**03/07/2025:**

Recursion is nothing but when the function calls itself is known as recursion.

1. Write a java code to find the factorial of a number.

Solution:

For finding a factorial of a number:

Step 1: We will create a base class.

Step 2: Recursion must stop somewhere.

Step 3: Factorial of 0 or 1 is defines as 1.

So we write:

If(n=0 || n==1){

` return 1;}

Step 4: No we will write a **recursive case**:

Return n\*fact(n-1);

It multiplies n by the factorial of the smaller number(n-1).

Step 5: Now we will see how it works for n=5.

**Working:**

Factorial(5)=5\*Factorial(4)

Factorial(4)=4\*Factorial(3)

Factorial(3)=3\*Factorial(2)

Factorial(2)=2\*Factorial(1)

Factorial(1)=1\*Factorial(0)

Then it returns Factorial(2) = 2\*Fact(1)

Factorial(3)=3\*Fact(2)

Factorial(4)=4\*Fact(3)

Factorial(5)=5\*Fact(4)

**What happens ins memory in stack?**

Java class stack handles this recursive factorial function. Every time we call a method in java. It creates a **stack frame** with its own local variables and the return address.

As a recursion goes deeper each call has its own stack frame.

Lets trace with factorial 5.

1. Main calls factorial(5): It means stack frame pushed fact(5), now n = 5.
2. Waiting for factorial(4) to return.
3. Factorial(5) call factorial(4), stack frame pushed factorial(4). Now, n=4.
4. Waiting for factorial(3) to return, factorial(4) call fact(3), now n= 3.
5. Waiting for fact(2) to return.
6. Now fact(3) call fact(2). Stack frame pushed fact(2). Now, n= 2.
7. Waiting for fact(1) to return.
8. Factorial(2) call fact(1) stack frame pushed fact(1) and **base case reached**, returns 1

**Unwinding the stack:**

Base condition reached.

Now, the stack unwinds in reverse order:  
SO, the factorial(1) returns 1.

Stack frame for n=1 is **POP**.

Result goes to n=2;

|  |
| --- |
| **Fact(0)** |
| **Fact(1)** |
| **Fact(2)** |
| **Fact(3)** |
| **Fact(4)** |
| **Fact(5)** |

Factorial(2) computes 2\*1=2

Returns 2;

Stack frame for n=2 is **POP.**

Factorial(3) computes 3\*2=6

Returns 6;

Stack frame for n=3 is **POP**

Factorial(4) computes 4\*6=24

Returns 24;

Stack frame for n=4 is **POP**

Factorial(5) computes 5\*24=120

Stack frame for n=5 is **POP**

Returns 120;

Why it is important?

The call stack holds separate local ‘n’ values for each call. Too many recursive calls(say factorial of a huge number) can cause **stack overflow error** because the stack is finite in size.

Question:

1. Print a string in a reverse order recursively.

Example: i/p: “Hello”

o/p:”olleH”

**Classes and Objects:**

1. Create a class employee with name and salary fields. Add a method to give a bonus of 10% to salary. Demonstrate the use of multiple objects with diff initial salaries.

Whenever we are using OOPs first features are classes and objects, which we have seen through multiple examples. The other feature of OOPs are:

1. Encapsulation
2. Abstraction
3. Polymorphism
4. Inheritance