

**NED UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

Formal Methods in Software Engineering

*Formal Specification Document for*

**“****Student Attendance Tracking System”**

**Group**

* Bakhtiar Ahmed (SE-21029)
* Fawad Tariq (SE-21050)

Table of Contents

1. [Problem Statement: Student Attendance Tracking System 2](#_bookmark0)
2. [4 + 1 Architectural View 3](#_bookmark2)
   1. [Logical View 3](#_bookmark3)
   2. [Process View 4](#_bookmark4)
   3. [Physical View 4](#_bookmark5)
   4. [Development View 5](#_bookmark6)

[2.5. +1 Scenarios 5](#_bookmark7)

1. The VDM specification of AttencdanceTracker System…………………………………….6
2. [Java Implementation 8](#_bookmark8)
3. [Testing Class 12](#_bookmark9)

**Student Attendance Tracking System**

# Problem Statement: Student Attendance Tracking System

**Background**: A student attendance tracking system is a critical component in educational institutions, facilitating efficient monitoring and management of students' class attendance. It's crucial for academic progress tracking and institutional compliance.

**Requirements:** Design and implement software for a student attendance tracking system. The system should:

* Allow the addition or update of attendance records for individual students.
* Display all student records, including their attendance percentages, classes attended, and classes held.
* Ensure accuracy in attendance percentage calculation and record management.

### Specifications:

### The system should store attendance records for each student, including their name, roll number (unique identifier), total classes held, and total classes attended.

### Calculate the attendance percentage based on the classes attended and classes held.

### Allow the addition or updating of attendance records by specifying the student's name, roll number, total classes held, and total classes attended.

### Display all student records, including their attendance percentages, classes attended, and classes held.

### Implement proper error handling for invalid inputs and ensure data integrity.

### Functionalities:

### 1) Add or Update Student Record:

### - Input: Student name, roll number, total classes held, total classes attended.

### - Result: The system updates or adds the student's attendance record.

### 

### 2) Display Student Attendance Records:

### - Input: Roll number of the student.

### - Output: Individual student record with attendance details.

### 3) Calculate Percentage:

### -Input: Number of classes attended and Number of classes held.

### - Output: Percentage of student’s attendance.

### Constraints:

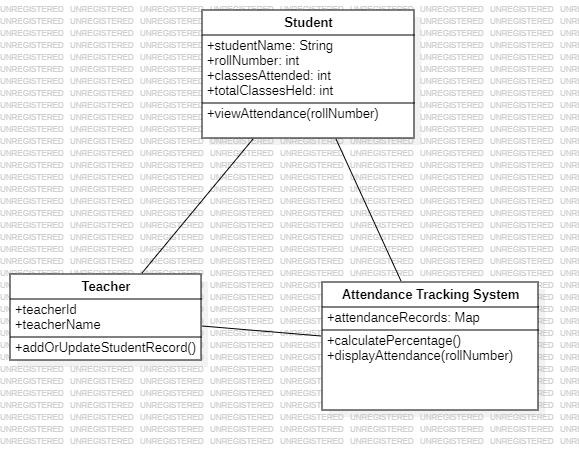
### • Ensure uniqueness of roll numbers to prevent conflicts in student records.

### • Validate inputs to maintain accurate attendance data.

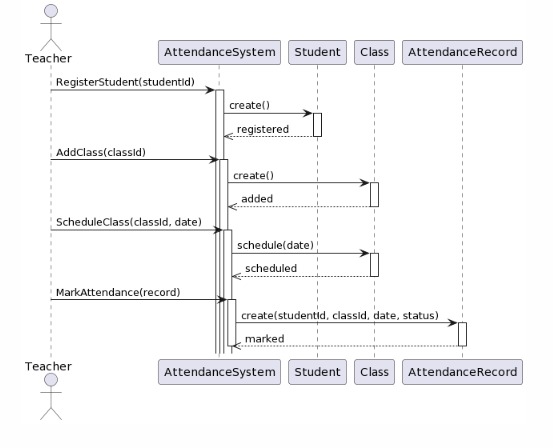
### • Implement error handling for scenarios like incorrect roll numbers or invalid attendance values.

