DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Experiment 3

Student Name: Annu **UID:22BCS10069**
Branch: BE-CSE **Section/Group: IOT-636/A**
Semester:6th **Date of Performance:27-01-25**
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).

b.) 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 occur

2. **Implementation/Code:**

a.)
```
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("
Square root of a negative
number is not valid.");
      }

      double squareRoot =
Math.sqrt(number);

System.out.println("Square
root: " + squareRoot);
  } catch
```

```
      (IllegalArgumentException
      e) {

      System.out.println("Error: "
      + e.getMessage());
        } catch (Exception e) {

      System.out.println("Invalid
      input! Please enter a valid
      number.");
        } finally {
            scanner.close();
        }
    }
}
```

3.Output

```
c exp3.java } ; if ($?) { java exp3 }
Enter a number: -25
Error: Square root of a negative number is not valid.
PS C:\Users\DELL\Downloads\java exp codes>
```

```
PS C:\Users\DELL\Downloads\java exp codes> cd "c:\Users\DELL\Downloads\java exp codes\" ; if ($?) { java
c exp3.java } ; if ($?) { java exp3 }
Enter a number: 25
Square root: 5.0
PS C:\Users\DELL\Downloads\java exp codes>
```

4.Code :b)

```
import java.util.Scanner;

public class ATM {
    private static final int
      CORRECT_PIN = 1234;  //
      Predefined PIN
    private static double balance
      = 5000.0;       // Initial
      balance

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

        try {
```

```
            System.out.print("Enter
        your PIN: ");
            int enteredPin =
        scanner.nextInt();

            if (enteredPin !=
        CORRECT_PIN) {
                throw new
        SecurityException("Invalid
        PIN! Access denied.");
            }

            System.out.print("Enter
        withdrawal amount: ");
            double amount =
        scanner.nextDouble();

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

            balance -= amount;
            System.out.println("Wit
        hdrawal successful!
        Amount withdrawn: " +
        amount);
        } catch (SecurityException
        e) {
            System.out.println("Erro
        r: " + e.getMessage());
        } catch
        (IllegalArgumentException
        e) {
            System.out.println("Erro
        r: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Inva
        lid input! Please enter a
        valid number.");
        } finally {
            System.out.println("Re
        maining balance: " +
        balance);
            scanner.close();
        }
    }
}
```

3b)Output

```
PS C:\Users\DELL\Downloads\java exp codes> cd "c:\Users\DELL\Downloads\java exp codes\" ; if ($?)
c ATM.java } ; if ($?) { java ATM }
Enter your PIN: 1234
Enter withdrawal amount: 250
Withdrawal successful! Amount withdrawn: 250.0
Remaining balance: 4750.0
```

```
PS C:\Users\DELL\Downloads\java exp codes> cd "c:\Users\DELL\Downloads\java exp codes\" ; if ($?) { j
c ATM.java } ; if ($?) { java ATM }
Enter your PIN: 4587
Error: Invalid PIN! Access denied.
Remaining balance: 5000.0
```

Learning Outcomes:

1. **Exception Handling** – Learn to use try-catch to handle invalid inputs, incorrect PINs, and insufficient balance errors.
2. **User Input & Validation** – Understand how to take user input, check for correct data types, and prevent negative or invalid entries.
3. **Mathematical & Financial Operations** – Perform square root calculations and simulate ATM withdrawals with balance checks.
4. **Security & Authentication** – Implement basic security by validating user PINs and handling authentication failures.
5. **Resource Management** – Use the finally block to ensure important actions, like displaying the remaining balance, always execute