**Learner Assignment Submission Format**

**Learner Details**

* **Name: Nithya B A**
* **Enrollment Number: SU625MR010**
* **Batch / Class: Mern stack**
* **Assignment: (Bridge Course Day 4)**
* **Date of Submission: 30/6/2025**

**Problem Solving Activity 1.1**

1. **Program Statement**

**Write a program to greet user by calling a function three times**

1. **Algorithm**

Step1:Start

Step2:Creation of function call

Step3:Call the function three times

Step4:Print the output

Step5:End

1. **Pseudocode**

FUNCTION GreetUser(name)

PRINT "Hello, " + name + "!"

END FUNCTION

BEGIN MAIN

DECLARE names = ["Jack", "Joe", "Moon"]

FOR EACH name IN names

CALL GreetUser(name)

END FOR

END MAIN

1. **Program Code**

**package** day4;

**public** **class** GreetingFunction {

**public** **static** **void** greetUser(String name) {

System.***out***.println("Hello, " + name + "!");

}

**public** **static** **void** main(String[] args) {

*greetUser*("Jack");

*greetUser*("Joe");

*greetUser*("Moon");

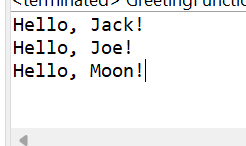
}

}

**5. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case No. | Input | Expected Output | Actual Output | Status (Pass/Fail) |
| 1 | Jack | Hello, Jack! | Hello, Jack! | Pass |
| 2 | Joe | Hello, Joe! | Hello, Joe! | Pass |
| 3 | Moon | Hello, Moon! | Hello, Moon! | Pass |

**6.Screenshots of Output**

****

**7. Observation / Reflection**

Calling a function using function call was a new concept

**Problem Solving Activity 1.2**

1. **Program Statement**

**Write a program to calculate square of a number by calling a function**

1. **Algorithm**

Step1:Start

Step2:Creation of function call

Step3:return number\*number

Step4:Input a number

Step5:Print the output

Step6:End

1. **Pseudocode**

FUNCTION CalculateSquare(number)

RETURN number \* number

END FUNCTION

BEGIN MAIN

PROMPT "Enter a number:"

INPUT n // user enters a value

square = CalculateSquare(n)

OUTPUT "Square of " + n + " is: " + square

END MAIN

1. **Program Code**

**package** day4;

**import** java.util.Scanner;

**public** **class** CalculateSq {

**public** **static** **int** calculateSquare(**int** number) {

**return** number \* number;

}

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

**int** n=sc.nextInt();

// int result = calculateSquare(n);

// System.out.println("Square of " +n+" is: " + result);

System.***out***.println("Square of " +n+" is: " +*calculateSquare*(n));

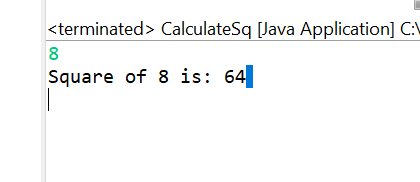
}

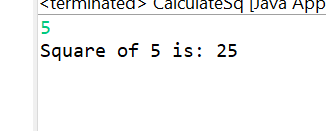
}

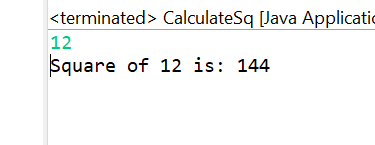
**5. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case No. | Input | Expected Output | Actual Output | Status (Pass/Fail) |
| 1 | 5 | 25 | 25 | Pass |
| 2 | 12 | 144 | 144 | Pass |
| 3 | 8 | 64 | 64 | Pass |

**6.Screenshots of Output**

****

****

****

**7. Observation / Reflection**

Using a function call with return type is challenging

**Problem Solving Activity 1.3**

1. **Program Statement**

**Write a program to calculate square of a number by calling a function**

1. **Algorithm**

Step1: **Start**

Step2: Prompt the user to enter two numbers.

Step3: Read the first number → a

Step4: Read the second number → b

Step5: Compute the sum: sum = a + b

Step6: Display the result: “Sum of a and b is: sum”

Step7: **End**

1. **Pseudocode**

FUNCTION AddNumbers(num1, num2)

RETURN num1 + num2

END FUNCTION

BEGIN MAIN

OUTPUT "Enter the 2 numbers:"

INPUT a

INPUT b

sum ← AddNumbers(a, b)

OUTPUT "Sum of " + a + " and " + b + " is: " + sum

END MAIN

1. **Program Code**

package day4;

import java.util.Scanner;

public class SumTwo {

public static double addNumbers(double num1, double num2) {

return num1 + num2;

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.*in*);

System.*out*.println("enter the 2 number: ");

double a=sc.nextDouble();

double b=sc.nextDouble();

double sum = *addNumbers*(a, b);

System.*out*.println("Sum of "+a+" and "+b+" is: " +sum);

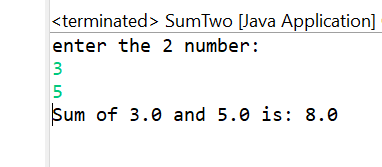
}

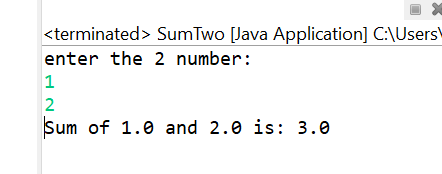
}

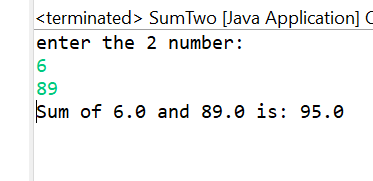
**5. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case No. | Input | Expected Output | Actual Output | Status (Pass/Fail) |
| 1 | 3,5 | 8 | 8 | Pass |
| 2 | 1,2 | 3 | 3 | Pass |
| 3 | 6,89 | 95 | 95 | Pass |

**6.Screenshots of Output**





****

**7. Observation / Reflection**

Used a return type correctly in the function call

**Problem Solving Activity 1.4**

1. **Program Statement**

**Write a program to calculate Celsius and Fahrenheit**

1. **Algorithm**

Step1: **Start**

Step2:Prompt user: *"Enter what you want to know: 1. Celsius value 2. Fahrenheit value"*

Step3: Read choice into choice

**If** choice == 1 (user wants Celsius):

Step4:Prompt: *"enter the Fahrenheit value:"*

Step5:Read tempF

Step6:Compute: result = (tempF − 32) × 5/9

Step8:Display: "tempF°F = result°C"

Step9: **Else** (user wants Fahrenheit):

Step10:Prompt: *"enter the Celsius value:"*

Step11:Read tempC

Step12:Compute: result = (tempC × 9/5) + 32

Step13:Display: "tempC°C = result°F"

Step14: **End**

1. **Pseudocode**

FUNCTION CelsiusToFahrenheit(celsius)

RETURN (celsius \* 9 / 5) + 32

END FUNCTION

FUNCTION FahrenheitToCelsius(fahrenheit)

RETURN (fahrenheit - 32) \* 5 / 9

END FUNCTION

BEGIN MAIN

OUTPUT "Enter what you want to know: 1. Celsius value 2. Fahrenheit value"

INPUT choice

IF choice = 1 THEN

OUTPUT "Enter the Fahrenheit value:"

INPUT tempF

celsius ← FahrenheitToCelsius(tempF)

OUTPUT tempF + "°F = " + celsius + "°C"

ELSE

OUTPUT "Enter the Celsius value:"

INPUT tempC

fahrenheit ← CelsiusToFahrenheit(tempC)

OUTPUT tempC + "°C = " + fahrenheit + "°F"

END IF

END MAIN

1. **Program Code**

package day4;

import java.util.Scanner;

public class TempCon {

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 sc=new Scanner(System.*in*);

System.*out*.println("Enter what you want to know 1.Celsius value 2.Fahrenheit value");

int a=sc.nextInt();

if(a==1) {

System.*out*.println("enter the Fahrenheit value:");

double tempF = sc.nextDouble();

System.*out*.println(tempF + "°F = " + *fahrenheitToCelsius*(tempF) + "°C");

}else {

System.*out*.println("enter the Celsius value:");

double tempC = sc.nextDouble();

System.*out*.println(tempC + "°C = " + *celsiusToFahrenheit*(tempC) + "°F");

}

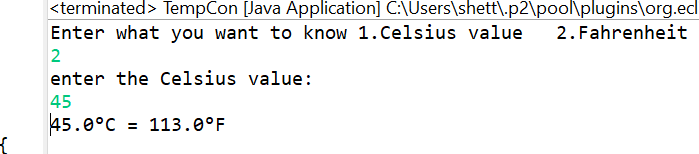
}

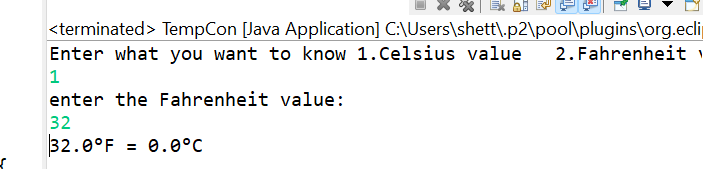
}

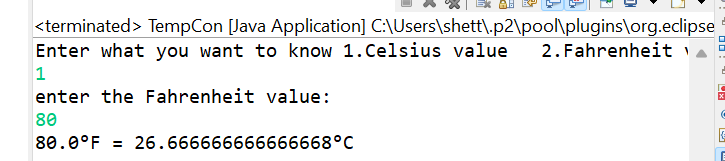
**5. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case No. | Input | Expected Output | Actual Output | Status (Pass/Fail) |
| 1 | 45 | 113 | 113 | Pass |
| 2 | 32 | 0 | 0 | Pass |
| 3 | 80 | 26.67 | 26.68 | Pass |

**6.Screenshots of Output**



****

****

**7. Observation / Reflection**

Derived two functions in the program and simultaneously called two functions

**Problem Solving Activity 2.2**

1. **Program Statement**

**Write a program to calculate the price of an item with discount and tax**

1. **Algorithm**

Step: 1  Start the program.  
Step: 2  Prompt "Enter the price of the item:"; input originalPrice.  
Step: 3  Prompt "Enter the discount percentage:"; input discountPercent.  
Step: 4  Prompt "Enter the tax percentage:"; input taxRate.  
Step: 5  Calculate discount = originalPrice × (discountPercent ÷ 100).  
Step: 6  Calculate discountedPrice = originalPrice − discount.  
Step: 7  Calculate tax = discountedPrice × (taxRate ÷ 100).  
Step: 8  Calculate finalPrice = discountedPrice + tax.  
Step: 9  Display "Final Price: ₹" + finalPrice.  
Step: 10  End the program.

1. **Pseudocode**

FUNCTION CalculateDiscount(originalPrice, discountPercentage)

RETURN originalPrice \* (discountPercentage / 100)

END FUNCTION

FUNCTION CalculateTax(amount, taxRate)

RETURN amount \* (taxRate / 100)

END FUNCTION

FUNCTION CalculateFinalPrice(itemPrice, discountPerc, taxRate)

discount ← CalculateDiscount(itemPrice, discountPerc)

discountedPrice ← itemPrice - discount

tax ← CalculateTax(discountedPrice, taxRate)

RETURN discountedPrice + tax

END FUNCTION

BEGIN MAIN

OUTPUT "Enter the price of the item:"

INPUT originalPrice

OUTPUT "Enter the discount percentage:"

INPUT discountPercent

OUTPUT "Enter the tax percentage:"

