Task 1: Data Types/Variables

Write a program that declares two integer variables, swaps their values without using a third variable, and prints the result.

```
public class SwapWithoutTemp {
    public static void main(String[] args) {
        // Declare two integer variables
        int a = 10;
        int b = 20;

        // Print the original values
        System.out.println("Before swap:");
        System.out.println("a = " + a);
        System.out.println("b = " + b);

        // Swap the values without using a third variable
        a = a + b; // a now becomes 30
        b = a - b; // b becomes 10 (30 - 20)
        a = a - b; // a becomes 20 (30 - 10)

        // Print the swapped values
        System.out.println("After swap:");
        System.out.println("a = " + a);
        System.out.println("b = " + b);
    }
}
```

Task 2: Operators

Create a program that simulates a simple calculator using command-line arguments to perform and print the result of addition, subtraction, multiplication, and division..

```
public class SimpleCalculator {
    public static void main(String[] args) {
        if (args.length != 3) {
            System.out.println("Usage: java SimpleCalculator <num1> <operation> <num2>");
            System.out.println("Operations: + for addition, - for subtraction, * for multiplication, / for division");
            return;
        }

        try {
            double num1 = Double.parseDouble(args[0]);
            String operation = args[1];
            double num2 = Double.parseDouble(args[2]);

            double result = 0;
            boolean validOperation = true;
```

```
switch (operation) {
    case "+":
        result = num1 + num2;
        break;
    case "-":
        result = num1 - num2;
        break;
    case "*":
        result = num1 * num2;
        break;
    case "/":
        if (num2 != 0) {
            result = num1 / num2;
        } else {
            System.out.println("Error: Division by zero is not allowed.");
            validOperation = false;
        }
        break;
    default:
        System.out.println("Error: Invalid operation. Use +, -, *, or /.");
        validOperation = false;
        break;
}
```

```
            if (validOperation) {
                System.out.println("Result: " + result);
            }
        } catch (NumberFormatException e) {
            System.out.println("Error: Please enter valid numbers.");
        }
    }
}
```

Task 3: Control Flow

Write a Java program that reads an integer and prints whether it is a prime number using a for loop and if statements.

```
import java.util.Scanner;

public class PrimeNumberChecker {
    public static void main(String[] args) {
        // Create a Scanner object to read input from the user
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter an integer
        System.out.print("Enter an integer: ");
        int number = scanner.nextInt();

        // Close the scanner to prevent resource leak
        scanner.close();

        boolean isPrime = true;

        // Check if the number is divisible by any integer from 2 to its square root
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                isPrime = false;
                break;
            }
        }

        // If the number is greater than 1 and not divisible by any integer other than itself and 1, it is prime
        if (number <= 1) {
            isPrime = false;
        }

        // Print the result
        if (isPrime) {
            System.out.println(number + " is a prime number.");
        } else {
            System.out.println(number + " is not a prime number.");
        }
    }
}
```

Task 4: Constructors

Implement a Matrix class that has a constructor which initializes the dimensions of a matrix and a method to fill the matrix with values.

```
public class Matrix {
    private int rows;
    private int columns;
    private int[][] matrix;

    // Constructor to initialize the dimensions of the matrix
    public Matrix(int rows, int columns) {
        this.rows = rows;
        this.columns = columns;
        this.matrix = new int[rows][columns];
    }
```

```
// Method to fill the matrix with values
public void fillMatrix(int[][] values) {
  if (values.length != rows || values[0].length != columns) {
    System.out.println("Error: Invalid dimensions of input values array.");
    return;
  }

  for (int i = 0; i < rows; i++) {
    for (int j = 0; j < columns; j++) {
      matrix[i][j] = values
```

Exception Handlind:
write a program that attempts to divide by zero, catches the arithmetic exception, and provides a custom error message;

1. **try-catch Block**:
   - The program contains a **try-catch** block to handle the potential division by zero.
   - Inside the **try** block, an attempt is made to divide an integer by zero, which would throw an **ArithmeticException**.
2. **Catching ArithmeticException**:
   - The **catch** block catches the **ArithmeticException** that occurs if division by zero is attempted.

- Inside the **catch** block, a custom error message is printed indicating that division by zero is not allowed.

```
public class DivideByZero {
    public static void main(String[] args) {
        try {
            int numerator = 10;
            int denominator = 0;
            int result = numerator / denominator;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero is not allowed.");
        }
    }
}
```