Experiment-3

Student Name: Tamanna **UID:** 22BCS17239
Branch: B.E.-C.S.E. **Section/Group:** 22BCS_IOT-635(A)
Semester: 6th **Date of Performance**
Subject Name: Project Based Learning in Java with Lab **Subject Code:** 22CSH-359

1. Aim:

a) Write a Java program to calculate the square root of a number entered by the user. Use try-catch to handle invalid inputs (e.g., negative numbers or non-numeric values).

2. Objective:

Square Root Calculator – Calculate the square root of a user-entered number with error handling for invalid inputs**.**

3. Procedure:

Square Root Calculator

1. Take user input.
2. Check if the number is negative; throw an exception if true.
3. Compute and display the square root.
4. Handle invalid inputs using try-catch.

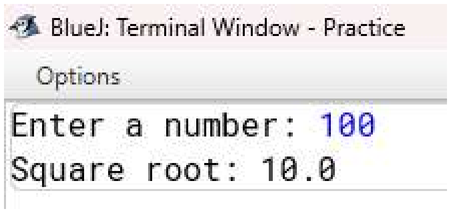
4. Implementation/Code:

a) Square Root Calculator

```
import java.util.Scanner;
public class SquareRootCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Enter a number: ");
            double number = scanner.nextDouble();
            if (number < 0) {
                throw new IllegalArgumentException("Cannot calculate the square root of a negative number.");
            }
            System.out.println("Square root: " + Math.sqrt(number));
        } catch (IllegalArgumentException e) {
            System.out.println("Error: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: Invalid input. Please enter a numeric value.");
        } finally {
            scanner.close();
        }
    }
}
```

5. Outputs:

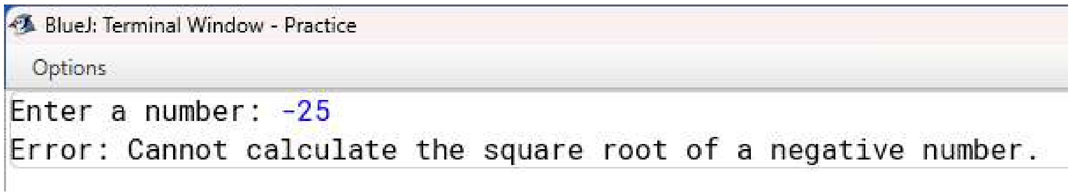
a) Output for Square Root Calculator



BlueJ: Terminal Window - Practice

Options

```
Enter a number: 100
Square root: 10.0
```



BlueJ: Terminal Window - Practice

Options

```
Enter a number: -25
Error: Cannot calculate the square root of a negative number.
```

6. Learning Outcomes:

- **Mastering Exception Handling:** Gain proficiency in using `try-catch` blocks to prevent runtime errors and ensure smooth program flow.
- **User Input Validation and Control Flow:** Develop the ability to take user inputs and validate them correctly while ensuring proper application behavior.
- **Designing Real-World Systems:** Understand how to simulate real-life scenarios like banking transactions and university enrollment with practical, manageable code.