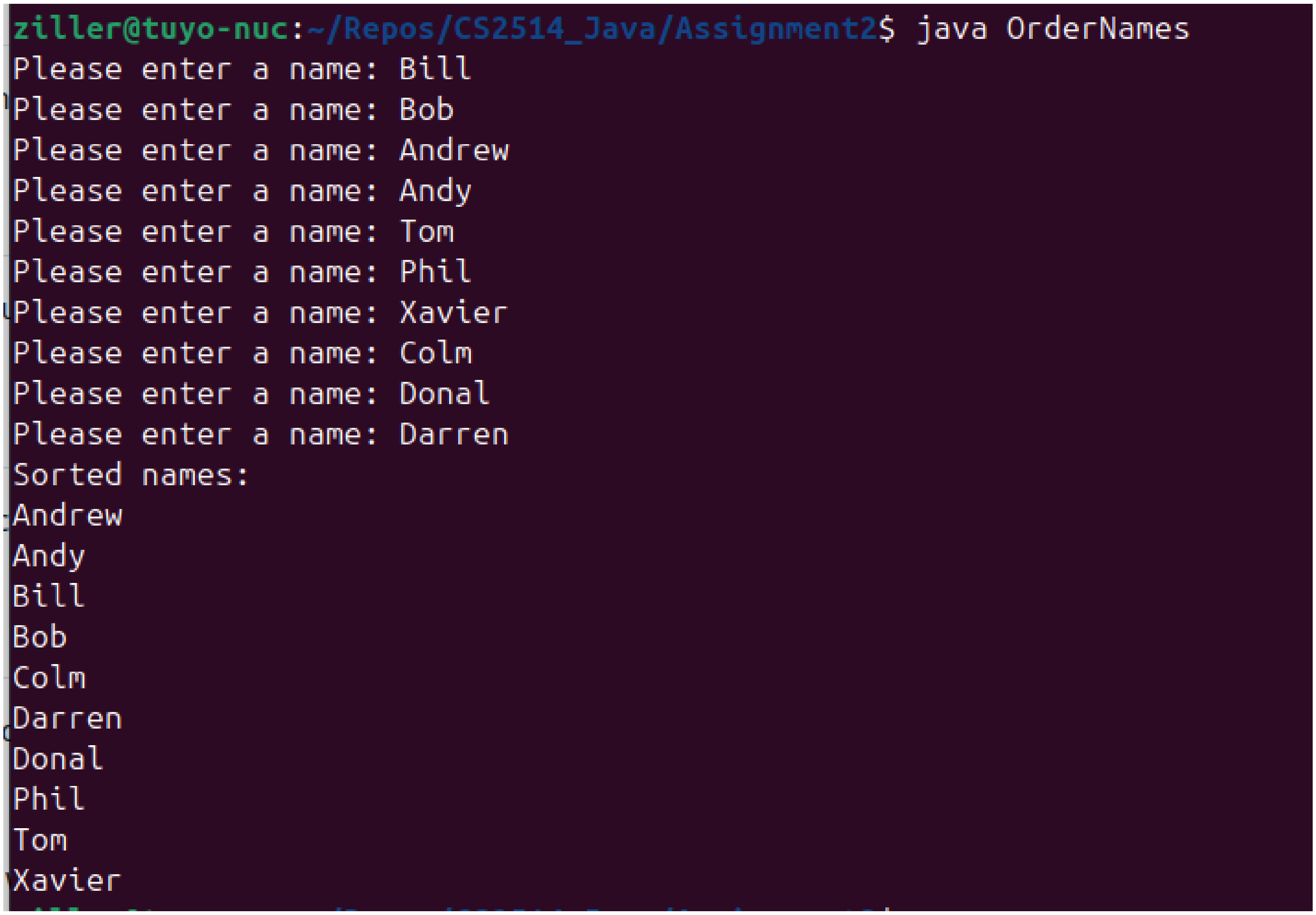
Program 1

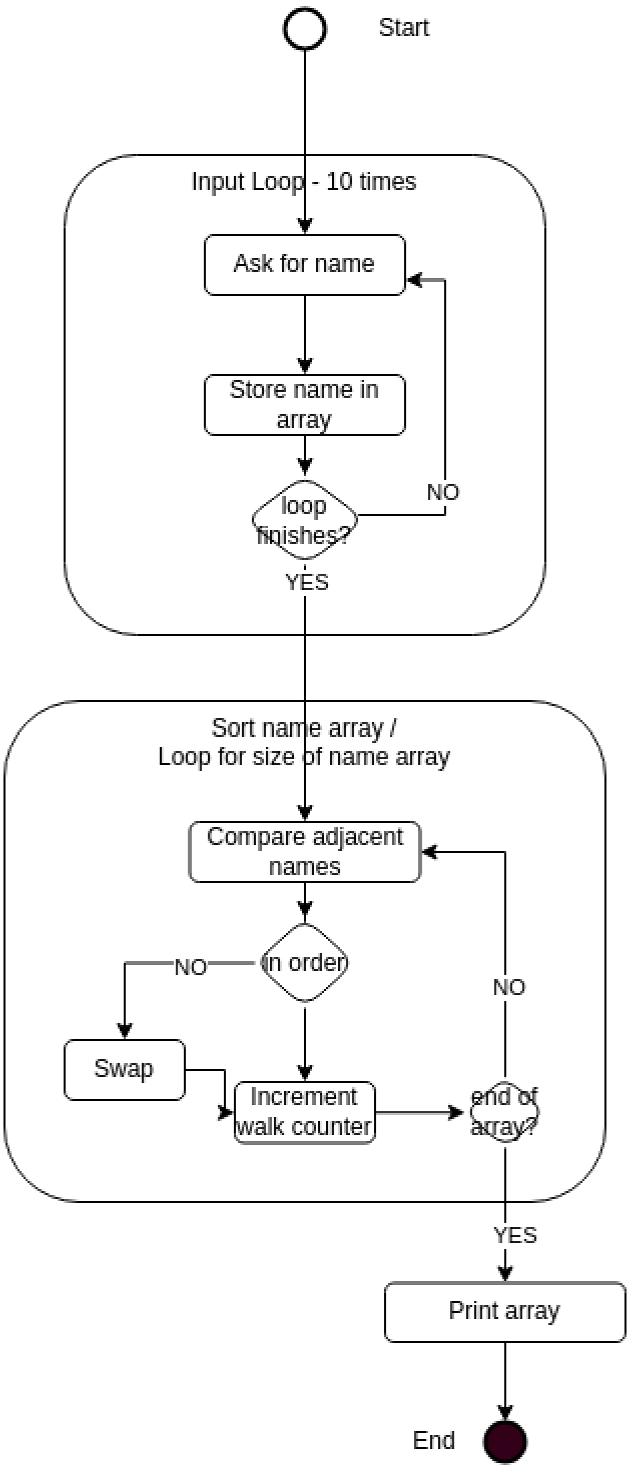
Write a program in JAVA that inputs 10 arbitrary names from the user and prints them in alphabetically.

Terminal Out



```
ziller@tuyo-nuc:~/Repos/CS2514_Java/Assignment2$ java OrderNames
Please enter a name: Bill
Please enter a name: Bob
Please enter a name: Andrew
Please enter a name: Andy
Please enter a name: Tom
Please enter a name: Phil
Please enter a name: Xavier
Please enter a name: Colm
Please enter a name: Donal
Please enter a name: Darren
Sorted names:
Andrew
Andy
Bill
Bob
Colm
Darren
Donal
Phil
Tom
Xavier
```

