Experiment 3

Student Name: Aniket	UID: 22BCS11646
Branch: BE-CSE	Section/Group: DL_903_A
Semester: 6th	Date of Performance: 11-02-2025
Subject Name: Program Based Learning in Java with Lab	Subject Code: 22CSH-359

1. Aim: Solving problems under the category of Exception handling in Easy, Medium and Hard

2. Objective: Introduction to Exceptions. Difference between error and exception. Use of try, catch and throw. Difference between throw and throws. Types of Exceptions, Exception handling in Java.

3.Implementation/Code:

1.) Easy: 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).

Code:

```
import java.util.*;

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

        try {
            double number = scanner.nextDouble();
            if (number < 0) {
                System.out.println("Error: Cannot calculate the square root of a negative number.");
            } else {
                double result = Math.sqrt(number);
                System.out.println("Square root: " + result);
            }
        } catch (InputMismatchException e) {
            System.out.println("Error: Invalid input. Please enter a numeric value.");
        } finally {
            scanner.close();
        }
    }
}
```

2.) Medium: 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.

Code:

```
import java.util.*;

class InvalidPinException extends Exception {
    public InvalidPinException(String message) {
        super(message);
    }
}

class InsufficientBalanceException extends Exception {
    public InsufficientBalanceException(String message) {
        super(message);
    }
}

public class ATMSystem {
    private static final int CORRECT_PIN = 1234;
    private static double balance = 5000.0;

    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 InvalidPinException("Error: Invalid PIN.");
            }

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

            if (amount > balance) {
                throw new InsufficientBalanceException("Error: Insufficient balance.");
            }

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

        } catch (InvalidPinException | InsufficientBalanceException e) {
            System.out.println(e.getMessage());
        } catch (InputMismatchException e) {
            System.out.println("Error: Invalid input. Please enter numeric values.");
        } finally {
            System.out.println("Current balance: " + balance);
            scanner.close();
        }
    }
}
```

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

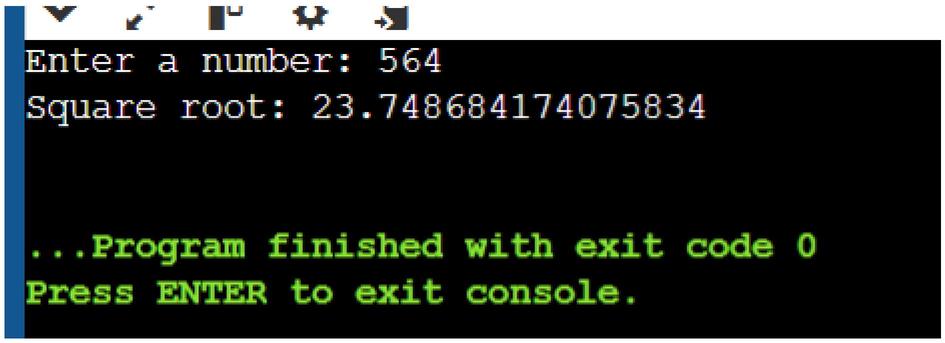
- Allow students to enroll in courses.
- Throw a CourseFullException if the maximum enrollment limit is reached.
- Throw a PrerequisiteNotMetException if the student hasn’t completed prerequisite courses.

Code:

```
import java.util.*;

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

class PrerequisiteNotMetException extends Exception {
    public PrerequisiteNotMetException(String message) {
        super(message);
    }
}

class Course {
    private final String courseName;
    private final int maxEnrollment;
    private final String prerequisite;
    private int enrolledStudents;

    public Course(String courseName, int maxEnrollment, String prerequisite) {
        this.courseName = courseName;
        this.maxEnrollment = maxEnrollment;
        this.prerequisite = prerequisite;
        this.enrolledStudents = 0;
    }

    public void enrollStudent(String studentName, Set<String> completedCourses) throws CourseFullException,
PrerequisiteNotMetException {
        if (enrolledStudents >= maxEnrollment) {
            throw new CourseFullException("Error: Course " + courseName + " is full.");
        }
        if (!prerequisite.isEmpty() && !completedCourses.contains(prerequisite)) {
            throw new PrerequisiteNotMetException("Error: Prerequisite " + prerequisite + " not met for course " + courseName +
".");
        }
        enrolledStudents++;
        System.out.println("Student " + studentName + " enrolled in " + courseName + " successfully.");
    }

    public String getCourseName() {
        return courseName;
    }
}

public class UniversityEnrollmentSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Set<String> completedCourses = new HashSet<>();
        Course course = new Course("Advanced Java", 2, "Intro to Java");

        try {
            System.out.print("Enter student name: ");
            String studentName = scanner.nextLine();
            System.out.print("Enter completed courses (comma separated): ");
            String[] courses = scanner.nextLine().split(",");
            for (String c : courses) {
                completedCourses.add(c.trim());
            }

            course.enrollStudent(studentName, completedCourses);
        } catch (CourseFullException | PrerequisiteNotMetException e) {
            System.out.println(e.getMessage());
        } finally {
            System.out.println("Enrollment process completed.");
            scanner.close();
        }
    }
}
```

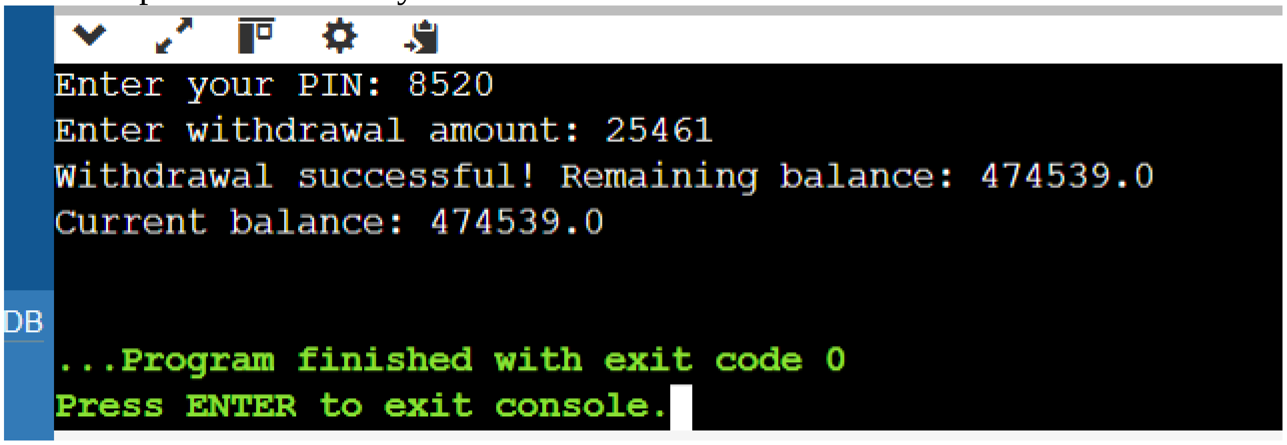
5. Output

1.) Easy problem: Square root calculator



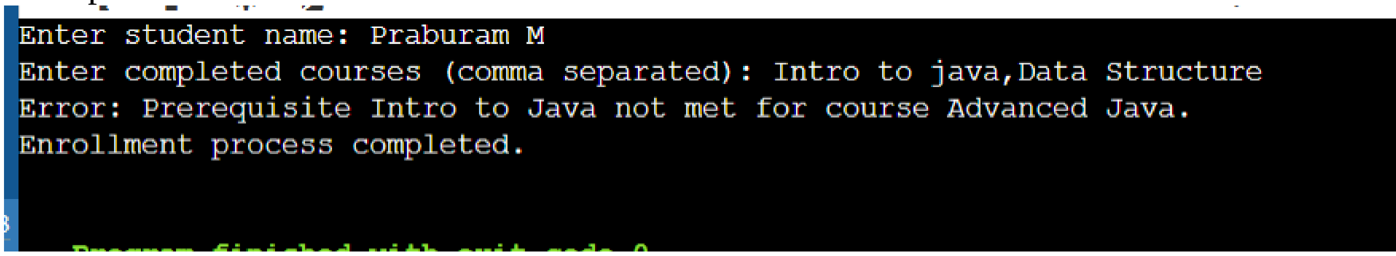
```
Enter a number: 564
Square root: 23.748684174075834


...Program finished with exit code 0
Press ENTER to exit console.
```

2.) Medium problem: ATM System



```
Enter your PIN: 8520
Enter withdrawal amount: 25461
Withdrawal successful! Remaining balance: 474539.0
Current balance: 474539.0


...Program finished with exit code 0
Press ENTER to exit console.
```

3.) Hard problem:



```
Enter student name: Praburam M
Enter completed courses (comma separated): Intro to java,Data Structure
Error: Prerequisite Intro to Java not met for course Advanced Java.
Enrollment process completed.


...Program finished with exit code 0
```

6. Learning Outcomes

1. Learned to use classes and objects for organizing employee and designation data in Java.
2. Implemented salary calculations using switch-case and array data handling.
3. Practiced input handling with the Scanner class and validating user input.
4. Gained experience in searching arrays and structuring conditional logic.
5. Displayed formatted output for real-world applications like employee management systems.