Table of Contents

## 0.1 ClassDef Main

# 1 Class: Main

## 1.1 Overview

The Main class serves as the entry point for the application and provides a collection of static utility methods for basic mathematical operations and console output.

## 1.2 Description

The Main class is a public class that contains several static methods. As these methods are static, they belong to the class itself and can be invoked directly without needing to create an instance of the Main class. The class includes functions for addition, factorial calculation, and printing a greeting, along with the standard main method that executes the program.

### 1.2.1 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.

### 1.2.2 public static int factorial(int n)

This method calculates the factorial of a non-negative integer n using recursion. - **Base Case**: If n is less than or equal to 1, the method returns 1. - **Recursive Step**: Otherwise, it returns the product of n and the result of calling factorial with n - 1. For example, factorial(5) is calculated as 5 \* 4 \* 3 \* 2 \* 1.

### 1.2.3 public static void greet(String name)

This method accepts a String parameter name and prints a personalized greeting message to the standard console output. Since its return type is void, it does not return any value.

### 1.2.4 public static void main(String[] args)

This is the primary entry point for the Java application. When the program is run, the code within this method is executed. It demonstrates the usage of the other three static methods (add, factorial, and greet) by calling them with predefined values and printing their results or output to the console.

## 1.3 Usage Notes

* All methods in this class are static, so they should be called directly on the Main class (e.g., Main.add(5, 10)) rather than on an instance of the class.
* The factorial method is recursive. Providing a very large integer as input may lead to a StackOverflowError due to excessive recursion depth.
* The greet method’s output is sent directly to System.out and cannot be captured as a return value from the method call.

## 1.4 Example

The following code demonstrates how to use the methods within the Main class. This is the content of the main method.

public static void main(String[] args) {  
 // Call the add method and print the sum  
 System.out.println("Sum: " + add(5, 10));  
  
 // Call the factorial method and print the result  
 System.out.println("Factorial: " + factorial(5));  
  
 // Call the greet method to print a message  
 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 calculates and returns 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 static method that performs a simple arithmetic addition. It takes two integer parameters, a and b. The core logic of the function is the expression a + b, which computes the sum of the two input values. The resulting integer sum is then returned by the function. Because it is a static method, it can be called directly on the Main class without needing to create an instance of the class.

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

## 2.4 Usage Notes

* This function is static and should be called on the class itself, 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, which is 2,147,483,647).

**Output Example**: A call to add(15, 7) would return the integer 22.

## 2.5 Example

// Example usage within the main method or another static context  
public class Main {  
 public static int add(int a, int b) {  
 return a + b;  
 }  
  
 public static void main(String[] args) {  
 int number1 = 25;  
 int number2 = 17;  
 int result = Main.add(number1, number2);  
 System.out.println("The sum is: " + result);  
 }  
}

**Output:**

The sum is: 42

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

## 3.3 Description

This function computes the factorial of an integer n, which is the product of all positive integers up to n (i.e., n!). It employs a recursive approach.

The logic is divided into two main parts:

1. **Base Case**: The recursion terminates when the input n is less than or equal to 1. In this scenario, the function returns 1. This is because the factorial of 1 (1!) is 1, and by convention, the factorial of 0 (0!) is also 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. This process continues, decrementing n by 1 in each subsequent call, until the base case is reached.

For example, a call to factorial(4) would unfold as follows: 4 \* factorial(3) 4 \* (3 \* factorial(2)) 4 \* (3 \* (2 \* factorial(1))) 4 \* (3 \* (2 \* 1)) The final result is 24.

## 3.4 Usage Notes

* The function is designed for non-negative integers. Providing a negative number will result in a StackOverflowError due to infinite recursion.
* The return type is int. Factorial values grow very rapidly. For larger values of n (typically n > 12), the result will exceed the maximum value of a standard int, leading to an integer overflow and an incorrect result. For larger numbers, a long or BigInteger return type would be more appropriate.

**Output Example**: A call to factorial(5) will return an integer value.

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) {  
 // Example usage  
 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 console.

## 4.2 parameters

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

## 4.3 Description

This public static method is designed to display a simple, formatted greeting. It accepts a single String argument named name.

The core functionality is handled by the System.out.println() method, which prints a complete line of text to the console. The text is constructed by concatenating the literal string "Hello, ", the value of the name parameter, and the literal string "!". For instance, if the name parameter is "Alice", the function will print the line "Hello, Alice!".

## 4.4 Usage Notes

* This is a void method, meaning it does not return any value. Its only effect is printing to the console.
* As a static method, it should be called directly on the class, for example, Main.greet("World"), rather than on an instance of the Main class.
* The output is always directed to the standard output stream.

## 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("Developer");  
 greet("User");  
 }  
}

**Output:**

Hello, Developer!  
Hello, User!

### 4.5.1 FunctionDef main

# 5 Function: main(String[] args)

## 5.1 Overview

The main function serves as the entry point for the Java application, demonstrating the functionality of the add, factorial, and greet methods by printing their results to the console.

## 5.2 parameters

* args: String[] - An array of String objects that allows passing command-line arguments to the application. This parameter is not utilized within the body of this function.

## 5.3 Description

The main method is the starting point of execution for the Java program. When the program is run, the Java Virtual Machine (JVM) calls this method.

The method performs the following operations in sequence:

1. It calls the add(5, 10) method to calculate the sum of two integers. The result is concatenated with the string "Sum: " and printed to the standard output console.
2. It calls the factorial(5) method to compute the factorial of the number 5. The result is then concatenated with the string "Factorial: " and printed to the console.
3. Finally, it calls the greet("Prateek") method, which is expected to print a greeting message to the console using the provided name.

This method serves as a demonstration and test driver for other utility functions within the Main class.

## 5.4 Usage Notes

* The main method must have the exact signature public static void main(String[] args) to be recognized as the application’s entry point by the JVM.
* This method is not intended to be called directly from other parts of the code; it is invoked by the JVM when the program starts.
* The args parameter can be used to provide configuration or data to the program at runtime from the command line, although it is unused in this specific implementation.

## 5.5 Example

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

// 1. Compile the Java code  
javac Main.java  
  
// 2. Run the compiled code  
java Main

**Output:**

Assuming the add, factorial, and greet methods are implemented to perform their respective tasks, the expected output would be:

Sum: 15  
Factorial: 120  
Hello, Prateek!