INFORMATICS INSTITUTE OF TECHNOLOGY

Software Development II

Coursework Report 2023/2024

Perumpulli Arachchige Prasansa Ramidu

w2084743

20233161

Task 01 – Source Code

```java
import java.util.Scanner;
import java.util.InputMismatchException;
import java.util.Arrays;
import java.util.Comparator;
import java.util.NoSuchElementException;
import java.lang.NumberFormatException;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
//class
public class Task_01 {
    private static Scanner input = new Scanner(System.in);//Scanner for user
input
    private static final int capacity = 100;//Maximum capacity of students
    private static String[][] students = new String[100][2];//2D-Array to
store each student name and student ID
    private static File new_file = new
File("Students_Management.txt");//Creating an instance of the File class
representing the file "Students_Management.txt"

    //main method
    public static void main(String[] args){
        initializer(students);//Calling initializer method
        create_file(new_file);//Calling create_file method
        try{
            boolean exit = false;
            while (!exit){
                int option = menu();
                switch (option){
                    case (1):
                        available_seats(students);//Calling available seats
method
                        break;
                    case (2):
                        register_student(students);//Calling register student
method
                        break;
                    case (3):
                        delete_student(students);//Calling delete students
method
                        break;
                    case (4):
                        find_student(students);//Calling find student method
                        break;
                    case (5):
                        dump_details_to_file(new_file, students);//Calling
dump students details to the file method
                        break;
                    case (6):
                        load_details_from_file(new_file, students);//Calling
load students details from the file method
                        break;
                    case (7):
                        view_details(students);//Calling view student details
method
                        break;
                    case (8):
                        exit = true;
                        System.out.println("You have successfully exited from
the university intake system");
                        break;
                    default:
                        System.out.println("Your choice is incorrect.Please
enter a given option that been given above");
                }
            }
        }
        catch (InputMismatchException e){
            System.out.println("Invalid input.please enter a valid input");
        }
    }
    //menu method
    public static int menu(){
        try {
            //Menu:
            System.out.println();
            System.out.println("***MENU***");
            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.To exit from this university intake
system");
            System.out.println();
            System.out.println("Please enter your option that you prefer: ");
            String value_01 = input.nextLine();
            int value = Integer.parseInt(value_01);
            return value;
        }
        catch (NumberFormatException e){
            System.out.println("Error occurred you cannot convert string that
contain data(text) in to integer or a string that contain double value to
integer: " + e);
        }
        return 0;
    }

    //available_seats method
    public static void available_seats(String[][] students){
        for (int K = 0;K < students.length;K++){
            if (students[K][0].equals("empty") &&
students[K][1].equals("empty")){
                System.out.println("Student slot " + (K + 1) + " is available
for the student registration");
            }
            else {
                System.out.println("Student slot " + (K + 1) + " is already
been occupied by someone. Please try again!");
            }
        }
    }

    //register_student method
    public static void register_student(String[][] students){
        try {
            System.out.println("Enter the student name that you want
register: ");
            String student = input.nextLine();
            System.out.println("Enter the student id to register a student:
");
            String student_id = input.nextLine();
            for (int R = 0;R < students.length;R++){
                if (0 < Integer.parseInt(student_id) &&
Integer.parseInt(student_id) <= 100){
                    boolean result = containsId(students, student_id);
                    if (students[R][0].equals("empty") &&
students[R][1].equals("empty")){
                        if (!(result)){
                            students[R][0] = student;
                            students[R][1] = student_id;
                            System.out.println("You have successfully
register the student.");
                            break;
                        }
                        else {
                            System.out.println("This student id is already
existing");
                            break;
                        }
                    }
                    else{
                        if (student.equals(students[R][0]) &&
student_id.equals(students[R][1])){
                            System.out.println("Actually this student slot "
+ (R + 1) + " has been occupied by student name called " + students[R][0] + "
And it's student id is " + students[R][1]);
                            break;
                        }
                    }
                }
                else {
                    System.out.println("You have exceed the bound that the
student id can get. So please enter a valid bound value for the input.");
                    break;
                }
            }
        }
        catch (NumberFormatException e){
            System.out.println("Error occurred you cannot convert string that
contain data(text) in to integer or a string that contain double value to
integer: " + e);
        }
```

```java
    }

    //delete_student method
    public static void delete_student(String[][] students){
        System.out.println("Enter the student name that you want to delete:
");
        String del_student = input.nextLine();
        System.out.println("Enter the student id that you want to delete: ");
        String del_student_id = input.nextLine();
        int iterator = 0;
        for (int d = 0;d < students.length;d++) {
            if (del_student.equals(students[d][0]) &&
del_student_id.equals(students[d][1])){
                students[d][0] = "empty";
                students[d][1] = "empty";
                System.out.println("You have successfully deleted the student
slot " + (d + 1) + " contain details");
                break;
            }
            else {
                iterator = iterator + 1;
            }
        }
        if ( iterator == students.length ) {
            System.out.println("You entered name " + del_student + " and you
entered student id " + del_student_id + " actually are invalid inputs and
these are not exist in the array.");
        }
    }

    //find_student method
    public static void find_student(String[][] students){
        try {
            System.out.println("Enter the student id that belong to the
student that you want to find: ");
            String student_to_find_id = input.nextLine();
            int iterator = 0;
            for (int Z = 0;Z < students.length;Z++){
                if (0 < Integer.parseInt(student_to_find_id) &&
Integer.parseInt(student_to_find_id) <= 100){
                    if (!(students[Z][0].equals("empty") &&
students[Z][1].equals("empty") ) &&
student_to_find_id.equals(students[Z][1])){
                        System.out.println();
                        System.out.println("Student slot " + (Z + 1) + "
finds the student that you looking for and his/her name is " +
students[Z][0]);
                        System.out.println("Student find successfully
completed");

                        System.out.println();
                        break;
                    }
                    else {
                        iterator = iterator + 1;
                    }
                }
                else {
                    System.out.println("You just entered id called " +
student_to_find_id + " is an invalid input and it is not exist in the
array.");
                    break;
                }
            }
            if (iterator == students.length) {
                System.out.println("No student with this id represent in the
array.");
            }
        }
        catch (NumberFormatException e){
            System.out.println("Error occurred you cannot convert a string
that contain data(text) to Integer or a string that contain double value to
Integer: " + e);
        }
    }

    //dump_details_to_file method
    public static void dump_details_to_file(File new_file, String[][]
students){
        try {
            FileWriter student_write = new FileWriter(new_file);
            for (int Y = 0;Y < students.length;Y++){
                if (students[Y][0].equals("empty") &&
students[Y][1].equals("empty")){
                    student_write.write("empty : empty\n");
                    if (Y == (students.length - 1)){
                        System.out.println("You have wrote the student
details successfully to the file.");
                        break;
                    }
                }
                else {
                    student_write.write(students[Y][0] + " : " +
students[Y][1] + "\n");
                    if (Y == (students.length - 1)){
                        System.out.println("You have wrote the student
details successfully to the file.");
                        break;
                    }
                }
            }
            student_write.close();
        }
        catch (IOException e){
            System.out.println("Error occurred while dumping data to the
file: " + e);
        }
    }

    //load_details_from_file method
    public static void load_details_from_file(File new_file, String[][]
students){
        try {
            Scanner read_file = new Scanner(new_file);
            int x = 0;
            while (read_file.hasNext() || x < students.length ) {
                String file_line = read_file.nextLine();
                String[] line_parts = file_line.split(" : ");
                students[x][0] = line_parts[0];
                students[x][1] = line_parts[1];
                x = x + 1;
                System.out.println(file_line);
            }
            System.out.println("\nYou have successfully got the student
details from the file.");
            read_file.close();
        }
        catch (NoSuchElementException e){
            System.out.println("Error occurred No data been found at the
start when you select the load data from the file option: " + e);
        }
        catch (IOException e){
            System.out.println("Error occurred while loading data from the
file : " + e);
        }
    }

    //view_details method
    public static void view_details(String[][] students){
        Arrays.sort(students, Comparator.comparing(I -> I[0]));
        for (String[] intaker : students){
            System.out.println(Arrays.toString(intaker));
        }
    }

    //initializer method
    public static void initializer(String[][] students){
        for (int i = 0;i < students.length;i++){
            for (int j = 0;j < students[i].length;j++){
                students[i][j] = "empty";
            }
        }
    }

    public static boolean containsId(String[][] students, String id){
        for (String[] start : students){
            if (start[1].equals(id)){
                return true;
            }
        }
        return false;
    }

    //create_file method
    public static void create_file(File new_file){
        try {
            boolean file_created = new_file.createNewFile();
            if (new_file.exists()){
                System.out.println("This " + new_file.getName() + " file is
already existing");
            }
            else {
                if (file_created){
                    System.out.println(new_file.getName() + "file created 
Successfully");
                }
                else {
                    System.out.println("While creating " + new_file.getName() 
+ " error occurred.");
                }
            }
        }
        catch (IOException e){
            System.out.println("Error occurred while creating the file: " + 
e);
        }
    }
}
```

