Name : Chandrakanth HV
Assignment : Bridge Course Day 4
Date : 27-06-2025

SECTION 1 :

Problem Statement 1 :Create `greetUser(String name)` to greet the user. Call it three times with different names.

Pseudo code :

- Function greetUser(name):
- Display "Hello " + name + "! Welcome.
- Start Main:
- Initialize Scanner
- FOR i = 1 to 3 DO
- Display "Enter name i: "
- Read userName
- Call greetUser(userName)
- END FOR
- Close Scanner
- End Main

Algorithm: steps

1.Start the program
2.Define the method `greetUser(name)` to print a greeting with the provided name.
3.In the `main()` method:
4.Create a `Scanner` object to read user input.
5.Repeat the following 3 times:
6.Prompt the user to enter a name.
7.Read the name from the user.
8.Call `greetUser(name)` with the entered name.
9.Close the scanner.
10.End the program.

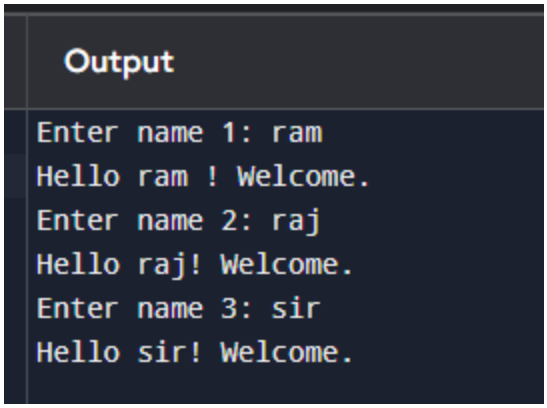
Code :

```
import java.util.Scanner;
public class GreetingExample {
```

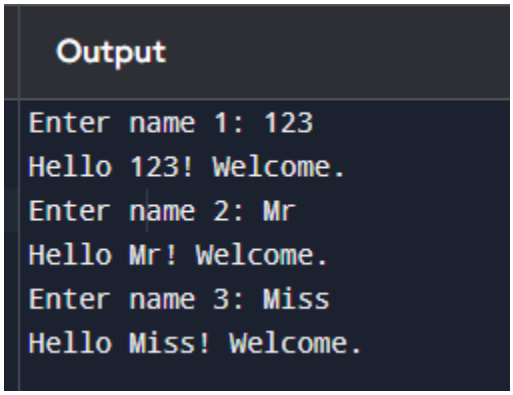
```
    public static void greetUser(String name) {
        System.out.println("Hello " + name + "! Welcome.");
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        for (int i = 1; i <= 3; i++) {
            System.out.print("Enter name " + i + ": ");
            String userName = scanner.nextLine();
            greetUser(userName);
        }

        scanner.close();
    }
}
```

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter name 1 : ram Enter name 2 : raj Enter name 3 : sir	Enter name 1: ram Hello ram ! Welcome. Enter name 2: raj Hello raj! Welcome. Enter name 3: sir Hello sir! Welco	Enter name 1: ram Hello ram ! Welcome. Enter name 2: raj Hello raj! Welcome. Enter name 3: sir Hello sir! Welco	Pass
TC2	Enter name 1: 123 Enter name 2: Mr Enter name 3: Miss	Enter name 1: 123 Hello 123! Welcome. Enter name 2: Mr Hello Mr! Welcome. Enter name 3: Miss Hello Miss! Welcome.	Enter name 1: 123 Hello 123! Welcome. Enter name 2: Mr Hello Mr! Welcome. Enter name 3: Miss Hello Miss! Welcome.	Pass
TC3	Enter name 1: Enter name 2: @ Enter name 3: K	Enter name 1: Hello ! Welcome. Enter name 2: @ Hello @! Welcome. Enter name 3: K Hello K! Welcome.	Enter name 1: Hello ! Welcome. Enter name 2: @ Hello @! Welcome. Enter name 3: K Hello K! Welcome.	Pass

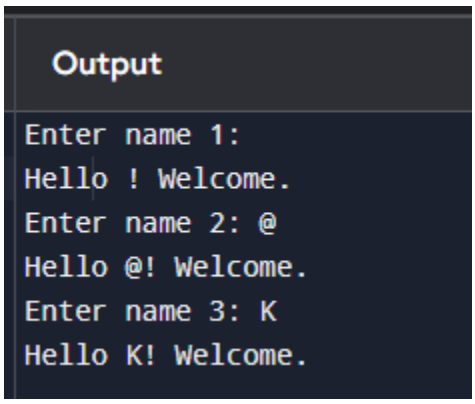
TC1 :



TC2 :



TC2 :



Observation:

In this i had a challenge how to create a user input because for n number of inputs we have to declare the input in those we can interface the multiple inputs by declaring the integer i value to it import the scanner for user input also declares the public class and create the static method for greeting the the user by taking string parameter , By the initialization of for loop it iterates the 3 user and print the output.

Problem Statement 2 : Create a method named `calculateSquare` that:
Takes an integer as a parameter.
Returns its square (i.e., `number * number`).
Accept a number from the user as input.
Store the result and print it.
Directly print the result from the method call**.**

Pseudo code :

- Function calculateSquare(number):
- Return number * number
- Start Main:
- Create scanner for user input
- Display "Enter a number to calculate its square: "
- Read num
- result = calculateSquare(num)
- Display "Square: " + result
- Close scanner
- End Main

Algorithm: steps

1. Start the program.
2. Define a method `calculateSquare` that returns the square of the input number.
3. In the `main` method:
4. Create a scanner to accept user input.
5. Prompt the user to enter an integer.
6. Read the input and store it in a variable `num`.
7. Call the `calculateSquare` method with `num` and store the result.
8. Print the square value.
9. Close the scanner.
10. End the program

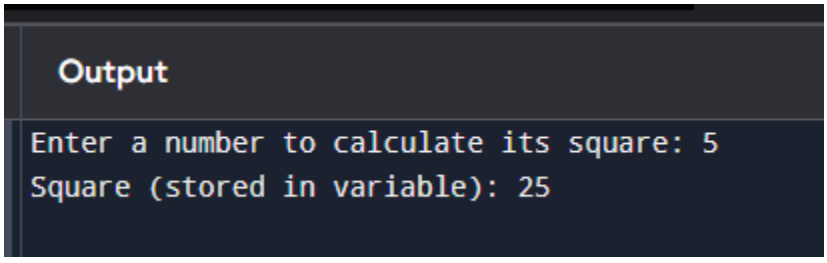
Code :

```
import java.util.Scanner;

public class SquareCalculator {
   public static int calculateSquare(int number) {
      return number * number;
```

