INFORMATICS INSTITUTE OF TECHNOLOGY

Software Development II

Coursework Report 2023/2024

M.Z.M. Juzail

w2083187

20222336

Task 01 – Source Code

StudentActivityManagementSystem.java

```
import java.io.*;
import java.util.*;

public class StudentActivityManagementSystem {
    private static final int MAXIMUM_CAPACITY = 100;
    private static Student[] students = new Student[MAXIMUM_CAPACITY];
    private static int studentCount = 0;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            printMenu();
            int choice = getValidChoice(scanner);

            switch (choice) {
                case 1 -> checkAvailableSeats();
                case 2 -> registerStudent(scanner);
                case 3 -> deleteStudent(scanner);
                case 4 -> findStudent(scanner);
                case 5 -> storeStudentDetails();
                case 6 -> loadStudentDetails();
                case 7 -> viewStudentsByName();
                case 8 -> manageStudentResults(scanner);
                case 9 -> {
                    System.out.println("Exiting the program..........");
                    return; // Exit the program
                }
            default -> System.out.println("Invalid choice. Please try again.");
        }
    }
}

private static int getValidChoice(Scanner scanner) {
    int choice;
    while (true) {
        System.out.print("Enter your choice: ");
        if (scanner.hasNextInt()) {
            choice = scanner.nextInt();
            if (choice >= 1 && choice <= 9) {
                scanner.nextLine(); // Consume newline
                break;
            }
        } else {
            scanner.nextLine(); // Consume invalid input
        }
        System.out.println("Invalid choice. Please enter a number between 1 and 9.");
    }
    return choice;
}

private static void printMenu() {
    System.out.println("******************************************");
    System.out.println("*              MENU OPTION               *");
    System.out.println("******************************************");
    System.out.println("1. Check available seats");
    System.out.println("2. Register student (with ID)");
    System.out.println("3. Delete student");
    System.out.println("4. Find student (with student ID)");
    System.out.println("5. Store student details into a file");
    System.out.println("6. Load student details from the file to the system");
    System.out.println("7. View the list of students based on their names");
    System.out.println("8. Manage student results");
    System.out.println("9. Exit");
    System.out.println("******************************************");
}

private static void checkAvailableSeats() {
    int availableSeats = MAXIMUM_CAPACITY - studentCount;
    System.out.println("Available seats: " + availableSeats);
}

private static void registerStudent(Scanner scanner) {
    if (studentCount >= MAXIMUM_CAPACITY) {
        System.out.println("No available seats. Registration is full.");
        return;
    }
    System.out.print("Enter Student ID: ");
    String id = scanner.nextLine();
    System.out.print("Enter Student Name: ");
    String name = scanner.nextLine();

    students[studentCount++] = new Student(id, name);
    System.out.println("Student registered successfully.");
}

private static void deleteStudent(Scanner scanner) {
    System.out.print("Enter Student ID to delete: ");
```

```
        String id = scanner.nextLine();
        for (int i = 0; i < studentCount; i++) {
            if (students[i].getId().equals(id)) {
                students[i] = students[--studentCount];
                students[studentCount] = null;
                System.out.println("Student deleted successfully.");
                return;
            }
        }
        System.out.println("Student not found.");
    }

    private static void findStudent(Scanner scanner) {
        System.out.print("Enter Student ID to find: ");
        String id = scanner.nextLine();
        for (Student student : students) {
            if (student != null && student.getId().equals(id)) {
                System.out.println("Student found: " + student);
                return;
            }
        }
        System.out.println("Student not found.");
    }

    private static void storeStudentDetails() {
        try (PrintWriter writer = new PrintWriter(new FileWriter("student_details.txt"))) {
            for (Student student : students) {
                if (student != null) {
                    writer.println(student.getId() + "," + student.getName());
                }
            }
            System.out.println("Student details stored successfully.");
        } catch (IOException e) {
            System.out.println("Error storing student details: " + e.getMessage());
        }
    }

    private static void loadStudentDetails() {
        try (Scanner fileScanner = new Scanner(new File("student_details.txt"))) {
            studentCount = 0; // Reset the student count
            while (fileScanner.hasNextLine()) {
                String[] details = fileScanner.nextLine().split(",");
                students[studentCount++] = new Student(details[0], details[1]);
            }
            System.out.println("Student details loaded successfully.");
        } catch (FileNotFoundException e) {
            System.out.println("Error loading student details: " + e.getMessage());
        }
    }

    private static void viewStudentsByName() {
        Arrays.sort(students, 0, studentCount, Comparator.comparing(Student::getName));
        for (Student student : students) {
            if (student != null) {
                System.out.println(student);
            }
        }
    }
}
```

