System Requirements Specification Index

For

Using Methods

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

1 Project Abstract

This project assesses knowledge of Java **methods** and their usage.
The tasks involve implementing and calling methods with no arguments, with arguments, returning values, using variable arguments, and calling one method from another method.

2 Assessment Tasks

Task 1: Calling a Method with No Arguments:

- Print the message:
  `"Task 1: Calling a Method with No Arguments"`.
- Call the method `printMessage()`.
- Create the method `printMessage()`:
  ➔ Print the message: `"Hello, welcome to method examples!"`.
- This demonstrates calling a method that doesn't take any arguments and performs an action (printing a message).

Expected Output:

```
Task 1: Calling a Method with No Arguments
Hello, welcome to method examples!
```

Task 2: Calling a Method with Arguments:

- Print the message:
  `"Task 2: Calling a Method with Arguments"`.
- Create and call the method `addNumbers()` with arguments `5` and `10`.
- The method `addNumbers(int a, int b)`:
  ➔ Accepts two integer parameters `a` and `b`.
  ➔ Returns the sum of `a` and `b`.
- Store the result in an integer variable `result`.
- Print the result:
  `"Sum of 5 and 10: <result>"`.
- This shows how to pass arguments to a method and receive a result.

Expected Output:

```
Task 2: Calling a Method with Arguments
Sum of 5 and 10: 15
```

Task 3: Calling a Method that Returns a Value:

- Print the message:
  `"Task 3: Calling a Method that Returns a Value"`.

- Create and call the method `findMax()` with arguments `15` and `20`.
- The method `findMax(int x, int y)`:
  - ➔ Accepts two integer parameters `x` and `y`.
  - ➔ Uses a ternary operator to return the maximum value between `x` and `y`.
- Store the result in an integer variable `max`.
- Print the result:
  `"Maximum Value: <max>"`.
- This demonstrates returning a value based on conditions.

Expected Output:

```
Task 3: Calling a Method that Returns a Value
Maximum Value: 20
```

Task 4: Calling a Method with Variable Arguments (Varargs):

- Print the message:
  `"Task 4: Calling a Method with Variable Arguments"`.
- Create and call the method `calculateSum()` with arguments `1, 2, 3, 4, 5`.
- The method `calculateSum(int... numbers)`:
  - ➔ Accepts a variable number of integer arguments.
  - ➔ Uses a **for-each loop** to iterate through each `num` in `numbers` and adds them to `sum`.
- Store the result in an integer variable `sum`.
- Print the result:
  `"Sum of numbers: <sum>"`.
- This demonstrates how to handle multiple arguments using varargs.

Expected Output:

```
Task 4: Calling a Method with Variable Arguments
Sum of numbers: 15
```

Task 5: Calling a Method from Another Method:

- Print the message:
  `"Task 5: Calling a Method from Another Method"`.
- Create and call the method `printMessageAndSum()` with arguments `3` and `4`.
- Inside the method `printMessageAndSum(int a, int b)`:
  - ➔ Call the method `printMessage()` to print a welcome message.
  - ➔ Call the method `addNumbers(a, b)` to calculate the sum of `a` and `b`.
  - ➔ Print the result:
    `"The sum of <a> and <b> is: <sum>"`.
- This shows how one method can call other methods.

Expected Output:

```
Task 5: Calling a Method from Another Method
Hello, welcome to method examples!
```

```
The sum of 3 and 4 is: 7
```

3 Template Code Structure

3.1 Package: com.yaksha.assignment.MethodAssignment

Resources

Class/Interface	Description	Status
MethodAssignment (class)	● Main class demonstrating usage of **methods** in Java. ● Includes examples of: - Defining and calling a method with no arguments. - Calling a method with arguments. - Calling a method that returns a value. - Using variable arguments in a method. - Calling a method from another method.	Need to be implemented.

4 Execution Steps to Follow

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) Terminal New Terminal.**
3. **This editor Auto Saves the code.**
4. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**

5. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
6. **To run your project use command:**
   ```
   mvn compile exec:java -Dexec.mainClass="com.yaksha.assignment.MethodAssignment"
   ```
7. **To test your project test cases, use the command**
   ```
   mvn test
   ```
8. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**