Table of Contents

## 0.1 ClassDef Main

# 1 Class: Main

## 1.1 Overview

The Main class is a container for several static utility methods, including a function for addition, a recursive function for factorial calculation, and a procedure to print a greeting.

## 1.2 Description

The Main class provides a set of standalone, static methods that can be called without creating an instance of the class. It also contains the primary main method, which serves as the entry point for the Java application.

The class includes the following methods:

* public static int add(int a, int b): This function takes two integer parameters, a and b, and returns their sum.
* public static int factorial(int n): This function calculates the factorial of a non-negative integer n using recursion. The base case for the recursion is when n is less than or equal to 1, in which case it returns 1. Otherwise, it returns n multiplied by the factorial of n - 1.
* public static void greet(String name): This is a void method that takes a String parameter name and prints a formatted greeting message, “Hello, [name]!”, to the standard output console.
* public static void main(String[] args): This is the main entry point of the program. It demonstrates the usage of the add, factorial, and greet methods and prints their results to the console.

## 1.3 Usage Notes

* All methods in this class are static, meaning they should be called directly on the class itself (e.g., Main.add(5, 10)), and no object instantiation is required.
* The factorial method is recursive. Inputting a very large number can lead to a StackOverflowError.
* The greet method prints directly to the console and does not return a value.

## 1.4 Example

The main method within the class provides a clear example of how to use the other static methods.

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

**Output:**

Sum: 15  
Factorial: 120  
Hello, Prateek!

### 1.4.1 FunctionDef add

# 2 Function: add

## 2.1 Overview

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

## 2.2 parameters

| Parameter | Type | Description |
| --- | --- | --- |
| a | int | The first integer to be added. |
| b | int | The second integer to be added. |

## 2.3 Description

The add function is a public static method that performs a simple arithmetic addition. It takes two integer parameters, a and b. Inside the function, the addition operator (+) is used to compute the sum of these two numbers. The resulting integer value is then returned by the function.

Because the method is static, it can be called directly on the Main class without needing to create an instance of the class.

return a + b;

## 2.4 Usage Notes

* This function is designed specifically for int data types. Providing arguments of other types will cause a compile-time error.
* The return value is also an int. Be mindful of potential integer overflow if the sum of a and b exceeds the maximum value for an int (2,147,483,647). If an overflow occurs, the value will wrap around to the negative range.

**Output Example**: A single integer representing the sum.

## 2.5 Example

// Example usage within the Main class  
public static void main(String[] args) {  
 int number1 = 15;  
 int number2 = 10;  
 int result = add(number1, number2);  
 System.out.println("The sum is: " + result);  
}

**Output:**

The sum is: 25

### 2.5.1 FunctionDef factorial

# 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 the factorial will be computed.

## 3.3 Description

This function provides a classic recursive implementation to compute the factorial of a number n (denoted as n!).

The logic follows the mathematical definition of a factorial: 1. **Base Case**: The factorial of 0 (0!) and 1 (1!) is 1. The function checks if the input n is less than or equal to 1. If it is, the function terminates the recursion and returns 1. 2. **Recursive Step**: If n is greater than 1, the function multiplies n by the result of calling itself with the argument n - 1.

For example, factorial(4) is computed as follows: - factorial(4) returns 4 \* factorial(3) - factorial(3) returns 3 \* factorial(2) - factorial(2) returns 2 \* factorial(1) - factorial(1) returns 1 (base case is met)

The results are then multiplied up the call stack: 2 \* 1 = 2, then 3 \* 2 = 6, and finally 4 \* 6 = 24.

// Base case for n=1 or n=0  
if (n <= 1)  
 return 1;  
// Recursive step for n > 1  
else  
 return n \* factorial(n - 1);

## 3.4 Usage Notes

* This function is designed for non-negative integers. Passing a negative value for n will cause infinite recursion, leading to a java.lang.StackOverflowError.
* The return type is int. Factorial values grow very quickly, and an integer overflow will occur for n > 12. The result of an overflow will be an incorrect, often negative, value. For calculating factorials of larger numbers, consider using a data type with a larger range, such as long or java.math.BigInteger.

**Output Example**: A single integer representing the calculated factorial. 120

## 3.5 Example

// Example usage of the factorial function  
public class Main {  
 public static int factorial(int n) {  
 if (n <= 1)  
 return 1;  
 else  
 return n \* factorial(n - 1);  
 }  
  
 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

# 4 Function: greet(String name)

## 4.1 Overview

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

## 4.2 parameters

* name: String - The name of the person or entity to greet. This string will be incorporated into the output message.

## 4.3 Description

This static method provides a simple way to display a standardized greeting. It accepts a single String parameter, name. The core logic involves string concatenation: it combines the literal string "Hello, ", the value passed in the name parameter, and the literal string "!". The resulting string is then printed to the standard console output using System.out.println(), which also appends a newline character at the end. Because the method is declared as void, it does not return any value.

## 4.4 Usage Notes

* This is a static method and should be called on the class, for example, Main.greet("World").
* The method prints directly to the console; it does not return the greeting string.
* If a null value is passed as the name, the output will be “Hello, null!”.

## 4.5 Example

public class Main {  
 public static void greet(String name) {  
 System.out.println("Hello, " + name + "!");  
 }  
  
 public static void main(String[] args) {  
 // Example usage  
 greet("Alice");  
 greet("Developer");  
 }  
}

**Output:**

Hello, Alice!  
Hello, Developer!

### 4.5.1 FunctionDef main

# 5 Function: main

## 5.1 Overview

The main function serves as the entry point for the Java application, executing a sequence of calls to other methods and printing their results to the console.

## 5.2 parameters

| Parameter | Type | Description |
| --- | --- | --- |
| args | String[] | An array of String objects that can hold command-line arguments passed to the application upon execution. This parameter is not utilized in this function’s body. |

## 5.3 Description

The main method is the starting point for the execution of the program. It is declared as public and static, meaning it can be called by the Java Virtual Machine (JVM) without creating an instance of the Main class.

The method performs the following operations in sequence:

1. It calls the add(5, 10) method. The return value of this call is concatenated with the string "Sum: " and the entire resulting string is printed to the standard output console.
2. It calls the factorial(5) method. The integer result is concatenated with the string "Factorial: " and printed to the console.
3. 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, as it is a standalone statement.

This function demonstrates a simple procedural execution flow and the integration of different helper methods (add, factorial, greet) to perform specific tasks.

## 5.4 Usage Notes

* The main method is a special method in Java that the JVM looks for as the starting point of any Java program.
* The method signature must be public static void main(String[] args) for the application to be launchable.
* This implementation depends on the existence of three other methods: add(int, int), factorial(int), and greet(String). These methods must be defined within the Main class or be otherwise accessible at compile time.

## 5.5 Example

// This is the code for the main method itself. It is executed when the  
// Java application is run.  
  
public static void main(String[] args) {  
 // Assuming add(5, 10) returns 15  
 System.out.println("Sum: " + add(5, 10));  
  
 // Assuming factorial(5) returns 120  
 System.out.println("Factorial: " + factorial(5));  
  
 // Assuming greet("Prateek") prints a greeting to the console  
 greet("Prateek");  
}

**Output:**

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

Sum: 15  
Factorial: 120  
Hello, Prateek!