Java Recursion

Topics : JAVA
Written on April 07, 2023

In Java, recursion is a technique in which a method calls itself to solve a problem. Recursion is a powerful tool that allows you to solve complex problems by breaking them down into simpler sub-problems.

To implement recursion in Java, you need to follow these steps:

1. Identify the base case: The base case is the condition where the recursion ends. It is the simplest possible case that the method can handle. You need to define the base case to avoid infinite recursion.

2. Identify the recursive case: The recursive case is the condition where the method calls itself with a smaller version of the original problem. This step allows you to break down the problem into smaller sub-problems.

3. Define the recursive method: The recursive method should call itself with the smaller version of the problem until it reaches the base case.

Here is an example of how to use recursion in Java to calculate the factorial of a number:

Example :

```
public static int factorial(int n) {
   // Base case: If n is 0 or 1, return 1
   if (n == 0 || n == 1) {
      return 1;
   }
   // Recursive case: Call the factorial method with n-1
   else {
      return n * factorial(n - 1);
   }
}
```

In this example, the factorial method calls itself with n-1 until it reaches the base case where n is 0 or 1. The method then returns 1, which allows the recursion to unwind and return the final result.

Recursion can be a powerful tool, but it can also be dangerous if not used correctly. If you don't define a base case or the base case is not reachable, you can end up with infinite recursion, which can cause your program to crash. It is important to carefully design your recursive methods and test them thoroughly.

Example :

```
import java.util.*;

public class Aryatechno {
public static void main(String[] args) {
int result = factorial(10);
System.out.println(result);

}

public static int factorial(int n) {
// Base case: If n is 0 or 1, return 1
if (n == 0 || n == 1) {
return 1;
}
// Recursive case: Call the factorial method with n-1
else {
return n * factorial(n - 1);
}
}


}
```

Output :

```
3628800
```