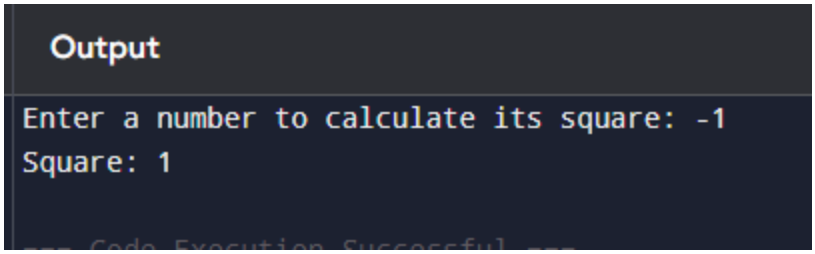
```
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number to calculate its square: ");
        int num = scanner.nextInt();
        int result = calculateSquare(num);
        System.out.println("Square: " + result);
        scanner.close();
    }
}
```

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter a number to calculate its square: 5	Enter a number to calculate its square: 5 Square : 25	Enter a number to calculate its square: 5 Square : 25	Pass
TC2	Enter a number to calculate its square: -1	Enter a number to calculate its square: -1 Square: 1	Enter a number to calculate its square: -1 Square: 1 .	Pass
TC3	Enter a number to calculate its square: 5	Enter a number to calculate its square: 0 Square: 0 .	Enter a number to calculate its square: 0 Square: 0	Pass

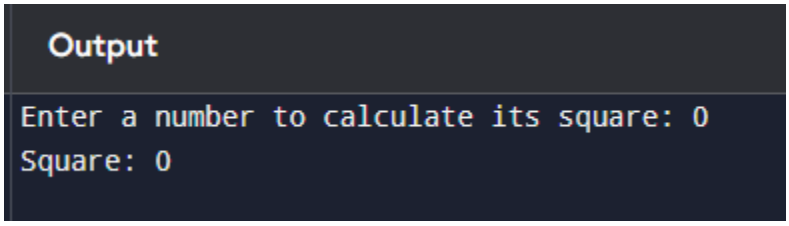
TC1 :



TC2 :



TC2 :



Observation:

While doing this program, I learned how to use the `Scanner` class to take input from the user in Java.I created a separate method called `calculateSquare` to keep the logic for squaring a number separate from the main program.It helped me understand how to pass parameters to a method and return values from it.I faced a small challenge remembering to close the `Scanner` at the end to avoid warnings.Testing the program with different inputs helped me confirm that the method works correctly for various numbers.I realized how using a method improves the clarity and structure of the program.

Problem Statement 3 : Create a method named `addNumbers` that:
Accepts two double numbers as input parameters.
Returns their sum as a `double`.

Pseudo code :

- Function addNumbers(num1, num2):
- return num1 + num2
- Start
- Create Scanner object
- Print "Enter the first number"
- Read number1
- Print "Enter the second number"
- Read number2
- result = addNumbers(number1, number2)
- Print "Sum : " + result
- Close Scanner

- End

Algorithm: steps

1. Start the program.
2. Define a method `addNumbers` that takes two double numbers and returns their sum.
3. In the `main` method:
4. Create a `Scanner` object to read user input.
5. Prompt the user to enter the first number.
6. Read the first number and store it in `number1`.
7. Prompt the user to enter the second number.
8. Read the second number and store it in `number2`.
9. Call the `addNumbers` method with `number1` and `number2` as arguments.
10. Store the result in a variable named `result`.
11. Display the sum using `System.out.println()`.
12. Close the `Scanner`
13. End the program.

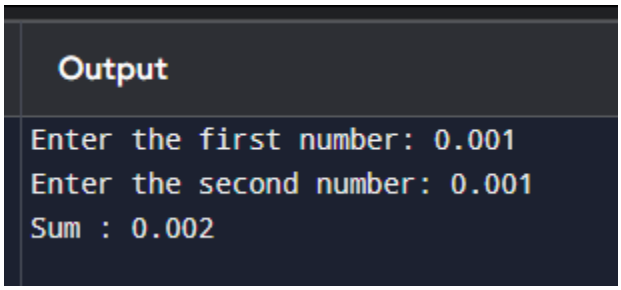
Code :

```
import java.util.Scanner;
public class AddTwoNumbers {
   public static double addNumbers(double num1, double num2) {
      return num1 + num2;
   }
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the first number: ");
      double number1 = scanner.nextDouble();
      System.out.print("Enter the second number: ");
      double number2 = scanner.nextDouble();
      double result = addNumbers(number1, number2);
      System.out.println("Sum : " + result);
      scanner.close();
   }
}
```

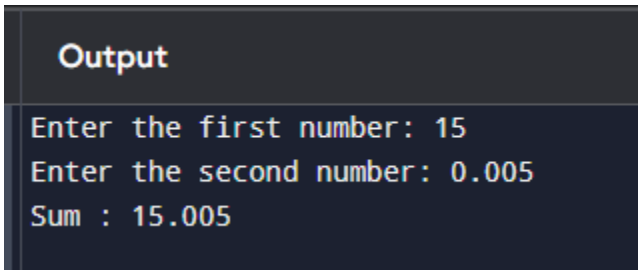
Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter the first number: 0.001	Enter the first number: 0.001	Enter the first number: 0.001 Enter the second number: 0.001 Sum : 0.002	Pass

	Enter the second number: 0.001	Enter the second number: 0.001 Sum : 0.002		
TC2	Enter the first number: 15 Enter the second number: 0.005	Enter the first number: 15 Enter the second number: 0.005 Sum : 15.005	Enter the first number: 15 Enter the second number: 0.005 Sum : 15.005	Pass
TC3	Enter the first number: 1.1 Enter the second number: 1.2	Enter the first number: 1.1 Enter the second number: 1.2 Sum : 2.3	Enter the first number: 1.1 Enter the second number: 1.2 Sum : 2.3	Pass

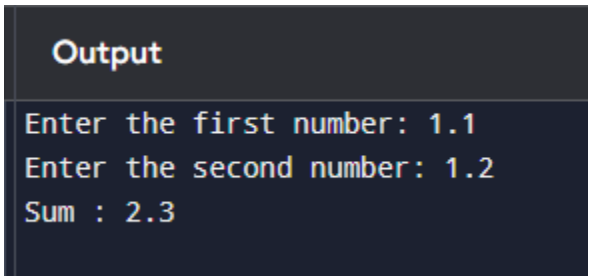
TC1 :



TC2 :



TC2 :



Observation:

While doing this program, I learned how to create and use a method that takes two parameters and returns a result.I understood how to take double input from the user using the `Scanner` class.The `addNumbers` method helped me separate the logic for addition from the `main()` method.I realized how to store the result of a method in a variable and print it.This activity helped me get comfortable with using methods, parameters, return values, and user input handling in Java.I also saw the importance of closing the `Scanner` to avoid resource leaks.

Problem Statement 4 : Create two methods:

1. `double celsiusToFahrenheit(double celsius)`
2. `double fahrenheitToCelsius(double fahrenheit`

Each method should convert temperature between Celsius and Fahrenheit using standard formulas.Print both converted values.

