

Learner Assignment Submission Format

Learner Details

- **Name:** Adnan Sameer
 - **Enrollment Number:** ---
 - **Batch / Class:** ---
 - **Assignment:** (Bridge Course Day 4)
 - **Date of Submission:** 27-06-2025
-

Problem Solving Activity 4.1

1. Identify Repetition

Write a Java program that calculates the area of three rectangles using repeated code. Then, identify the lines that are repeated.

Create a function calculateArea (int length, int width) and call it for each rectangle.

2. Algorithm

1. Define a method to calculate the area of a rectangle.
 2. Take two parameters: width (w) and height (h).
 3. Calculate the area by multiplying the width and height.
 4. Print the calculated area.
 5. In the main method, call the area calculation method with different sets of width and height values.
-

3. Pseudocode

FUNCTION calculateArea(width, height)

 area = width * height

 PRINT "The area of the rectangle is " + area

MAIN

 calculateArea(4, 2)

calculateArea(3, 4)

calculateArea(5, 3)

4. Program Code

```
public class D41{  
    static void Area(int w,int h){  
        //area of rectangle=w*h  
        int res=(w*h);  
        System.err.println("The area of rectangle is "+res);  
    }  
    public static void main(String[] args) {  
        Area(4,2);  
        Area(3,4);  
        Area(5,3);  
    }  
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	4,2	8	8	pass
2	3,4	12	12	pass
3	5,3	15	15	pass

6. Screenshots of Output

```
workspaceStorage\6aae71e0b3be4421
The area of rectangle is 8
The area of rectangle is 12
The area of rectangle is 15
PS C:\Users\Admin\OneDrive\Desktop
```

7. Observation / Reflection

I was able to identify repetition. I got desired results using function. Using function make some code optimized

Problem Solving Activity 4.2

1. Program Breakdown

Think of a simple ATM program with tasks like:

- o Checking balance
- o Depositing money
- o Withdrawing money

Define at least three functions to modularize this program.

2. Algorithm

1. Initialize a static balance (₹1000).

2. Display a menu for ATM operations:

- Check Balance
- Deposit Money
- Withdraw Money
- Exit
- • Take user input for choice.

3. Based on choice:

- If **1**, show the current balance.
- If **2**, prompt user for amount, add it to balance.
- If **3**, prompt for amount, check if balance is sufficient. If yes, deduct it.
- If **4**, exit the program.
- Else, show invalid choice message.

4. Repeat the process until user chooses to exit.

3. Pseudocode

START

Set balance = 1000

REPEAT

DISPLAY ATM Menu

PROMPT user to enter choice

IF choice == 1 THEN

DISPLAY current balance

ELSE IF choice == 2 THEN

PROMPT user to enter amount to deposit

ADD amount to balance

DISPLAY deposit confirmation

ELSE IF choice == 3 THEN

PROMPT user to enter amount to withdraw

IF amount <= balance THEN

SUBTRACT amount from balance

DISPLAY withdrawal confirmation

ELSE

DISPLAY "Insufficient balance"

ELSE IF choice == 4 THEN

DISPLAY "Thank you" message

ELSE

DISPLAY "Invalid choice"

UNTIL choice == 4

END

4. Program Code

```
import java.util.Scanner;

public class D42 {
    static double balance = 1000.0;
    public static void checkBalance() {
        System.out.println("Current Balance: ₹" + balance);
    }
    public static void depositMoney(double amount) {
        balance += amount;
        System.out.println("Deposited: ₹" + amount);
    }
    public static void withdrawMoney(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: ₹" + amount);
        } else {
            System.out.println("Insufficient balance!");
        }
    }
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int choice;
    do {
        System.out.println("\n===== ATM Menu =====");
        System.out.println("1. Check Balance");
        System.out.println("2. Deposit Money");
        System.out.println("3. Withdraw Money");
        System.out.println("4. Exit");
        System.out.print("Enter choice: ");
        choice = sc.nextInt();
    }
}
```

```

switch (choice) {
    case 1:
        checkBalance();
        break;
    case 2:
        System.out.print("Enter amount to deposit: ");
        double deposit = sc.nextDouble();
        depositMoney(deposit);
        break;
    case 3:
        System.out.print("Enter amount to withdraw: ");
        double withdraw = sc.nextDouble();
        withdrawMoney(withdraw);
        break;
    case 4:
        System.out.println("Thank you for using the ATM.");
        break;
    default:
        System.out.println("Invalid choice.");
}
} while (choice != 4);

sc.close();
}
}

```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	1	1000	1000	pass
2	2	2000/2000	2000/2000	pass
3	3	200/withdraw	200/withdraw	pass

4

If we enter 4 it will come out of the loop

7. Observation / Reflection

The Simple ATM program effectively demonstrates basic banking operations like balance checks, deposits, and withdrawals. Key observations include:

1. Menu-driven interface: The program uses a simple and intuitive menu-driven interface, making it user-friendly.
 2. Basic banking operations: The program implements fundamental banking operations, showcasing conditional statements and loops.
 3. Error handling: The program handles insufficient balance errors and invalid user input, demonstrating basic error handling.
 4. Code organization in structured way
-

Problem Solving Problem 1.1

1. Greeting Function

Create greetUser (String name) to greet the user.

Call it three times with different names.

2. Algorithm

1. Define a method greetUser that takes a name parameter.
 2. Inside the greetUser method, print a greeting message with the provided name.
 3. In the main method, call the greetUser method with different names.
-

3. Pseudocode

```
FUNCTION greetUser(name)
```

```
    PRINT "Hello.... " + name
```

```
MAIN
```

```
    greetUser("Adnan")
```

```
    greetUser("Sariya")
```

greetUser("Afreen")

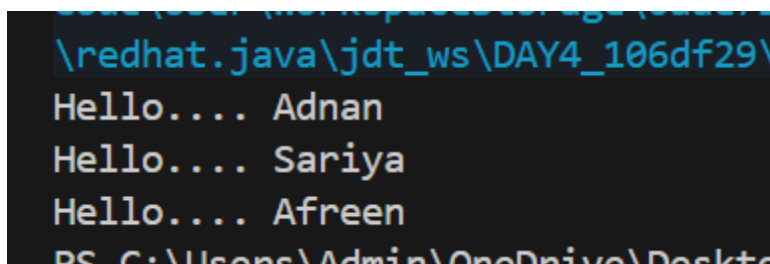
4. Program Code

```
public class P1_1 {
    static void greetUser(String name){
        System.out.println("Hello.... "+name);
    }
    public static void main(String[] args) {
        greetUser("Adnan");
        greetUser("Sariya");
        greetUser("Afreen");
    }
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	Adnan	Hello...Adnan	Hello...Adnan	pass
2	Sariya	Hello...Sariya	Hello...Sariya	pass
3	Afreen	Hello...Afreen	Hello...Afreen	pass

6. Screenshots of Output



```
\redhat.java\jdt_ws\DAY4_106df29\
Hello.... Adnan
Hello.... Sariya
Hello.... Afreen
PS C:\Users\Admin\OneDrive\Desktop
```


7. Observation / Reflection

- The code promotes code reusability by allowing multiple greetings without duplication.

It is simple and readable, demonstrating basic programming principles.

Problem Solving Problem 1.2

1. Price Calculator (Function Composition)

double calculateDiscount (double originalPrice, double discountPercentage);

double calculate Tax (double amount, double taxRate);

double calculate Final Price (double itemPrice, double discountPerc, double taxRate);

Call and print the result.

2. Algorithm

- Define a method `calculateSquare(int num)` that:
 - Takes an integer `num` as input.
 - Calculates the square of `num` (i.e., `num * num`).
 - Returns the result.
 - In the `main` method:
 - Call `calculateSquare(5)` and store the result in variable `a`.
 - Print the value of `a`.
 - (Optional) The commented lines show other possible method calls.
-

3. Pseudocode

FUNCTION `calculateSquare(num)`:

`result` \leftarrow `num` \times `num`

 RETURN `result`

BEGIN MAIN

a ← CALL calculateSquare(5)

PRINT a

END

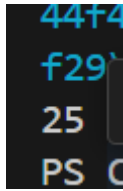
4. Program Code

```
public class P1_2 {  
    static int calculateSquare(int num){  
        int res=num*num;  
        return res;  
    }  
    public static void main(String[] args) {  
        int a=calculateSquare(5);  
        System.out.println(a);  
        // calculateSquare(3);  
        // calculateSquare(2);  
    }  
}
```

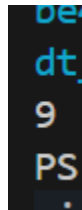
5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	5	25	25	pass
2	3	9	9	pass
3	2	4	4	pass

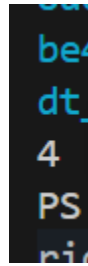
6. Screenshots of Output



```
44+4  
f29  
25  
PS C
```



```
be4  
dt_  
9  
PS
```



```
be4  
dt_  
4  
PS  
nic
```

7. Observation / Reflection

- **Purpose:** The program demonstrates how to define and use a method in Java.
- **Output:** The output will be 25, as 5 squared is 25.
- **Modularity:** The use of a method (`calculateSquare`) makes the code reusable and clean.



Stemup
A Unit of Pragnova Pvt Ltd

Problem Solving Problem 1.3

1. Sum Two Numbers

Create double add Numbers (double num1, double num2).

Call it and print the sum.

2. Algorithm

- Define a method `addNumbers(double num1, double num2):`
 - Accepts two double numbers as arguments.
 - Calculates their sum (`num1 + num2`).
 - Returns the result.

- In the `main` method:
- Call the `addNumbers` method with values `12` and `12.2`.
- Store the result in a variable `sum`.
- Print the sum with a message.

3. Pseudocode

FUNCTION `addNumbers(num1, num2)`:

`result` \leftarrow `num1 + num2`

 RETURN `result`

BEGIN MAIN

`sum` \leftarrow CALL `addNumbers(12, 12.2)`

 PRINT "The sum of two numbers is " + `sum`

END

4. Program Code

```
class P1_3{
    static double addNumbers(double num1,double num2){
        double res=num1+num2;
        return res;
    }
    public static void main (String[]args){
        double sum=addNumbers(12,12.2);
        System.out.println("The sum of two numbers is "+sum);
    }
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	12	12.2	24.2	pass
2	10	1	11	pass
3	14	15	29	pass

6. Screenshots of Output

```
68\redhat.java\jdt_ws\DAY4_106df29\
The sum of two numbers is 24.2
```

```
9\bin' 'P1_3'
The sum of two numbers is 11.0
PS C:\Users\Admin\OneDrive\Desktop
```

```
8\redhat.java\jdt_ws\DAY4_106df29\
The sum of two numbers is 29.0
PS C:\Users\Admin\OneDrive\Desktop
```

7. Observation / Reflection

- **Good Practices:**

- Use of a method keeps logic modular and reusable.
- Using `double` allows support for both whole numbers and decimals.

- **Readability:** Code is clean, well-structured, and readable.

A Unit of Pragnova Pvt Ltd

Problem Solving problem 1.4

1. Temperature Converter

Create double celsius To Fahrenheit (double celsius).

Create double fahrenheit To Celsius (double fahrenheit).

Test with sample values.

2. Algorithm

- **Method 1: celsiusToFahrenheit(double c)**
 - Input: Temperature in Celsius.
 - Process: Apply formula $F = (C \times 1.8) + 32$.
 - Output: Temperature in Fahrenheit.
- **Method 2: fahrenheitToCelsius(double F)**
 - Input: Temperature in Fahrenheit.
 - Process: Apply formula $C = (F - 32) / 1.8$.
 - Output: Temperature in Celsius.
- In main method:
 - Call `celsiusToFahrenheit(45)` and store the result in a.
 - Call `fahrenheitToCelsius(113.0)` and store the result in b.
 - Print both results.

3. Pseudocode

FUNCTION `celsiusToFahrenheit(c)`:

Fahrenheit $\leftarrow (c \times 1.8) + 32$

RETURN Fahrenheit

FUNCTION `fahrenheitToCelsius(F)`:

Celsius $\leftarrow (F - 32) / 1.8$

RETURN Celsius

BEGIN MAIN

a \leftarrow CALL `celsiusToFahrenheit(45)`

b \leftarrow CALL `fahrenheitToCelsius(113.0)`

PRINT "The Fahrenheit is " + a

PRINT "The Celsius is " + b

END

4. Program Code

```
public class p1_4 {  
    static double celsiusToFahrenheit(double c){  
        //F = (°C × 1.8) + 32  
        double Fahrenheit=(c*1.8)+32;  
        return Fahrenheit;  
    }  
    static double fahrenheitToCelsius(double F){  
        //C = (°F - 32) / 1.8  
        double Celsius=(F - 32) / 1.8;  
        return Celsius;  
    }  
    public static void main(String[]args){  
        double a=celsiusToFahrenheit(45);  
        double b=fahrenheitToCelsius(113.0);  
        System.out.println("The Fahrenheit is "+a);  
        System.out.println("The Celsius is "+b);  
    }  
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	45,113	113,45	113,45	pass
2	466,155.0	852.8,68	852.8,68	pass
3	123,134	253.4,56.6	253.4,56.6	pass

6. Screenshots of Output

```
\Users\Admin\AppData\Roam
The Fahrenheit is 113.0
The Celsius is 45.0
PS C:\Users\Admin\OneDrive\
```

```
8\redhat.java\jdt_ws\DAY4_106df29\bin'
The Fahrenheit is 852.8000000000001
The Celsius is 68.33333333333333
```

```
The Fahrenheit is 253.4
The Celsius is 56.666666666666664
PS C:\Users\Admin\OneDrive\Desktop\Bride
```

7. Observation / Reflection

- **Purpose:** The program accurately converts temperatures between Celsius and Fahrenheit.
- **Output:**

```
csharp
CopyEdit
The Fahrenheit is 113.0
The Celsius is 45.0
```

Problem Solving Problem 2.1

1. Scope Experiment

```
public class ScopeTest {
    static String globalMessage = "I am global!";
    static void displayMessages() {
    }
    String localMessage = "I am local!";
    System.out.println(globalMessage);
```



```
public static void main(String[] args) {  
    displayMessages();  
    // Try to print localMessage here and observe the error.  
}  
}
```

4. Program Code

Before solving error

```
public class p2_1 {  
    static String globalMessage = "I am global!";  
    static void displayMessages() {  
    }  
    String localMessage = "I am local!";  
    System.out.println(globalMessage);  
    public static void main(String[] args) {  
        displayMessages();  
        // Try to print localMessage here and observe the error.  
    }  
}
```

A Unit of Pragnova Pvt Ltd

After solving error

Code:

```
public class p2_1 {  
    // static String globalMessage = "I am global!";  
    static void displayMessages() {  
        String localMessage = "I am local!";  
        System.out.println(localMessage);  
    }  
  
    // System.out.println(globalMessage);  
    public static void main(String[] args) {  
        displayMessages();  
        // Try to print localMessage here and observe the error.  
    }  
}
```

```
}  
}
```

6. Screenshots of Output

```
p2_1.java
❌ <identifier> expected (errors(1): 7:1-7:35) [Ln 7, Col 1]
❌ cannot find symbol (errors(3): 7:8-7:11) [Ln 7, Col 8] ^
   symbol:   class out
   location: class System
❌ Syntax error on token ":", @ expected after this token Java(1610612976) [Ln 7, Col 11]
❌ cannot find symbol (errors(2): 7:20-7:33) [Ln 7, Col 20] ^
   symbol:   class globalMessage
   location: class p2_1
❌ Syntax error, insert "SimpleName" to complete QualifiedName Java(1610612976) [Ln 7, Col 33]
❌ Syntax error, insert "Identifier (" to complete MethodHeaderName Java(1610612976) [Ln 7, Col 33]
❌ Syntax error, insert ")" to complete MethodDeclaration Java(1610612976) [Ln 7, Col 33]
⚠ Variable globalMessage is never read (hints(1): 3:15-3:28) [Ln 3, Col 15]
⚠ Variable localMessage is never read (hints(2): 6:8-6:20) [Ln 6, Col 8]
p2_1.java
```

After solving error

```
PS C:\Users\Admin\OneDrive\Desktop> cd ridgecourseStemup\Stemup
PS C:\Users\Admin\OneDrive\Desktop> X:+ShowCodeDetailsInException
PS C:\Users\Admin\OneDrive\Desktop> 6a2d4144f4368\redhat.java
I am local!
PS C:\Users\Admin\OneDrive\Desktop>
```



7. Observation / Reflection

In the above code we can't able to call the global msg.

Because it is out of the scope

Problem Solving Problem 2.3

1. Refactor Repetitive Cod

Refactor your Day 2 calculator to use:

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

Discuss improvements in clarity and reuse.

2. Algorithm

- Start the program and import the `Scanner` class for user input.
 - Prompt the user to input:
 - The **first number** (num1)
 - The **second number** (num2)
 - Prompt the user to choose an **operator**: +, -, *, or /.
 - Use a `switch` statement to:
 - Call the corresponding method (add, subtract, multiply, divide) based on the chosen operator.
 - If division is chosen and the second number is 0, display an error and return 0.
 - Print the result.
 - Close the scanner
-

3. Pseudocode

START

PROMPT "Enter first number"

READ num1

PROMPT "Enter second number"

READ num2

PROMPT "Select operation: +, -, *, /"

READ operator

SWITCH (operator)

CASE '+':

result \leftarrow num1 + num2

CASE '-':

result \leftarrow num1 - num2

CASE '*':

result \leftarrow num1 \times num2

CASE '/':

IF num2 == 0 THEN

PRINT "Cannot divide by zero"

result \leftarrow 0

ELSE

result \leftarrow num1 / num2

DEFAULT: A Unit of Pragnova Pvt Ltd

PRINT "Invalid operator"

EXIT

PRINT "Result: " + result

END

4. Program Code

```
import java.util.Scanner;
```

```
public class p2_2 {
    public static double add(double num1, double num2) {
        return num1 + num2;
    }
    public static double subtract(double num1, double num2) {
        return num1 - num2;
    }
    public static double multiply(double num1, double num2) {
        return num1 * num2;
    }
    public static double divide(double num1, double num2) {
        if (num2 == 0) {
            System.out.println("Error: Cannot divide by zero.");
            return 0;
        }
        return num1 / num2;
    }
}

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

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

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

    System.out.println("Select operation: +, -, *, /");
    char operator = sc.next().charAt(0);

    double result;

    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.");
            return;
    }

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

```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	4,4(/)	1.0	1	pass
2	162,78(*)	12636.0	12636.0	pass
3	50,33(-)	17.0	17	pass

6. Screenshots of Output

A Unit of Pragnova Pvt Ltd

```
4_106df29\bin' 'p2_2
Enter first number: 4
Enter second number: 4
Select operation: +, -, *, /
/
Result: 1.0
```

```
dt_ws\DAY4_106df29\bin' 'p2_2
Enter first number: 162
Enter second number: 78
Select operation: +, -, *, /
*
Result: 12636.0
PS C:\Users\Admin\OneDrive\Desktop
```

```
Enter first number: 50
Enter second number: 33
Select operation: +, -, *, /
-
Result: 17.0
PS C:\Users\Admin\OneDrive\Desktop
```

7. Observation / Reflection

In Day 2 calculator we only used the switch case now we implement the method with parameter ,and we use user i/p also and if any number is divided by 0 it will show ("Error: Cannot divide by zero.");

Division by Zero: Though the error is handled, returning 0 could be misleading

Problem Solving Problem 2.2

1. Price Calculator (Function Composition)

double calculateDiscount (double originalPrice, double discountPercentage);

double calculate Tax (double amount, double taxRate);

double calculate Final Price (double itemPrice, double discountPerc, double taxRate);

Call and print the result.

2. Algorithm

Step 1: Start

Step 2: Define function `calculateDiscount(originalPrice, discountPercentage)`

→ Calculate `discountAmount = originalPrice * discountPercentage / 100`

→ Return `originalPrice - discountAmount`

Step 3: Define function `calculateTax(amount, taxRate)`

→ Calculate `taxAmount = amount * taxRate / 100`

→ Return `amount + taxAmount`

Step 4: Define function `calculateFinalPrice(itemPrice, discountPerc, taxRate)`

→ Call `calculateDiscount(itemPrice, discountPerc)` and store in `discountedPrice`

→ Call `calculateTax(discountedPrice, taxRate)` and return as `finalPrice`

Step 5: In `main()`, initialize:

→ `itemPrice = 1000.0`

→ `discountPercentage = 10.0`

→ `taxRate = 8.0`

Step 6: Call `calculateFinalPrice()` with above values and store result in `finalPrice`

Step 7: Print "Final Price after Discount and Tax: ₹" + `finalPrice`

Step 8: End

3. Pseudocode

Start

Function `calculateDiscount(originalPrice, discountPercentage)`:

`discountAmount = originalPrice * discountPercentage / 100`

`return originalPrice - discountAmount`

Function `calculateTax(amount, taxRate)`:

`taxAmount = amount * taxRate / 100`

`return amount + taxAmount`

Function `calculateFinalPrice(itemPrice, discountPerc, taxRate)`:


```
discountedPrice = calculateDiscount(itemPrice, discountPerc)

finalPrice = calculateTax(discountedPrice, taxRate)

return finalPrice
```

Main:

```
Set itemPrice to 1000.0

Set discountPercentage to 10.0

Set taxRate to 8.0
```

```
finalPrice = calculateFinalPrice(itemPrice, discountPercentage, taxRate)

Display "Final Price after Discount and Tax: ₹" + finalPrice
```

End



4. Program Code

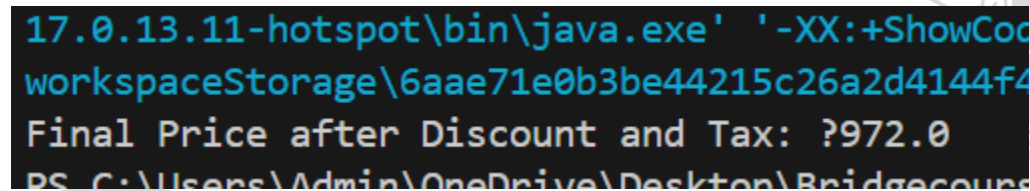
```
public class P2_3 {

    public static double calculateDiscount(double originalPrice, double
discountPercentage) {
        return originalPrice - (originalPrice * discountPercentage / 100);
    }
    public static double calculateTax(double amount, double taxRate) {
        return amount + (amount * taxRate / 100);
    }
    public static double calculateFinalPrice(double itemPrice, double
discountPerc, double taxRate) {
        double priceAfterDiscount = calculateDiscount(itemPrice, discountPerc);
        double finalPrice = calculateTax(priceAfterDiscount, taxRate);
        return finalPrice;
    }
    public static void main(String[] args) {
        double itemPrice = 1000.0;
        double discountPercentage = 10.0;
```

```
double taxRate = 8.0;

double finalPrice = calculateFinalPrice(itemPrice, discountPercentage,
taxRate);
System.out.println("Final Price after Discount and Tax: ₹" + finalPrice);
}
}
```

5. Screenshots of Output



```
17.0.13.11-hotspot\bin\java.exe' '-XX:+ShowCodeDetails -ea workspaceStorage\6aae71e0b3be44215c26a2d4144f4\Main.class'
Final Price after Discount and Tax: ₹972.0
PS C:\Users\Admin\OneDrive\Desktop\Bridgeworks>
```

Problem Solving Problem 3.1

1. Customizable Greeting (Overloading)

```
void customGreet(String name, String greeting);
```

```
void customGreet(String name);
```

```
void customGreet();
```

Demonstrate calling all variants.

2. Algorithm

- Define three overloaded methods named `customGreet`:

- One takes **name** and **greeting** → prints: "<greeting>, <name>!"
 - One takes only **name** → prints: "Hello, <name>!"
 - One takes **no parameters** → prints: "Hello, Guest!"
- In `main()`:
 - Call all three versions of `customGreet` with appropriate arguments.
 - Output the result of each call.

3. Pseudocode

FUNCTION `customGreet(name, greeting)`:

 PRINT `greeting + ", " + name + "!"`

FUNCTION `customGreet(name)`:

 PRINT `"Hello, " + name + "!"`

FUNCTION `customGreet()`:

 PRINT `"Hello, Guest!"`

MAIN:

 CALL `customGreet("Sariya", "Good Morning")`

 CALL `customGreet("Sariya")`

 CALL `customGreet()`

4. Program Code

```
public class p3_1{

    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("Hello, Guest!");
}

public static void main(String[] args) {

    customGreet("Sariya", "Good Morning");
    customGreet("Sariya");
    customGreet();
}
}

```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	Adnan ,good night	Good Morning, Adnan!	Good Morning, Adnan!	pass
2	Sameer	Hello, Sameer!	Hello, Sameer!	pass
3		Hello, Guest!	Hello, Guest!	pass

6. Screenshots of Output

```

_Bridge_course\DAY4'; & 'C:\Users\Admin\AppData\Roam
Good Night, Adnan!
Hello, Sameer!
Hello, Guest!
PS C:\Users\Admin\OneDrive

```

7. Observation / Reflection

According to the parameter giving it will executes the output when we take the user input

But now we already given the input for the method that's why it is printing all the output

Problem Solving Problem 3.2

1. Power Calculator

2. Algorithm

- Define a method `myPower(int base, int exponent):`
 - Initialize `result = 1.`
 - Loop from 1 to `exponent.`
 - Multiply `result` by `base` in each iteration.
 - Return the final `result.`
- In `main():`
 - Set `base = 2, exponent = 5.`
 - Call `myPower(base, exponent)` and store result.
 - Also calculate power using `Math.pow(base, exponent).`
 - Print both results for comparison.

