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 arithmetic, factorial calculation, and printing greetings.

## 1.2 attributes

This class does not have any instance attributes as it primarily consists of static methods. The methods themselves accept parameters which are detailed in the description below.

## 1.3 Description

The Main class is a container for several independent, static helper functions and the primary main method which executes the program.

**Methods:**

* public static int add(int a, int b): This function takes two integer arguments, a and b, and returns their sum. It performs a simple addition operation.
* public static int factorial(int n): This function calculates the factorial of a given non-negative integer n using recursion.
  + **Base Case**: If n is less than or equal to 1, the function returns 1.
  + **Recursive Step**: Otherwise, it returns the product of n and the factorial of n - 1.
* public static void greet(String name): This function takes a String argument name and prints a formatted greeting message to the console. It has a void return type, meaning it does not return any value.
* public static void main(String[] args): This is the main entry point of the Java application. When the program is run, this method is executed. It demonstrates the usage of the other three methods by:
  1. Calling add(5, 10) and printing the result.
  2. Calling factorial(5) and printing the result.
  3. Calling greet("Prateek") to print a message.

The main method serves as a self-contained example of how to use the utility functions within the class.

## 1.4 Usage Notes

* All methods in this class are static and should be invoked directly on the class, e.g., Main.add(5, 3). There is no need to instantiate the Main class.
* The factorial method is recursive. Providing a very large integer as input may lead to a StackOverflowError.
* The greet method prints its output directly to the standard output (console) and does not return the greeting string.

## 1.5 Example

The following code shows the complete class and its main method, which demonstrates the functionality.

public class Main {  
  
 // Function to add two numbers  
 public static int add(int a, int b) {  
 return a + b;  
 }  
  
 // Function to find factorial of a number  
 public static int factorial(int n) {  
 if (n <= 1)  
 return 1;  
 else  
 return n \* factorial(n - 1);  
 }  
  
 // Function to print a greeting message  
 public static void greet(String name) {  
 System.out.println("Hello, " + name + "!");  
 }  
  
 public static void main(String[] args) {  
 System.out.println("Sum: " + add(5, 10));  
 System.out.println("Factorial: " + factorial(5));  
 greet("Prateek");  
 }  
}

**Output:**

When the main method is executed, the following output is printed to the console:

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 computes the sum of two integer numbers.

## 2.2 parameters

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

## 2.3 Description

The add function is a public static method that takes two integer parameters, a and b. The core logic of the function is to perform a standard arithmetic addition operation on these two integers. The expression a + b calculates their sum. The resulting integer value 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 simply returns the result of the addition  
return a + b;

## 2.4 Usage Notes

* As a static method, it should be invoked directly on the class, for example, Main.add(5, 10).
* The function operates within the standard range of a Java int. If the sum of a and b exceeds Integer.MAX\_VALUE or is less than Integer.MIN\_VALUE, an integer overflow will occur, and the result will wrap around.
* The function only accepts int types. Passing other numeric types will require an explicit cast to int.

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

## 2.5 Example

// Example usage within a main method  
public static void main(String[] args) {  
 int number1 = 15;  
 int number2 = 7;  
 int result = Main.add(number1, number2);  
 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 the factorial will be computed.

## 3.3 Description

This function implements the mathematical factorial operation using recursion. The factorial of a non-negative integer n, denoted by n!, is the product of all positive integers less than or equal to n.

The logic follows two main paths: 1. **Base Case**: If the input n is less than or equal to 1 (i.e., 0 or 1), the function returns 1. This is because 0! and 1! are both defined as 1, and 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 the argument n - 1. This process repeats, decrementing n by 1 in each subsequent call, until the base case is reached.

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, yielding 4 \* 3 \* 2 \* 1 = 24.

## 3.4 Usage Notes

* This function is intended for non-negative integers. Providing a negative number will lead to infinite recursion and result in a StackOverflowError.
* Due to the recursive implementation, very large values of n can also cause a StackOverflowError by exceeding the call stack depth limit.
* The return type is int. Factorials grow very rapidly, and the result will overflow the standard 32-bit int data type for n > 12. For larger numbers, a long or java.math.BigInteger implementation would be necessary.

**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) {  
 // 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(name)

# 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 to be included in the greeting message.

## 4.3 Description

This static function provides a simple way to display a standardized greeting. It takes a single String argument, name. The core logic involves string concatenation: it combines the literal string "Hello, ", the value passed in the name parameter, and the literal string "!". This complete string is then passed to the System.out.println() method, which prints the message to the console and appends a newline character.

For example, if the name parameter is "World", the function will concatenate "Hello, ", "World", and "!" to form the final string "Hello, World!" before printing it.

// The function concatenates strings and prints the result  
System.out.println("Hello, " + name + "!");

## 4.4 Usage Notes

* This is a static method, so it should be called directly on the Main class, like Main.greet("Alice").
* The function has a void return type, meaning it does not return any value. Its only effect is printing to the console.
* If null is passed as the argument, 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("Developer");  
 }  
}

**Output:**

Hello, Developer!

### 4.5.1 FunctionDef main(args)

# 5 Function: main

## 5.1 Overview

The main function serves as the primary entry point for the Java application, demonstrating the functionality of other methods within the class by calling them with predefined values and printing the results to the console.

## 5.2 parameters

| Parameter | Type | Description |
| --- | --- | --- |
| args | String[] | An array of strings that accepts command-line arguments passed to the program at runtime. This parameter is not utilized within the current implementation of the method. |

## 5.3 Description

The main method is the starting point of execution for the Java program. The Java Virtual Machine (JVM) invokes this method when the program is run. The method is declared as public and static, allowing it to be called without creating an instance of the Main class.

The method’s execution flow is as follows:

1. It first calls the add(5, 10) method. The return value (the sum of 5 and 10) is concatenated with the string "Sum: " and printed to the standard output console.
2. Next, it calls the factorial(5) method. The calculated factorial of 5 is then concatenated with the string "Factorial: " and 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 to the console.

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 with a name  
 greet("Prateek");  
}

## 5.4 Usage Notes

* This method is the designated entry point for the application and is automatically called by the JVM. It should not be called directly from other parts of the code.
* The functionality of this method is dependent on the add, factorial, and greet methods being defined and accessible within the same scope.
* While the args parameter is defined, it is not used in this specific implementation. However, it can be used to pass external data to the application from the command line.

## 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.

**Command:**

# First, compile the Java source file  
javac Main.java  
  
# Then, run the compiled class  
java Main

**Output:**

The expected output printed to the console would be:

Sum: 15  
Factorial: 120  
Hello, Prateek!

*(Note: The output of the greet method is assumed based on its name and argument.)*