Pseudo code :

- Function celsiusToFahrenheit(celsius):
- return (celsius * 9 / 5) + 32
- Function fahrenheit
- return (fahrenheit - 32) * 5 / 9
- Start
- Create Scanner object
- Print "Enter temperature in Celsius: "
- Read celsius
- Fahrenheit Result = celsius
- Print celsius + "C = " + fahrenheitResult + "F"
- Print "Enter temperature in Fahrenheit: "
- Read fahrenheit
- celsiusResult = fahrenheitToCelsius(fahrenheit)
- Print fahrenheit + "F = " + celsiusResult + "C"
- Close Scanner
- End

Algorithm: steps

1. Start the program.
2. Define method `celsiusToFahrenheit` which converts Celsius to Fahrenheit using the formula:
   `F = (C * 9 / 5) + 32`

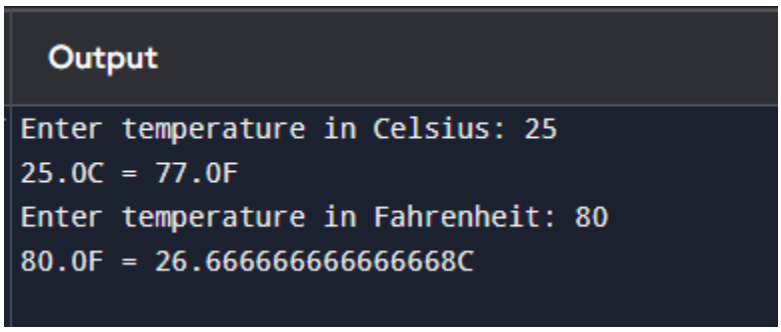
3. Define method `fahrenheitToCelsius` which converts Fahrenheit to Celsius using the formula:
   `C = (F - 32) * 5 / 9`
4. In the `main()` method:
5. Create a `Scanner` object to read user input.
6. Prompt the user to enter temperature in Celsius.
7. Read Celsius value and store it in a variable.
8. Call `celsiusToFahrenheit()` with the Celsius value.
9. Print the converted Fahrenheit result.
10. Prompt the user to enter temperature in Fahrenheit.
11. Read Fahrenheit value and store it in a variable.
12. Call `fahrenheitToCelsius()` with the Fahrenheit value.
13. Print the converted Celsius result.
14. Close the Scanner object
15. End the program.

Code :

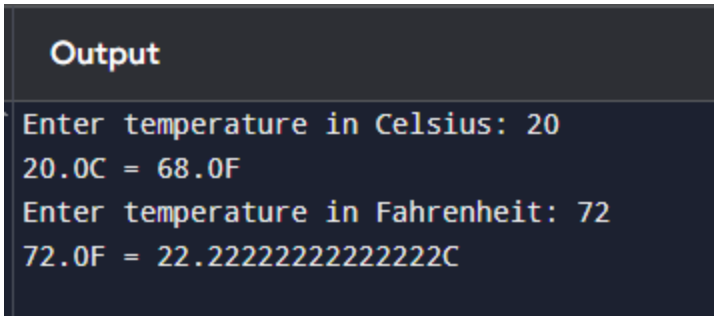
```
import java.util.Scanner;

public class TemperatureConverter {
    public static double celsiusToFahrenheit(double celsius) {
        return (celsius * 9 / 5) + 32;
    }
    public static double fahrenheitToCelsius(double fahrenheit) {
        return (fahrenheit - 32) * 5 / 9;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter temperature in Celsius: ");
        double celsius = scanner.nextDouble();
        double fahrenheitResult = celsiusToFahrenheit(celsius);
        System.out.println(celsius + "C = " + fahrenheitResult +"F");

        System.out.print("Enter temperature in Fahrenheit: ");
        double fahrenheit = scanner.nextDouble();
        double celsiusResult = fahrenheitToCelsius(fahrenheit);
        System.out.println(fahrenheit + "F = " + celsiusResult + "C");

        scanner.close();
    }
}
```

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter temperature in Celsius: 25 Enter temperature in Fahrenheit: 80	Enter temperature in Celsius: 25 25.0C = 77.0F Enter temperature in Fahrenheit: 80 80.0F = 26.666666666666668C	Enter temperature in Celsius: 25 25.0C = 77.0F Enter temperature in Fahrenheit: 80 80.0F = 26.666666666666668C	Pass
TC2	Enter temperature in Celsius: 20 Enter temperature in Celsius: 20	Enter temperature in Celsius: 20 20.0C = 68.0F Enter temperature in Fahrenheit: 72 72.0F = 22.22222222222222C	Enter temperature in Celsius: 20 20.0C = 68.0F Enter temperature in Fahrenheit: 72 72.0F = 22.22222222222222C	Pass
TC3	Enter temperature in Celsius: 30 Enter temperature in Fahrenheit: 96	Enter temperature in Celsius: 30 30.0C = 86.0F Enter temperature in Fahrenheit: 96 96.0F = 35.55555555555556C	Enter temperature in Celsius: 30 30.0C = 86.0F Enter temperature in Fahrenheit: 96 96.0F = 35.55555555555556C	Pass

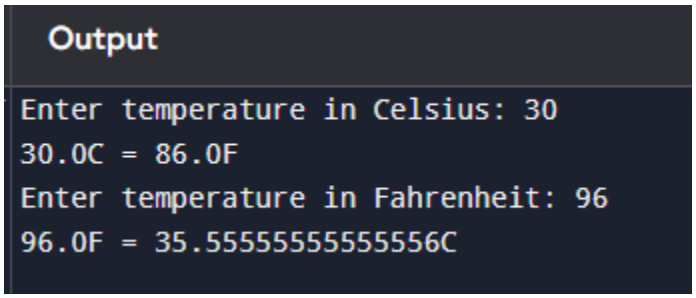
TC1 :



TC2 :



TC2 :



Observation:

While doing this program, I learned how to write and use multiple methods for specific tasks.I understood how to convert temperatures between Celsius and Fahrenheit using standard formulas.The program accepts user input using the `Scanner` class and uses double for decimal accuracy.I was able to apply parameter passing and return values effectively.I learned how separating logic into methods like `celsiusToFahrenheit` and `fahrenheitToCelsius` makes the code cleaner and reusable.I faced an issue with using the degree symbol (°), so I replaced it with plain text like `C` and `F` to avoid invalid output symbols.

SECTION 2:

Problem Statement 1 : Scope Experiment

Task:

Create a class to understand the concept of variable scope.

Define a global variable (`globalMessage`).

Inside a method, define a local variable (`localMessage`) and print both.

Try accessing `localMessage` outside its scope and observe the error.

