
INFORMATICS INSTITUTE OF TECHNOLOGY

# Software Development II

Coursework Report 2023/2024

Sundaralingam Gokula Nandhan

W2082091

20233027

# Task 01 – Source Code

## 1.Available seats method

**Note :** This lines of code starts with the main class file(Program).

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

public class Program {

    //maximum places to register students
    static Student[] studentArray = new Student[100];
    //count of the registered students
    static int studentCount = 0;



    /*the following method calculates the available seats to register students
     which means the total count of remaining index in student array*/
    public static void availableSeats() {
        int availableSeats = studentArray.length - studentCount;
        System.out.println("\n" + availableSeats + " seat(s) available!");
    }
```

## 2.Register student method

```
/*this method checks the registered student count with comparing to length
     of student array.if there is a place in student array the entered details
will
     be stored.Otherwise prints error message*/
    public static void registerStudent() {
        Scanner studentDetails = new Scanner(System.in);
        try {
            if (studentCount <= studentArray.length) {
                //gets user inputs to store Name and ID
                System.out.print("Ent er student name :");
                String studentName = studentDetails.nextLine();

                System.out.print("Enter student ID (e.g.w1234567) :");
                String studentID = studentDetails.nextLine();

                //creates new student object and stores inputs to the
studentArray
```

```
                Student student = new Student(studentID, studentName);
                studentArray[studentCount] = student;
                System.out.println("Student registered successfully!");
                studentCount++;
                storeDetails();
            }
        } catch (Exception ex1) {
            System.out.println(ex1);
        }
    }
```

## 3.Delete student method

```
/*the method prompts the student id from user and checks from student array
     to delete.if it is there,student delails will be deleted from the array and
     student count decreased.otherwise prints error message*/
    public static void deleteStudent() {
        //gets input as student ID and checks
        Scanner deleteStudent = new Scanner(System.in);
        System.out.print("Enter student ID :");
        String deleteDetails = deleteStudent.nextLine();

        //to detect if the student found for deletion
        boolean found=false;

        //deletes the student details and decreases the student count
        for (int i = 0; i < studentCount; i++) {
            if (studentArray[i].getStudentID().equals(deleteDetails)) {
                studentArray[i] = null;
                found=true;

                //shift the elements to the left index
                for (int j = i; j < studentCount - 1; j++) {
                    studentArray[j] = studentArray[j + 1];
                }
                studentArray[studentCount - 1] = null;
                studentCount--;
                System.out.println("Student details deleted successfully!");
                storeDetails();
                break;
            }
        }
        //to print the error message if student not identified
        if(!found){
            System.out.print("please enter correct Student ID!");
        }
    }
```

## 4.Find student method

```
    /*this method prompts the student ID to genarate the details of the relavent
     student.if it available from student array,the details will be displayed.
     otherwise prints error message*/
    public static void findStudent() {
        //gets input as student ID to check the student is already registered
from the system
        Scanner findStudent = new Scanner(System.in);
        System.out.print("Enter student ID :");
        String findDetails = findStudent.nextLine();

        //to track the student was found
        boolean found=false;

        //prints the details of found student or not found
        for (int i = 0; i < studentCount; i++) {
            if (studentArray[i].getStudentID().equals(findDetails)) {
                System.out.println("\nStudent details as follows,");
                System.out.println("Student name :" +
studentArray[i].getStudentName());
                System.out.print("Student ID :" +
studentArray[i].getStudentID());
                found=true;
                break;
            }
        }
        //prints the error if the student not found
        if(!found){
            System.out.println("Student ID not found!");
        }
    }
```

## 5.Store details method

```
/*the method stores the details from the student array to another text file.if
     there is a problem to write the file the error message will be displayed*/
    public static void storeDetails() {
        try {
            //creates a new text file calls studentDetails.txt as file
            File writeFile = new File("StudentDetails.txt");
            boolean fileCreated = writeFile.createNewFile();
            FileWriter storedDetails = new FileWriter("StudentDetails.txt");

            for (int i = 0; i < studentCount; i++) {
```

```
                if (studentArray[i] == null) {
                    continue;
                } else {
                    //writes the student name and ID to the file
                    storedDetails.write(studentArray[i].getStudentID() + " ");
                    storedDetails.write(studentArray[i].getStudentName() + " ");