1. **4+1 View Architecture:**

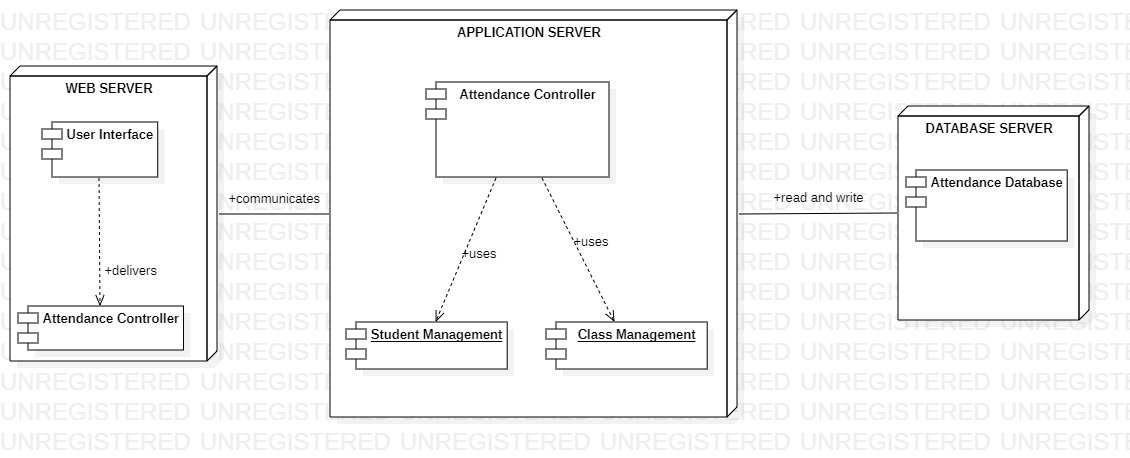
**2.1. Logical View:**

****

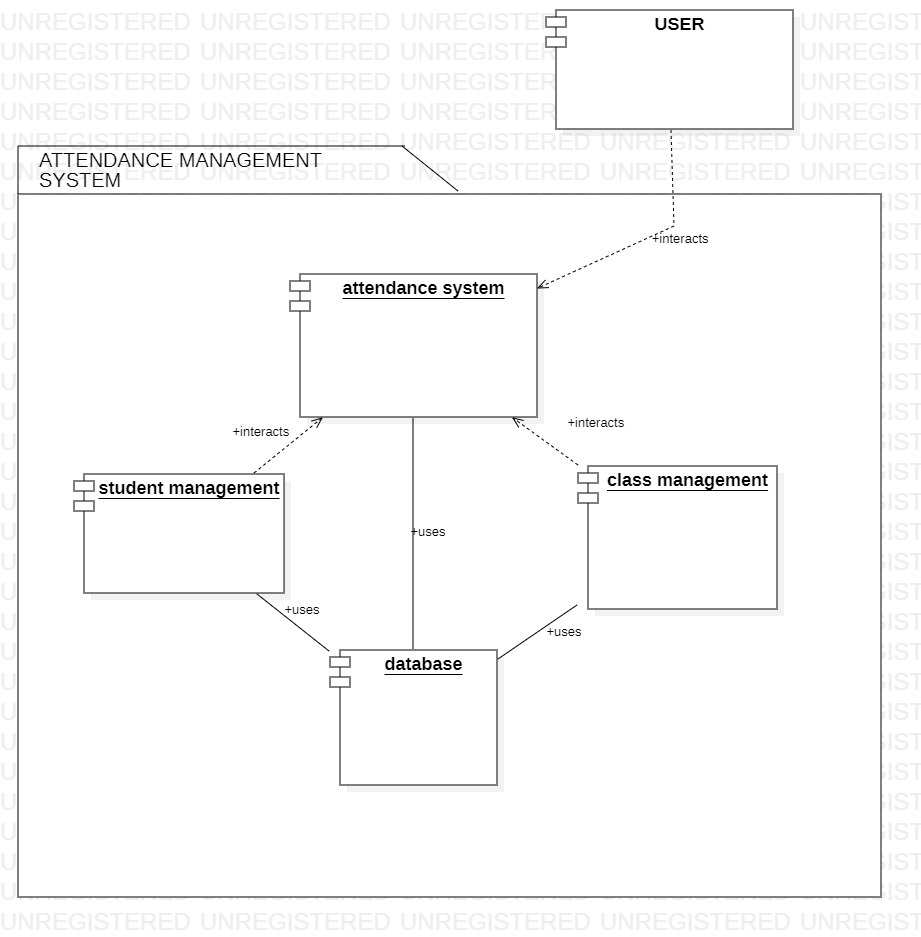
**2.2. Process View:**

****

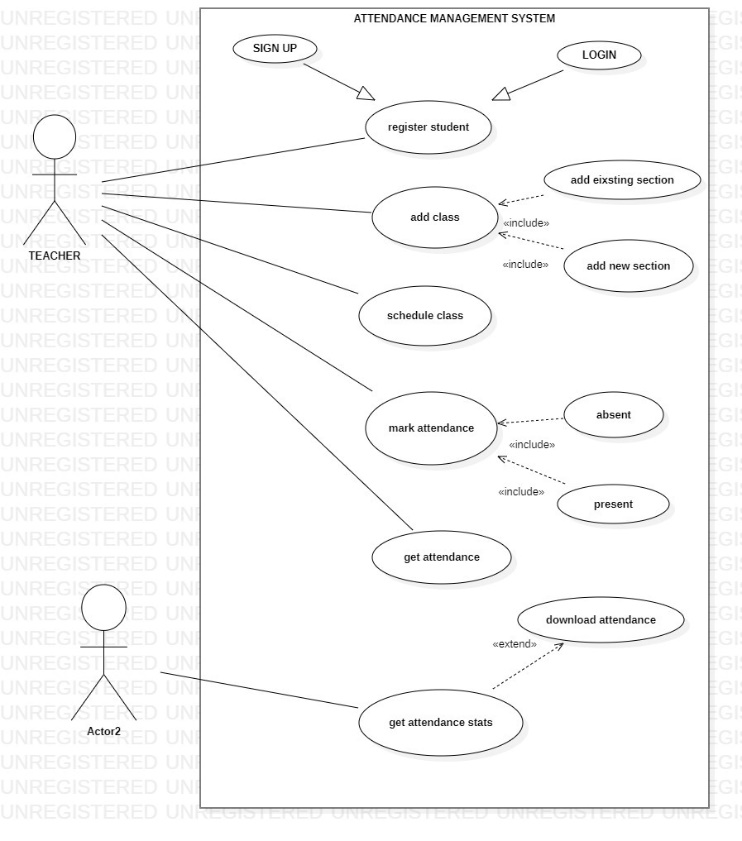
**2.3. Physical View:**

****

**2.4. Development View:**

****

**2.5. +1 Scenario:**

****

1. **T****he VDM specification of *AttencdanceTracker* System**

**types**

Name = seq of char;

**state** *AttendanceTracker* **of** studentName : Name rollNumber : ℤ

totalClassesHeld : **ℤ**

classesAttended : **ℤ**

-- Total Classes Held and Classes Attended must not be less than zero and Classes Held must be greater than or equal to Classes Attended

**inv mk**-AttendanceTracker (t, c) (inRange(t) ˅ = **nil**) ˄ (inRange(c) ˅ = **nil**) ˄ (c ≤ t)

-- totalClassesHeld and classesAttended are undefined when the system is initialized

**init mk**-AttendanceTracker (t, c) t = **nil** ˄ c = **nil**

**end**

**functions**

*inRange:*(val: totalClassesHeld ) result: B

**pre** True

**post** result ⇔ 0 ≤ val ;

*calculatePercentage:*(val1: totalClassesHeld, val2: ClassesAttended ) result: B

**pre** inRange(val1) ˄ inRange(val2)

**post** result ⇔ (val2/ val1) \* 100;

**operations**

addOrUpdateStudentRecord: (stName : Name, rollNo : **ℤ**, classHeld: **ℤ**, classAttended: **ℤ**)

**ext wr** studentName : Name

rollNumber: **ℤ**

totalClassesHeld: **ℤ**

classesAttended : **ℤ**

**pre** inRange(classHeld) ˄ inRange(classAttended)

**post** studentName = stName ˄ RollNumber = rollNo ˄ totalClassesHeld = classHeld ˄ classesAttended = classAttended;

getAttendanceRecord(rollNo: **ℤ**)

**ext rd** studentName : Name

RollNumber: **ℤ**

totalClassesHeld: **ℤ**

classesAttended: **ℤ**

**pre**  rollNo = rollNumber ˄ rollNumer != **nil**

**post**  rollNumber ˄ studentName ˄ totalClassesHeld ˄ classesAttended ˄

calculatePercentage(totalClassesHeld, classesAttended)

getStudentName: (stName : Name, rollNo : **ℤ**)