Task 02 & Task 03 – Source Code

```
import java.util.Scanner;
import java.util.InputMismatchException;
import java.util.Arrays;
import java.util.Comparator;
import java.util.NoSuchElementException;
import java.lang.NumberFormatException;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
//class
public class Task_01 {
    private static Scanner input = new Scanner(System.in);//Scanner for user 
input
    private static final int capacity = 100;//Maximum capacity of students
    private static String[][] students = new String[100][2];//2D-Array to 
store each student name and student ID
    private static Student[] studentsArray = new Student[100];//Array to 
store student details
    private static File new_file = new 
File("Students_Management.txt");//Creating an instance of the File class 
representing the file "Students_Management.txt"

    //main method
    public static void main(String[] args){
        initializer(students);//Calling initializer method

        initializerArray(studentsArray);//Calling initializerArray method

        create_file(new_file);//Calling create_file method
        try{
            boolean exit = false;
            while (!exit){
                int option = menu();
                switch (option){
                    case (1):
                        available_seats(students);//Calling available seats
method
                        break;
                    case (2):
                        register_student(students);//Calling register student
method
                        break;
                    case (3):
                        delete_student(students);//Calling delete students
method
                        break;
                    case (4):
                        find_student(students);//Calling find student method
                        break;
                    case (5):
                        dump_details_to_file(new_file, students);//Calling
dump students details to the file method
                        break;
                    case (6):
                        load_details_from_file(new_file, students,
studentsArray);//Calling load students details from the file method
                        break;
                    case (7):
                        view_details(students);//Calling view student details
method
                        break;
                    case (8):
                        additional_students_controls(students);//Calling
additional students control method
                        break;
                    case (9):
                        generate_system_summery();//Calling generate system
summery method
                        break;
                    case (10):
                        generate_students_report();//Generate students report
method
                        break;
                    case (11):
                        exit = true;
                        System.out.println("You have successfully exited from
the university intake system");//To exit from the university intake system
                        break;
                    default:
                        System.out.println("Your choice is incorrect.Please
enter a given option that been given above");//To represent the user entered
an incorrect option
                }
            }
        }
        catch (InputMismatchException e){
            System.out.println("Invalid input.please enter a valid input");
        }
    }
```

```java
    //menu method
    public static int menu(){
        try {
            //Menu:
            System.out.println();
            System.out.println("***MENU***");
            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.To handle additional students details");
            System.out.println("9.Generate system summary");
            System.out.println("10.Generate students report");
            System.out.println("11.To exit from this university intake
system");
            System.out.println();
            System.out.println("Please enter your option that you prefer: ");
            String value_01 = input.nextLine();
            int value = Integer.parseInt(value_01);
            return value;
        }
        catch (NumberFormatException e){
            System.out.println("Error occurred you cannot convert string that
contain data(text) in to integer or a string that contain double value to
integer: " + e);
        }
        return 0;
    }

    //available_seats method
    public static void available_seats(String[][] students){
        for (int K = 0;K < students.length;K++){
            if (students[K][0].equals("empty") &&
students[K][1].equals("empty")){
                System.out.println("Student slot " + (K + 1) + " is available
for the student registration");
            }
            else {
                System.out.println("Student slot " + (K + 1) + " is already
been occupied by someone. Please try again!");
            }
        }
    }

    //register_student method
    public static void register_student(String[][] students){
        try {
            System.out.println("Enter the student name that you want to
register: ");
            String student = input.nextLine();
            System.out.println("Enter the student id to register a student:
");
```

```
            String student_id = input.nextLine();

            for (int R = 0;R < students.length;R++){
                if (0 < Integer.parseInt(student_id) &&
Integer.parseInt(student_id) <= 100){
                    boolean result = containsId(students, student_id);
                    if (students[R][0].equals("empty") &&
students[R][1].equals("empty") &&
studentsArray[R].get_student_name().equals("empty") &&
studentsArray[R].get_student_id().equals("empty")){
                        if (!(result) ){
                            students[R][0] = student;
                            students[R][1] = student_id;

                            studentsArray[R].set_student_name( student );
                            studentsArray[R].set_student_id( student_id );

                            System.out.println("You have successfully
register the student.");
                            break;
                        }
                        else {
                            System.out.println("This student id already
existing");
                            break;
                        }
                    }
                    else{
                        if (student.equals(students[R][0]) &&
student_id.equals(students[R][1]) && student.equals(
studentsArray[R].get_student_name() ) && student_id.equals(
studentsArray[R].get_student_id() )){
                            System.out.println("Actually this student slot "
+ (R + 1) + " has been occupied by student name called " + students[R][0] + "
And it's student id is " + students[R][1]);
                            break;
                        }
                    }
                }
                else {
                    System.out.println("You have exceed the bound that the
student id can get. So please enter a valid bound value for the input.");
                    break;
                }
            }
        }
        catch (NumberFormatException e){
            System.out.println("Error occurred you cannot convert string that
contain data(text) in to integer or a string that contain double value to
integer: " + e);
        }
    }

    //delete_student method
    public static void delete_student(String[][] students){
        System.out.println("Enter the student name that you want to delete:
");
```

```
        String del_student = input.nextLine();
        System.out.println("Enter the student id that you want to delete: ");
        String del_student_id = input.nextLine();
        int iterator = 0;
        for (int d = 0;d < students.length;d++) {
            if (del_student.equals(students[d][0]) &&
del_student_id.equals(students[d][1]) &&
studentsArray[d].get_student_name().equals(students[d][0]) &&
studentsArray[d].get_student_id().equals(students[d][1])){
                students[d][0] = "empty";
                students[d][1] = "empty";

                studentsArray[d].set_student_name("empty");
                studentsArray[d].set_student_id("empty");
                System.out.println("You have successfully deleted the student
slot " + (d + 1) + " contain details");
                break;
            }
            else {
                iterator = iterator + 1;
            }
        }
        if ( iterator == students.length ) {
            System.out.println("You entered name " + del_student + " and you
entered student id " + del_student_id + " actually are invalid inputs and
these are not exist in the array.");
        }
    }

    //find_student method
    public static void find_student(String[][] students){
        try {
            System.out.println("Enter the student id that belong to the
student that you want to find: ");
            String student_to_find_id = input.nextLine();
            int iterator = 0;
            for (int Z = 0;Z < students.length;Z++){
                if (0 < Integer.parseInt(student_to_find_id) &&
Integer.parseInt(student_to_find_id) <= 100){
                    if (!(students[Z][0].equals("empty") &&