                    //get the module marks to write
                    int[]
modulemarks=studentArray[i].getModule().getModuleMarks();
                    storedDetails.write(modulemarks[0]+" ");
                    storedDetails.write(modulemarks[1]+" ");
                    storedDetails.write(+modulemarks[2]+"\n");
                }
            }
            //close the filewriter after writing
            storedDetails.close();
            System.out.println("Details stored Successfully!");

        } catch (IOException ex2) {
            System.out.println("Something went wrong in store details!");
        }
    }
```

## 6.Load details method

```
/*the method loads the stored details from the text file to the system.
     if there is any problem to load file,error message will be displayed*/
    public static void loadDetails() {
        try {
            //creates a file object to read from text file
            File readFile = new File("StudentDetails.txt");
            Scanner toReadFile = new Scanner(readFile);

            //read the text file line by line
            while (toReadFile.hasNextLine()) {
                String fileData = toReadFile.nextLine();
                //split each line into an array
                String[] dataArray = fileData.split(" ");

                //checks the data array has minimum 5 elements
                if (dataArray.length >= 5) {
                    //inisialize variables from data array to store
                    String studentID = dataArray[0];
                    String studentName = dataArray[1];
                    int moduleMark1=Integer.parseInt(dataArray[2]);
                    int moduleMark2=Integer.parseInt(dataArray[3]);
                    int moduleMark3=Integer.parseInt(dataArray[4]);
```

```
                    //stores the element to an object
                    Student loadStudent= new Student(studentID, studentName);
                    loadStudent.getModule().setModuleMarks1(moduleMark1);
                    loadStudent.getModule().setModuleMarks2(moduleMark2);
                    loadStudent.getModule().setModuleMarks3(moduleMark3);

                    //add the student details to student array
                    studentArray[studentCount]=loadStudent;
                    studentCount++;
                }
            }
            //close the scanner after readings
            toReadFile.close();
            System.out.print("Details loaded successfully!");
        } catch (IOException ex3) {
            System.out.println("Something went wrong with load file to system!");
        }
    }
```

## 7.Sort student list method

```
/*the method sort the list of students by their name's alphabetical order
     and displays the student ID and name */
    public static void viewList() {
        //checks if there are students available to display
        if (studentCount > 0) {
            System.out.println("\n------Student Details------");
            Student swapElement;

            //bubble sort algorithm for student details by the student name
alphebatical order
            for (int i = 0; i < studentCount; i++) {
                for (int j = 0; j < studentCount - i - 1; j++) {
                    //compares the studentName elements to sort
                    if (studentArray[j].getStudentName().compareTo(studentArray[j
+ 1].getStudentName()) > 0) {
                        //swaps the elements to the alphabetical ascending order
of StudentName
                        swapElement = studentArray[j];
                        studentArray[j] = studentArray[j + 1];
                        studentArray[j + 1] = swapElement;
                    } else {
                        continue;
                    }
                }
            }

            //prints the sorted list of students details
```

```
            for (int i = 0; i < studentCount; i++) {
                System.out.println("\nStudent No." + (i + 1));
                System.out.println("Student Name :" +
studentArray[i].getStudentName());
                System.out.println("Student ID :" +
studentArray[i].getStudentID());
            }
        }else{
            System.out.print("No more students to list out!");
        }
    }
```

## 8.Main menu method

```
/*method for main menu of the programme.if there are any issue with user
         input,the exception message will be displayed and iterates until exit*/
        public static void main (String[] args){
            int choice = 0;
            while (choice != 9) {
                Scanner mainMenuScanner = new Scanner(System.in);
                System.out.print("""
                        \n
                        ============================================================
                                    STUDENT ACTIVITY MANAGEMENT SYSTEM
                        ============================================================


                       Numbers of the choices as follows,

                        1 - Check available seats
                        2 - Register student(With student ID)
                        3 - Delete student
                        4 - Find student(With student ID)
                        5 - Store student details into a file
                        6 - Load student details from the file to the system
                        7 - View the list of students based on their names
                        8 - Additional options
                        9 - Exit

                       Enter number of the  choice :""");

                try {
                    choice = mainMenuScanner.nextInt();

                    switch (choice) {
                        case 1:
                            availableSeats();
                            break;
                        case 2:
                            registerStudent();
                            break;
```