Task 02 – Source Code

```
private static void manageStudentResults(Scanner scanner) {
    System.out.println("a. Add student name");
    System.out.println("b. Enter module marks (1, 2, and 3)");
    System.out.print("Enter your choice: ");
    String choice = scanner.nextLine();

    switch (choice) {
        case "a" -> addStudentName(scanner);
        case "b" -> addModuleMarks(scanner);
        default -> System.out.println("Invalid choice.");
    }
}

private static void addStudentName(Scanner scanner) {
    System.out.print("Enter Student ID: ");
    String id = scanner.nextLine();
    for (Student student : students) {
        if (student != null && student.getId().equals(id)) {
            System.out.print("Enter Student Name: ");
            String name = scanner.nextLine();
            student.setName(name);
            System.out.println("Student name added successfully.");
            return;
        }
    }
    System.out.println("Student not found.");
}
```

```
private static void addModuleMarks(Scanner scanner) {
    System.out.print("Enter Student ID: ");
    String id = scanner.nextLine();
    for (Student student : students) {
        if (student != null && student.getId().equals(id)) {
            int[] marks = new int[3];
            for (int i = 0; i < 3; i++) {
                System.out.print("Enter marks for Module " + (i + 1) + ": ");
                marks[i] = scanner.nextInt();
            }
            student.setModuleMarks(marks);
            System.out.println("Module marks added successfully.");
            return;
        }
    }
    System.out.println("Student not found.");
}
```

Student.java

```
public class Student {
    private String studentID;
    private String studentName;
    private int[] moduleMarks;
    private String grade;

    public Student(String studentID, String studentName) {
        this.studentID = studentID;
        this.studentName = studentName;
        this.moduleMarks = new int[3]; // Assuming 3 modules
        this.grade = "N/A"; // Default grade
```

```
}

public String getStudentID() {
    return studentID;
}

public String getStudentName() {
    return studentName;
}

public void setStudentName(String studentName) {
    this.studentName = studentName;
}

public int[] getModuleMarks() {
    return moduleMarks;
}

public void setModuleMarks(int[] moduleMarks) {
    this.moduleMarks = moduleMarks;
    this.grade = calculateGrade(moduleMarks);
}

public String getGrade() {
    return grade;
}

public double calculateAverage(int[] marks) {
    int total = 0;
    for (int mark : marks) {
            total += mark;
        }
        return total / 3.0;
    }

    private String calculateGrade(int[] marks) {
        double average = calculateAverage(marks);
        if (average >= 80) {
            return "Distinction";
        } else if (average >= 70) {
            return "Merit";
        } else if (average >= 40) {
            return "Pass";
        } else {
            return "Fail";
        }
    }
}
```

Module.java

```
import java.io.Serializable;

public class Module implements Serializable {
    private String moduleName;
    private int marks;

    public Module(String moduleName, int marks) {
        this.moduleName = moduleName;
        this.marks = marks;
    }
    public String getModuleName() {
        return moduleName;
    }

    public void setModuleName(String moduleName) {
        this.moduleName = moduleName;
    }

    public int getMarks() {
        return marks;
    }

    public void setMarks(int marks) {
        this.marks = marks;
    }

    @Override
    public String toString() {
        return "Module Name: " + moduleName + ", Marks: " + marks;
    }
}
```

