Experiment 3

Student Name: Sarthak Arora	**UID** 23BCS12984
Branch: CSE	**Section/Group:** KRG_2B
Semester: 5th	**Date of Performance:** 10/9/25
Subject Name: PBLJ	**Subject Code:** 23CSH-304

1. Aim:

Develop Java programs with exception handling for user input validation, ATM systems, and university enrollment management.

A) Easy Level:

❖ 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) Medium Level:

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.

C) Hard Level:

Create a Java program for a university enrollment system with exception handling. The program should:

- Allow students to enroll in courses.
- Throw a CourseFullException if the maximum enrollment limit is reached.
- Throw a PrerequisiteNotMetException if the student hasn't completed prerequisite courses.

2. Objectives:

❖ To calculate the square root of a number and handle invalid inputs using exceptions.
❖ To simulate an ATM system with PIN validation and withdrawal using exception handling.

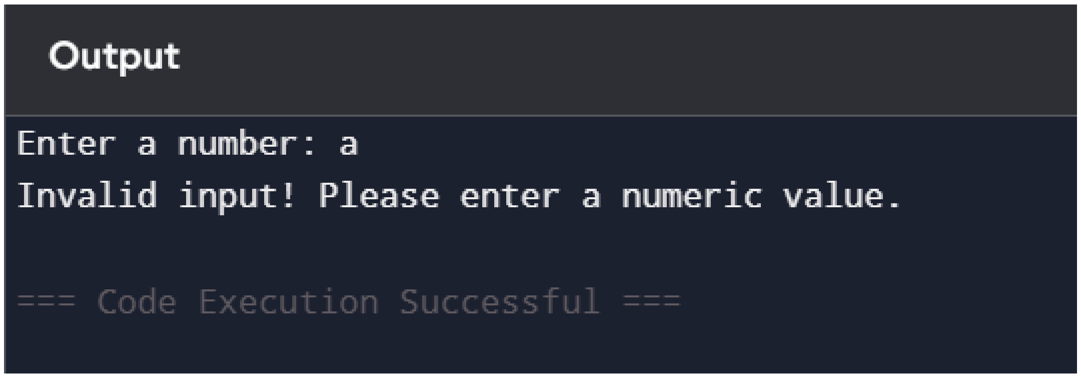
❖ To manage course enrollment and demonstrate custom exceptions for full courses and unmet prerequisites.

3. JAVA script and output:

EASY-LEVEL PROBLEM

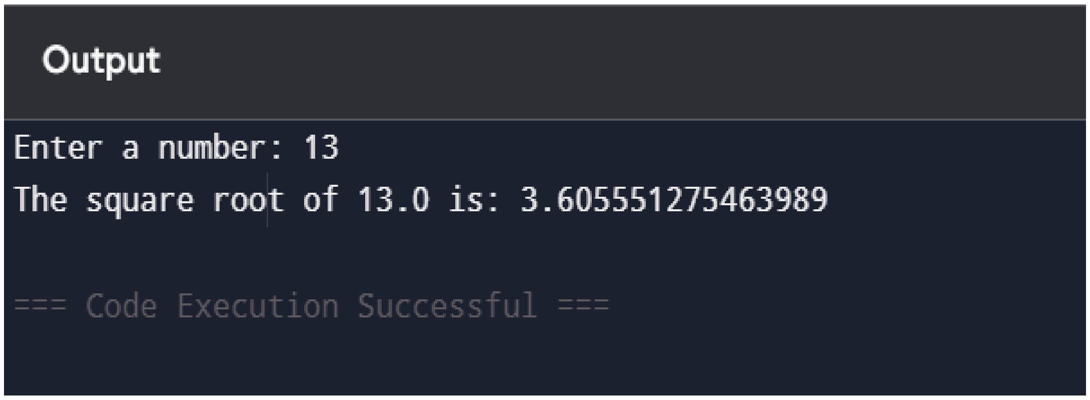
```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        try {
            System.out.print("Enter a number: ");
            String input = scan.nextLine();
            double n = Double.parseDouble(input);

            if (n<0) {
                throw new IllegalArgumentException("Square root of a negative number is not
real.");
            }

            double res= Math.sqrt(n);
            System.out.println("The square root of "+n+" is: "+res);

        } catch (NumberFormatException e) {
            System.out.println("Invalid input! Please enter a numeric value.");
        } catch (IllegalArgumentException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            scan.close();
        }
    }
}
```