Pseudo code :

- START
- DECLARE globalMessage as String (global)
- FUNCTION showMessages
- CREATE new Scanner
- PROMPT "Enter a local message"

- READ localMessage
- PRINT "Global Message: " + globalMessage
- PRINT "Local Message: " + localMessage
- END FUNCTION
- MAIN FUNCTION
- CREATE Scanner
- PROMPT "Enter a global message"
- READ globalMessage
- CALL showMessages()
- CLOSE Scanner
- END MAIN
- END

Algorithm: steps

1. Start the program.
2. Define method `celsiusToFahrenheit` which converts Celsius to Fahrenheit using the formula:
   `F = (C * 9 / 5) + 32`
3. Define method `fahrenheitToCelsius` which converts Fahrenheit to Celsius using the formula:
   `C = (F - 32) * 5 / 9`
4. In the `maStart the program.`
5. `Declare a global variable globalMessage.`
6. `In the main() method:`
7. `Create a Scanner object.`
8. `Prompt the user to enter a global message.`
9. `Store the input in the globalMessage.`
10. `Call the method showMessages().`
11. `In the showMessages() method:`
12. `Create another Scanner object.`
13. `Prompt the user to enter a local message.`
14. `Store the input in localMessage.`
15. `Print both globalMessage and localMessage.`
16. `Close the scanner in the main() method.`
17. `End program.`

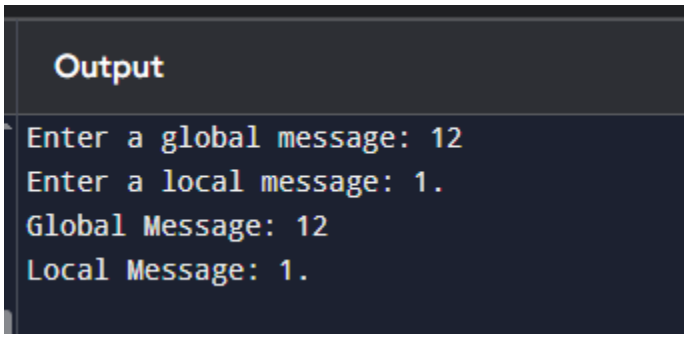
Code :

```
import java.util.Scanner;
public class ScopeExample {

   static String globalMessage;
```

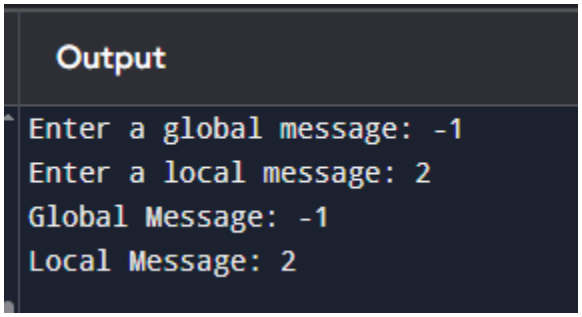
```
    public static void showMessages() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a local message: ");
        String localMessage = scanner.nextLine();

        System.out.println("Global Message: " + globalMessage);
        System.out.println("Local Message: " + localMessage);

    }
public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a global message: ");
        globalMessage = scanner.nextLine();

        showMessages();

        scanner.close();
    }
}
```

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter a global message: 12 Enter a local message: 1.	Enter a global message: 12 Enter a local message: 1. Global Message: 12 Local Message: 1.	Enter a global message: 12 Enter a local message: 1. Global Message: 12 Local Message: 1.	Pass
TC2	Enter a global message: -1 Enter a local message: 2	Enter a global message: -1 Enter a local message: 2 Global Message: -1 Local Message: 2	Enter a global message: -1 Enter a local message: 2 Global Message: -1 Local Message: 2	Pass
TC3	Enter a global message: abc Enter a local message: 2	Enter a global message: abc Enter a local message: 2 Global Message: abc Local Message: 2	Enter a global message: abc Enter a local message: 2 Global Message: abc Local Message: 2	Pass

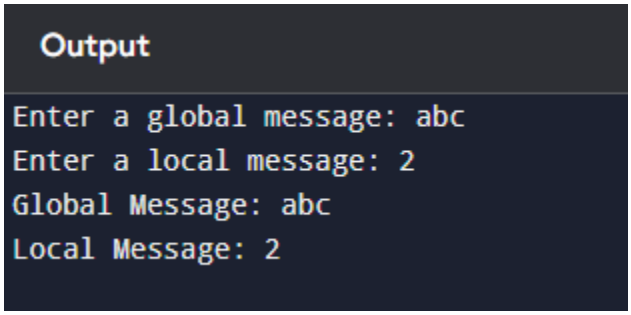
TC1 :



TC2 :



TC3 :



Observation:

This program shows how global and local variables work in Java.The `globalMessage` is declared outside methods, so it can be used anywhere inside the class.The `localMessage` is declared inside the `showMessages()` method, so it can only be used there.The program takes two inputs – one for the global message and one for the local message.It prints both messages to show the difference in variable scope.Two Scanner objects are used – one in `main()` and one in `showMessages()` – but we can also use the same Scanner by passing it.

Problem Statement 2:Price Calculator (Function Composition)

implement the following functions in Java

```
double calculateDiscount(double originalPrice, double
discountPercentage);
double calculateTax(double amount, double taxRate);
double calculateFinalPrice(double itemPrice, double
discountPerc, double taxRate);
```

Pseudo code :

- START
- FUNCTION calculateDiscount(originalPrice, discountPercentage)
- RETURN originalPrice * (discountPercentage / 100)
- FUNCTION calculateTax(amount, taxRate)
- RETURN amount * (taxRate / 100)
- FUNCTION calculateFinalPrice(itemPrice, discountPerc, taxRate)
- discount = calculateDiscount(itemPrice, discountPerc)
- priceAfterDiscount = itemPrice - discount
- tax = calculateTax(priceAfterDiscount, taxRate)
- finalPrice = priceAfterDiscount + tax
- RETURN finalPric
- MAIN:
- CREATE Scanner
- PROMPT "Enter item price"
- READ itemPrice
- PROMPT "Enter discount percentage"
- READ discountPercent
- PROMPT "Enter tax rate"
- READ taxRate
- CALL calculateFinalPrice(itemPrice, discountPercent, taxRate)
- PRINT final price
- CLOSE Scanner
- END

Algorithm: steps

1. Start the program.
2. Dene a function calculateDiscount that takes original price and discount percentage and returns the discount amount.
3. Define a function calculateTax that takes amount and tax rate and returns the tax amount.

4. Define a function calculateFinalPrice that:
5. Calls calculateDiscount to get the discount amount.
6. Subtracts the discount from the item price.
7. Calls calculateTax on the discounted price.
8. Adds tax to the discounted price to get the final price.
9. In main():
10. Create a Scanner object.
11. Ask the user to enter the item price, discount percentage, and tax rate.
12. Call calculateFinalPrice() with the user inputs.
13. Print the final price.
14. Close the Scanner.
15. End the program.

