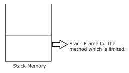
What is recursive function/method in java ?
---

* If a function/method is calling **itself directly OR in-directly then It is called reursive function.**
* As we know **methods are executed** in a special memory area Stack memory area using LIFO.
* If we call the method itself which is recursive call then there is chance of getting StackOverflowError which is runtime error.



The program which throws StackoverFlowError due to recursive call.

```
package com.nit.blc;

public class FactorialUsingRecursive
{
   public static int getFactorial(int num)  //num = 5 4 3 2 1
   {


      //Recursive call
     return  num * getFactorial(num -1);  //5 X 4 X 3 X 2 X 1
   }
}

package com.nit.elc;

import com.nit.blc.FactorialUsingRecursive;


//ELC
public class ELC
{
    public static void main(String[] args)
    {
        int number = Integer.parseInt(IO.readLn("Enter a number :"));
        IO.println("The factorial of "+number+" is
:"+FactorialUsingRecursive.getFactorial(number));


    }
}
```

* In order to avoid the StackOverflowError we must need to pass the **Base condition**

```
package com.nit.blc;

public class FactorialUsingRecursive
{
   public static int getFactorial(int num)  //num = 5 4 3 2 1
   {
       if(num==1)
       {
           return 1;  //Terminate the logic & return the value 1 by using Base condition
       }


       //Recursive call
      return  num * getFactorial(num -1);  //5 X 4 X 3 X 2 X 1
   }
}

package com.nit.elc;

import com.nit.blc.FactorialUsingRecursive;


//ELC
public class ELC
{
    public static void main(String[] args)
    {
        int number = Integer.parseInt(IO.readLn("Enter a number :"));
        IO.println("The factorial of "+number+" is
:"+FactorialUsingRecursive.getFactorial(number));


    }
}
```

//Find the reverse of a number using concatenation (String return type)

```
package com.nit.blc;

public class ReverseByConcatenation
{
    public static String getReverseOfTheNumber(int num)  //1
    {
        String reverse = ""; //321

        while(num !=0)
        {
            int digit = num%10;  //digit = 1
            reverse = reverse + digit;
            num = num/10;
        }

        return reverse;
    }
}

package com.nit.elc;

import com.nit.blc.ReverseByConcatenation;

//ELC
public class ELC
{
    public static void main(String[] args)
    {
        int number = Integer.parseInt(IO.readLn("Enter a number :"));
        IO.println("Reverse of "+number+" is
:"+ReverseByConcatenation.getReverseOfTheNumber(number));


    }
}
```

//Find the reverse of a number without String (without concatenation)

```
package com.nit.blc;

public class ReverseOfNumber
{
  public static int getReverseOfNumber(int num)
  {
      int reverse = 0;

      while(num !=0)
      {
          int digit = num % 10;
          reverse = reverse * 10 + digit;
          num = num /10;
      }
       return reverse;
  }
}

/*
num = 123
reverse = 0 -> 3 -> 32

digit 3  ->  reverse = 3  //Pass
digit 2  ->  reverse = 32  //Pass
digit 1  ->  reverse = 321  //pass

reverse = reverse * 10 + digit

*/

package com.nit.elc;

import com.nit.blc.ReverseOfNumber;

//ELC
public class ELC
{
    public static void main(String[] args)
    {
        int number = Integer.parseInt(IO.readLn("Enter a number :"));
        IO.println("Reverse of "+number+" is :"+ReverseOfNumber.getReverseOfNumber(number));


    }
}
```

Method return type as a String with Data Validation :

```
package com.nit.blc;

public class Circle
{
    public static String getAreaOfCircle(double radius)
    {
        //Data Validation
        if(radius<=0)
        {
            System.err.println("Error :Radius cannot be zero OR Nagative");
            System.exit(0);  //shut down the JVM
        }

        final double PI = 3.14;
        double area = PI * radius * radius;
        return ""+area;

    }
}

package com.nit.elc;

import com.nit.blc.Circle;

//ELC
public class ELC
{
    public static void main(String[] args)
    {
      double radius = Double.parseDouble(IO.readLn("Enter the radius :"));
      String areaOfCircle = Circle.getAreaOfCircle(radius);

      //Converting the String into double value
      double area = Double.parseDouble(areaOfCircle);

      System.out.printf("Area of circle is :%.2f",area);

    }
}
```

Method return type as a String

If a method return type is String so we can append OR concatnate **multiple values of different types in a single statement as shown in the program below**

```
package com.nit.blc;

public class Employee
{
  public static String getEmployeeDetails(int eno, String name, double salary)
  {
      //[Employee number is : 101, Name is : Scott, Salary is :45000];

      return "[Employee number is :"+eno+", Name is :"+name+", Salary is :"+salary+"]";
  }
}

package com.nit.elc;

import com.nit.blc.Employee;

//ELC
public class ELC
{
    public static void main(String[] args)
    {
      int eno = Integer.parseInt(IO.readLn("Enter Employee Id:"));
      String name = IO.readLn("Enter Employee Name :");
      double salary = Double.parseDouble(IO.readLn("Enter the Salary :"));

      String employeeDetails = Employee.getEmployeeDetails(eno, name, salary);
      IO.println(employeeDetails);

    }
}
```

/*Check a number is Palindrome or not.
A number is a palindrome If it reads same forward and backward.
Example 121
***/**

```
package com.nit.blc;

public class FindingPalindrome
{
    public static  boolean isPalindrome(int num)  //121
    {
       int temp = num;    //temp = 121

       int reverse = 0;   //reverse = 121

       while(num !=0)
       {
          int digit = num % 10;
          reverse = (reverse * 10) + digit;
          num = num/10;
       }

       return temp==reverse;

    }
}

package com.nit.elc;

import com.nit.blc.FindingPalindrome;

//ELC
public class ELC
{
    public static void main(String[] args)
    {
      int number = Integer.parseInt(IO.readLn("Enter a number :"));
      boolean isPalindrome = FindingPalindrome.isPalindrome(number);

      if(isPalindrome)
      {
        IO.println(number+" is a palindrome number");
      }
      else
      {
         IO.println(number+" is a not palindrome number");
      }

    }
}
```