Output:



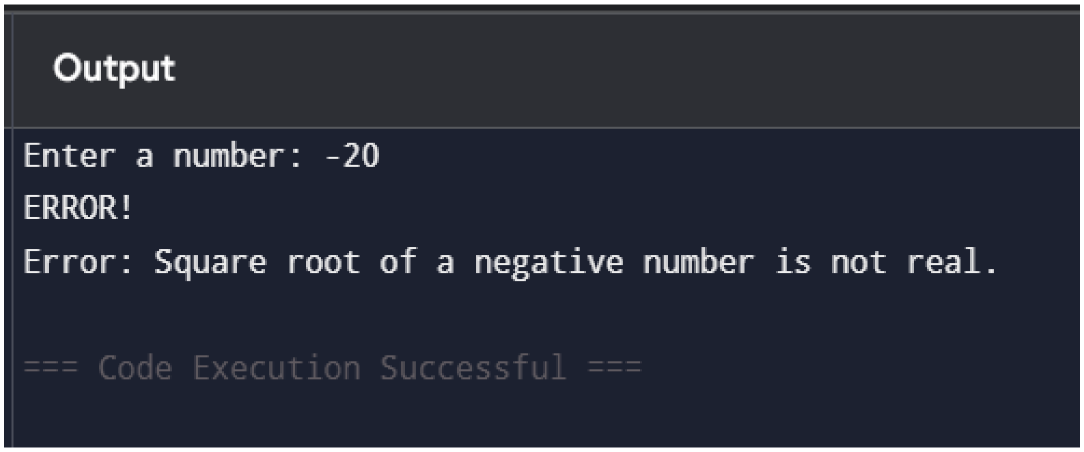
Output

```
Enter a number: a
Invalid input! Please enter a numeric value.

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



Output