```
                        case 3:
                            deleteStudent();
                            break;
                        case 4:
                            findStudent();
                            break;
                        case 5:
                            storeDetails();
                            break;
                        case 6:
                            loadDetails();
                            break;
                        case 7:
                            viewList();
                            break;
                        case 8:
                            additionalOptions();
                            break;
                        case 9:
                            System.out.println("The programme is being
terminated!");
                            break;
                        default:
                            System.out.println("Please enter a valid choice!");
                    }
                } catch (Exception ex4) {
                    System.out.print("Invalid input.please try again!");
                }
            }
        }
    }
```

# Task 02 – Source Code

# 1.Class files

## 1.1.Student Class

```
public class Student {
    private String studentName;
    private String studentID;
    private Module module;
```

```
    public Student(String studentID,String studentName){
        this.studentName=studentName;
        this.studentID=studentID;
        this.module=new Module();
    }


    public String getStudentName() {
        return studentName;
    }

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

    public String getStudentID() {
        return studentID;
    }

    public Module getModule() {
        return module;
    }
}
```

## 1.2. Module Class

```
public class Module {

    private int[] moduleMarks=new int[3];

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

    public void setModuleMarks1(int Marks1) {
        this.moduleMarks[0] = Marks1;
    }
    public void setModuleMarks2(int Marks2) {
        this.moduleMarks[1] = Marks2;
    }
    public void setModuleMarks3(int Marks3) {
        this.moduleMarks[2] = Marks3;
    }

    public int total(){
        int total=moduleMarks[0]+moduleMarks[1]+moduleMarks[2];
        return total;
    }
```

```
    public float average(){
        int totalMarks=total();
        float average=(float)totalMarks/3;
        return average;
    }

    public String grade(){
        float average=average();

        if(average>=80){
            return "Distinction";
        }else if(average>=70){
            return "Merit";
        }else if(average>=40){
            return "Pass";
        }else{
            return "Fail";
        }
    }

}
```

# 2.Methods

# 2.1.Sub menu method

```
//submenu for another list of choices to generate summary of the system and
student progress
    private static void additionalOptions() {
        //variable for subchoice selection
        String subChoice;

        //iterates until the choice equals to "e"
        do{
            Scanner subMenuScanner = new Scanner(System.in);
            System.out.print("""
                    \n
                    \n
                     ******** ADDITIONAL OPTIONS ********

                    Choices as follows,

                    a - Recorrect name
                    b - Add module marks for Students
                    c - Summary of Student Activity Management System
                    d - Progress Report of the students
                    e - Main Menu

                    Enter choice :""");
```

```
            //stores chosen choice with the lowercase letter
            subChoice = subMenuScanner.nextLine().toLowerCase();

            switch (subChoice) {
                case "a":
                    addStudent();
                    break;
                case "b":
                    addModuleMarks();
                    break;
                case "c":
                    systemSummary();
                    break;
                case "d":
                    studentSummary();
                    break;
                case "e":
                    break;
                default:
                    System.out.print("Please enter a valid choice!");
            }
        }while(!(subChoice.equals("e")));

    }
```

## 2.2.Add student method

```
/*the following method can update the student name when the registered name
incorrect.
    it takes student id as input.if the id matched user can update the
name.otherwise system
    prints the error message*/
    public static void addStudent() {

        Scanner toCheck = new Scanner(System.in);

        System.out.print("Enter student ID :");
        String checkID = toCheck.nextLine();
        boolean found=false;

        //checks student id of stunet with iteration
        for (int i = 0; i < studentCount; i++) {
            if (studentArray[i].getStudentID().equals(checkID)) {
                System.out.print("Enter student name :");
                String updatedName = toCheck.nextLine();
                studentArray[i].setStudentName(updatedName);
                found=true;
                System.out.print("Student name entered successfully!");
                storeDetails();
```

```
            }
        }
        if(!found){
            System.out.print("Student not found!");
        }

    }
```

## 2.3.Add module marks method

```
/*the method get user input as student id.if the student id is available from
    student array the marks for every three modules will be prompted from user
and stored*/
    public static void addModuleMarks(){
        try {
            Scanner addMarks = new Scanner(System.in);
            boolean found=false;

            System.out.print("Enter student ID :");
            String ID = addMarks.nextLine();


            //iteration for each student from array
            for (int i = 0; i < studentCount; i++) {
                if (studentArray[i].getStudentID().equals(ID)) {

                    //initialize varibales for marks validation
                    int module1Marks=101;
                    int module2Marks=101;
                    int module3Marks=101;

                    //validation of marks for module 1
                    while (0 > module1Marks || 100 < module1Marks) {
                        System.out.print("Enter marks for Module-1 :");
                        module1Marks = addMarks.nextInt();