Flowchart



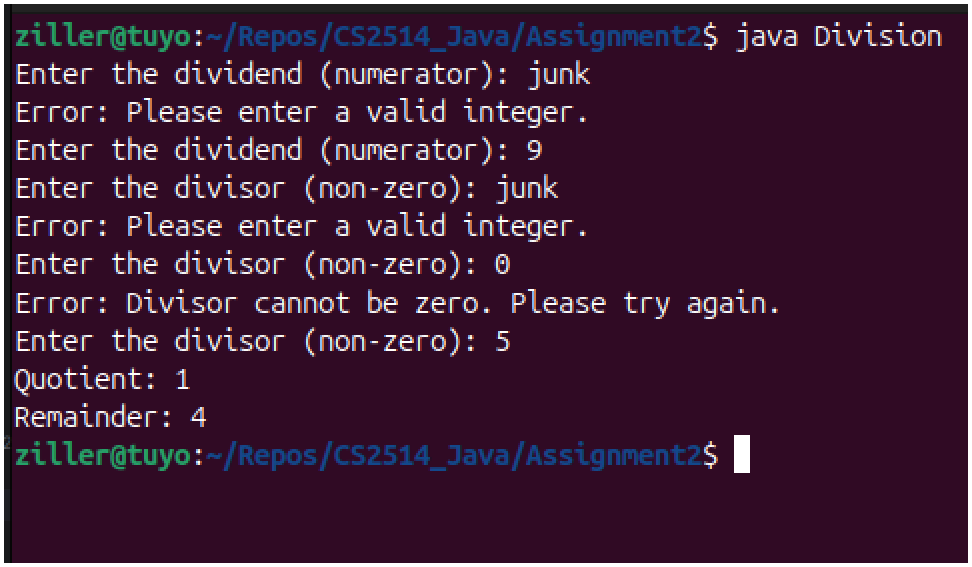
Source Code

```
import java.util.*;

// Class to take 10 names from user and print out in order
public class OrderNames{
    public static void main(String[] args){
        // make list buffer
        ArrayList<String> names = new ArrayList<>();
        // Create Scanenr Object
        Scanner scanner = new Scanner(System.in);

        // Make a loop of 10, ask for name -> store in list buffer
        for (int i = 1; i <= 10; i++){
            System.out.print("Please enter a name: ");
            String name = scanner.nextLine();
            names.add(name);    // Appending to array.
        }

        scanner.close();

        // Sort list of names -> Alphabetically
        //Collections.sort(names);
        bubbleSort(names);

        System.out.println("Sorted names: ");
        // Iterate over names and print
        for (String name : names){
            System.out.println(name);
        }
    }

    // Instead of using Collections.sort I'm going to try and
    // create my own sort algo here as recommended by question.
    // So bubblesort seems a good idea from CS2516:Algorithms and Data Structures

    public static void bubbleSort(ArrayList<String> names) {
        int n = names.size();
        // Outer loop for each pass
        for (int i = 0; i < n - 1; i++) {
            // Inner loop for comparing adjacent elements
            for (int j = 0; j < n - i - 1; j++) {
                // Compare adjacent strings
                // This should handle Adrian vs Aidan scenarios..
                if (names.get(j).compareTo(names.get(j + 1)) > 0) {
                    // Swap if elements are in the wrong order
                    String temp = names.get(j);
                    names.set(j, names.get(j + 1));
                    names.set(j + 1, temp);
                }
            }
        }
    }
}
```

Program 2

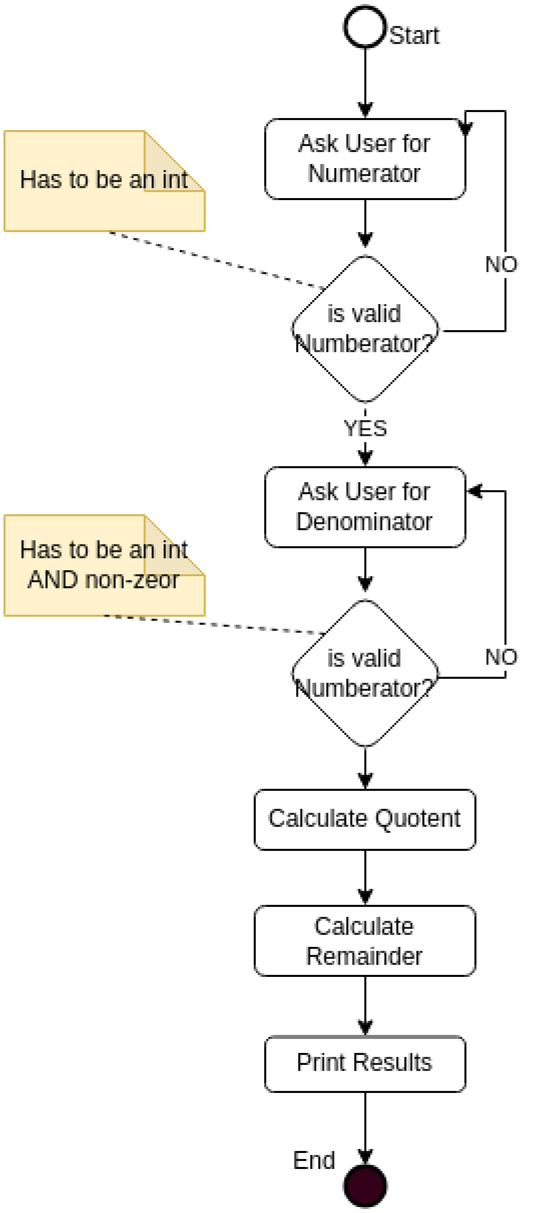
Write a program in JAVA to input the dividend and divisor from the user and find the quotient and remainder

