)Implement the following class hierarchy. In the Cuboid class, override the method computeArea() and computePerimeter() of Rectangle class to compute the surface area and perimeter of a rectangle cuboid. Add a method computeVolume() in Cuboid class to compute volume of the cuboid. Assuming length, width and height as l, w and h respectively,

- formula to find the surface area = 2(lw) + 2(hl) + 2(hw)
- formula to find the perimeter = 2l + 2w
- formula to find the volume = l x w x h

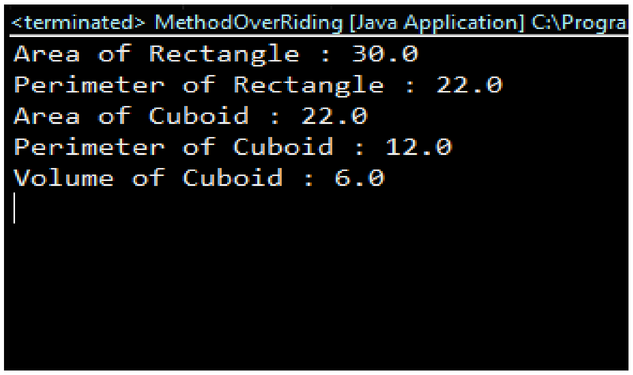


Program source code:

```
import java.io.*;
import java.lang.*;
import java.util.*;

class Rectangle
{
  protected double width,length;

  public Rectangle(double width, double length){
    this.width = width;
    this.length=length;
  }

  public double computeArea()
  {
    return(length*width);
  }

  public double computePeri()
  {
    return(2*(length+width));
  }
}

class Cuboid extends Rectangle
{
  double height;
  public Cuboid(double height,double width, double length)
  {
    super(width,length);
    this.height=height;
  }
  public double computeArea()
  {
    return(2*(length*width+height*length+height*width));
  }
  public double computePeri()
  {
    return(2*(length+height+width));
  }
  public double computeVolume()
  {
    return(length*height*width);
  }

}

public class MethodOverRiding
{
  public static void main(String[]args)
  {
    Rectangle r1=new Rectangle(5,6);
    System.out.println("Area of Rectangle : "+r1.computeArea());
    System.out.println("Perimeter of Rectangle : "+r1.computePeri());

    Cuboid c1= new Cuboid(1,2,3);
    System.out.println("Area of Cuboid : "+c1.computeArea());
    System.out.println("Perimeter of Cuboid : "+c1.computePeri());
    System.out.println("Volume of Cuboid : "+c1.computeVolume());
```

```
   }}
```
INPUT&OUTPUT:

CASE1:



```
<terminated> MethodOverRiding [Java Application] C:\Progra
Area of Rectangle : 30.0
Perimeter of Rectangle : 22.0
Area of Cuboid : 22.0
Perimeter of Cuboid : 12.0
Volume of Cuboid : 6.0
```