NAME : Agrima Sharma	**UID :** 22BCS15314	**SECTION :** 607-A

//Problem 1: Square Root Calculation (Easy Level)

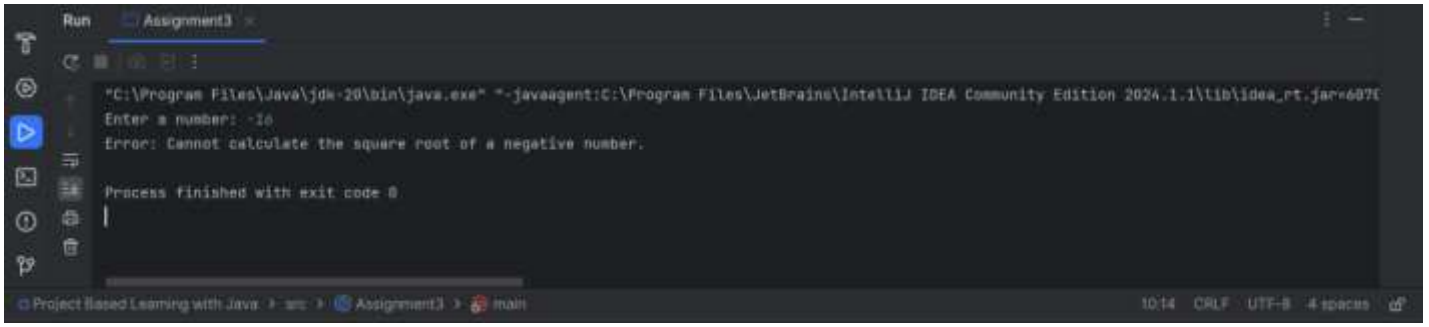
```
import java.util.Scanner;

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

        try {
            double number = Double.parseDouble(scanner.nextLine());

            if (number < 0) {
                throw new IllegalArgumentException("Error: Cannot calculate the square
root of a negative number.");
            }

            double result = Math.sqrt(number);
            System.out.println("Square Root: " + result);
        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid input. Please enter a numeric value.");
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```



Problem 2: ATM Withdrawal System (Medium Level)

```
import java.util.Scanner;

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

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

class ATM {
    private final int correctPin = 1234;
    private double balance = 3000;

    public void withdraw(int pin, double amount) throws InvalidPINException,
InsufficientBalanceException {
        if (pin != correctPin) {
            throw new InvalidPINException("Error: Invalid PIN.");
        }
        if (amount > balance) {
            throw new InsufficientBalanceException("Error: Insufficient balance.");
        }
        balance -= amount;
        System.out.println("Withdrawal successful! Current Balance: " + balance);
    }

    public void displayBalance() {
        System.out.println("Current Balance: " + balance);
    }
}

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

        try {
            System.out.print("Enter PIN: ");
```

```
            int pin = scanner.nextInt();

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

            atm.withdraw(pin, amount);
        } catch (InvalidPINException | InsufficientBalanceException e) {
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: Invalid input.");
        } finally {
            atm.displayBalance();
            scanner.close();
        }
    }
}
```

//Problem 3: University Enrollment System (Hard Level)

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

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

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

```
}

class University {
    private final Map<String, Integer> courses = new HashMap<>();
    private final Map<String, String> prerequisites = new HashMap<>();
    private final Map<String, Boolean> completedCourses = new HashMap<>();

    public University() {
        courses.put("Advanced Java", 2);
        courses.put("Data Structures", 3);
        courses.put("Algorithms", 2);

        prerequisites.put("Advanced Java", "Core Java");
    }

    public void completeCourse(String course) {
        completedCourses.put(course, true);
    }

    public void enroll(String course) throws CourseFullException,
PrerequisiteNotMetException {
        if (!courses.containsKey(course)) {
            throw new IllegalArgumentException("Error: Course not found.");
        }

        if (prerequisites.containsKey(course)) {
            String prerequisite = prerequisites.get(course);
            if (!completedCourses.getOrDefault(prerequisite, false)) {
                throw new PrerequisiteNotMetException("Error: Complete " + prerequisite
+ " before enrolling in " + course + ".");
            }
        }

        if (courses.get(course) == 0) {
            throw new CourseFullException("Error: " + course + " is full.");
        }

        courses.put(course, courses.get(course) - 1);
        System.out.println("Enrollment successful in " + course + "!");
    }
}

public class Assignment3 {
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
        University university = new University();

        try {
            System.out.print("Enroll in Course: ");
            String course = scanner.nextLine();

            university.enroll(course);
        } catch (CourseFullException | PrerequisiteNotMetException e) {
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.out.println("Error: Invalid input.");
        } finally {
            scanner.close();
        }
    }
}
```