Terminal Out



```
ziller@tuyo:~/Repos/CS2514_Java/Assignment2$ java Division
Enter the dividend (numerator): junk
Error: Please enter a valid integer.
Enter the dividend (numerator): 9
Enter the divisor (non-zero): junk
Error: Please enter a valid integer.
Enter the divisor (non-zero): 0
Error: Divisor cannot be zero. Please try again.
Enter the divisor (non-zero): 5
Quotient: 1
Remainder: 4
ziller@tuyo:~/Repos/CS2514_Java/Assignment2$
```

Flowchart



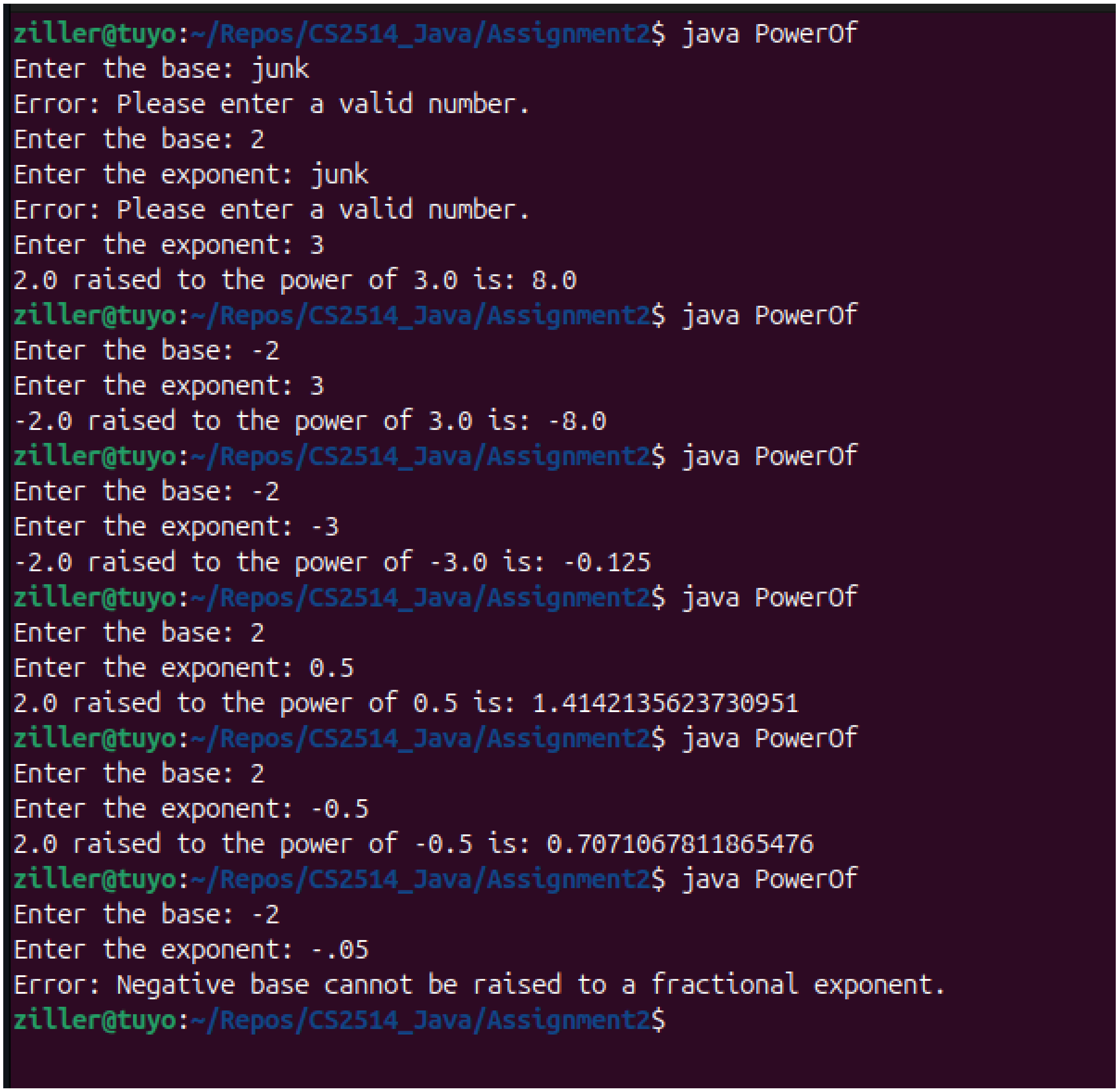
Source Code

```
import java.util.Scanner;

// Class to find the quotent and reminder for a division request
public class Division {

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

        // When I ask for the user input I need to make sure its Real numbers
        // and also ensure the Divisor is not zero.
        // So we introduce two methors to handle this checking..

        // Get the dividend (numerator)
        int dividend = getIntegerInput(scanner, "Enter the dividend (numerator): ");

        // Get the divisor (denominator)
        int divisor = getDivisorInput(scanner);

        // Perform the division and output the result
        int quotient = dividend / divisor;
        int remainder = dividend % divisor;

        System.out.println("Quotient: " + quotient);
        System.out.println("Remainder: " + remainder);

        scanner.close();
    }

    // Method to get a valid integer input
    public static int getIntegerInput(Scanner scanner, String prompt) {
        int number = 0;
        boolean valid = false;

        // Going to loop until I get valid integer from user
        while (!valid) {
            System.out.print(prompt);
            if (scanner.hasNextInt()) {
                number = scanner.nextInt();
                valid = true;
            } else {
                System.out.println("Error: Please enter a valid integer.");
                scanner.next(); // clear the invalid input
            }
        }
        return number;
    }

    // Method to get a valid divisor.
    // This also has to handle the divide by zero issue where the numerator did not.
    public static int getDivisorInput(Scanner scanner) {
        int divisor = 0;
        boolean valid = false;

        // Going to loop until I get valid integer from user
        while (!valid) {
            divisor = getIntegerInput(scanner, "Enter the divisor (non-zero): ");

            if (divisor == 0) {
                System.out.println("Error: Divisor cannot be zero. Please try again.");
            } else {
                valid = true;
            }
        }
        return divisor;
    }
}
```

