**ASSIGNMENT NO.8**

Name: NEIL CARDOZ Roll no: 2307012079

Batch: AIML B1

Title: Student-Exception-

1. Main.java

//Main.java

/\*

 \* Name: Neil Cardoz

 \* PRN: 23070126079

 \* Batch: 23-27

 \*

 \*/

import java.util.\*;

// import StudentExceptions.EmptyStudentListException; // Ensure this package and class exist, or remove this line if unnecessary

public class Main {

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        StudentOperations operations = new StudentOperations();

        int choice;

        // Main menu loop

        do {

            // Displaying menu options

            System.out.println("\n===== Student Management System =====");

            System.out.println("1. Add Student");

            System.out.println("2. Display All Students");

            System.out.println("3. Search Student by PRN");

            System.out.println("4. Search Student by Name");

            System.out.println("5. Search Student by Position");

            System.out.println("6. Update Student");

            System.out.println("7. Delete Student");

            System.out.println("8. Exit");

            System.out.print("Enter your choice: ");

            choice = sc.nextInt();

            sc.nextLine();

            try {

                switch (choice) {

                    case 1:

                        handleAddStudent(sc, operations);

                        break;

                    case 2:

                        handleDisplayStudents(operations);

                        break;

                    case 3:

                        handleSearchByPrn(sc, operations);

                        break;

                    case 4:

                        handleSearchByName(sc, operations);

                        break;

                    case 5:

                        handleSearchByPosition(sc, operations);

                        break;

                    case 6:

                        handleUpdateStudent(sc, operations);

                        break;

                    case 7:

                        handleDeleteStudent(sc, operations);

                        break;

                    case 8:

                        System.out.println("Exiting the program. Goodbye!");

                        break;

                    default:

                        System.out.println("Invalid choice. Please try again.");

                }

            } catch (Exception e) {

                System.out.println("Error: " + e.getMessage());

            }

        } while (choice != 8);

        sc.close();

    }

    private static void handleAddStudent(Scanner sc, StudentOperations operations) throws Exception {

        System.out.print("Enter PRN: ");

        String prn = sc.nextLine();

        System.out.print("Enter Name: ");

        String name = sc.nextLine();

        System.out.print("Enter Date of Birth (DD-MM-YYYY): ");

        String dob = sc.nextLine();

        System.out.print("Enter Marks: ");

        double marks = sc.nextDouble();

        Student newStudent = new Student(prn, name, dob, marks);

        operations.addStudent(newStudent);

    }

    private static void handleDisplayStudents(StudentOperations operations) {

        try {

            operations.displayStudents();

        } catch (StudentExceptions.EmptyStudentListException e) {

            System.out.println("Error: " + e.getMessage());

        }

    }

    private static void handleSearchByPrn(Scanner sc, StudentOperations operations) {

        System.out.print("Enter PRN to search: ");

        String searchPrn = sc.nextLine();

        try {

            Student foundStudent = operations.searchByPrn(searchPrn);

            if (foundStudent != null) {

                System.out.println("Student Found: " + foundStudent);

            } else {

                System.out.println("Student with PRN " + searchPrn + " not found.");

            }

        } catch (StudentExceptions.InvalidSearchCriteriaException e) {

            System.out.println("Error: Invalid search criteria. " + e.getMessage());

        } catch (StudentExceptions.StudentNotFoundException e) {

            System.out.println("Error: Student not found. " + e.getMessage());

        }

    }

    private static void handleSearchByName(Scanner sc, StudentOperations operations) {

        System.out.print("Enter Name to search: ");

        String searchName = sc.nextLine();

        try {

            Student foundByName = operations.searchByName(searchName);

            if (foundByName != null) {

                System.out.println("Student Found: " + foundByName);

            } else {

                System.out.println("Student with Name " + searchName + " not found.");

            }

        } catch (StudentExceptions.InvalidSearchCriteriaException e) {

            System.out.println("Error: Invalid search criteria. " + e.getMessage());

        } catch (StudentExceptions.StudentNotFoundException e) {

            System.out.println("Error: Student not found. " + e.getMessage());