students[Z][1].equals("empty") ) &&
student_to_find_id.equals(students[Z][1])){
                        System.out.println();
                        System.out.println("Student slot " + (Z + 1) + "
finds the student that you looking for and his/her name is " +
students[Z][0]);
                        System.out.println("Student find successfully
completed");
                        System.out.println();
                        break;
                    }
                    else {
                        iterator = iterator + 1;
                    }
                }
                else {
                    System.out.println("You just entered id called " +
```

```java
student_to_find_id + " is an invalid input and it is not exist in the 
array.");
                    break;
                }
            }
            if (iterator == students.length) {
                System.out.println("No student with this id represent in the 
array.");
            }
        }
        catch (NumberFormatException e){
            System.out.println("Error occurred you cannot convert a string 
that contain data(text) to Integer or a string that contain double value to 
Integer: " + e);
        }
    }

    //dump_details_to_file method
    public static void dump_details_to_file(File new_file, String[][] 
students){
        try {
            FileWriter student_write = new FileWriter(new_file);
            for (int Y = 0;Y < studentsArray.length;Y++){
                if (studentsArray[Y].get_student_name().equals("empty") && 
studentsArray[Y].get_student_id().equals("empty")){
                    student_write.write("empty : empty : 0 : 0 : 0\n");
                    if (Y == (studentsArray.length - 1)){
                        System.out.println("You have wrote the student 
details successfully to the file.");
                        break;
                    }
                }
                else {
                    student_write.write(studentsArray[Y].get_student_name() + 
" : " + studentsArray[Y].get_student_id() + " : " + 
studentsArray[Y].modules[0].get_module_marks() +  " : " + 
studentsArray[Y].modules[1].get_module_marks() + " : " + 
studentsArray[Y].modules[2].get_module_marks()+"\n");
                    if (Y == (studentsArray.length - 1)){
                        System.out.println("You have wrote the student 
details successfully to the file.");
                        break;
                    }
                }
            }
            student_write.close();
        }
        catch (IOException e){
            System.out.println("Error occurred while dumping data to the 
file: " + e);
        }
    }

    //load_details_from_file method
    public static void load_details_from_file(File new_file, String[][] 
students, Student[] studentsArray){
        try {
            Scanner read_file = new Scanner(new_file);
            int x = 0;
            while (read_file.hasNext() || x < studentsArray.length ) {
                String file_line = read_file.nextLine();
                String[] line_parts = file_line.split(" : ");

                studentsArray[x].set_student_name( line_parts[0] );
                studentsArray[x].set_student_id( line_parts[1] );
                studentsArray[x].modules[0].set_module_marks(
Double.parseDouble( line_parts[2] ) );
                studentsArray[x].modules[1].set_module_marks(
Double.parseDouble( line_parts[3] ) );
                studentsArray[x].modules[2].set_module_marks(
Double.parseDouble( line_parts[4] ) );

                students[x][0] = line_parts[0];
                students[x][1] = line_parts[1];

                x = x + 1;
                System.out.println(file_line);
            }
            System.out.println("\nYou have successfully got the student
details from the file.");
            read_file.close();
        }
        catch (NoSuchElementException e){
            System.out.println("Error occurred No data been found at the
start when you select the load data from the file option: " + e);
        }
        catch (IOException e){
            System.out.println("Error occurred while loading data from the
file : " + e);
        }
    }

    //view_details method
    public static void view_details(String[][] students){
        Arrays.sort(students, Comparator.comparing(I -> I[0]));
        for (String[] intaker : students){
            System.out.println(Arrays.toString(intaker));
        }
    }

    //initializer method
    public static void initializer(String[][] students){
        for (int i = 0;i < students.length;i++){
            for (int j = 0;j < students[i].length;j++){
                students[i][j] = "empty";
            }
        }
    }

    //initializerArray method
    public static void initializerArray(Student[] studentsArray){
        for (int i = 0; i < studentsArray.length; i++) {
            studentsArray[i] = new Student("empty", "empty");
            for (int x = 0; x < studentsArray[i].modules.length; x++) {
                studentsArray[i].modules[x] = new Module(0);
            }
        }
    }

    //containId method
    public static boolean containsId(String[][] students, String id){
        for (String[] start : students){
            if (start[1].equals(id)){
                return true;
            }
        }
        return false;
    }

    //additional_students_controls method
    public static void additional_students_controls(String[][] students){
        while (true) {
            System.out.println("Enter the student id number: ");
            String student_id = input.nextLine();

            Student requested_student = null;
            for (Student s: studentsArray) {
                if (student_id.equals(s.get_student_id())) {
                    requested_student = s;
                    break;
                }
            }

            if (requested_student == null) {
                System.out.println("Wrong Student id Entered...Please check 
again!");
                break;
            } else {
                for (int i=1; i<4; i++) {
                    System.out.println("Enter module " + i + " marks: ");
                    while (true) {
                        try {
                            double marks = 
Double.parseDouble(input.nextLine());
                            if (marks >= 0 && marks <= 100) {
                                Module m = new Module(marks);
                                requested_student.set_module_marks(i - 1, m);
                                break;
                            } else {
                                System.out.println("Marks should be between 0 
& 100...Please enter module " + i + " marks again!");
                            }
                        } catch (Exception ex) {
                            System.out.println("Marks are not in expected 
format...Please enter module " + i + " marks again!");
                        }
                    }
                }
                requested_student.calculate_module_results();
                break;
            }
        }
    }

    //generate_system_summery method
    public static void generate_system_summery(){
        int registered_count = 0;
        for (Student s: studentsArray) {
            if (s.get_student_id().equals("empty")) {
                break;
            } else {
                registered_count++;
            }
        }
        System.out.println("No. of registered students: " +
registered_count);
        System.out.println("List of students passed all modules");
        for (Student s: studentsArray) {
            if (s.check_all_modules_passed()) {
                System.out.println("id: " + s.get_student_id() + " === name:
" + s.get_student_name());
            }
        }
    }

    //generate_students_report method
    public static void generate_students_report(){
        for (Student s: studentsArray) {
            if (!s.get_student_id().equals("empty")) {
                System.out.println("id: " + s.get_student_id() + " === name:
" + s.get_student_name());
                System.out.println("module 1: " +
s.get_modules()[0].get_module_marks() + " === module 2: " +
s.get_modules()[1].get_module_marks() + " === module 3: " +
s.get_modules()[2].get_module_marks());
                System.out.println("total: " + s.get_total_marks() + " ===
average: " + s.get_average() + " === grade: " + s.get_grade());
                System.out.println("\n");
            }
        }
    }

    //create_file method
    public static void create_file(File new_file){
        try {
            boolean file_created = new_file.createNewFile();
            if (new_file.exists()){
                System.out.println("This " + new_file.getName() + " file is
already existing");
            }
            else {
                if (file_created){
                    System.out.println(new_file.getName() + "file created
Successfully");
                }
                else {
                    System.out.println("While creating " + new_file.getName()
+ " error occurred.");
```

```
                }
            }
        }
        catch (IOException e){
            System.out.println("Error occurred while creating the file: " +
e);
        }
    }
}
```

Student.java

```
public class Student {
    private String student_id;
    private String student_name;
    public Module[] modules = new Module[3];
    private double total_marks;
    private double average;
    private String grade;

    public Student(String student_id, String student_name){
        this.student_id = student_id;
        this.student_name = student_name;
    }
    public String get_student_id(){
        return student_id;
    }
    public String get_student_name(){
        return student_name;
    }
    public void set_student_id(String student_id){
        this.student_id = student_id;
    }
    public void set_student_name(String student_name){
        this.student_name = student_name;
    }
    public String get_grade() {
        return grade;
    }
    public double get_average() {
        return average;
    }
    public double get_total_marks() {
        return total_marks;
    }

    public Module[] get_modules(){
        return modules;
    }
    public void set_module_marks(double[] module_marks){
        for (int Y = 0;Y < modules.length;Y++){
            modules[Y] = new Module(module_marks[Y]);
        }
    }
```

```java
    public void set_module_marks(int position, Module module){
        modules[position] = module;
    }
    public void calculate_module_results(){
        for (Module module : modules){
            total_marks += module.get_module_marks();
        }
        average = total_marks / modules.length;
        if (average >= 80){
            grade = "Distinction";
        }
        else if (average >= 70){
            grade = "Merit";
        }
        else if (average >= 40){
            grade = "Pass";
        }
        else {
            grade = "Fail";
        }
    }

    public boolean check_all_modules_passed() {
        boolean all_passed = true;
        for (Module module : modules){
            if (module.get_module_marks() < 40) {
                all_passed = false;
                break;
            }
        }
        return all_passed;
    }
}
```