Program 3

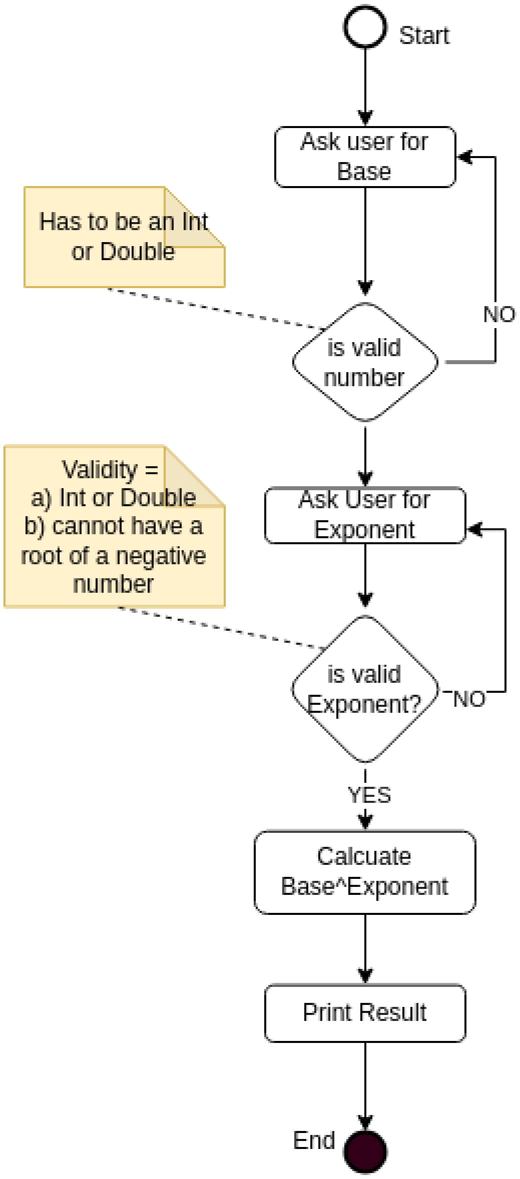
Write a program in JAVA to calculate the power of a number, where the base and exponent are taken as input from the user

