ChatGpt Analysis:

Here is a detailed analysis of the provided solution for Exercise 5, graded across **correctness**, **accuracy**, and **style**:

**1. Correctness (Grade: 9/10)**

* **Positive Points**:
  + The program handles basic arithmetic operations (addition, subtraction, multiplication, and division) correctly using BiFunction instances.
  + The switch statement accurately maps user input to the appropriate operation.
  + The result of the operation is displayed correctly.
* **Minor Issue**:
  + Division by zero is not handled. If the user inputs 0 as the second number and selects division, the program will throw a java.lang.ArithmeticException.

**2. Accuracy (Grade: 8.5/10)**

* **Positive Points**:
  + The use of BiFunction is appropriate for this task and demonstrates an understanding of functional interfaces.
  + User input is correctly obtained using Scanner, and the prompts for input are clear.
  + The switch statement is used effectively to apply the correct operation.
* **Suggestions for Improvement**:
  + **Division by Zero Handling**: Add a check to prevent division by zero:

java

CopyEdit

BiFunction<Integer, Integer, Integer> division = (num1, num2) -> {

if (num2 == 0) {

throw new ArithmeticException("Division by zero is not allowed");

}

return num1 / num2;

};

Alternatively, warn the user and exit gracefully if division by zero is attempted.

* + **Error Handling for Invalid Inputs**:  
    Add error handling for invalid inputs (e.g., non-integer values or unsupported operations) to avoid crashes or unexpected behavior.

**3. Style (Grade: 8/10)**

* **Positive Points**:
  + Code is well-structured and easy to follow.
  + The variable names (input1, input2, input3, output) are functional and understandable for this simple example.
* **Suggestions for Improvement**:
  + **Method Extraction**: The main method is slightly cluttered. Extract the logic for user input and operation handling into separate methods to improve readability:

java

CopyEdit

private static int getUserInput(String prompt) {

System.out.println(prompt);

return scan.nextInt();

}

* + **Default Case Output**: The default case in the switch statement should terminate early (e.g., with return) instead of proceeding to print an incorrect result of 0.
  + **Avoid Global Scanner**: Declaring Scanner as a global constant (private static final Scanner scan) is unnecessary. It could instead be initialized and closed in the main method.

**Revised Solution with Improvements**

Here’s a version of the code that incorporates the suggested improvements:

java

CopyEdit

import java.util.Scanner;

import java.util.function.BiFunction;

public class Main {

public static void main(String[] args) {

try (Scanner scan = new Scanner(System.in)) {

int input1 = getUserInput(scan, "Please Insert First Number:");

int input2 = getUserInput(scan, "Please Insert Second Number:");

int operation = getUserInput(scan, "Please Choose an Operation from the following list:\n1 - Addition\n2 - Subtraction\n3 - Multiplication\n4 - Division");

BiFunction<Integer, Integer, Integer> addition = Integer::sum;

BiFunction<Integer, Integer, Integer> subtraction = (num1, num2) -> num1 - num2;

BiFunction<Integer, Integer, Integer> multiplication = (num1, num2) -> num1 \* num2;

BiFunction<Integer, Integer, Integer> division = (num1, num2) -> {

if (num2 == 0) {

throw new ArithmeticException("Division by zero is not allowed");

}

return num1 / num2;

};

int result;

switch (operation) {

case 1 -> result = addition.apply(input1, input2);

case 2 -> result = subtraction.apply(input1, input2);

case 3 -> result = multiplication.apply(input1, input2);

case 4 -> result = division.apply(input1, input2);

default -> {

System.out.println("Unsupported Operation. Exiting...");

return;

}

}

System.out.println("The Result of the Chosen Operation on the Two Numbers is: " + result);

}

}

private static int getUserInput(Scanner scan, String prompt) {

System.out.println(prompt);

while (!scan.hasNextInt()) {

System.out.println("Invalid input. Please enter an integer:");

scan.next(); // Consume invalid input

}

return scan.nextInt();

}

}

**Execution Example (Output)**

**Valid Input Scenario**:

sql

CopyEdit

Please Insert First Number:

8

Please Insert Second Number:

4

Please Choose an Operation from the following list:

1 - Addition

2 - Subtraction

3 - Multiplication

4 - Division

3

The Result of the Chosen Operation on the Two Numbers is: 32

**Invalid Input Scenario**:

sql

CopyEdit

Please Insert First Number:

abc

Invalid input. Please enter an integer:

5

Please Insert Second Number:

0

Please Choose an Operation from the following list:

1 - Addition

2 - Subtraction

3 - Multiplication

4 - Division

4

Exception in thread "main" java.lang.ArithmeticException: Division by zero is not allowed

**Overall Grades**

| **Aspect** | **Grade** | **Comments** |
| --- | --- | --- |
| **Correctness** | 9/10 | Handles operations well but fails to address division by zero. |
| **Accuracy** | 8.5/10 | Could improve error handling and edge case management. |
| **Style** | 8/10 | Clean code, but modularization and better input handling would help. |

Let me know if you'd like further refinements or more exercises!