Task 03 – Source Code

```
private static void manageStudentResults(Scanner scanner) {
    System.out.println("a. Add student name");
    System.out.println("b. Enter module marks (1, 2, and 3)");
    System.out.println("c. Generate a summary of the system");
    System.out.println("d. Generate complete report");
    System.out.print("Enter your choice: ");
    String choice = scanner.nextLine();

    switch (choice) {
        case "a" -> addStudentName(scanner);
        case "b" -> addModuleMarks(scanner);
        case "c" -> generateSummary();
        case "d" -> generateCompleteReport();
        default -> System.out.println("Invalid choice.");
    }
}

private static void generateSummary() {
    int totalRegistrations = studentCount;
    int studentsPassedModule1 = 0;
    int studentsPassedModule2 = 0;
    int studentsPassedModule3 = 0;

    for (Student student : students) {
        if (student != null) {
            int[] marks = student.getModuleMarks();
            if (marks[0] >= 40) studentsPassedModule1++;
            if (marks[1] >= 40) studentsPassedModule2++;
            if (marks[2] >= 40) studentsPassedModule3++;
        }
    }

    System.out.println("Total student registrations: " + totalRegistrations);
    System.out.println("Students passed Module 1: " + studentsPassedModule1);
    System.out.println("Students passed Module 2: " + studentsPassedModule2);
    System.out.println("Students passed Module 3: " + studentsPassedModule3);
}

private static void generateCompleteReport() {
    Arrays.sort(students, 0, studentCount, (s1, s2) -> {
        if (s1 == null || s2 == null) return 0;
        return Double.compare(s2.getAverageMarks(), s1.getAverageMarks());
    });

    for (Student student : students) {
        if (student != null) {
            System.out.println(student);
        }
    }
}
```

Task 04 – Testing

Test case	Expected Output	Actual Result	Pass/Fail
Check available seats before registration	Press “1” to check number of available seats	Displays the number of available seats (100)	PASS
Register a new student	Enter student ID (8 characters): w2083187 Enter student name: Juzail Student registered successfully.	Enter student ID (8 characters): w2083187 Enter student name: Juzail Student registered successfully.	PASS
Delete a student	Enter student ID to delete: w2083187 Student deleted successfully.	Enter student ID to delete: w2083187 Student deleted successfully.	PASS
Find a student	Enter student ID to find (8 characters): w2083187 Student found: Juzail	Enter student ID to find (8 characters): w2083187 Student found: Juzail	PASS
Store details	Student details saved successfully. 1. W2083187, Juzail.0,0,0 2. W2036566, Aadil.0,0,0 3. W2084568, Raazi.0,0,0 4. W2084789, Usman.0,0,0	Student details saved successfully. 1. W2083187, Juzail.0,0,0 2. W2036566, Aadil.0,0,0 3. W2084568, Raazi.0,0,0 4. W2084789, Usman.0,0,0	PASS
Load details	Student details loaded successfully.	Student details loaded successfully.	PASS
View students by their name	1. W2083187, Juzail. 2. W2036566, Aadil. 3. W2084568, Raazi. 4. W2084789, Usman.	1. W2083187, Juzail. 2. W2036566, Aadil. 3. W2084568, Raazi. 4. W2084789, Usman	PASS
Manage student results	Press “8” to manage students results	Press “8” to manage students results	PASS
Update student name	Enter student ID to add name: w2083187 Enter new student name: Juzail Student name updated successfully.	Enter student ID to add name: w2083187 Enter new student name: Juzail Student name updated successfully.	PASS

Enter module marks	Enter student ID to add marks: w2083187 Enter marks for Module 1 (0-100): 45 Enter marks for Module 2 (0-100): 50 Enter marks for Module 3 (0-100): 70 Module marks and grade updated successfully.	Enter student ID to add marks: w2083187 Enter marks for Module 1 (0-100): 45 Enter marks for Module 2 (0-100): 50 Enter marks for Module 3 (0-100): 70 Module marks and grade updated successfully.	PASS
Generate summary report	Complete Report: ID: w2083187 Name: Juzail Module 1 Marks: 45.0 Module 2 Marks: 50.0 Module 3 Marks: 70.0 Average Marks: 55.0 Grade: Pass	Complete Report: ID: w2083187 Name: Juzail Module 1 Marks: 45.0 Module 2 Marks: 50.0 Module 3 Marks: 70.0 Average Marks: 55.0 Grade: Pass	PASS
Generate complete report	Complete Report: ID: w2083187 Name: Juzail Module 1 Marks: 45.0 Module 2 Marks: 50.0 Module 3 Marks: 70.0 Average Marks: 55.0 Grade: Pass	Complete Report: ID: w2083187 Name: Juzail Module 1 Marks: 45.0 Module 2 Marks: 50.0 Module 3 Marks: 70.0 Average Marks: 55.0 Grade: Pass	PASS
Go to main menu	Press “8”, “e” to exit from the 8th choice	Press “8”, “e” to exit from the 8th choice	PASS
Exit the program	Press “9”, to exit from the program Exiting program	Press “9”, to exit from the program Exiting program	PASS

Task 04 – Testing – Discussion

Talking about test cases

I selected the test cases based on the expected inputs for the main program.

1. Verify the availability of seats
2. Register the student
3. Delete the student;
4. Locate the student;
5. Save the student's information in a file.
6. Open the file and load the student data.
7. View students by their names
8. Change the student's name
    a. Add the student's name;
    b. Add the module marks;
    c. Create a report summary;
    d. Create a complete report;
    e. Return to the main menu.
9. Exit

End situation and error handling skills were also examined and then corrected when a defect was discovered, in addition to these tests. These input validations are so numerous that the report does not include a record of each one separately. But by addressing these topics, the test cases guarantee that the software is reliable, can appropriately manage a range of user interactions, and preserves data integrity.

Self-Evaluation form

Task 1:

1. Check available seats - fully implemented and working
2. Register student - fully implemented and working
3. Delete student - fully implemented and working
4. Find student - fully implemented and working
5. Save student details into a text file - fully implemented and working
6. Load student details from the text file to the system - fully implemented and working
7. View the list of students based on their names - fully implemented and working
8. Manage student details - fully implemented and working
9. Exit - fully implemented and working

Task 2:

a. Add/Update student name - fully implemented and working

b. Add module marks - fully implemented and working

Task 3:

c. Generate summary report - fully implemented and working

d. Generate complete report - fully implemented and working

e. Return to main menu - fully implemented and working

Criteria		Allocated marks	Expected marks	Total
Task 1	Three marks for each option (1,2,3,4,5,6,7,8)	24	**20**	**(30)**
	Menu works correctly	6	**4**	
Task 2	Student class works correctly	14	**14**	**(30)**
	Module class works correctly	10	8	
	Sub menu (A and B works well)	6	6	
Task 3	Report – Generate a summary	7	6	**(20)**
	Report – Generate the complete report	10	9	
	Implementation of Bubble sort	3	3	
Task 4	Test case coverage and reasons	6	4	**(10)**
	Writeup on which version is better and why.	4	2	
Coding Style (Comments, indentation, style)		7	6	**(10)**
Complete the self-evaluation form indicating what you have accomplished to ensure appropriate feedback.		3	3	
Totals		100	**85**	**(100)**

Demo: At the discretion of your tutor, you may be called on to give a demo of your work to demonstrate understanding of your solutions. If you cannot explain your code and are unable to point to a reference within your code of where this code was found (i.e., in a textbook or on the internet) then significant marks will be lost for that marking component. If you do not attend a requested demo your mark will be capped at 50%.

References

W3Schools - https://www.w3schools.com/java/default.asp

Sample CRUD based system - https://www.geeksforgeeks.org/crud-operations-in-studentmanagement-system-in-java/

Stack Overflow thread of a issue I faced during the programming process - https://stackoverflow.com/questions/218384/what-is-anullpointerexception-and-how-do-i-fix-it/218510#218510