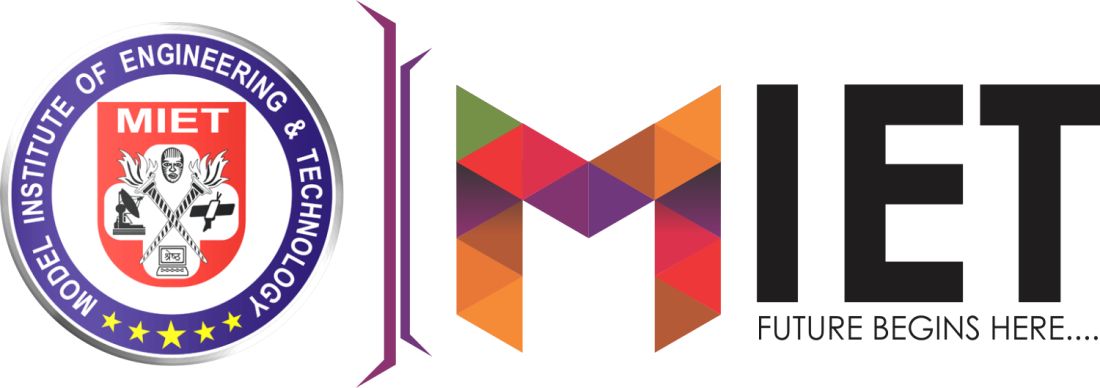
JAVA ASSIGNMENT : 2



SUBMITTED TO : Mrs. Sheerin Zadoo

SUBMITTED BY : Muskan Arora

CLASS : CSE(A1)

SEMESTER : 1638/17

ROLLNO. : 4th

CONTENTS

1.REQUIREMENTS SPECIFICATION

2.DESIGN SPECIFICATION

3.PROGRAM SPECIFICATION

4.IMPLEMENTATION

5.TESTING

1.REQUIREMENTS SPECIFICATION

Use Case 1: Uploading details , in this the details of the student is uploaded including their name and backlog count.

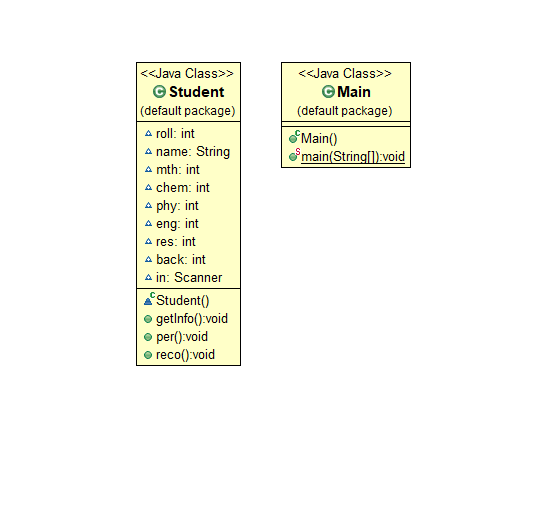
Use Case 2: Sorting, in this step the sorting of the students with 0 backlogs is done and are recommended for the further placement drive.

2. DESIGN SPECIFICATION

Class Diagram

In [software engineering](https://en.wikipedia.org/wiki/Software_engineering), a  **class diagram** in the [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML) is a type of static structure diagram that describes the structure of a system by showing the system's [classes](https://en.wikipedia.org/wiki/Class_(computer_science)), their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) modeling. It is used for general [conceptual modeling](https://en.wikipedia.org/wiki/Conceptual_model) of the structure of the application, and for detailed modeling translating the models into [programming code](https://en.wikipedia.org/wiki/Programming_code). Class diagrams can also be used for [data modeling](https://en.wikipedia.org/wiki/Data_modeling).[[1]](https://en.wikipedia.org/wiki/Class_diagram#cite_note-1) The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

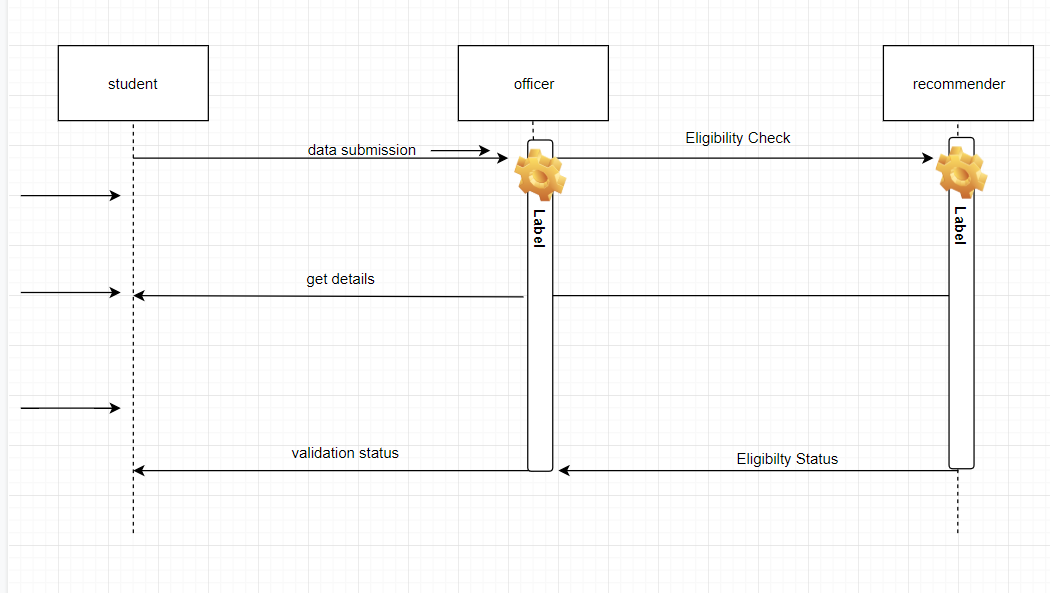


Sequence Diagram

A **sequence diagram** simply depicts interaction between objects in a **sequential** order i.e. the order in which these interactions take place. We can also use the terms event **diagrams** or event scenarios to refer to a **sequence diagram**. **Sequence diagrams** describe how and in what order the objects in a system function.

1. Student : submit all his details for placements drive to the officer for placement.
2. Officer : which further makes use of the recommender system for sorting of the students.
3. Recommender : which checks the eligibility of the student whether the student is eligible to sit for the placement or not.

After that the Recommender get the details of the student and pass the eligibility result to the officer who further pass on the enrollment status to the student for placement drive.



3. PROGRAM SPECIFICATION

Important Algorithms/ Function Logic

This programme is a student recommender program which is used to recommend students who are eligible to sit in the placement drive on the basis of the marks obtained by them and backlog count. This program works for the sorting of the students who have more than 60% or having 0 backlogs in any of the semesters.

This is a basic input/output program which is used to take the input from the user about the marks and the name of the student and print out whether the user is valid to sit for the placement drive or not.

The main logic of code for sorting of students with more than 60% is as follows:

public void reco()

{

if (mth>=40 && chem>=40 && phy>=40 && eng>=40 && back==0 && res>=60)

{

System.out.println (name+" is recommended to the company");

}

else

{

System.out.println (name+" is not recommended to the company");

}

}

This is the main code for the program created which would help in recommending the students with 0 backlog and more than 60% and withmarks minimum 40 as passing marks for the placement drive.

The code is as follows:

import java.util.\*;

class Student

{

int roll;

String name;

int mth;

int chem;

int phy;

int eng;

int res;

int back;

Scanner in = new Scanner(System.in);

public void getInfo()

{

System.out.println("enter the name and roll no. :");

name = in.next();

roll = in.nextInt();

System.out.println("enter the marks for each subject (maximum marks 100)");

mth = in.nextInt();

chem = in.nextInt();

phy = in.nextInt();

eng = in.nextInt();

System.out.println("Enter the no. of the backlogs ");

back = in.nextInt();

}

public void per()

{

int add= mth+chem+phy+eng;

res=add/4;

System.out.println("the result of the student is ="+res+"%");

}

public void reco()

{

if (mth>=40 && chem>=40 && phy>=40 && eng>=40 && back==0 && res>=60)

{

System.out.println (name+" is recommended to the company");

}

else

{

System.out.println (name+" is not recommended to the company");

}

}

}

public class Main

{

public static void main (String[]args)

{

Scanner in = new Scanner(System.in);

System.out.println("enter the number of Student in database");

int n = in.nextInt();

ArrayList<Student> al = new ArrayList<Student>();

for(int i=1; i<=n; i++)

{

Student s = new Student();

s.getInfo();

s.per();

s.reco();

al.add(s);

}

}

}

# IMPLEMENTATION

# GitHub Link : [https://github.com/muskanarora12/-](https://github.com/muskanarora12/-%20%20%20%20%20%20%20%20)

# studentrcomenders-

1. TESTING

Unit Testing Specifications and Execution Logs

1. While Entering the Name:

While entering the name if we had entered the input name as numeric input or some special character input, it should have given an error but instead of giving an error the program has accepted the error input.

1. While Entering the Marks:

While entering the marks if we had entered the input marks as alphabetic input or some special character input, it should have given an error but instead of giving an the program has the error input.

# 