Module.java

```java
public class Module {
    private double module_marks;

    public Module(double module_marks){
        this.module_marks = module_marks;
    }
    public double get_module_marks(){
        return module_marks;
    }
    public void set_module_marks(double module_marks){
        this.module_marks = module_marks;
    }
}
```

Task 04 – Testing

Test Case	Expected Result	Actual Result	Pass/Fail
Overall Program Test Cases			
At the start of the program execution if the (Students_Management.txt) file does not exist in the system directory, This particular file gonna be created.	Students_Management.txt file created Successfully.	Students_Management.txt file created Successfully.	Pass
At the start of the program if the (Students_Management.txt) file does exist in the system directory, Then it's gonna print file already exist message.	This Students_Management.txt file is already existing.	This Students_Management.txt file is already existing.	Pass
After the menu function got executed, System asking from the user "Please enter your option that you prefer: ", so if the user enter input value: my name	Error occurred you cannot convert string that contain data(text) into integer or a string that contain double value to integer: java.lang.NumberFormatException: For input string: "my name" Your choice is incorrect. Please enter a given option that been given above	Error occurred you cannot convert string that contain data(text) into integer or a string that contain double value to integer: java.lang.NumberFormatException: For input string: "my name" Your choice is incorrect. Please enter a given option that been given above	Pass

According to the previous scenario the user enter as the user input value: 0	**Your choice is incorrect. Please enter a given option that been given above**	**Your choice is incorrect. Please enter a given option that been given above**	**Pass**
To check whether there any student slot(seat) available in String [][] students array and Student [] studentsArray, user should select the option "1" in the menu function once this function got executed.	**Student slot 1 is available for the student registration** **"""""""""""""""""""""""""""""""""** **"""""""""""""""""""""""""""""""""** **"""""""""""""""""""""""""""""""""** **Student slot 100 is available for the student registration**	**Student slot 1 is available for the student registration** **"""""""""""""""""""""""""""""""""** **"""""""""""""""""""""""""""""""""** **"""""""""""""""""""""""""""""""""** **Student slot 100 is available for the student registration**	**Pass**
Think the user already registered a student, So now if the user select the option "1" again.	**Student slot 1 is already been occupied by someone. Please try again!** **Student slot 2 is available for the student registration** **""""""""""""""""""""""""""""""""** **""""""""""""""""""""""""""""""""** **""""""""""""""""""""""""""""""""** **Student slot 100 is available for the student registration**	**Student slot 1 is already been occupied by someone. Please try again!** **Student slot 2 is available for the student registration** **""""""""""""""""""""""""""""""""** **""""""""""""""""""""""""""""""""** **""""""""""""""""""""""""""""""""** **Student slot 100 is available for the student registration**	**Pass**
Now the user choose to select option "2" in the menu function after that it's gonna call the register_student function so now the system asking from the user "Enter the student name that you want to register: "and "Enter the student id to register a student: "	**You have successfully register the student.**	**You have successfully register the student.**	**Pass**

So now the user implement as =>name: Ramidu, id:1			
So now the user trying to enter the previously entered id:1 and different name: Shehan	This student id already existing	This student id already existing	Pass
So now the user trying to enter the previously entered student id:1 and with the same name: Ramidu	Actually this student slot 1 has been occupied by student name called Ramidu And it's student id is 1	Actually this student slot 1 has been occupied by student name called Ramidu And it's student id is 1	Pass
If the user enter student id:101 name: Shawn	You have exceed the bound that the student id can get. So please enter a valid bound value for the input.	You have exceed the bound that the student id can get. So please enter a valid bound value for the input.	Pass
So now the user entered student id: hello name: Ramidu	Error occurred you cannot convert string that contain data(text) in to integer or a string that contain double value to integer: java.lang.NumberFormatException: For input string: "Ramidu"	Error occurred you cannot convert string that contain data(text) in to integer or a string that contain double value to integer: java.lang.NumberFormatException: For input string: "Ramidu"	Pass
Think a name called "Ramidu" and id "1" existing in the arrays so if the user choose a menu function option as "3" and give to the system that the student name and the particular id that user want to delete so the user give user inputs as, name: Ramidu id: 1	You have successfully deleted the student slot 1 contain details	You have successfully deleted the student slot 1 contain details	Pass

If the user trying to delete a student name and a student id that dosen't exist in the array like, name: Sahan id: 3	You entered name Sahan and you entered student id 3 actually are invalid inputs and these are not exist in the array.	You entered name Sahan and you entered student id 3 actually are invalid inputs and these are not exist in the array.	Pass
User registering a student name: Shane id: 2	You have successfully register the student.	You have successfully register the student.	Pass
User registering another student name: Devin id: 3	You have successfully register the student.	You have successfully register the student.	Pass
Now the user choose to select option "4" in the menu function. After the user choose the option system asking "Enter the student id that belong to the student that you want to find: ", So the user enters the id as, id: 2	Student slot 2 finds the student that you looking for and his/her name is Shane Student find successfully completed	Student slot 2 finds the student that you looking for and his/her name is Shane Student find successfully completed	Pass
If the user enter an id that exceed the id bound like the user enter id as, id: 101	You just entered id called 101 is an invalid input and it is not exist in the array.	You just entered id called 101 is an invalid input and it is not exist in the array.	Pass
Now the user enter an id like that dosen't even exist in the array like, id: 4	No student with this id represent in the array.	No student with this id represent in the array.	Pass

So if the user enter user input id as invalid input like, Id: world	**Error occurred you cannot convert a string that contain data(text) to Integer or a string that contain double value to Integer: java.lang.NumberFormatException: For input string: "world"**	**Error occurred you cannot convert a string that contain data(text) to Integer or a string that contain double value to Integer: java.lang.NumberFormatException: For input string: "world"**	**Pass**
Now the user choose to select option “5” in the menu function. When selects it the data that assign in to the arrays are gonna write to the file	**You have wrote the student details successfully to the file. (This is how the data that I have written to the .txt file displaying) Shane : 2 : 0.0 : 0.0 : 0.0 Devin : 3 : 0.0 : 0.0 : 0.0**	**You have wrote the student details successfully to the file. (This is how the data that I have written to the .txt fille displaying) Shane : 2 : 0.0 : 0.0 : 0.0 Devin : 3 : 0.0 : 0.0 : 0.0**	**Pass**
Now the user choose to select option “6” in the menu function. When selects it the data that been wrote to the file gonna get those data by the system and put them back into the arrays.	**You have successfully got the student details from the file.**	**You have successfully got the student details from the file.**	**Pass**
Now the user choose to select option “7” in the menu function. When selects it the data that stored in the array are been sorted according to the student name. It’s actually print on the console like this	**[Devin, 3] [Shane, 2] [empty, empty] “”””””””””””””””” “”””””””””””””””” “”””””””””””””””” “”””””””””””””””” “”””””””””””””””” “”””””””””””””””” “”””””””””””””””” “”””””””””””””””” “”””””””””””””””””**	**[Devin, 3] [Shane, 2] [empty, empty] “”””””””””””””””” “”””””””””””””””” “”””””””””””””””” “”””””””””””””””” “”””””””””””””””” “”””””””””””””””” “”””””””””””””””” “”””””””””””””””” “””””””””””””””””**	**Pass**

Now the user choose to select option "8" in the menu function. Now the system asking from the user "Enter the student id number: " and that particular student "Enter the module 1 marks: ", "Enter the module 2 marks: " and "Enter the module 3 marks: " So think the user enters the user inputs as for student id: 2 module1 mark:60 module2 mark:70 module3 mark:80 Now if I write this data to the file it's displaying like this,	**Shane : 2 : 60.0 : 70.0 : 80.0**	**Shane : 2 : 60.0 : 70.0 : 80.0**	**Pass**
I am gonna execute the same option again to add the marks to the other person also, student id: 3 module1 mark:50 module2 mark:65 module3 mark:65 Now if I write this data again to the file it's displaying like this,	**Shane : 2 : 60.0 : 70.0 : 80.0 Devin : 3 : 50.0 : 65.0 : 65.0**	**Shane : 2 : 60.0 : 70.0 : 80.0 Devin : 3 : 50.0 : 65.0 : 65.0**	**Pass**
If the user enter the masks beyond the bound range like student id that you want to add marks to student id : 3 module 1 mark: 120	**Marks should be between 0 & 100...Please enter module 1 marks again!**	**Marks should be between 0 & 100...Please enter module 1 marks again!**	**Pass**

