Table of Contents

## 0.1 ClassDef Main

# 1 Class: Main

## 1.1 Overview

The Main class serves as a container for several static utility methods and the primary entry point for the Java application.

## 1.2 attributes

This class does not have any instance attributes. It is a collection of static methods.

## 1.3 Description

The Main class is a utility class that provides a set of simple, independent functions. As all methods are declared static, they belong to the class itself and can be called directly without needing to create an instance of the Main class.

The class contains the following methods:

* add(int a, int b): This function takes two integers, a and b, as input and returns their sum.
* factorial(int n): This function calculates the factorial of a given integer n using recursion. The base case for the recursion is when n is less than or equal to 1, at which point it returns 1. For any n greater than 1, it returns the product of n and the factorial of n - 1.
* greet(String name): This is a void function that accepts a String name and prints a personalized greeting message to the standard output console.
* main(String[] args): This is the standard entry point for a Java application. The code within this method is executed when the program starts. It demonstrates the usage of the other three methods by calling them with sample values and printing the results.

// Example of the factorial logic  
if (n <= 1)  
 return 1;  
else  
 return n \* factorial(n - 1);

## 1.4 Usage Notes

* All methods are static and should be called directly on the class (e.g., Main.add(5, 10)).
* The factorial method is recursive. Using a very large input value for n can cause a StackOverflowError.
* The greet method does not return a value; its purpose is to produce a side effect by printing to the console.

## 1.5 Example

The main method within the class provides a clear example of how to use the utility functions.

public static void main(String[] args) {  
 // Calling the add method  
 System.out.println("Sum: " + Main.add(5, 10));  
  
 // Calling the factorial method  
 System.out.println("Factorial: " + Main.factorial(5));  
  
 // Calling the greet method  
 Main.greet("Prateek");  
}

**Output:**

Sum: 15  
Factorial: 120  
Hello, Prateek!

### 1.5.1 FunctionDef add(a, b)

# 2 Function: add(int a, int b)

## 2.1 Overview

The add function calculates and returns the sum of two integer numbers.

## 2.2 parameters

| Parameter | Type | Description |
| --- | --- | --- |
| a | int | The first integer operand. |
| b | int | The second integer operand. |

## 2.3 Description

This function provides a straightforward implementation of integer addition. It accepts two integer parameters, a and b. Internally, it uses the standard arithmetic addition operator (+) to compute the sum of these two values. The resulting integer sum is then returned by the function.

// The function returns the result of a + b  
return a + b;

## 2.4 Usage Notes

* This is a static method, meaning it should be called on the class itself (e.g., Main.add(5, 10)) rather than on an instance of the class.
* The function operates within the standard range of Java’s int type. Be aware of potential integer overflow if the sum of a and b exceeds Integer.MAX\_VALUE or falls below Integer.MIN\_VALUE, which will cause the value to wrap around.

**Output Example**: The function returns a single integer value representing the sum of the inputs.

## 2.5 Example

public class Main {  
 public static int add(int a, int b) {  
 return a + b;  
 }  
  
 public static void main(String[] args) {  
 // Example usage  
 int result = add(15, 7);  
 System.out.println("The sum is: " + result);  
 }  
}

**Output:**

The sum is: 22

### 2.5.1 FunctionDef factorial(n)

# 3 Function: factorial(int n)

## 3.1 Overview

The factorial function recursively calculates the factorial of a given non-negative integer.

## 3.2 parameters

* n: int - The non-negative integer for which to calculate the factorial.

## 3.3 Description

This function provides a classic recursive implementation for calculating the factorial of a number n, denoted as n!.

The logic operates based on two conditions: 1. **Base Case**: If the input n is less than or equal to 1, the function returns 1. This is the termination condition for the recursion, as the factorial of 1 (1!) and 0 (0!) is defined as 1. 2. **Recursive Step**: If n is greater than 1, the function returns the product of n and the result of calling itself with the argument n - 1. This process continues to break down the problem into smaller subproblems until the base case is reached.

For example, factorial(4) would be calculated as 4 \* factorial(3), which expands to 4 \* 3 \* factorial(2), then 4 \* 3 \* 2 \* factorial(1), and finally 4 \* 3 \* 2 \* 1, yielding 24.