**ext rd** studentName : Name

**pre** studentName != **nil** ˄ rollNumber != **nil** ˄ inRange(rollNumber)

**post** stName = studentName

getClassesHeld: (classHeld: **ℤ,** rollNo : **ℤ** )

**ext rd** totalClassesHeld: **ℤ**

**pre** inRange(totalClassesHeld) ˄ totalClassesHeld != nil ˄ (rollNumber != **nil** ˄ inRange(rollNumber))

**post** classHeld = totalClassesHeld

getClassesAttended: (classAttended: **ℤ,** rollNo : **ℤ**)

**ext rd** classesAttended : **ℤ**

**pre** inRange(classHeld) ˄ inRange(classAttended) ˄ (rollNumber != **nil** ˄ inRange(rollNumber))

**post** classesAttended = classAttended

# Java Implementation

Code:

import java.util.\*;

class Student {

private String studentName;

private int rollNumber;

private int totalClassesHeld;

private int totalClassesAttended;

public Student(String studentName, int rollNumber, int totalClassesHeld, int totalClassesAttended) {

this.studentName = studentName;

this.rollNumber = rollNumber;

this.totalClassesHeld = totalClassesHeld;

this.totalClassesAttended = totalClassesAttended;

}

public void updateAttendance(int totalClassesHeld, int totalClassesAttended) {

this.totalClassesHeld = totalClassesHeld;

this.totalClassesAttended = totalClassesAttended;

}

public String getStudentName() {

return studentName;

}

public int getRollNumber() {

return rollNumber;

}

public int getTotalClassesHeld() {

return totalClassesHeld;

}

public int getTotalClassesAttended() {

return totalClassesAttended;

}

public double getAttendancePercentage() {

double percentage = ((double) totalClassesAttended / totalClassesHeld \* 100);

return Math.round(percentage \* 100.0) / 100.0;

}

}

class StudentAttendanceSystem {

private Map<Integer, Student> attendanceRecords; // Map<RollNumber, Student>

public StudentAttendanceSystem() {

attendanceRecords = new HashMap<>();

}

public void addOrUpdateStudentRecord(String studentName, int rollNumber, int totalClassesHeld, int totalClassesAttended) {

if (attendanceRecords.containsKey(rollNumber)) {

Student existingStudent = attendanceRecords.get(rollNumber);

if (!existingStudent.getStudentName().equals(studentName)) {

System.out.println("Error: A record with the same roll number but different name already exists.");

return;

}

existingStudent.updateAttendance(totalClassesHeld, totalClassesAttended);

} else {

Student newStudent = new Student(studentName, rollNumber, totalClassesHeld, totalClassesAttended);

attendanceRecords.put(rollNumber, newStudent);

}

}

public void displayAllAttendance() {

if (attendanceRecords.isEmpty()) {

System.out.println("No records found.");

return;

}

System.out.println("All Student Records:");

for (Map.Entry<Integer, Student> entry : attendanceRecords.entrySet()) {

Student student = entry.getValue();

System.out.println("Roll Number: " + student.getRollNumber());

System.out.println("\nStudent Name: " + student.getStudentName());

System.out.println("\nAttendance Percentage: " + student.getAttendancePercentage() + "%");

System.out.println("\nClasses Attended: " + student.getTotalClassesAttended());

System.out.println("\nClasses Held: " + student.getTotalClassesHeld());

System.out.println("\n--------------------------");

}

}

public void displayAttendance(int rollNumber) {

if (!attendanceRecords.containsKey(rollNumber)) {

System.out.println("Student not found.");

return;

}

Student student = attendanceRecords.get(rollNumber);

System.out.println("Attendance for Student " + student.getStudentName() + ":");

System.out.println("Attendance Percentage: " + student.getAttendancePercentage() + "%");

System.out.println("Classes Attended: " + student.getTotalClassesAttended());

System.out.println("Classes Held: " + student.getTotalClassesHeld());

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

StudentAttendanceSystem attendanceSystem = new StudentAttendanceSystem();

while (true) {

System.out.println("\nChoose an option:");

System.out.println("1. Add or update a student record");

System.out.println("2. View all student records");

System.out.println("3. View an individual student record");