Now the user choose to select option "9" in the menu function. It's gonna generate the summary of the the students	**No. of registered students: 2 List of students passed all modules id: 2 === name: Shane id: 3 === name: Devin**	**No. of registered students: 2 List of students passed all modules id: 2 === name: Shane id: 3 === name: Devin**	**Pass**
Now the user choose to select option "10" in the menu function. It's gonna generate the report of the the students	**id: 2 === name: Shane module 1: 60.0 === module 2: 70.0 === module 3: 80.0 total: 210.0 === average: 70.0 === grade: Merit id: 3 === name: Devin module 1: 50.0 === module 2: 65.0 === module 3: 65.0 total: 180.0 === average: 60.0 === grade: Pass**	**id: 2 === name: Shane module 1: 60.0 === module 2: 70.0 === module 3: 80.0 total: 210.0 === average: 70.0 === grade: Merit id: 3 === name: Devin module 1: 50.0 === module 2: 65.0 === module 3: 65.0 total: 180.0 === average: 60.0 === grade: Pass**	**Pass**
At the last the user want to exit from this university intake system, so the user only have one option to do this that is selectins option "11" from the menu function. If he achieve this goal user can exit from the system.	**You have successfully exited from the university intake system.**	**You have successfully exited from the university intake system.**	**Pass**

Task 04 – Testing - Discussion

To ensure comprehensive coverage of Students Activity Management System, I selected test cases based on various scenarios the system must handle. For basic functionalities like check available seats, register student, delete student, find student, write student details to the file, load student details from the file, view details of the student, additional controls, system summary, students report and exit from the university intake system. These tests confirmed the core operations of the system. I also included edge cases to validate system robustness. For instance, I tested the system response to registering more than 100 students, ensuring it handles full capacity carefully. To handle the errors that pop up, I attempted to test register students with duplicate IDs to ensure the system prevents(handles) such duplicates. Deleting non-existent students tested how the system responds to invalid operations. Additionally, I tested invalid data entries, such as non-numeric marks or incorrectly formatted IDS, to check input validation.

The array solution is straightforward and easy to implement. Using basic array operations familiar to most programmers. Arrays provides quick access to elements due to their consumer wise memory allocation, which can be beneficial for performance in fixed-size collections. For small-scale applications, array-based implementation can be easier to read and understand due to their simplicity. However, the array solution has significant limitations. The fixed size of arrays limits scalability. Adding new features or modifying existing ones can be challenging. Leading cluttered code with low-level array management tasks. Additionally, arrays lack the flexibility of more advanced data structures. The class solutions leverages object-oriented principles, promoting modularity and encapsulation. This approach makes the codebase easier to maintain and extend. The class-based approach is more readable and intuitive. Especially those who are familiar with OOP concepts they can make the code more simple, easy to handle and self-explanatory.

Classes provides greater flexibility, making it more natural to add features like calculating grades and handling dynamic collections. Object oriented design also promotes code reuse through inheritance and polymorphisms. So, while the array solution is suitable for every basic and small-scale application. The class solutions is superior for more complex, scalable, and maintainable applications. And It will provide modern Programming Practices that making the code easier to understand and to develop maintain process.

Self-Evaluation form

Criteria		Allocated marks	Expected marks	Total
Task 1	Three marks for each option (1,2,3,4,5,6,7,8)	24	**24**	**(30)**
	Menu works correctly	6	**6**	
Task 2	Student class works correctly	14	**10**	**(30)**
	Module class works correctly	10	**8**	
	Sub menu (A and B works well)	6	**4**	
Task 3	Report – Generate a summary	7	**6**	**(20)**
	Report – Generate the complete report	10	**6**	
	Implementation of Bubble sort	3	-	
Task 4	Test case coverage and reasons	6	**5**	**(10)**
	Writeup on which version is better and why.	4	**3**	
Coding Style (Comments, indentation, style)		7	**3**	**(10)**
Complete the self-evaluation form indicating what you have accomplished to ensure appropriate feedback.		3	**3**	
Totals		100	**78**	**(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

Lecture materials………