```
Enter a number: 13
The square root of 13.0 is: 3.605551275463989

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



Output

```
Enter a number: -20
ERROR!
Error: Square root of a negative number is not real.

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

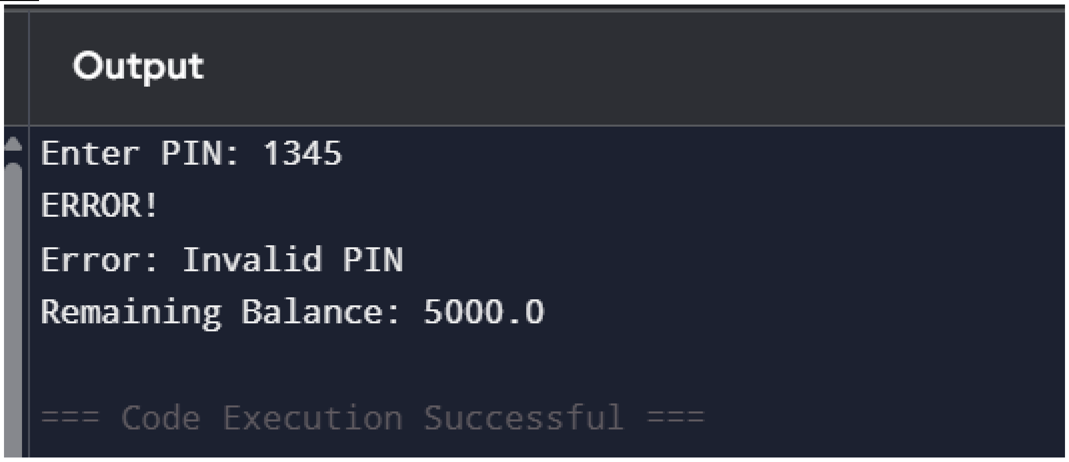
MEDIUM LEVEL PROBLEM:

import java.util.Scanner;

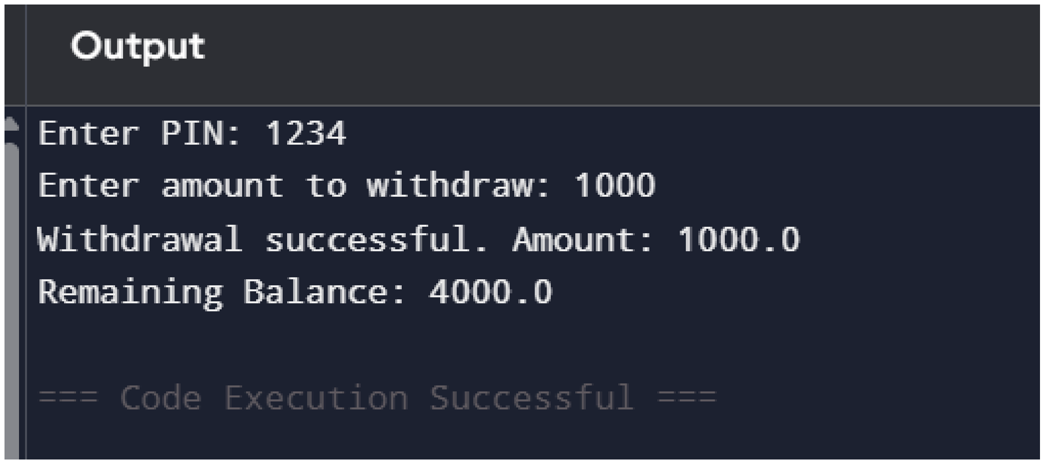
class InvalidPinException extends Exception {

```
    public InvalidPinException(String msg) {
        super(msg);
    }
}
class InsufficientBalanceException extends Exception {
    public InsufficientBalanceException(String msg) {
        super(msg);
    }
}
public class ATM {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int pin = 1234;
        double balance = 5000;
        try {
            System.out.print("Enter PIN: ");
            int enteredPin = sc.nextInt();
            if (enteredPin != pin) {
                throw new InvalidPinException("Invalid PIN");
            }
            System.out.print("Enter amount to withdraw: ");
            double amount = sc.nextDouble();
            if (amount > balance) {
                throw new InsufficientBalanceException("Insufficient Balance");
            }
            balance -= amount;
            System.out.println("Withdrawal successful. Amount: " + amount);
        } catch (InvalidPinException | InsufficientBalanceException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            System.out.println("Remaining Balance: " + balance);
            sc.close();
        }
    }
}
```

Output:



```
Output
Enter PIN: 1345
ERROR!
Error: Invalid PIN
Remaining Balance: 5000.0

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

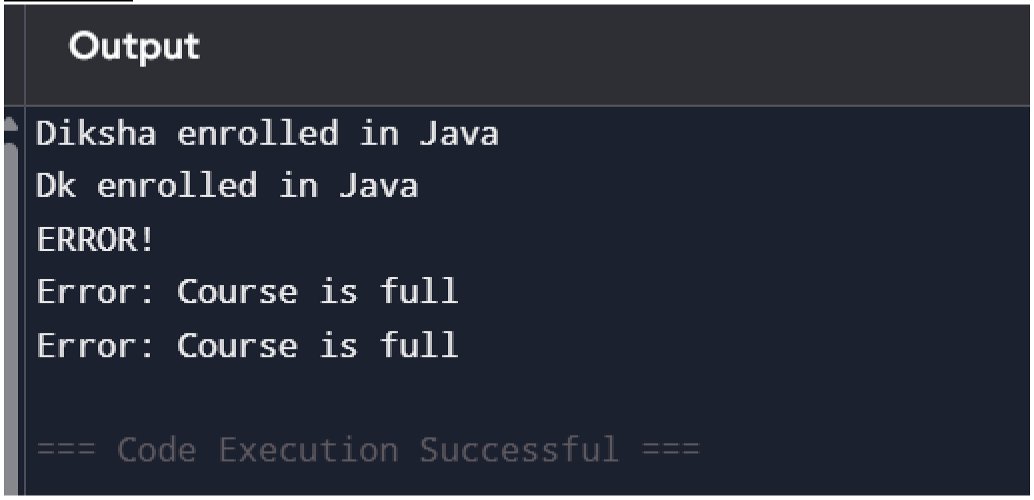


```
Output
Enter PIN: 1234
Enter amount to withdraw: 1000
Withdrawal successful. Amount: 1000.0
Remaining Balance: 4000.0

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

HARD LEVEL PROBLEM

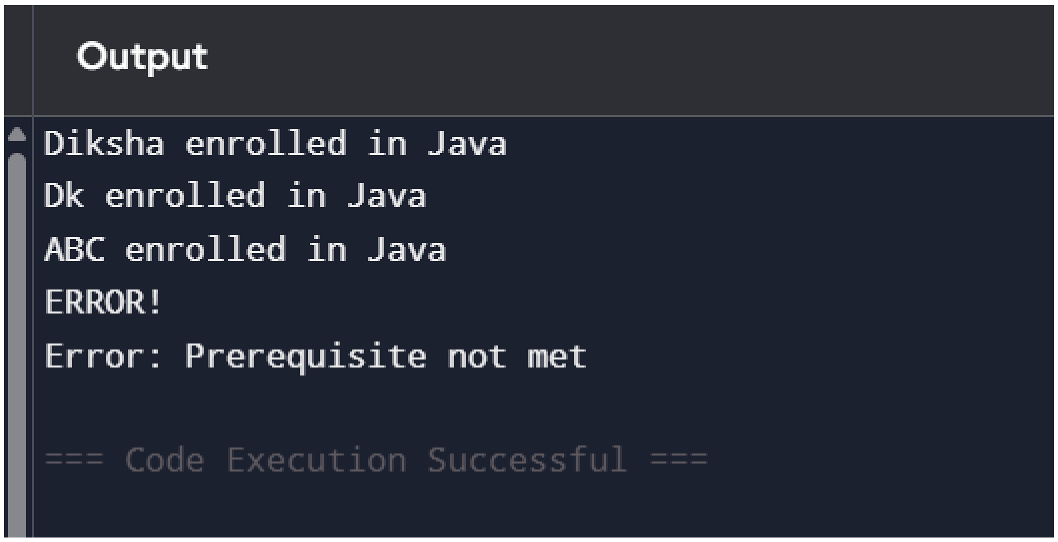
```
class CourseFullException extends Exception {
    public CourseFullException(String msg) {
        super(msg);
    }
}

class PrerequisiteNotMetException extends Exception {
    public PrerequisiteNotMetException(String msg) {
        super(msg);
    }
}
class Course {
    String name;
    int max;
    int count = 0;
    String prereq;

    Course(String name, int max, String prereq) {
        this.name = name;
        this.max = max;
        this.prereq = prereq;
    }

    void enroll(String student, boolean hasPrereq) throws CourseFullException,
PrerequisiteNotMetException {
        if (count >= max) throw new CourseFullException("Course is full");
        if (!hasPrereq) throw new PrerequisiteNotMetException("Prerequisite not met");
        count++;
        System.out.println(student + " enrolled in "+name);
    }
}
public class University {
    public static void main(String[] args) {
        Course java=new Course("Java", 2, "OOP");//do “Java”,4,”OOP” for 2nd output pasted below

        try {
            java.enroll("Diksha", true);
            java.enroll("Dk", true);
            java.enroll("ABC", true);
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }

        try {
            java.enroll("XYZ", false);
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:



Output

```
Diksha enrolled in Java
Dk enrolled in Java
ERROR!
Error: Course is full
Error: Course is full

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



Output

```
Diksha enrolled in Java
Dk enrolled in Java
ABC enrolled in Java
ERROR!
Error: Prerequisite not met

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