**Assignment Title**

Name of Student: Student ID

Affiliation

Course Code: Course Name   
Instructor Name

Due Date

Contents

[Assessment 3](#_Toc147007659)

[Function 3 – Pseudo-Code 3](#_Toc147007660)

[Function 4 - Pseudo-Code 3](#_Toc147007661)

[StudentMarksProcessor Class 4](#_Toc147007662)

[Student Class 10](#_Toc147007663)

[Output Code 13](#_Toc147007664)

[Read and Display Student Marks 13](#_Toc147007665)

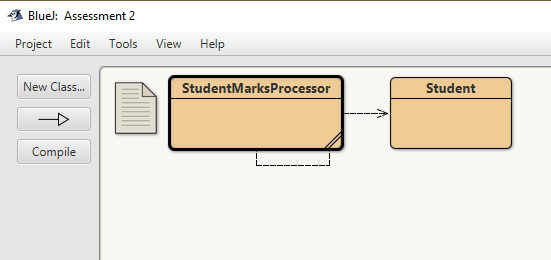
[Calculate Total Marks 13](#_Toc147007666)

[Print Student Below a threshold 14](#_Toc147007667)

[Print top 5 and bottom 5 Students 15](#_Toc147007668)

[Exit 16](#_Toc147007669)

# Assessment



# Function 3 – Pseudo-Code

Algorithm 3 - Print Students Below Threshold

Input: threshold (an integer)

1. FOR each student in the list of students:

2. Calculate the total marks for the student by summing a1, a2, and a3.

3. IF total marks < threshold THEN

4. Print "Last Name: " + student.getLastName()

5. Print "First Name: " + student.getFirstName()

6. Print "Student ID: " + student.getStudentID()

7. Print "Total Marks: " + total marks

8. Print a blank line

9. END IF

10. END FOR

# Function 4 - Pseudo-Code

Algorithm 4 - Print Top 5 and Bottom 5 Students

1. Sort the list of students in descending order based on their total marks.

2. Calculate the total number of students in the list (size).

3. Determine the count of top students and bottom students to display (topCount and bottomCount).

4. Print "Top 5 Students:".

5. FOR i from 0 to topCount - 1:

6. Get the student at index i from the sorted list.

7. Print student's first name, last name, and total marks.

8. END FOR

9. Print a blank line.

10. Print "Bottom 5 Students:".

11. FOR i from size - 1 down to size - bottomCount:

12. Get the student at index i from the sorted list.

13. Print student's first name, last name, and total marks.

14. END FOR

# StudentMarksProcessor Class

/\*\*

\* Write a description of class StudentMarks here.

\*

\* @author -Jashanpreet Kaur

\* @studentID -

\*/

import java.io.\*;

import java.util.\*;

public class StudentMarksProcessor {

private static List<Student> students = new ArrayList<>();

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

BufferedReader br = new BufferedReader(new FileReader("C:\\Users\\sony\\OneDrive\\Desktop\\prog5001\_students\_grade\_2022.txt"));

String line;

while ((line = br.readLine()) != null) {

// Skip lines that are comments or empty lines

if (line.startsWith("Unit") || line.startsWith("Last Name")|| line.isEmpty()) {

continue;

}

String[] parts = line.split("\t");

String lastName = parts[0];

String firstName = parts[1];

String studentID = parts[2];

String a1 = parts.length > 3 ? parts[3] : "N/A";

String a2 = parts.length > 4 ? parts[4] : "N/A";

String a3 = parts.length > 5 ? parts[5] : "N/A";

// Create a Student object and add it to the list

Student student = new Student(lastName, firstName, studentID, a1, a2, a3);

students.add(student);

}

br.close();

} catch (IOException e) {

System.err.println("Error reading the file.");

}

while (true) {

System.out.println("Choose an option:");

System.out.println("1. Read and display student marks");

System.out.println("2. Calculate total marks");

System.out.println("3. Print students below a threshold");

System.out.println("4. Print top 5 and bottom 5 students");

System.out.println("5. Exit");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume the newline character

switch (choice) {

case 1:

readAndDisplayStudentMarks();

break;

case 2:

calculateTotalMarks();

break;

case 3:

System.out.println("Enter the threshold: ");

int threshold = scanner.nextInt();

scanner.nextLine(); // Consume the newline character

printStudentsBelowThreshold(threshold);

break;

case 4:

printTopAndBottomStudents();

break;

case 5:

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

System.exit(0);

default:

System.out.println("Invalid choice. Please select a valid option.");

}

}

}

// Function to read and display student marks

public static void readAndDisplayStudentMarks() {

for (Student student : students) {

System.out.println("Last Name: " + student.getLastName());

System.out.println("First Name: " + student.getFirstName());

System.out.println("Student ID: " +student.getStudentID());

System.out.println("A1: " + student.getA1());

System.out.println("A2: " + student.getA2());

System.out.println("A3: " + student.getA3());

System.out.println();

}

}

// Function to calculate total marks

public static void calculateTotalMarks() {

for (Student student : students) {

double total = 0.0; // Initialize total to 0.0

// Parse and sum A1, A2, and A3 only if they are valid numbers

if (student.getA1().isEmpty()||student.getA1().equals("N/A"))

total += 0;

else

total += Double.parseDouble(student.getA1());

if (student.getA2().isEmpty()||student.getA2().equals("N/A"))

total += 0;

else

total += Double.parseDouble(student.getA2());

if (student.getA3().isEmpty()||student.getA3().equals("N/A"))

total += 0;

else

total += Double.parseDouble(student.getA3());