INPUT taxRate

finalPrice ← CalculateFinalPrice(originalPrice, discountPercent, taxRate)

OUTPUT "Final Price: ₹" + finalPrice

END MAIN

1. **Program Code**

**package** day4;

**import** java.util.Scanner;

**public** **class** PriceCal {

**public** **static** **double** calculateDiscount(**double** originalPrice, **double** discountPercentage) {

**return** originalPrice \* (discountPercentage / 100);

}

**public** **static** **double** calculateTax(**double** amount, **double** taxRate) {

**return** amount \* (taxRate / 100);

}

**public** **static** **double** calculateFinalPrice(**double** itemPrice, **double** discountPerc, **double** taxRate) {

**double** discount = *calculateDiscount*(itemPrice, discountPerc);

**double** discountedPrice = itemPrice - discount;

**double** tax = *calculateTax*(discountedPrice, taxRate);

**return** discountedPrice + tax;

}

**public** **static** **void** main(String[] args) {

Scanner c=**new** Scanner(System.in);

System.out.println("enter the price of the item: ");

**double** originalPrice = c.nextDouble();

System.out.println("enter the disciunt %: ");

**double** discountPercent = c.nextDouble();

System.out.println("enter the tax %: ");

**double** taxRate = c.nextDouble();

**double** finalPrice = calculateFinalPrice(originalPrice, discountPercent, taxRate); System.out.println("Final Price: ₹" + finalPrice);

}

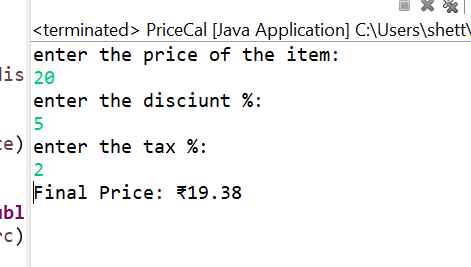
    }

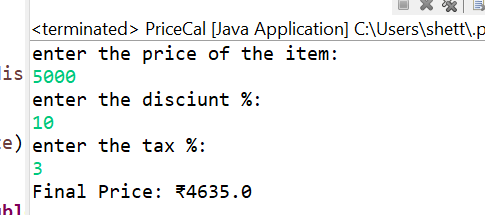
}

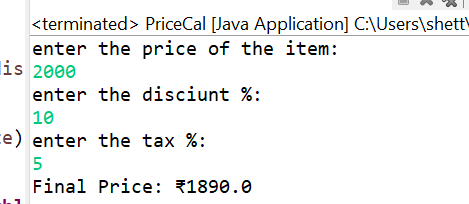
**5. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case No. | Input | Expected Output | Actual Output | Status (Pass/Fail) |
| 1 | 20  5  2 | 19,38 | 19.38 | Pass |
| 2 | 5000  10  3 | 4635 | 4635 | Pass |
| 3 | 2000  10  5 | 1890 | 1890 | Pass |

**6.Screenshots of Output**

****

****

****

**7. Observation / Reflection**

Created function call for all the different tasks and took user input

**Problem Solving Activity 3.1**

1. **Program Statement**

**Write a program to greet the user with custom input and by using overloading**

1. **Algorithm**

Step1:Start the program.

Step2:Call customGreet("Ratan", "Welcome").

Step3:Call customGreet("Tata").

Step4:Call customGreet().

Step5:End the program.

1. **Pseudocode**

FUNCTION customGreet(name, greeting)

OUTPUT greeting + ", " + name + "!"

END FUNCTION

FUNCTION customGreet(name)

OUTPUT "Hello, " + name + "!"

END FUNCTION

FUNCTION customGreet()

OUTPUT "Hi there!"

END FUNCTION

BEGIN MAIN

CALL customGreet("Ratan", "Welcome")

CALL customGreet("Tata")

CALL customGreet()

END MAIN

1. **Program Code**

**package** day4;

**import** java.util.Scanner;

**public** **class** GreetCus {

**public** **static** **void** customGreet(String name, String greeting) {

System.***out***.println(greeting + ", " + name + "!");

}

**public** **static** **void** customGreet(String name) {

System.***out***.println("Hello, " + name + "!");

}

**public** **static** **void** customGreet() {

System.***out***.println("Hi there!");

}

**public** **static** **void** main(String[] args) {

*customGreet*("Joe", "Welcome");

*customGreet*("Biden");

*customGreet*();

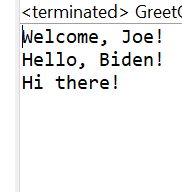
}

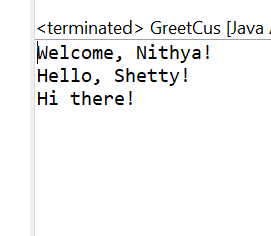
}

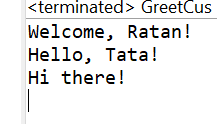
**5. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case No. | Input | Expected Output | Actual Output | Status (Pass/Fail) |
| 1 | Joe  Biden | Welcome,Joe!  Hello,Biden!  Hi there! | Welcome,Joe!  Hello,Biden!  Hi there! | Pass |
| 2 | Nithya  Shetty | Welcome, Nithya!  Hello, Shetty!  Hi there! | Welcome, Nithya!  Hello, Shetty!  Hi there! | Pass |
| 3 | Ratan  Tata | Welcome, Ratan!  Hello, Tata!  Hi there! | Welcome, Ratan!  Hello, Tata!  Hi there! | Pass |

**6.Screenshots of Output**

****

****

****

**7. Observation / Reflection**

Understood the concept of function overloading

**Problem Solving Activity 3.2**

1. **Program Statement**

**Write a program to calculate the power of a number using functions**

1. **Algorithm**

Step: 1  Start the program – indicates algorithm entry.  
Step: 2  Read base from the user – input value.  
Step: 3  Read exponent from the user – input power.  
Step: 4  Set result = 1 – initialize accumulator.  
Step: 5  FOR i FROM 1 TO exponent – begin loop for multiplication.  
Step: 6    Set result = result \* base – multiply in each iteration.  
Step: 7  END FOR – end loop after exponent multiplications.  
Step: 8  Display result – output the computed power.  
Step: 9  Stop the program – indicates algorithm termination.

1. **Pseudocode**

FUNCTION myPower(base, exponent)

result ← 1

FOR i FROM 1 TO exponent DO

result ← result \* base

END FOR

RETURN result

END FUNCTION

BEGIN MAIN

INPUT base

INPUT exponent

OUTPUT "myPower(" + base + ", " + exponent + ") = " + myPower(base, exponent)

END MAIN

1. **Program Code**

package day4;

import java.util.Scanner;

public class PowerC {

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 sc=new Scanner(System.*in*);

System.*out*.println("Enter your base");

int base = sc.nextInt();

System.*out*.println("enter the power");

int exponent = sc.nextInt();

int customPower = *myPower*(base, exponent);

System.*out*.println("myPower(" + base + ", " + exponent + ") = " + customPower);

double mathPower = Math.*pow*(base, exponent);

System.*out*.println("Math.pow(" + base + ", " + exponent + ") = " + mathPower);