System.out.println("4. Exit");

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

String input = scanner.nextLine().toLowerCase(); // Convert input to lowercase

if (input.equals("1")) {

System.out.print("Enter student name: ");

String studentName = scanner.nextLine().toLowerCase(); // Convert input to lowercase

int rollNumber = 0;

int totalClassesHeld = 0;

int totalClassesAttended = 0;

while (true) {

try {

System.out.print("Enter roll number: ");

rollNumber = scanner.nextInt();

System.out.print("Enter total classes held: ");

totalClassesHeld = scanner.nextInt();

System.out.print("Enter total classes attended: ");

totalClassesAttended = scanner.nextInt();

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

if (totalClassesAttended > totalClassesHeld) {

System.out.println("Error: Attended classes cannot be greater than total classes held. Please enter again.");

continue;

}

break;

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter integers for roll number and total classes.");

scanner.nextLine(); // Clear the input buffer

}

}

attendanceSystem.addOrUpdateStudentRecord(studentName, rollNumber, totalClassesHeld, totalClassesAttended);

} else if (input.equals("2")) {

attendanceSystem.displayAllAttendance();

} else if (input.equals("3")) {

System.out.print("Enter roll number to view attendance: ");

int rollNumber = 0;

try {

rollNumber = scanner.nextInt();

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

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter an integer for roll number.");

scanner.nextLine(); // Clear the input buffer

continue;

}

attendanceSystem.displayAttendance(rollNumber);

} else if (input.equals("4")) {

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

break;

} else {

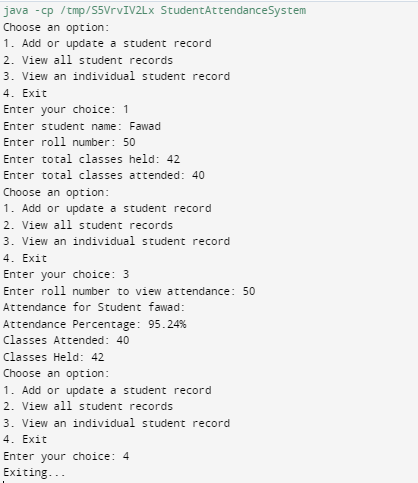
System.out.println("Invalid choice. Please enter 1, 2, 3, or 4.");

}

}

scanner.close();

}

}

Output:

# Testing Class

Code:

import java.util.Scanner;

public class TestStudentAttendanceSystem {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

StudentAttendanceSystem attendanceSystem = new StudentAttendanceSystem();

while (true) {

System.out.println("\nChoose an option:");

System.out.println("1. Add or update a student record");

System.out.println("2. View all student records");

System.out.println("3. View an individual student record");

System.out.println("4. Exit");

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

String input = scanner.nextLine().toLowerCase(); // Convert input to lowercase

if (input.equals("1")) {

System.out.print("Enter student name: ");

String studentName = scanner.nextLine().toLowerCase(); // Convert input to lowercase

int rollNumber = 0;

int totalClassesHeld = 0;

int totalClassesAttended = 0;

while (true) {

try {

System.out.print("Enter roll number: ");

rollNumber = scanner.nextInt();

System.out.print("Enter total classes held: ");

totalClassesHeld = scanner.nextInt();

System.out.print("Enter total classes attended: ");

totalClassesAttended = scanner.nextInt();

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

if (totalClassesAttended > totalClassesHeld) {

System.out.println("Error: Attended classes cannot be greater than total classes held. Please enter again.");

continue;

}

break;

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter integers for roll number and total classes.");

scanner.nextLine(); // Clear the input buffer

}

}

attendanceSystem.addOrUpdateStudentRecord(studentName, rollNumber, totalClassesHeld, totalClassesAttended);

} else if (input.equals("2")) {

attendanceSystem.displayAllAttendance();

} else if (input.equals("3")) {

System.out.print("Enter roll number to view attendance: ");

int rollNumber = 0;

try {

rollNumber = scanner.nextInt();

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

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter an integer for roll number.");

scanner.nextLine(); // Clear the input buffer

continue;

}

attendanceSystem.displayAttendance(rollNumber);

} else if (input.equals("4")) {

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

break;

} else {

System.out.println("Invalid choice. Please enter 1, 2, 3, or 4.");

}

}

scanner.close();

}

}

Output:

