Experiment-3

Student Name: Prateek Pratap Singh **UID: 22BCS10036**
Branch: BE-CSE **Section/Group: IOT_631-A**
Semester:6th **Date of Performance: 31/01/2025**
Subject Name: Project Based Learning in Java with Lab **Subject Code: 22CSH-359**

1. **(a) Aim:** 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:**
   - To create a Java program that calculates the square root of a user-input number.
   - To handle exceptions using try-catch, ensuring the program does not crash for invalid inputs like negative numbers or non-numeric values.
   - To display appropriate error messages for incorrect inputs.

3. **Implementation:**

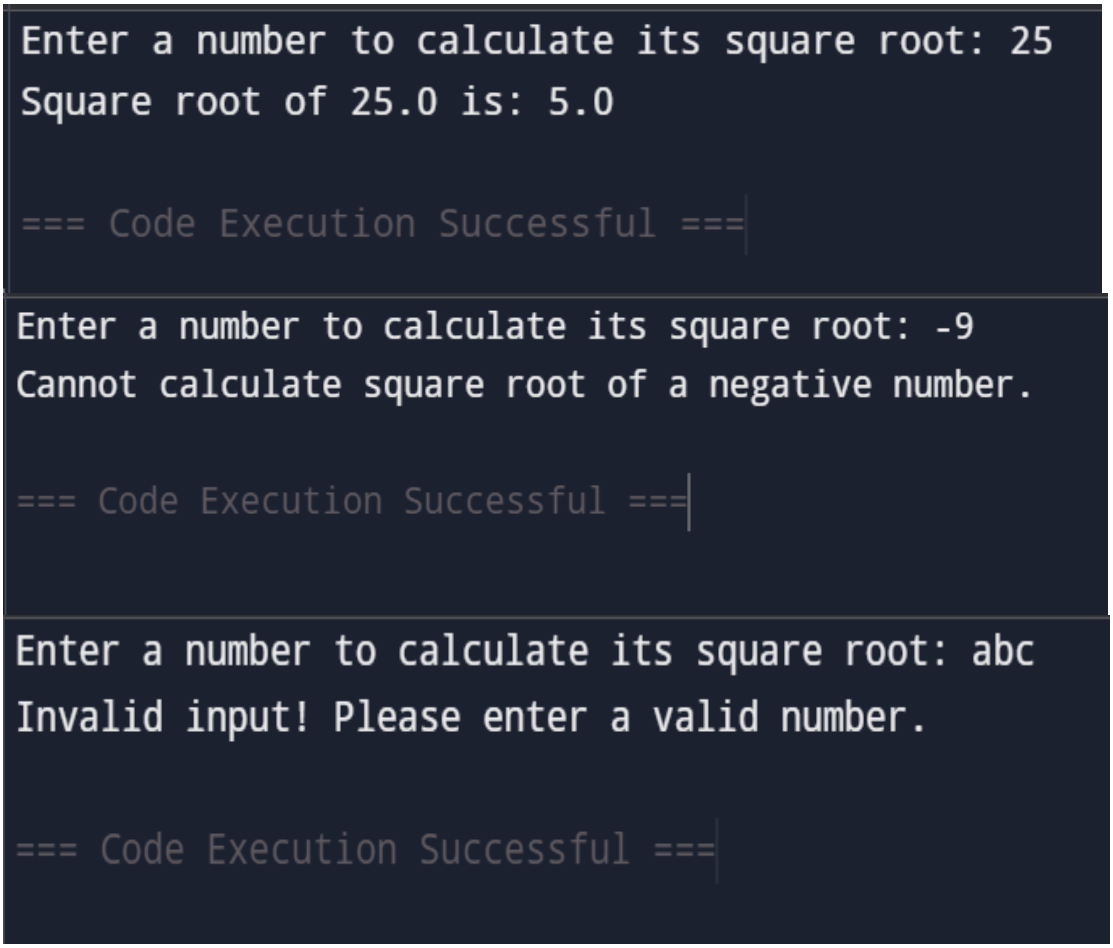
```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number to calculate its square root: ");

        try {
            double number = Double.parseDouble(scanner.nextLine());
            if (number < 0) {
                throw new IllegalArgumentException("Cannot calculate square root
of a negative number.");
```

```
            }
            double squareRoot = Math.sqrt(number);
            System.out.println("Square root of " + number + " is: " + squareRoot);
        } catch (NumberFormatException e) {
            System.out.println("Invalid input! Please enter a valid number.");
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

4. **Output:**



5. Learning Outcomes:

- Understanding how to take user input in Java using Scanner.
- Learning to parse user input from String to double using Double.parseDouble().
- Implementing exception handling using try-catch to manage invalid inputs.
- Handling **negative numbers** with custom exceptions.
- Using the Math.sqrt() method to compute square roots.

1. **(b) Aim:** Write a Java program to simulate an ATM withdrawal system. The program should:
   - Ask the user to enter their PIN.
   - Allow withdrawal if the PIN is correct and the balance is sufficient.
   - Throw exceptions for invalid PIN or insufficient balance.
   - Ensure the system always shows the remaining balance, even if an exception occurs.

2. **Objective :**
   - To simulate a simple ATM system that allows users to withdraw money after entering a correct PIN.
   - To check if the entered PIN matches the stored PIN and deny access if incorrect.
   - To ensure withdrawal happens only if the balance is sufficient.
   - To use try-catch for handling incorrect PINs, insufficient balance, and non-numeric inputs.
   - To always display the remaining balance, even in case of an error.

3. **Implementation :**

```
import java.util.Scanner;

public class Main {
    private static int pin = 1234;
    private static double balance = 1000.0;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your PIN: ");

        try {
            int enteredPIN = Integer.parseInt(scanner.nextLine());
            if (enteredPIN != pin) {
                throw new IllegalArgumentException("Invalid PIN entered.");
            }
```

```
        System.out.print("Enter amount to withdraw: ");
        double amount = Double.parseDouble(scanner.nextLine());

        if (amount > balance) {
            throw new IllegalArgumentException("Insufficient balance.");
        }

        balance -= amount;
        System.out.println("Withdrawal successful. Remaining balance: " +
balance);

    } catch (NumberFormatException e) {
        System.out.println("Invalid input! Please enter a valid number.");
    } catch (IllegalArgumentException e) {
        System.out.println(e.getMessage());
    } finally {
        System.out.println("Remaining balance: " + balance);
    }
}
}
```

4. **Output :**

```
Enter your PIN: 1234
Enter amount to withdraw: 500
Withdrawal successful. Remaining balance: 500.0
Remaining balance: 500.0

=== Code Execution Successful ===
```

```
Enter your PIN: 1122
Invalid PIN entered.
Remaining balance: 1000.0

=== Code Execution Successful ===
```

```
Enter your PIN: 1234
Enter amount to withdraw: 1500
Insufficient balance.
Remaining balance: 1000.0

=== Code Execution Successful ===
```

```
Enter your PIN: 1234
Enter amount to withdraw: xyz
Invalid input! Please enter a valid number.
Remaining balance: 1000.0

=== Code Execution Successful ===
```

5. **Learning Outcomes:**
   - Understanding user authentication using a PIN-based system.
   - Implementing conditional statements to verify PIN correctness and balance availability.
   - Handling incorrect PIN entries and insufficient balance using exceptions.
   - Parsing numeric input using Integer.parseInt() and Double.parseDouble().
   - Using a finally block to ensure the remaining balance is always displayed.
   - Learning basic banking logic implementation in Java.