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 includes the main entry point for the Java application.

## 1.2 Description

The Main class is designed as a utility class containing static methods, meaning they can be called directly on the class without needing to create an instance. It demonstrates basic programming concepts through its methods and serves as the executable entry point for the program.

The class includes the following static methods:

**public static int add(int a, int b)** This method takes two integer parameters, a and b, and returns their sum. It performs a simple addition operation.

// Example of the add method  
int result = Main.add(5, 10); // result will be 15

**public static int factorial(int n)** This method calculates the factorial of a given non-negative integer n using recursion. The base case for the recursion is when n is less than or equal to 1, at which point the method returns 1. For any n greater than 1, it returns n multiplied by the factorial of n - 1.

// Example of the factorial method  
int fact = Main.factorial(5); // fact will be 120 (5 \* 4 \* 3 \* 2 \* 1)

**public static void greet(String name)** This method takes a String parameter name and prints a formatted greeting message to the console. It is a void method, so it does not return any value.

// Example of the greet method  
Main.greet("Prateek"); // Prints "Hello, Prateek!" to the console

**public static void main(String[] args)** This is the main entry point of the application. When the program is executed, the Java Virtual Machine (JVM) calls this method. The main method in this class demonstrates the usage of the add, factorial, and greet methods and prints their results to the standard output.

## 1.3 Usage Notes

* All methods in this class are static, so they should be invoked directly on the class itself (e.g., Main.add(x, y)), not on an instance of the class.
* The factorial method uses recursion. Providing a very large integer as input may lead to a java.lang.StackOverflowError.
* The main method is the standard entry point for any Java application and is used here to demonstrate the functionality of the other methods.

## 1.4 Example

The main method within the class provides a clear example of how to use the utility functions. To run this example, compile and execute the Main.java file.

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

**Output:**

Sum: 15  
Factorial: 120  
Hello, Prateek!

### 1.4.1 FunctionDef add

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

## 2.1 Overview

The add function computes the sum of two integer numbers.

## 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 takes two integer parameters, a and b. It performs a standard arithmetic addition operation on these two integers. The core logic is simply return a + b;, which calculates the sum and immediately returns the resulting integer value. Because it is a static method, it can be called directly on the Main class without needing to create an object instance.

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

## 2.4 Usage Notes

* As a static method, it should be invoked using the class name, for example, Main.add(5, 10).
* The function operates on primitive int types. Be aware of potential integer overflow if the sum of a and b exceeds the maximum value for an int (Integer.MAX\_VALUE) or falls below the minimum value (Integer.MIN\_VALUE).
* The function does not perform any null checks or input validation.

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

15

## 2.5 Example

// Example usage  
public class Main {  
 public static int add(int a, int b) {  
 return a + b;  
 }  
  
 public static void main(String[] args) {  
 int number1 = 7;  
 int number2 = 8;  
 int sum = add(number1, number2);  
 System.out.println("The sum is: " + sum);  
 }  
}

**Output:**

The sum is: 15

### 2.5.1 FunctionDef factorial

# 3 Function: factorial

## 3.1 Overview

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

## 3.2 parameters

* n: The non-negative integer for which the factorial will be calculated.

## 3.3 Description

This function computes the factorial of a number n, which is the product of all positive integers up to n (n!). It employs a recursive approach to solve the problem.

The logic follows two main paths: 1. **Base Case**: If the input n is less than or equal to 1, the function returns 1. This is because the factorial of 0 (0!) and 1 (1!) are both defined as 1. This condition serves as the termination point for the recursion. 2. **Recursive Step**: If n is greater than 1, the function returns the product of n and the result of calling itself with n - 1.

For example, a call to factorial(4) would execute 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)

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

## 3.4 Usage Notes

* The function is intended for non-negative integers. Providing a negative integer will cause infinite recursion, leading to a StackOverflowError.
* The return type is int. Factorial values grow very rapidly. The largest factorial that can be stored in a standard 32-bit signed int is 12!. Any input greater than 12 will result in an integer overflow, producing an incorrect value. For larger numbers, a data type like long or BigInteger would be required.

**Output Example**: A call to factorial(5) would return the integer 120.

## 3.5 Example

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

## 4.1 Overview

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

## 4.2 parameters

* name (String): The name of the person or entity to be greeted.

## 4.3 Description

The greet function is a public static method, meaning it can be called directly on the Main class without needing to create an instance of the class. It is designed to perform a simple action: displaying a greeting.

The function accepts a single parameter, name, which is a String. It then uses the System.out.println() method to print a formatted string to the console. The output string is constructed by concatenating the literal string "Hello, ", the value passed in the name parameter, and the literal string "!". Because its return type is void, the function does not return any value upon completion.

For example, if the string "World" is passed as the name argument, the function will print "Hello, World!" to the console.

## 4.4 Usage Notes

* This function prints directly to the standard output and does not return the greeting string. If you need to store or manipulate the greeting message, you will need to modify the function to return a String instead.
* The function will concatenate the provided name as-is. If an empty string is passed, the output will be “Hello, !”.
* If null is passed as the argument, Java’s string concatenation will convert it to the string “null”, resulting in the output “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 primary entry point for the Java application, executing a series of method calls to demonstrate the program’s functionality.

## 5.2 parameters

| Parameter | Type | Description |
| --- | --- | --- |
| args | String[] | An array of String objects that holds any command-line arguments passed to the program upon execution. This parameter is not utilized within the current implementation. |

## 5.3 Description

The main method is the starting point of execution for the program. Its logic proceeds as follows:

1. It first calls the static method add(5, 10) to calculate the sum of two integers. The result is concatenated with the string "Sum: " and printed to the standard console output.
2. Next, it calls the factorial(5) method to compute the factorial of the number 5. The returned value is then concatenated with the string "Factorial: " and displayed on the console.
3. Finally, it invokes the greet("Prateek") method, passing a string literal as an argument. This method is expected to perform an action, such as printing a greeting message directly to the console.

// The main method demonstrates calls to other utility functions.  
System.out.println("Sum: " + add(5, 10));  
System.out.println("Factorial: " + factorial(5));  
greet("Prateek");

## 5.4 Usage Notes

* This method has a specific signature (public static void main(String[] args)) that is recognized by the Java Virtual Machine (JVM) as the entry point of the application.
* The program must be executed for this method to run; it is not intended to be called directly from other parts of the code.
* The functionality demonstrated depends on the implementations of the add, factorial, and greet methods, which are called from within main.

## 5.5 Example

To run this main method, you would compile the Main.java file and then execute the resulting class file from the command line.

**Execution:**

# Compile the Java source file  
javac Main.java  
  
# Run the compiled Java class  
java Main

**Output:**

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

Sum: 15  
Factorial: 120  
Hello, Prateek!