Terminal Out



```
ziller@tuyo:~/Repos/CS2514_Java/Assignment2$ java PowerOf
Enter the base: junk
Error: Please enter a valid number.
Enter the base: 2
Enter the exponent: junk
Error: Please enter a valid number.
Enter the exponent: 3
2.0 raised to the power of 3.0 is: 8.0
ziller@tuyo:~/Repos/CS2514_Java/Assignment2$ java PowerOf
Enter the base: -2
Enter the exponent: 3
-2.0 raised to the power of 3.0 is: -8.0
ziller@tuyo:~/Repos/CS2514_Java/Assignment2$ java PowerOf
Enter the base: -2
Enter the exponent: -3
-2.0 raised to the power of -3.0 is: -0.125
ziller@tuyo:~/Repos/CS2514_Java/Assignment2$ java PowerOf
Enter the base: 2
Enter the exponent: 0.5
2.0 raised to the power of 0.5 is: 1.4142135623730951
ziller@tuyo:~/Repos/CS2514_Java/Assignment2$ java PowerOf
Enter the base: 2
Enter the exponent: -0.5
2.0 raised to the power of -0.5 is: 0.7071067811865476
ziller@tuyo:~/Repos/CS2514_Java/Assignment2$ java PowerOf
Enter the base: -2
Enter the exponent: -.05
Error: Negative base cannot be raised to a fractional exponent.
ziller@tuyo:~/Repos/CS2514_Java/Assignment2$
```

Flowchart



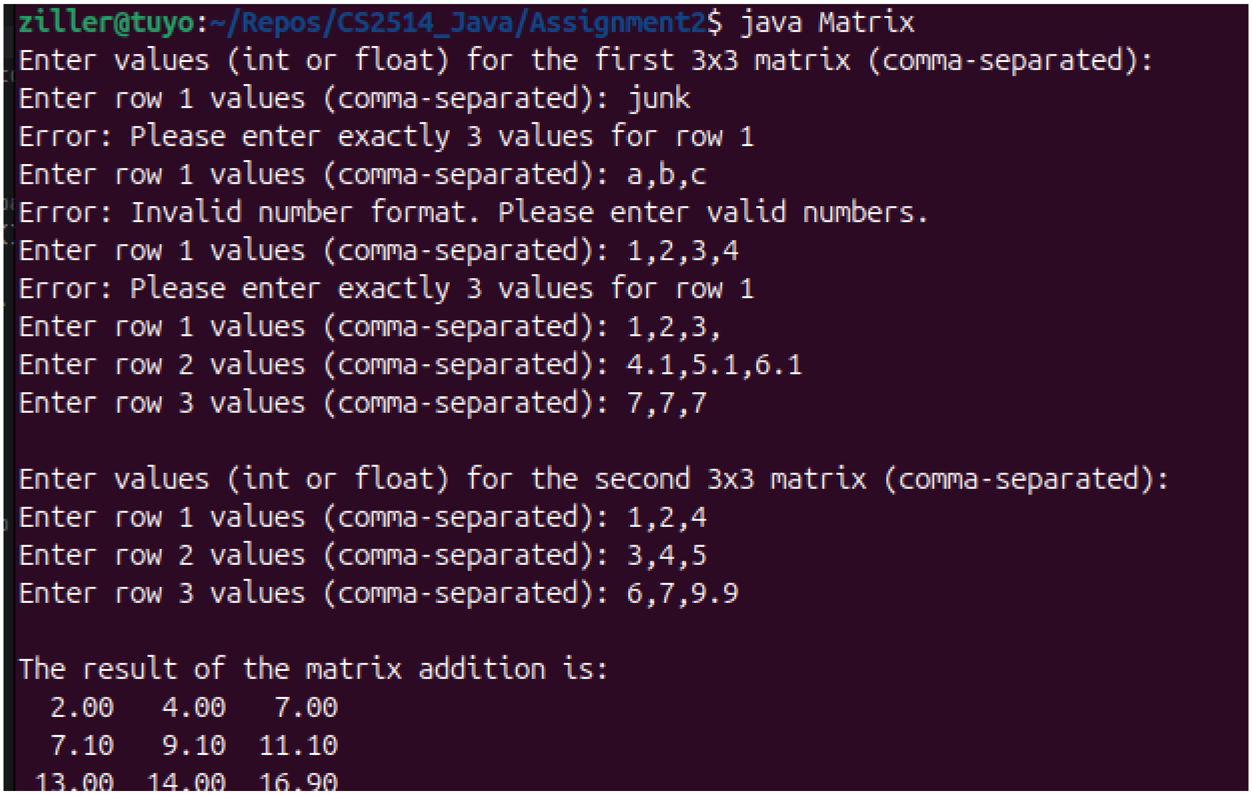
Source Code

```
import java.util.Scanner;

// While this program is simple the challenge is ensuring
// The user inputs correct values.
// Int is obvious, but double is a valid input here as well.
// So we create a method to handle ensureing what comes in is
// valid.

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

        // Get valid base input
        double base = getDoubleInput(scanner, "Enter the base: ");

        // Get valid exponent input (now a double)
        double exponent = getDoubleInput(scanner, "Enter the exponent: ");
        // Here we handle an interesting input permutation.
        // What if the user enters a negative number, but then a negative fraction as
        // the exponent. We essentailly end up with trying to get the root of
        // a negative number - which mathematically not possible.
        // so we try and catch it here....
        if (base < 0 && exponent % 1 != 0) {
            System.out.println("Error: Negative base cannot be raised to a fractional exponent.");
        } else {
            double result = Math.pow(base, exponent);
            System.out.println(base + " raised to the power of " + exponent + " is: " + result);
        }

        // Close scanner
        scanner.close();
    }

    // Method to get a valid double input from the user
    public static double getDoubleInput(Scanner scanner, String prompt) {

        // Keep asking until user enters valid data...
        while (true) {
            System.out.print(prompt);
            if (scanner.hasNextDouble()) {
                return scanner.nextDouble();
            } else {
                System.out.println("Error: Please enter a valid number.");
                scanner.next();
            }
        }
    }
}
```