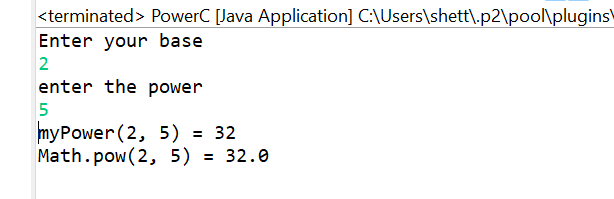
}

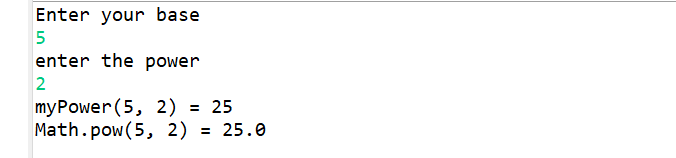
}

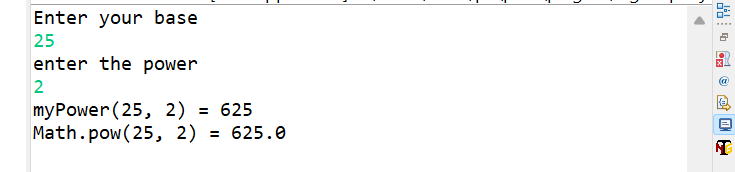
**5. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case No. | Input | Expected Output | Actual Output | Status (Pass/Fail) |
| 1 | 2  5 | 32  32 | 32  32 | Pass |
| 2 | 5  2 | 25  25 | 25  25 | Pass |
| 3 | 25  2 | 625  625 | 625  625 | Pass |

**6.Screenshots of Output**

****

****

****

**7. Observation / Reflection**

Used function call to calculate the power of a number and understood the concept by using math built in operations

**Problem Solving Activity 3.3**

1. **Program Statement**

**Write a program to trace a program which is to multiply by 2**

1. **Algorithm**

Step: 1  Start the program — marks entry of execution.  
Step: 2  CALL functionA() — kick off core logic.  
Step: 3  In functionA, CALL functionB() — get a transformed user input.  
Step: 4  In functionB, PROMPT user to “enter the number:” — request input.  
Step: 5  INPUT value and RETURN value \* 2 — double the input.  
Step: 6  Back in functionA, STORE returned value in result — hold doubled number.  
Step: 7  CALL functionC(result) — pass result for final processing.  
Step: 8  In functionC, DISPLAY “Final result from function C: ” + number — output result.  
Step: 9  END functionC and return to functionA — finish processing.  
Step: 10 END functionA and return to main — complete workflow.  
Step: 11 END the program — exit execution.

1. **Pseudocode**

FUNCTION functionA()

result ← CALL functionB()

CALL functionC(result)

END FUNCTION

FUNCTION functionB()

OUTPUT "enter the number:"

INPUT value

RETURN value \* 2

END FUNCTION

FUNCTION functionC(number)

OUTPUT "Final result from function C: " + number

END FUNCTION

BEGIN MAIN

CALL functionA()

END MAIN

1. **Program Code**

package day4;

import java.util.Scanner;

public class TraceF {

static Scanner *sc*=new Scanner(System.*in*);

public static void functionA() {

int result = *functionB*();

*functionC*(result);

}

public static int functionB() {

System.*out*.println("enter the number: ");

int value =*sc*.nextInt();

return value \* 2;

}

public static void functionC(int number) {

System.*out*.println("Final result from function C: " + number);

}

public static void main(String[] args) {

*functionA*();

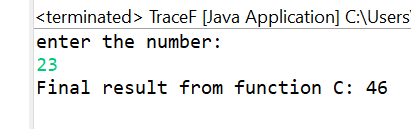
}

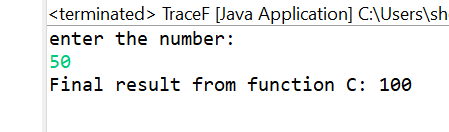
}

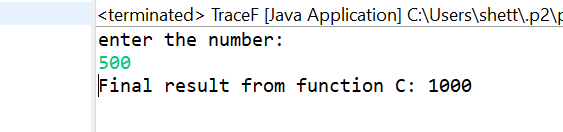
**5. Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case No. | Input | Expected Output | Actual Output | Status (Pass/Fail) |
| 1 | 23 | 46 | 46 | Pass |
| 2 | 50 | 100 | 100 | Pass |
| 3 | 500 | 1000 | 1000 | Pass |

**6.Screenshots of Output**

****

****

****

**7. Observation / Reflection**

Understood the flow of the program