                        if(module1Marks<0 || module1Marks>100){
                            System.out.println("\nInvalid marks!The marks should
be between 0 to 100!");
                        }
                    }
                    //stores the module 1 marks student array
                    studentArray[i].getModule().setModuleMarks1(module1Marks);


                    //validation of marks for module 2
                    while (0 > module2Marks || 100 < module2Marks) {
                        System.out.print("Enter marks for Module-2 :");
```

```
                        module2Marks = addMarks.nextInt();

                        if(module2Marks<0 || module2Marks>100){
                            System.out.println("\nInvalid marks!The marks should
be between 0 to 100!");
                        }
                    }
                    //stores the module 2 marks to student array
                    studentArray[i].getModule().setModuleMarks2(module2Marks);

                    //validation of marks for module 3
                    while (0 > module3Marks || 100 < module3Marks) {
                        System.out.print("Enter marks for Module-3 :");
                        module3Marks = addMarks.nextInt();

                        if(module3Marks<0 || module3Marks>100){
                            System.out.println("\nInvalid marks!The marks should
be between 0 to 100!");
                        }
                    }
                    //stores the module 3 marks to student array
                    studentArray[i].getModule().setModuleMarks3(module3Marks);

                    found=true;
                    storeDetails();
                    break;
                }
            }
            if(!found){
                System.out.print("Invalid Student ID!");
            }
        }catch (Exception ex5){
            System.out.print("Error!");
        }
    }
```

# Task 03 – Source Code

## 1.Summary of the system method

```
/*this method provides the complete progress report of the students
     and the system summary*/
    public static void systemSummary(){
        //print the summary of the system
```

```
        System.out.println("\n");
        System.out.println("\nSummary of the system");
        System.out.println("-----------------------------");

        System.out.println("    The system can store 100 students at one time.");
        System.out.println("    "+studentCount+" Student(s) already registered to
the system.");


        //initialize counters for module summary
        int count1=0;
        int count2=0;
        int count3=0;
        int count4=0;

        //iteration for every registered  students
        for(int i=0;i<studentCount;i++){
            int[] marks=studentArray[i].getModule().getModuleMarks();
            boolean validMarks=false;



            //count students scoring above 40 in each and every modules
            if(marks[0]>40){
                count1++;
            }
            if(marks[1]>40){
                count2++;
            }
            if(marks[2]>40){
                count3++;
            }
            if(marks[0]>40 && marks[1]>40 && marks[2]>40){
                count4++;
            }
        }

        //prints the summary of module marks
        System.out.println("\nSummary of module marks");
        System.out.println("-----------------------------");

        System.out.print("    Total count of students scored above 40 marks in
module 1     : ");
        System.out.print(count1);

        System.out.print("\n    Total count of students scored above 40 marks in
module 2     : ");
        System.out.print(count2);

        System.out.print("\n    Total count of students scored above 40 marks in
module 3     : ");
        System.out.print(count3);
```

```
        System.out.print("\n    Total count of students scored above 40 marks in
all modules  : ");
        System.out.print(count4);
    }
```

## 2.Student progress report method

```
/*the method gets the student details from student array and displays
     the student summary.if there are no students registered prints
     the relevant message*/
    public static void studentSummary(){
        if (studentCount > 0) {
            System.out.println("\n");
            System.out.println("\n      Student Progress Report");
            System.out.print("................................");

            //intialize temporary variable to swap
            Student swapAverage;

            //bubble sort algorithm for student details by the average discending
order
            for (int i = 0; i < studentCount - 1; i++) {
                for (int j = 0; j < studentCount - i - 1; j++) {
                    if (studentArray[j].getModule().average() < studentArray[j +
1].getModule().average()) {
                        //swaps the elements to the discending order of the
average
                        swapAverage = studentArray[j];
                        studentArray[j] = studentArray[j + 1];
                        studentArray[j + 1] = swapAverage;
                    }
                }
            }

            //prints the sorted list of students details
            for (int i = 0; i < studentCount; i++) {
                System.out.println("");
                System.out.println("\nStudent No." + (i + 1));

                //stores the student detail by initializing variables and array
                int total=studentArray[i].getModule().total();
                float average=studentArray[i].getModule().average();
                String grade=studentArray[i].getModule().grade();
                int[] moduleMarks=studentArray[i].getModule().getModuleMarks();