        }

    }

    private static void handleSearchByPosition(Scanner sc, StudentOperations operations) {

        System.out.print("Enter position index to search: ");

        int pos = sc.nextInt();

        sc.nextLine(); // Consume newline

        try {

            Student foundByPos = operations.searchByPosition(pos);

            if (foundByPos != null) {

                System.out.println("Student at position " + pos + ": " + foundByPos);

            }

        } catch (StudentExceptions.InvalidSearchCriteriaException e) {

            System.out.println("Error: Invalid search criteria. " + e.getMessage());

        } catch (StudentExceptions.StudentNotFoundException e) {

            System.out.println("Error: Student not found. " + e.getMessage());

        }

    }

    private static void handleUpdateStudent(Scanner sc, StudentOperations operations) throws StudentExceptions.StudentUpdateException, StudentExceptions.InvalidUpdateDataException {

        System.out.print("Enter PRN of the student to update: ");

        String updatePrn = sc.nextLine();

        try {

            Student existingStudent = operations.searchByPrn(updatePrn);

            if (existingStudent != null) {

                System.out.print("Enter new Name: ");

                String newName = sc.nextLine();

                System.out.print("Enter new Date of Birth (DD-MM-YYYY): ");

                String newDob = sc.nextLine();

                System.out.print("Enter new Marks: ");

                double newMarks = sc.nextDouble();

                sc.nextLine(); // Consume newline

                Student updatedStudent = new Student(updatePrn, newName, newDob, newMarks);

                try {

                    operations.updateStudent(updatePrn, updatedStudent);

                    System.out.println("Student updated successfully.");

                } catch (StudentExceptions.StudentUpdateException | StudentExceptions.InvalidUpdateDataException e) {

                    System.out.println("Error: " + e.getMessage());

                }

            } else {

                System.out.println("Student with PRN " + updatePrn + " not found.");

            }

        } catch (StudentExceptions.InvalidSearchCriteriaException e) {

            System.out.println("Error: Invalid search criteria. " + e.getMessage());

        } catch (StudentExceptions.StudentNotFoundException e) {

            System.out.println("Error: Student not found. " + e.getMessage());

        }

    }

    private static void handleDeleteStudent(Scanner sc, StudentOperations operations) {

        System.out.print("Enter PRN of the student to delete: ");

        String deletePrn = sc.nextLine();

        try {

            operations.deleteStudent(deletePrn);

            System.out.println("Delete operation completed.");

        } catch (StudentExceptions.StudentDeletionException e) {

            System.out.println("Error: Unable to delete student. " + e.getMessage());

        } catch (StudentExceptions.InvalidPRNException e) {

            System.out.println("Error: Invalid PRN provided. " + e.getMessage());

        }

    }

}

}

}

2. StudentOperations.java

/\*

 \* StudentOperations.java

 \* Description: This file contains methods to perform operations on an ArrayList of Student objects.

 \*/

import java.util.ArrayList;

public class StudentOperations {

    // ArrayList to store Student objects

    private ArrayList<Student> students;

    // Constructor to initialize the ArrayList

    public StudentOperations() {

        students = new ArrayList<>();

    }

    // Validation methods

    private void validatePRN(String prn) throws StudentExceptions.InvalidPRNException {

        if (prn == null || prn.trim().isEmpty() || !prn.matches("\\d{11}")) {

            throw new StudentExceptions.InvalidPRNException(prn);

        }

    }

    private void validateMarks(double marks) throws StudentExceptions.InvalidMarksException {

        if (marks < 0 || marks > 100) {

            throw new StudentExceptions.InvalidMarksException(marks);

        }

    }

    public void addStudent(Student student) throws StudentExceptions.DuplicatePRNException,

                                                 StudentExceptions.InvalidStudentDataException {

        // Validate student data

        try {

            validatePRN(student.getPrn());

            validateMarks(student.getMarks());

        } catch (StudentExceptions.InvalidPRNException | StudentExceptions.InvalidMarksException e) {

            throw new StudentExceptions.InvalidStudentDataException(e.getMessage());

        }

        // Check for duplicate PRN

        try {

            if (searchByPrn(student.getPrn()) != null) {

                throw new StudentExceptions.DuplicatePRNException(student.getPrn());

            }

        } catch (StudentExceptions.InvalidSearchCriteriaException | StudentExceptions.StudentNotFoundException e) {

            // If StudentNotFoundException is caught, it means the PRN is not a duplicate, so we can proceed.