Program 4

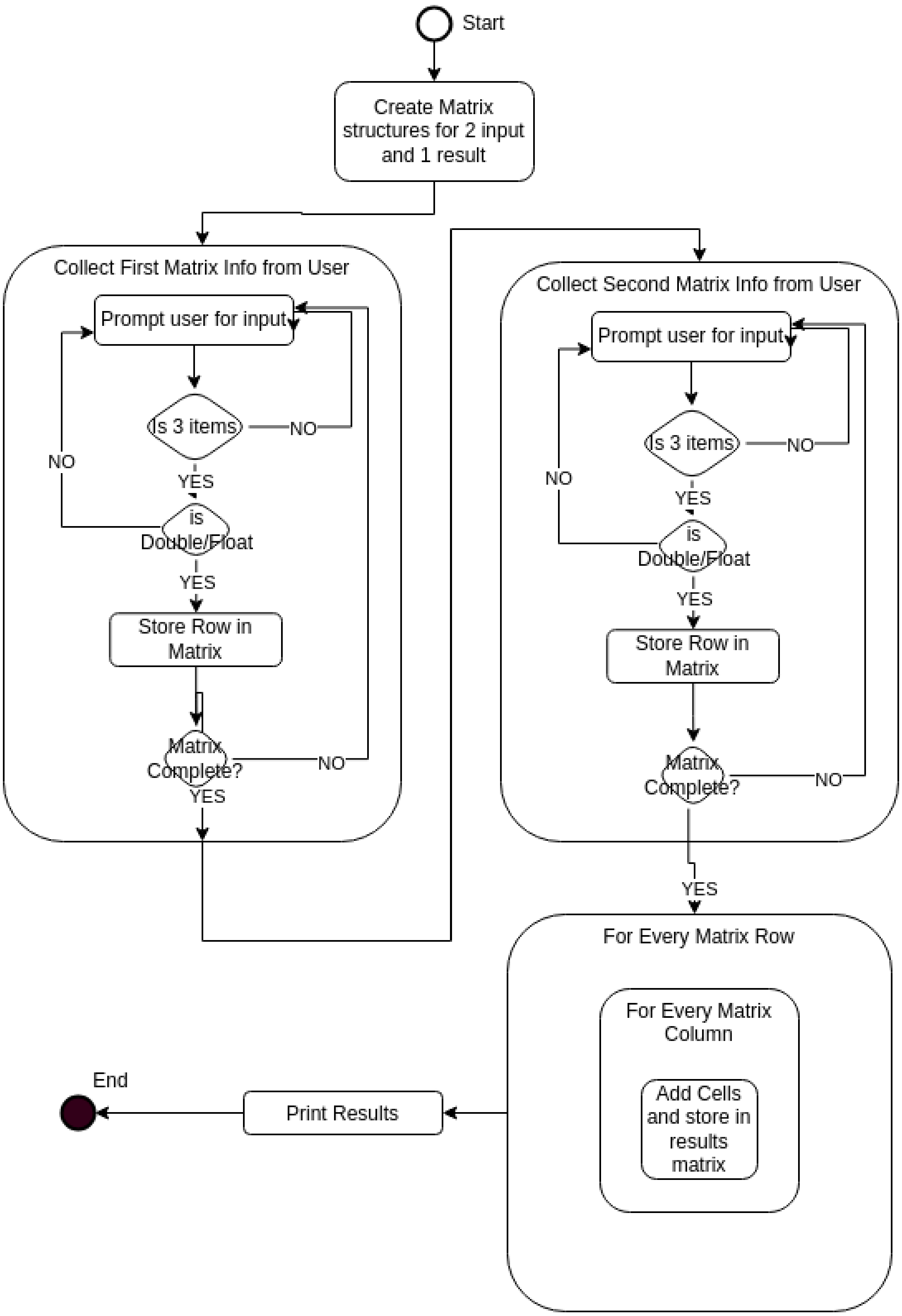
Write a program in JAVA that adds two 3x3 matrices. The inputs of the matrices are given by the user.

Terminal Out



```
ziller@tuyo:~/Repos/CS2514_Java/Assignment2$ java Matrix
Enter values (int or float) for the first 3x3 matrix (comma-separated):
Enter row 1 values (comma-separated): junk
Error: Please enter exactly 3 values for row 1
Enter row 1 values (comma-separated): a,b,c
Error: Invalid number format. Please enter valid numbers.
Enter row 1 values (comma-separated): 1,2,3,4
Error: Please enter exactly 3 values for row 1
Enter row 1 values (comma-separated): 1,2,3,
Enter row 2 values (comma-separated): 4.1,5.1,6.1
Enter row 3 values (comma-separated): 7,7,7

Enter values (int or float) for the second 3x3 matrix (comma-separated):
Enter row 1 values (comma-separated): 1,2,4
Enter row 2 values (comma-separated): 3,4,5
Enter row 3 values (comma-separated): 6,7,9.9

The result of the matrix addition is:
  2.00   4.00   7.00
  7.10   9.10  11.10
 13.00  14.00  16.90
```

Flowchart



Source Code

```
import java.util.*;

// Simple class for matrix addition of a fixed 3x3 matrix
```

```
public class Matrix {
    public static void main(String[] args) {
        // Create scanner object.
        Scanner scanner = new Scanner(System.in);

        // Create both matrix objects as double for better precision
        double[][] matrix1 = new double[3][3];
        double[][] matrix2 = new double[3][3];
        // Create a matrix for storing the result
        double[][] resultMatrix = new double[3][3];

        // Input values for the first matrix
        System.out.println("Enter values (int or float) for the first 3x3 matrix (comma-separated):");
        inputMatrix(scanner, matrix1);

        // Input values for the second matrix
        System.out.println("\nEnter values (int or float) for the second 3x3 matrix (comma-separated):
        inputMatrix(scanner, matrix2);

        // Perform matrix addition: Add corresponding elements
        // Addition is a simple double loop O(N^2)
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                resultMatrix[i][j] = matrix1[i][j] + matrix2[i][j]; // Correctly adding corresponding
            }
        }

        // Display the result of the matrix addition
        System.out.println("\nThe result of the matrix addition is:");
        printMatrix(resultMatrix);

        // Close the scanner
        scanner.close();
    }

    // Method to input a matrix
    // To make input easier, we take each row as comma-separated
    public static void inputMatrix(Scanner scanner, double[][] matrix) {
        for (int i = 0; i < 3; i++) {
            boolean validRow = false;
            while (!validRow) {
                System.out.print("Enter row " + (i + 1) + " values (comma-separated): ");
                String input = scanner.nextLine();

                // Split the input string into an array of strings, using commas as separators
                String[] values = input.split(",");

                // Checking if the user entered exactly 3 comma-separated numbers.
                // If not ask them to go again.
                if (values.length != 3) {
                    System.out.println("Error: Please enter exactly 3 values for row " + (i + 1));
                    continue;  // Ask for the row again if the input is not exactly 3 values
                }

                // Ok, we should have valid input at this point, so we..
                // Convert the string values to doubles and assign them to the matrix
                boolean validValues = true;
                for (int j = 0; j < 3; j++) {
                    try {
                        // Trim spaces to avoid errors with unexpected spaces in input
                        matrix[i][j] = Double.parseDouble(values[j].trim());
```

```java
                    } catch (NumberFormatException e) {
                        System.out.println("Error: Invalid number format. Please enter valid numbers."
                        validValues = false;
                        break;
                    }
                }

                if (validValues) {
                    validRow = true;  // If all values are valid, move to the next row
                }
            }
        }
    }

    // Method to print a matrix in a neat format
    public static void printMatrix(double[][] matrix) {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                // Print each element with two decimal places for better readability
                // This keeps our matrix print nice and neat.
                System.out.printf("%6.2f ", matrix[i][j]);
            }
            System.out.println(); // New line after each row
        }
    }
}
```