3. Pseudocode

FUNCTION `myPower(base, exponent):`

`result ← 1`

 FOR `i ← 1 TO exponent` DO

result \leftarrow result \times base

RETURN result

BEGIN MAIN

base \leftarrow 2

exponent \leftarrow 5

customResult \leftarrow CALL myPower(base, exponent)

mathResult \leftarrow CALL Math.pow(base, exponent)

PRINT "Result using myPower(): " + customResult

PRINT "Result using Math.pow(): " + mathResult

END



Stemup

4. Program Code

```
public class P3_2 {  
  
    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) {  
        int base = 2;  
        int exponent = 5;  
  
        int customResult = myPower(base, exponent);  
    }  
}
```

```
double mathResult = Math.pow(base, exponent);

System.out.println("Result using myPower(): " + customResult);
System.out.println("Result using Math.pow(): " + mathResult);
}
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	2,5	32	32.0	pass
2	5,4	625	625.0	pass
3	8,9	134217728	1. 134217728E8	pass

6. Screenshots of Output

```
workspaceStorage\6aae71e0b3be4421
Result using myPower(): 32
Result using Math.pow(): 32.0
```

```
X:\ShowCodeDetailsInExceptionMessages
6a2d4144f4368\redhat.java\jdt_ws\D
Result using myPower(): 625
Result using Math.pow(): 625.0
PS C:\Users\Admin\OneDrive\Desktop
```

```
X:\ShowCodeDetailsInExceptionMessages
6a2d4144f4368\redhat.java\jdt_ws\DAY4_106
Result using myPower(): 134217728
Result using Math.pow(): 1.34217728E8
PS C:\Users\Admin\OneDrive\Desktop\Bridge
```

7. Observation / Reflection

I felt more difficulty in this problem compare to other problems

Problem Solving Problem 3.3

1. Trace the Flow

Create 3 functions A, B, and C where:

- o A calls B
- o B returns a value used in A to call C
- o C prints the result

Manually trace the call order and explain execution.

2. Algorithm

- Define method `C(int value):`
 - Prints the final result.
- Define method `B():`
 - Calculates $10 + 5$ and returns the result (15).
- Define method `A():`
 - Calls `B()` and stores the result.
 - Passes the result to method `C()`.
- In `main():`
 - Call method `A()` to initiate the process.

3. Pseudocode

FUNCTION C(value):

 PRINT "Final Result is : " + value

FUNCTION B():

 result \leftarrow 10 + 5

 RETURN result

FUNCTION A():

 bResult \leftarrow CALL B()

 CALL C(bResult)

MAIN:

 CALL A()

4. Program Code

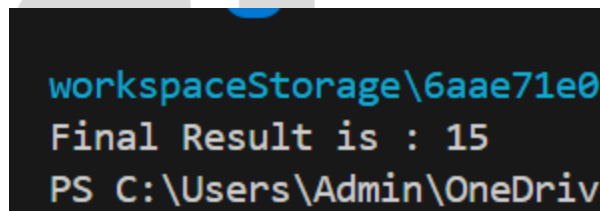
```
public class p3_3 {  
  
    public static void C(int value) {  
        System.out.println("Final Result is : " + value);  
    }  
  
    public static int B() {  
        int result = 10 + 5;  
        return result;  
    }  
  
    public static void A() {  
        int bResult = B();  
        C(bResult);  
    }  
}
```

```
public static void main(String[] args) {
    A();
}
}
```

5. Test Cases

Test Case No.	Input	Expected Output	Actual Output	Status (Pass/Fail)
1	10+5	15	15	pass

6. Screenshots of Output



```
workspaceStorage\6aae71e0
Final Result is : 15
PS C:\Users\Admin\OneDrive
```

7. Observation / Reflection

In this we Create a 3 Methods in that we put output Statement in C Method . and in b method we put operation like 10+5.and we store b method in the bresult .and we call A method in main Method