        }

        students.add(student);

        System.out.println("Student added successfully.");

    }

    public void displayStudents() throws StudentExceptions.EmptyStudentListException {

        if (students.isEmpty()) {

            throw new StudentExceptions.EmptyStudentListException();

        }

        for (int i = 0; i < students.size(); i++) {

            System.out.println("Position " + i + ": " + students.get(i));

        }

    }

    public Student searchByPrn(String prn) throws StudentExceptions.InvalidSearchCriteriaException,

                                                 StudentExceptions.StudentNotFoundException {

        try {

            validatePRN(prn);

        } catch (StudentExceptions.InvalidPRNException e) {

            throw new StudentExceptions.InvalidSearchCriteriaException(e.getMessage());

        }

        for (Student student : students) {

            if (student.getPrn().equals(prn)) {

                return student;

            }

        }

        throw new StudentExceptions.StudentNotFoundException("PRN: " + prn);

    }

    public Student searchByName(String name) throws StudentExceptions.InvalidSearchCriteriaException,

                                                  StudentExceptions.StudentNotFoundException {

        if (name == null || name.trim().isEmpty()) {

            throw new StudentExceptions.InvalidSearchCriteriaException("Name cannot be empty");

        }

        for (Student student : students) {

            if (student.getName().equalsIgnoreCase(name)) {

                return student;

            }

        }

        throw new StudentExceptions.StudentNotFoundException("Name: " + name);

    }

    public Student searchByPosition(int position) throws StudentExceptions.InvalidSearchCriteriaException,

                                                       StudentExceptions.StudentNotFoundException {

        if (position < 0) {

            throw new StudentExceptions.InvalidSearchCriteriaException("Position cannot be negative");

        }

        if (position >= students.size()) {

            throw new StudentExceptions.StudentNotFoundException("Position: " + position);

        }

        return students.get(position);

    }

    public void updateStudent(String prn, Student updatedStudent) throws StudentExceptions.StudentUpdateException,

                                                                       StudentExceptions.InvalidUpdateDataException {

        try {

            validatePRN(prn);

            validatePRN(updatedStudent.getPrn());

            validateMarks(updatedStudent.getMarks());

        } catch (StudentExceptions.InvalidPRNException | StudentExceptions.InvalidMarksException e) {

            throw new StudentExceptions.InvalidUpdateDataException(e.getMessage());

        }

        for (int i = 0; i < students.size(); i++) {

            if (students.get(i).getPrn().equals(prn)) {

                students.set(i, updatedStudent);

                return;

            }

        }

        throw new StudentExceptions.StudentUpdateException("Student with PRN " + prn + " not found");

    }

    public void deleteStudent(String prn) throws StudentExceptions.StudentDeletionException,

                                               StudentExceptions.InvalidPRNException {

        validatePRN(prn);

        for (int i = 0; i < students.size(); i++) {

            if (students.get(i).getPrn().equals(prn)) {

                students.remove(i);

                return;

            }

        }

        throw new StudentExceptions.StudentDeletionException("Student with PRN " + prn + " not found");

    }

}

}

3. Student.java

/\*

 \* Student.java

 \* Description: This file contains the definition of the Student class with attributes: prn, name, dob, and marks.