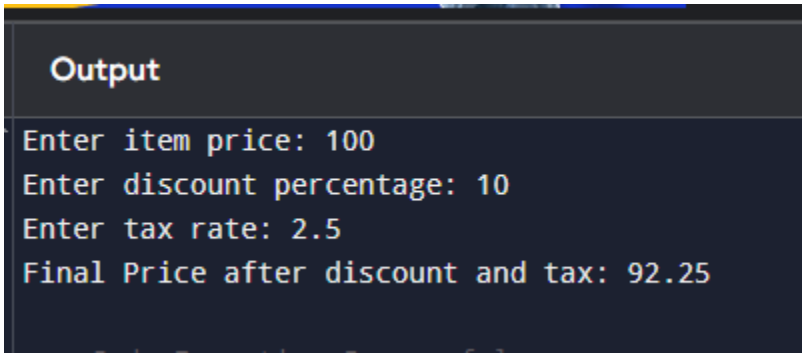
Code :

```
import java.util.Scanner;

public class PriceCalculator {

    static double calculateDiscount(double originalPrice, double discountPercentage) {
        return originalPrice * (discountPercentage / 100);
    }
    static double calculateTax(double amount, double taxRate) {
        return amount * (taxRate / 100);
    }

    static double calculateFinalPrice(double itemPrice, double discountPerc, double taxRate) {
        double discount = calculateDiscount(itemPrice, discountPerc);
        double priceAfterDiscount = itemPrice - discount;
        double tax = calculateTax(priceAfterDiscount, taxRate);
        double finalPrice = priceAfterDiscount + tax;
        return finalPrice;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter item price: ");
        double itemPrice = scanner.nextDouble();

        System.out.print("Enter discount percentage: ");
        double discountPercent = scanner.nextDouble();
```

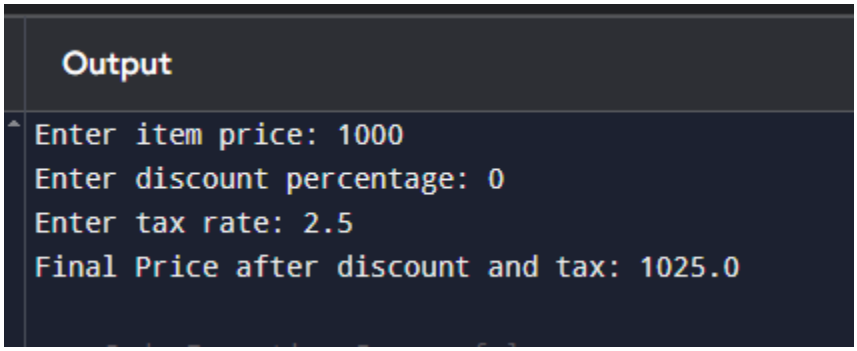
```
        System.out.print("Enter tax rate: ");
        double taxRate = scanner.nextDouble();

        double finalPrice = calculateFinalPrice(itemPrice, discountPercent, taxRate);

        System.out.println("Final Price after discount and tax: ₹" + finalPrice);

        scanner.close();
    }
}
```

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter item price: 100 Enter discount percentage: 10 Enter tax rate: 2.5	Enter item price: 100 Enter discount percentage: 10 Enter tax rate: 2.5 Final Price after discount and tax: 92.25	Enter item price: 100 Enter discount percentage: 10 Enter tax rate: 2.5 Final Price after discount and tax: 92.25	Pass
TC2	Enter a global message: -1 Enter a local message: 2	Enter item price: 1000 Enter discount percentage: 0 Enter tax rate: 2.5	Enter item price: 1000 Enter discount percentage: 0 Enter tax rate: 2.5	Pass
TC3	Enter item price: 120 Enter discount percentage: 50 Enter tax rate: 2.5	Enter item price: 120 Enter discount percentage: 50 Enter tax rate: 2.5 Final Price after discount and tax: 61.5	Enter item price: 120 Enter discount percentage: 50 Enter tax rate: 2.5 Final Price after discount and tax: 61.5	Pass

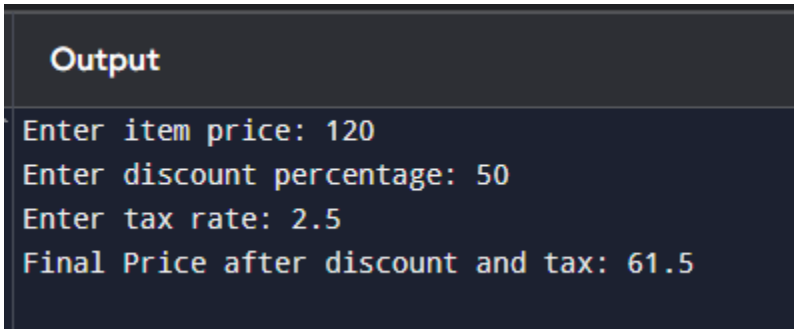
TC1 :



TC2 :



TC3 :



Observation:

This program helps to calculate the final price of a product after applying discount and tax.It uses three functions: one for discount, one for tax, and one to calculate the final price using both.We use `Scanner` to take user input for price, discount, and tax rate.This shows function composition and how smaller functions can be reused inside a bigger function.The logic is broken into parts, so it's easier to read, reuse, and debug.

Problem Statement 3 :Refactor your Day 2 calculator to use:

- `add(num1, num2)`
- `subtract(num1, num2)`
- `multiply(num1, num2)`
- `divide(num1, num2)`

Pseudo code :

- START
- FUNCTION add(a, b):
- RETURN a + b
- FUNCTION subtract(a, b):
- RETURN a - b

- FUNCTION multiply(a, b):
- RETURN a * b
- FUNCTION divide(a, b):
- IF b ≠ 0 THEN
- RETURN a / b
- ELSE
- PRINT "Error: Division by zero!"
- RETURN 0
- MAIN FUNCTION:
- CREATE Scanner
- PROMPT "Enter first number"
- READ num1
- PROMPT "Enter second number"
- READ num2
- PROMPT "Choose operation (+, -, *, /)"
- READ operator
- INITIALIZE result = 0
- SWITCH(operator)
- CASE '+': result = add(num1, num2)
- CASE '-': result = subtract(num1, num2)
- CASE '*': result = multiply(num1, num2)
- CASE '/': result = divide(num1, num2)
- DEFAULT: PRINT "Invalid operator!"
- PRINT result
- CLOSE Scanner
- END

Algorithm: steps

1. Start the program.
2. Define four functions: `add`, `subtract`, `multiply`, and `divide` to perform arithmetic operations.
3. In the `main()` method:
4. Create a `Scanner` object.
5. Ask the user to input the first number and store it in `num1`.
6. Ask the user to input the second number and store it in `num2`.
7. Ask the user to choose an arithmetic operator (`+`, `-`, `*`, or `/`).
8. Use a `switch` statement to:
9. Call the corresponding function based on the operator.
10. Handle division by zero inside the `divide` function.
11. Print an error message for an invalid operator.
12. Print the final result.
13. Close the `Scanner`.