## 3.4 Usage Notes

* **Input Domain**: This function is intended for non-negative integers. Providing a negative number will result in infinite recursion, leading to a StackOverflowError.
* **Integer Overflow**: The return type is int. Factorial values grow very quickly. For inputs greater than 12, the result will exceed the maximum value of a Java int (2,147,483,647), causing an integer overflow and producing an incorrect result.
* **Recursion Depth**: As a recursive function, very large input values (even if they did not cause an overflow) could potentially lead to a StackOverflowError due to excessive recursion depth.

**Output Example**: The function returns a single integer value representing the calculated factorial. For an input of 5, the output would be 120.

## 3.5 Example

// Example usage within the Main class  
public static void main(String[] args) {  
 int number = 5;  
 int result = factorial(number);  
 System.out.println("The factorial of " + number + " is " + result);  
}

**Output:**

The factorial of 5 is 120

### 3.5.1 FunctionDef greet(name)

# 4 Function: greet

## 4.1 Overview

The greet function prints a personalized greeting message to the standard console output.

## 4.2 parameters

* name (String): The name to be included in the greeting message.

## 4.3 Description

This public static function is designed to provide a simple greeting. It accepts a single String parameter called name.

The core logic of the function involves string concatenation. It combines the literal string "Hello, ", the value of the name parameter, and the literal string "!" into a single message. This resulting string is then printed to the console using System.out.println(), which also appends a newline character at the end.

Because the function is declared as void, it does not return any value. Its sole purpose is to produce output on the console.

## 4.4 Usage Notes

* This is a static method, so it should be called on the class itself (e.g., Main.greet("World")) rather than on an instance of the class.
* The function’s output is sent directly to the standard output stream. It does not return the greeting string for further processing.
* If a null value is passed for the name parameter, the output will be “Hello, null!”.

## 4.5 Example

// Example usage within the same project  
Main.greet("Alice");

**Output:**

Hello, Alice!

### 4.5.1 FunctionDef main(args)

# 5 Function: main(String[] args)

## 5.1 Overview

The main function serves as the primary entry point for the Java application, executing a sequence of method calls to demonstrate various functionalities and print their results to the console.

## 5.2 parameters

| Parameter | Type | Description |
| --- | --- | --- |
| args | String[] | An array of strings that can receive command-line arguments passed to the application. This parameter is not utilized within the body of this specific function. |

## 5.3 Description

The main method is the starting point that the Java Virtual Machine (JVM) calls when the program begins execution. Its logic proceeds in three distinct steps:

1. It first calls the add(5, 10) method. The integer result from this method is concatenated with the string "Sum: " and the complete string is printed to the standard output console using System.out.println().
2. Next, it calls the factorial(5) method. Similar to the previous step, the returned value is concatenated with the string "Factorial: " and the result is printed to the console.
3. Finally, it calls the greet("Prateek") method, passing the string "Prateek" as an argument. This method is expected to perform an action, such as printing a greeting message directly to the console.

This method demonstrates a simple, sequential execution flow and serves as a testbed for the add, factorial, and greet functions.

## 5.4 Usage Notes

* This method is the designated entry point for a Java application and is automatically invoked by the JVM. It should not be called directly from other parts of your code.
* For this method to compile and run correctly, the add, factorial, and greet methods must be defined and accessible from within the Main class.
* The arguments passed to the called methods (5, 10, and "Prateek") are hardcoded. Therefore, the program will produce the same output each time it is executed, unless the code is modified.

## 5.5 Example

/\*\*  
 \* The main method serves as the entry point for the application.  
 \* It demonstrates calls to other methods and prints their results.  
 \*/  
public static void main(String[] args) {  
 // Calls the add method and prints the sum  
 System.out.println("Sum: " + add(5, 10));  
  
 // Calls the factorial method and prints the result  
 System.out.println("Factorial: " + factorial(5));  
  
 // Calls the greet method to print a message  
 greet("Prateek");  
}

**Output:**

Assuming standard implementations for the called methods, the console output would be:

Sum: 15  
Factorial: 120  
// Plus any output generated by the greet("Prateek") call.