 \*/

public class Student {

    private String prn;

    private String name;

    private String dob;

    private double marks;

    // Constructor to initialize Student object

    public Student(String prn, String name, String dob, double marks) {

        this.prn = prn;

        this.name = name;

        this.dob = dob;

        this.marks = marks;

    }

    // Getter and Setter methods for each attribute

    public String getPrn() {

        return prn;

    }

    public void setPrn(String prn) {

        this.prn = prn;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getDob() {

        return dob;

    }

    public void setDob(String dob) {

        this.dob = dob;

    }

    public double getMarks() {

        return marks;

    }

    public void setMarks(double marks) {

        this.marks = marks;

    }

    // Overriding toString() to display student details

    @Override

    public String toString() {

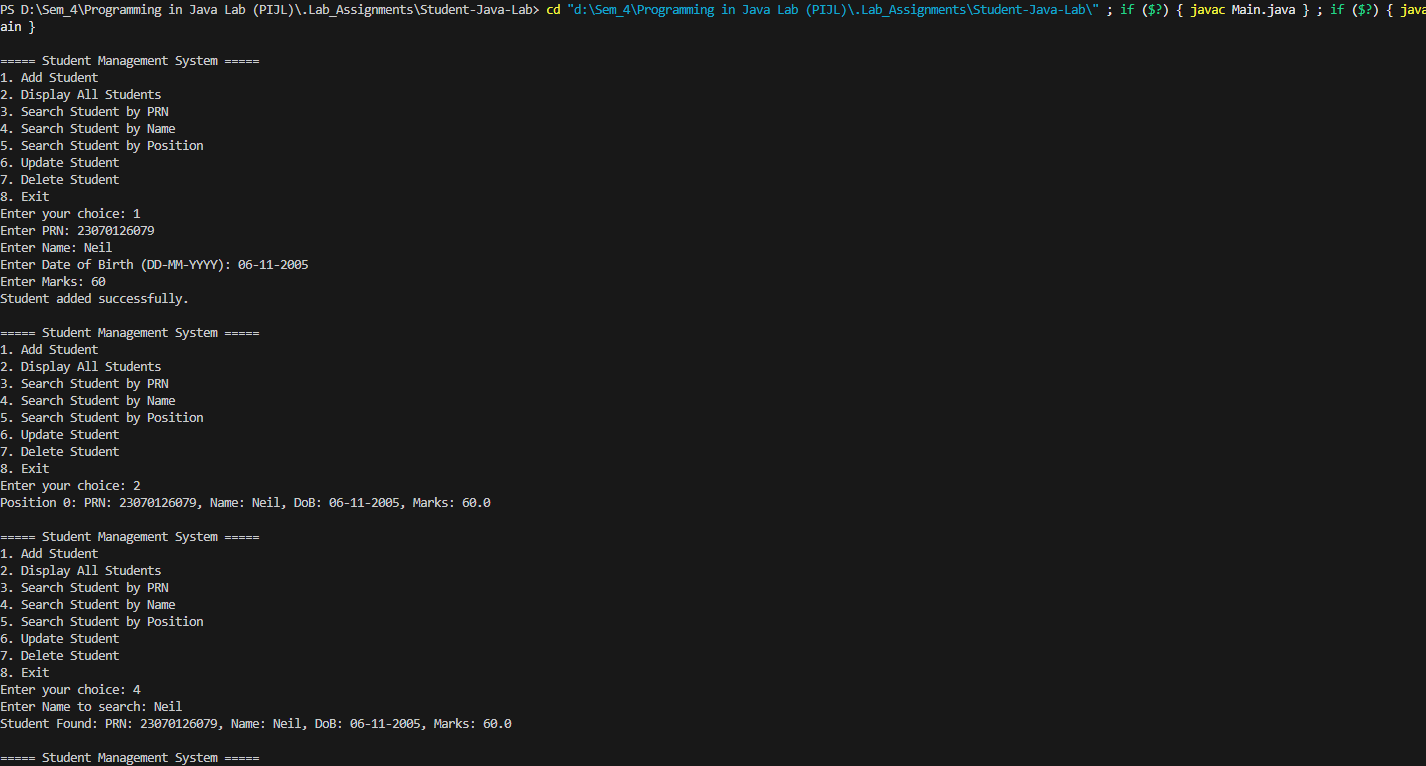
        return "PRN: " + prn + ", Name: " + name + ", DoB: " + dob + ", Marks: " + marks;

    }

}

}

4. Output

 A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect. A black screen with white text

AI-generated content may be incorrect.

5. Repository

https://github.com/Neil-Cardoz/Student-Exception-Java-Lab