14. End the program.

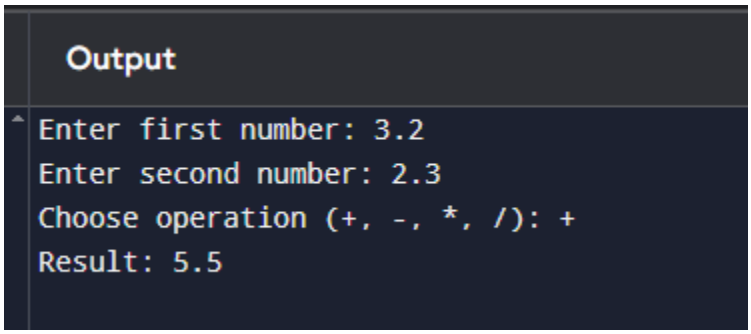
Code :

```
import java.util.Scanner;

public class RefactoredCalculator {

    static double add(double a, double b) {
        return a + b;
    }

    static double subtract(double a, double b) {
        return a - b;
    }

    static double multiply(double a, double b) {
        return a * b;
    }

    static double divide(double a, double b) {
        if (b != 0) {
            return a / b;
        } else {
            System.out.println("Error: Division by zero!");
            return 0;
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter first number: ");
        double num1 = scanner.nextDouble();

        System.out.print("Enter second number: ");
        double num2 = scanner.nextDouble();

        System.out.print("Choose operation (+, -, *, /): ");
        char operator = scanner.next().charAt(0);

        double result = 0;

        switch (operator) {
            case '+':
```

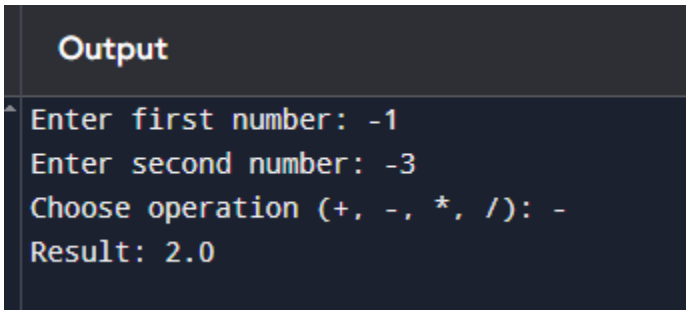
```
                result = add(num1, num2);
                break;
            case '-':
                result = subtract(num1, num2);
                break;
            case '*':
                result = multiply(num1, num2);
                break;
            case '/':
                result = divide(num1, num2);
                break;
            default:
                System.out.println("Invalid operator!");
                break;
        }

        System.out.println("Result: " + result);
        scanner.close();
    }
}
```

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter first number: 3.2 Enter second number: 2.3 Choose operation (+, -, *, /): +	Enter first number: 3.2 Enter second number: 2.3 Choose operation (+, -, *, /): + Result: 5.5	Enter first number: 3.2 Enter second number: 2.3 Choose operation (+, -, *, /): + Result: 5.5	Pass
TC2	Enter first number: -1 Enter second number: -3 Choose operation (+, -, *, /): -	Enter first number: -1 Enter second number: -3 Choose operation (+, -, *, /): - Result: 2.0	Enter first number: -1 Enter second number: -3 Choose operation (+, -, *, /): - Result: 2.0	Pass
TC3	Enter first number: 2 Enter second number: 0 Choose operation (+, -, *, /): /	Enter first number: 2 Enter second number: 0 Choose operation (+, -, *, /): / ERROR! Error: Division by zero! Result: 0.0	Enter first number: 2 Enter second number: 0 Choose operation (+, -, *, /): / ERROR! Error: Division by zero! Result: 0.0	Pass

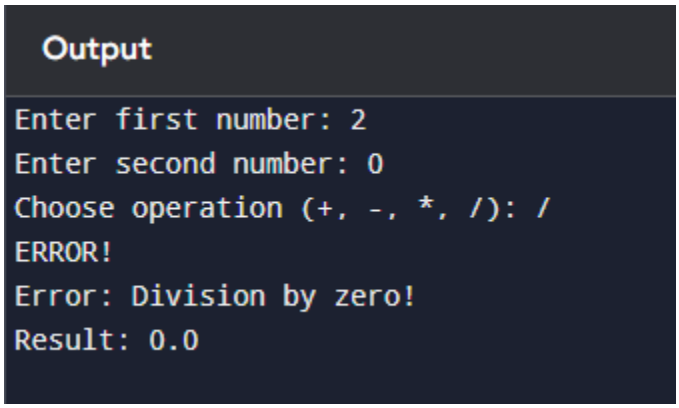
TC1 :



TC2 :



TC2 :



Observation:

This program acts as a simple calculator using user input.It uses functions for each operation (add, subtract, multiply, divide), which makes the code cleaner.It takes two numbers and an operator from the user.Based on the operator, it calls the right function and prints the result.It handles division by zero safely inside the divide function.The code is more readable, reusable, and easy to update.Using switch-case with function calls makes it structured and clear.

SECTION 3:

Problem Statement 1 : Customizable Greeting (overloading)

Create a Java program with a method named `customGreet` that demonstrates method overloading.`void customGreet(String name, String greeting)` – prints a custom greeting for the given name.
`void customGreet(String name)` – prints "Hello, [name]!".
`void customGreet()` – prints a default message like "Hello, man!".
In the `main` method, demonstrate calling all three versions of `customGreet`

Pseudo code :

- STAR
- DEFINE FUNCTION customGreet(name, greeting)
- PRINT greeting + ", " + name + "!"
- DEFINE FUNCTION customGreet(name)
- PRINT "Hello, " + name + "!
- DEFINE FUNCTION customGreet()
- PRINT "Hello, Guest!"
- MAIN FUNCTION:
- CREATE Scanner
- PROMPT "Enter your name"
- READ name
- PROMPT "Enter your greeting"
- READ greeting
- CALL customGreet(name, greeting)
- PROMPT "Enter your name again"
- READ nameOnly
- CALL customGreet(nameOnly)
- CALL customGreet() // No arguments
- CLOSE Scanner
- END

Algorithm: steps

1. Start the program
2. Define three overloaded methods named customGreet:
3. One that takes both name and greeting.
4. One that takes only name.
5. One with no parameters.
6. In the main() method:
7. Create a Scanner object for user input

8. Ask the user to enter their name and greeting, and call customGreet(name, greeting).
9. Ask the user to enter their name again, and call customGreet(name)
10. Call customGreet() with no arguments for default greeting.
11. Print the appropriate greeting message in each method.
12. Close the Scanner.
13. End the program.

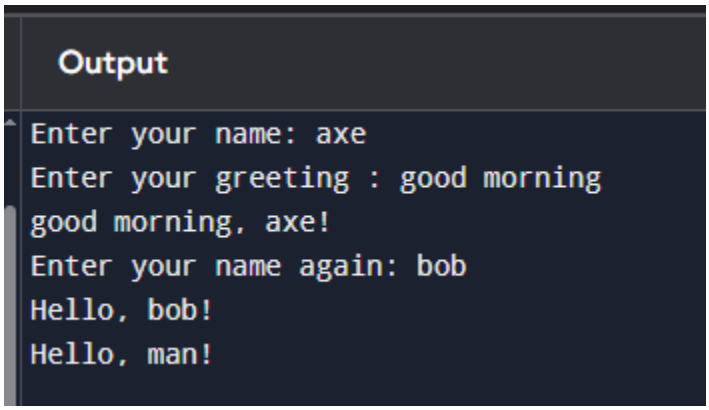
Code :

```
import java.util.Scanner;

public class GreetingOverload {

    static void customGreet(String name, String greeting) {
        System.out.println(greeting + ", " + name + "!");
    }

    static void customGreet(String name) {
        System.out.println("Hello, " + name + "!");
    }

    static void customGreet() {
        System.out.println("Hello, man!");
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String name = scanner.nextLine();
        System.out.print("Enter your greeting : ");
        String greeting = scanner.nextLine();
        customGreet(name, greeting);

        System.out.print("Enter your name again: ");
        String nameOnly = scanner.nextLine();
        customGreet(nameOnly);

        customGreet();

        scanner.close();
    }
```

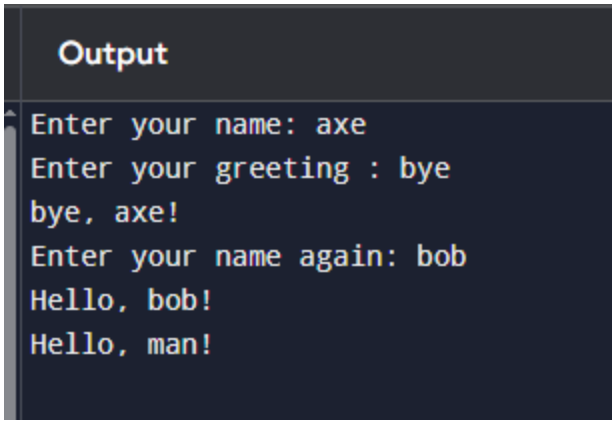
}

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter your name: axe Enter your greeting : good morning Enter your name again: bob.	Enter your name: axe Enter your greeting : good morning good morning, axe! Enter your name again: bob Hello, bob! Hello, man!	Enter your name: axe Enter your greeting : good morning good morning, axe! Enter your name again: bob Hello, bob! Hello, man!	Pass
TC2	Enter your name: axe Enter your greeting : bye Enter your name again: bob	Enter your name: axe Enter your greeting : bye bye, axe! Enter your name again: bob Hello, bob! Hello, man!	Enter your name: axe Enter your greeting : bye bye, axe! Enter your name again: bob Hello, bob! Hello, man!	Pass
TC3	Enter your name: bob Enter your greeting : good bye Enter your name again: axe	Enter your name: bob Enter your greeting : good bye good bye, bob! Enter your name again: axe Bye, axe! Hello, mister	Enter your name: bob Enter your greeting : good bye good bye, bob! Enter your name again: axe Bye, axe! Hello, mister	Pass

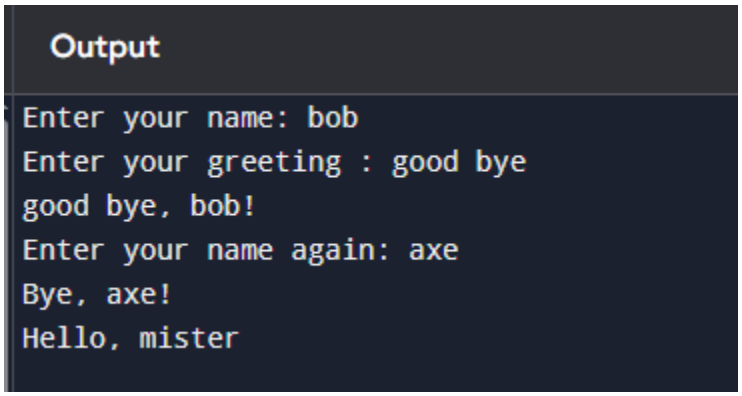
TC1 :



TC2 :



TC3 :



Observation:

This program shows method overloading using the same method name with different parameters. It take input from the user and shows how the same method name behaves differently based on arguments,Three versions of `customGreet()` are used:
One with name and greeting (personalized).
One with name only (default greeting).
One with no input (generic greeting)
It helps in understanding how Java supports multiple methods with same name but different parameter types/counts.The program is user-friendly and interactive with proper input and output.

Problem Statement 2 :

.Write a Java program that calculates the power of a number without using built-in method.

- Define a method `myPower(int base, int exponent)` that uses a loop to compute base raised to the power exponent.
- In the `main` method, accept the base and exponent as user inputs.

- Display the result using your custom `myPower` method.
- Also, display the result using Java's built-in `Math.pow()` method for comparison.

Pseudo code :

- FUNCTION myPower(base, exponent):
- SET result = 1
- FOR i FROM 1 TO exponent:
- result = result * base
- END FOR
- RETURN result
- START
- CREATE scanner object to read input
- PRINT "Enter the base"
- READ base
- PRINT "Enter the exponent"
- READ exponent
- CALL myPower(base, exponent) → store in customResult
- CALL Math.pow(base, exponent) → store in builtInResult
- PRINT "Using myPower():", customResult
- PRINT "Using Math.pow():", builtInResult
- END

Algorithm: steps

1. Start
2. Define a method `myPower(base, exponent)` that:
3. Initializes `result` to 1
4. Repeats a loop from 1 to `exponent`
5. Multiplies `result` by `base` each time
6. Returns `result`
7. In the `main()` method:
8. Create a Scanner object
9. Prompt user to enter `base` and `exponent`
10. Call `myPower()` and store the result
11. Use `Math.pow(base, exponent)` to get built-in result
12. Display both result
13. End

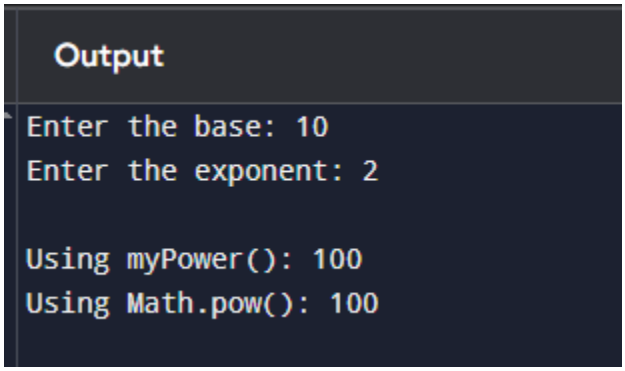
Code :

```
import java.util.Scanner;

public class PowerCalculator {

    public static int myPower(int base, int exponent) {
        int result = 1;
        for (int i = 1; i <= exponent; i++) {
            result *= base;
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        System.out.print("Enter the base: ");
        int base = scanner.nextInt();

        System.out.print("Enter the exponent: ");
        int exponent = scanner.nextInt();


        int customResult = myPower(base, exponent);
        double builtInResult = Math.pow(base, exponent);

        System.out.println("\nUsing myPower(): " + customResult);
        System.out.println("Using Math.pow(): " + (int)builtInResult);
    }
}
```

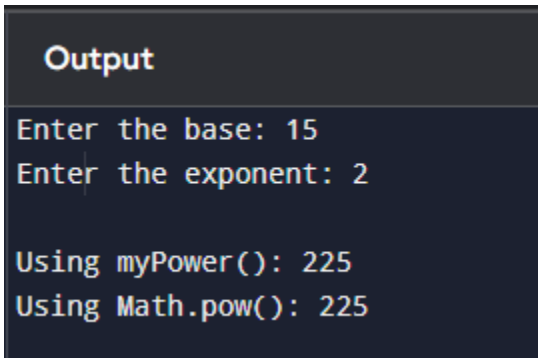
Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter the base: 10 Enter the exponent: 2	Enter the base: 10 Enter the exponent: 2 Using myPower(): 100 Using Math.pow(): 100	Enter the base: 10 Enter the exponent: 2 Using myPower(): 100 Using Math.pow(): 100	Pass
TC2	Enter the base: 15 Enter the exponent: 2	Enter the base: 15 Enter the exponent: 2 Using myPower(): 225	Enter the base: 15 Enter the exponent: 2 Using myPower(): 225	Pass

		Using Math.pow(): 225	Using Math.pow(): 225	
TC3	Enter your name: bob Enter your greeting : good bye Enter your name again: axe	Enter the base: 2 Enter the exponent: 2 Using myPower(): 4 Using Math.pow(): 4	Enter the base: 2 Enter the exponent: 2 Using myPower(): 4 Using Math.pow(): 4	Pass

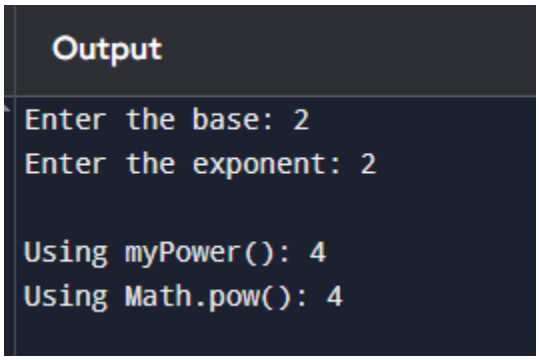
TC1 :



TC2 :



TC3 :



Observation:

The custom method `myPower()` correctly calculates the power using a loop, suitable for positive integersIt doesn't handle:
Negative exponents (would always return 1)
Exponent = 0 is handled correctly (returns 1 as expecte`Math.pow()` is more flexible:
It supports decimal bases
It works with negative exponents and returns floating point results

Problem Statement 3 :

Write a Java program that reads three integers from the user and determines the smallest among them using a method.

- Create a method `findMinimum(int a, int b, int c)` that returns the smallest value.
- Call the method from `main()` and print the result

Pseudo code :

- FUNCTION findMinimum(a, b, c)
- SET min = a
- IF b < min THEN
- min = b
- END IF
- IF c < min THEN
- min = c
- END IF
- RETURN min
- END FUNCTION
- START
- DECLARE Scanner
- PRINT "Enter first number"
- READ num1
- PRINT "Enter second number"
- READ num2
- PRINT "Enter third number"
- READ num3
- CALL findMinimum(num1, num2, num3) → smallest
- PRINT "The smallest number is: " + smallest
- END

Algorithm: steps

1. Start
2. Declare method findMinimum(int a, int b, int c)
3. Set min = a
4. If b < min, then set min = b
5. If c < min, then set min = c
6. Return min
7. In he main() method:
8. Create a Scanner object
9. Prompt user to input three integers
10. Call findMinimum(num1, num2, num3) and store the result
11. Print the smallest numbe
12. End

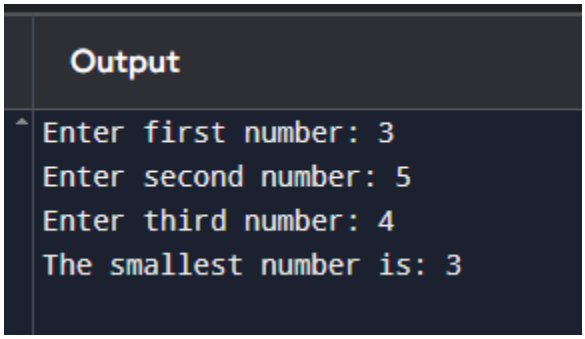
Code :

```
import java.util.Scanner;

public class MinimumOfThree {

    public static int findMinimum(int a, int b, int c) {
        int min = a;

        if (b < min) {
            min = b;
        }
        if (c < min) {
            min = c;
        }

        return min;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter first number: ");
        int num1 = scanner.nextInt();

        System.out.print("Enter second number: ");
        int num2 = scanner.nextInt();

        System.out.print("Enter third number: ");
        int num3 = scanner.nextInt();
```

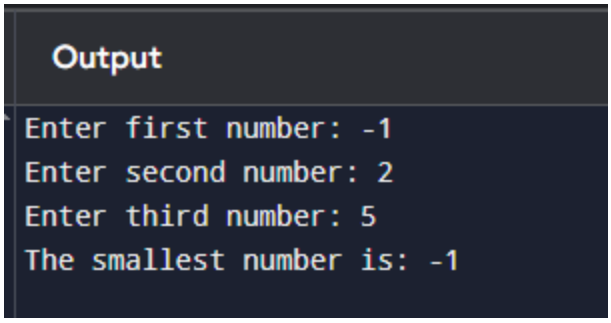
```
        int smallest = findMinimum(num1, num2, num3);

        System.out.println("The smallest number is: " + smallest);

        scanner.close();
    }}
```

Test cases	Input	Expected Output	Actual Output	Status
TC1	Enter first number: 3 Enter second number: 5 Enter third number: 4	Enter first number: 3 Enter second number: 5 Enter third number: 4 The smallest number is: 3	Enter first number: 3 Enter second number: 5 Enter third number: 4 The smallest number is: 3	Pass
TC2	Enter first number: -1 Enter second number: 2 Enter third number: 5	Enter first number: -1 Enter second number: 2 Enter third number: 5 The smallest number is: -1	Enter first number: -1 Enter second number: 2 Enter third number: 5 The smallest number is: -1	Pass
TC3	Enter first number: 25 Enter second number: 10 Enter third number: 0	Enter first number: 25 Enter second number: 10 Enter third number: 0 The smallest number is: 0	Enter first number: 25 Enter second number: 10 Enter third number: 0 The smallest number is: 0	Pass

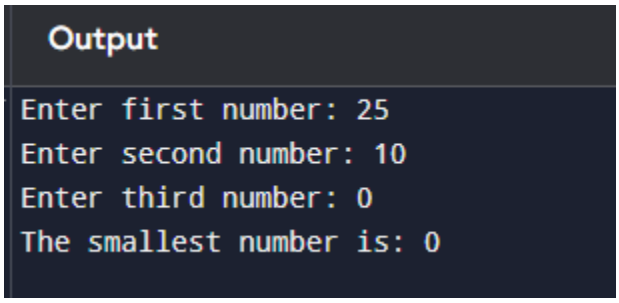
TC1 :



TC2 :



TC3 :



Observation:

In this i know that The program accurately identifies the minimum of three integers using basic conditional comparisons.It uses a modular approach by placing the comparison logic inside a method (`findMinimum()`), which promotes code reusability.The program handles positive and negative numbers.This code helps beginners understand:Method creation and calling,If conditions,User input,Return values,Does not use any built-in methods like `Math.min()` — which makes it ideal for learning basic logic building.