                //prints complete student progress report for each student
```

```
            System.out.print("Student Name        :");
            System.out.println(studentArray[i].getStudentName());

            System.out.print("Student ID          :");
            System.out.println(studentArray[i].getStudentID());

            System.out.print("Marks of Module 01  :");
            System.out.println(moduleMarks[0]);

            System.out.print("Marks of Module 02  :");
            System.out.println(moduleMarks[1]);

            System.out.print("Marks of Module 03  :");
            System.out.println(moduleMarks[2]);

            System.out.print("Total               :");
            System.out.println(total);

            System.out.print("Average             :");
            System.out.println(average);

            System.out.print("Grade               :");
            System.out.println(grade);
        }
    }
    else{
        //print the message for the empty stuent details
        System.out.print("No students to genarate progress report!");
    }
}
```

# Task 04 – Testing

| Test Case | Expected Result | Actual Result | Pass/Fail |
|---|---|---|---|
| Input invalid choice like alphabetical characters. | Press 'f'<br>Displays "Invalid input. Please try again!. Asks for input | Displays "Invalid input. Please try again! " and asks for the input again. | Pass |
| Check available seats before register students | Enter '1'<br>Displays "100 seats(s) available. | Displays expected result | Pass |
| Register 5 students to the system | Enter '2' 5 times after registering every students.<br>Asks student Name and ID to register. And prints the completion message. | Displays expected output | Pass |
| Check available seats after registering 5 students | Print "95 seat(s) available!". | Displays expected output | Pass |
| Deleting a student details from system with unregistered student | Enter"3"<br>Asks for student ID. Input unregistered student ID("w2345678").<br>Print the error message. | Displays "Please enter correct student ID". Displayed expected output | Pass |
| Deleting a student details from system with registered student | Enter"3"<br>Asks for student ID. Input registered student ID.<br>Prints the deletion success message.<br>Save the data | Displays "Student details deleted successfully!" and "Details stored successfully!" | Pass |
| Check student count after deleting a student | Prints "96 seat(s) available!" | Displayed expected message. | Pass |
| Finding a student details with unregistered Student ID | Press "4"<br>Asks for student ID. If input unregistered student ID, prints error message. | Displayed "Student ID not found!" | Pass |
| Finding a student details with registered Student ID | Enter "4"<br>If entered registered student ID prints the Student details | Displayed the student details for the relevant input | Pass |
| Store the system details to a text file | Enter "5"<br>Stores the whole student details to a newly created text file as "StudentDetails.txt", And Prints the message. | Did expected thing and displayed "Details stored successfully!" | Pass |
| View the list of all registered students | Enter "7"<br>Prints all the details of students(Name, ID) according to the alphabetical ascending order of their names. | Displayed expected output | Pass |
| Exit from the program | Enter "9"<br>Prints the relevant message and terminate the Program. | Displays "The program is being terminated" and did expected result | Pass |
| Checks the student count after rerunning the program. | Displays "100 seat(s) available!" | Displayed expected message. | Pass |
| Load the student details from the text file to system. | Enter "6"<br>Stores the saved student details from the text file and prints the success message. | Did expected thing and displayed "Details loaded successfully!" | Pass |

| | | | |
|---|---|---|---|
| Check the available seats | Displays “96 seat(s) available!” | Displayed expected message | Pass |
| To enter to the additional options menu | Enter “8”<br>Displays the additional options submenu | Displayed expected list | Pass |
| Input invalid choice(numbers) | Displays the error message | Displayed “Please enter a valid choice!” | Pass |
| Recorrect the name with unregistered student ID | Press “a”<br>If the student id mismatched, Displays “Student not found!” | Displayed the relevant message. | Pass |
| Recorrect the name with registered student ID | Ask for the updated name, prints the relevant message and stores the name | Did expected result | Pass |
| Add module marks for the students with invalid student ID | Press “b”<br>If entered the invalid student ID, The error message will displays. | Printed “Invalid student ID” | Pass |
| Enter a invalid marks for module 1 ( 0<=marks <=100.The marks system is same for all the three modules) | Enter marks as “-4,101”<br>Prints the error message. | Printed “Invalid marks! The marks should be between 0 to 100! ” | Pass |
| Enter a valid marks to module 1. | Enter marks as “56”<br>Asks input for the module 2 marks | Did expected result | Pass |
| Enter an invalid marks to module 2. | Enter marks as “-1, 300”<br>Prints the error message. | Printed the error message. | Pass |
| Enter a valid marks to module 2. | Enter marks as “78”<br>Asks input for the module 3 marks | Did expected result | Pass |
| Enter an invalid marks to module 3 | Enter marks as “-1,101”<br>Prints the error message. | Printed the relevant error message | Pass |
| Enter a valid marks to module 3 | Enter marks as “24”<br>Prints the relevant message. | Printed “Details stored successfully!” | Pass |
| Enter marks to all registered students | Prints the completion message for every inputs and stores the marks | Did expected result | Pass |
| Generate the summary of student management system | Gives the summary of the system and module marks | Did expected output | Pass |
| Generate the Student Progress Report | Enter “d”<br>Gives the progress reports of the registered student from the system by the average descending order | Did expected output | Pass |
| To go to the main menu | Press “e”<br>Submenu program terminates, Prints the message and will go to the main menu. | Did expected output | Pass |
| Load the file by rerun the program. | Stores the student details and marks to the system. Prints the success message. | Did expected output. | Pass |
| Check the student progress report | Print all the saved details with marks | Printed the details. | Pass |
| Register a student after loading a file | Student registered without marks. Prints the success message | Did expected output.<br>Printed the success message | Pass |
| Check the new student details | Details will print with the past student details. | Printed expected way of method. | Pass |

# Task 04 – Testing - Discussion

In my test case, I have considered these factors for the test. Input validation. Basic functional operations, file operations, such as loading and writing the files from another text file, sorting algorithm functionality (Bubble sort), deletion of student records, finding student records, usage of arrays, and usage of classes and objects, were divided into the test cases for the below types of requirements.

1. The new implementation of student details without marks.
2. Updating the student details already registered without marks.
3. The updating of marks for the already registered students.
4. The updating of new student data for the registered student details with marks.

I implemented the tests line by line according to the specifications of the student activity management system. Totally, I have 33 types of test cases for the above types of requirements. All the test cases passed from the student activity management system.

According to the coursework work specification, the student management system basically covers the array solution and the class solution. In my opinion, the array solution is Managing parallel arrays for student IDs, names, and marks can lead to errors and difficulties in understanding the overall logic. but the class solution makes it easy to identify the logic of the program. For example, in the student management system, there were separate classes for the students and modules. However, arrays are disorganized and store every element. The class solution has proper readability and maintainability of the codes. In conclusion, array methods can solve the basic programming projects. If we want to handle large amounts of data, like coursework, the class solution is better than the array solution. For the reusability of the student management system, a class solution is the better way. Overall, the class solution is generally better for this type of application filled with complex tasks.

# Self-Evaluation form

The student activity manager has several types of tasks covering the basics of the Java programming language. Task one discovered the Java array, conditions, operators, and loops.

File handling, error handling, and sorting algorithms. Task 01 is fully implemented using these basics and working. I have a better knowledge of these basics. So, I have a few days to implement my task 01 code .I am expecting a full marks for task 01.

The second task completely works with the classes and objects with other basics from the task 01 . I had a lack of knowledge of Java object-oriented programming. I had more time to spend on task 2. I studied object-oriented programming in a lot of ways. However, I have fully implemented task 02 and it is working .I am expecting full marks for task 02.

Task three covers the sorting methods, operations, loops and using classes and objects. I gained better knowledge after working on task 02. It makes it easier for me to implement the task 03 codes. Task 03 is also fully implemented and working. I am expecting full marks for task 03.

The test cases were implemented with my understanding of the coursework. And also I have commented single and multi line comment. I have a average knowledge with comment and providing test cases. I am expecting 4 marks for the coding style.

| 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 | **6** | |
| **Task 2** Student class works correctly | 14 | **14** | **(30)** |
| Module class works correctly | 10 | **10** | |
| Sub menu (A and B works well) | 6 | **6** | |
| **Task 3** Report – Generate a summary | 7 | **7** | **(20)** |
| Report – Generate the complete report | 10 | **10** | |
| 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 | **3** | |
| Coding Style (Comments, indentation, style) | 7 | **4** | **(10)** |
| Complete the self-evaluation form indicating what you have accomplished to ensure appropriate feedback. | 3 | **2** | |
| **Totals** | 100 | **93** | **(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

1. Weekly lecture materials and tutorials

2. Websites – w3schools, geeksforgeeks

3. Youtube videos – error makes clever academy , brocade, code.io