System.out.print("Student ID: " + student.getStudentID()+" ");

System.out.println("Total Marks: " + total);

System.out.println();

}

}

// Function to print students below a threshold

public static void printStudentsBelowThreshold(int threshold) {

for (Student student : students) {

double total = Double.valueOf(student.getA1()) + Double.valueOf(student.getA2()) + Double.valueOf(student.getA3());

if (total < threshold) {

System.out.println("Last Name: " + student.getLastName());

System.out.println("First Name: " + student.getFirstName());

System.out.println("Student ID: " + student.getStudentID());

System.out.println("Total Marks: " + total);

System.out.println();

}

}

}

// Function to print top 5 and bottom 5 students

public static void printTopAndBottomStudents() {

students.sort(Comparator.comparingDouble(Student::getTotalMarks).reversed()); // Sort in descending order

int size = students.size();

int topCount = Math.min(5, size);

int bottomCount = Math.min(5, size);

System.out.println("Top 5 Students:");

for (int i = 0; i < topCount; i++) {

Student student = students.get(i);

System.out.println(student.getFirstName() + " " + student.getLastName() + " - Total Marks: " + student.getTotalMarks());

}

System.out.println("\nBottom 5 Students:");

for (int i = size - 1; i >= size - bottomCount; i--) {

Student student = students.get(i);

System.out.println(student.getFirstName() + " " + student.getLastName() + " - Total Marks: " + student.getTotalMarks());

}

}

}

# Student Class

/\*\*

\* Write a description of class Student here.

\*

\* @author JashanPreet Kaur

\* @version (a version number or a date)

\*/

class Student {

private String lastName;

private String firstName;

private String studentID;

private String a1;

private String a2;

private String a3;

public Student(String lastName, String firstName, String studentID, String a1, String a2, String a3) {

this.lastName = lastName;

this.firstName = firstName;

this.studentID = studentID;

// Correctly check the parameters passed to the constructor

if (a1 == null || a1.isEmpty() || a1.equals("N/A"))

this.a1 = "0";

else

this.a1 = a1;

if (a2 == null || a2.isEmpty() || a2.equals("N/A"))

this.a2 = "0";

else

this.a2 = a2;

if (a3 == null || a3.isEmpty() || a3.equals("N/A"))

this.a3 = "0";

else

this.a3 = a3;

}

public String getLastName() {

return lastName;

}

public String getFirstName() {

return firstName;

}

public String getStudentID() {

return studentID;

}

public String getA1() {

return a1;

}

public String getA2() {

return a2;

}

public String getA3() {

return a3;

}

public double getTotalMarks() {

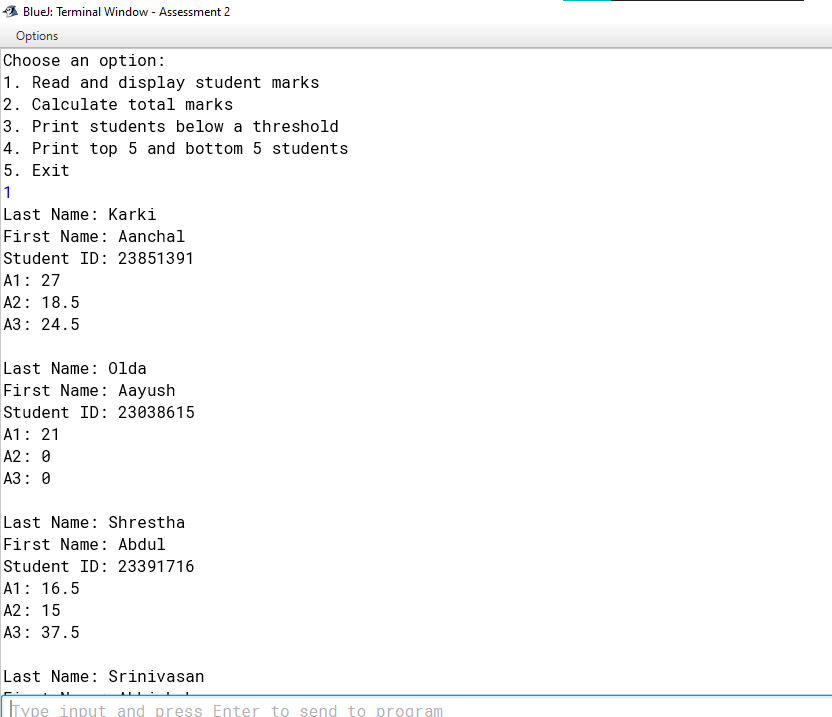
return Double.parseDouble(a1) + Double.parseDouble(a2) + Double.parseDouble(a3);

}

}

# Output Code

## **Read and Display Student Marks**



## **Calculate Total Marks**

A screenshot of a computer

Description automatically generated

## **Print Student Below a threshold**

A screenshot of a computer

Description automatically generated

## **Print top 5 and bottom 5 Students**

A screenshot of a computer

Description automatically generated

## **Exit**

A screenshot of a computer program

Description automatically generated