Name of the Experiment :

Write a Java program which includes a base student class to hold five private data members : name, id and name of dept. A class will be derived from the base class which holds marks of three courses and the corresponding credits as private. In addition, another class will be further derived having the corresponding CTMA marks privately. The next derived class computes the letter grades for each course. The last derived class calculate total CPA and stores as a private data member. Write a test application to demonstrate the explained problem.

Introduction :

We have to define a base Student class and four derived class as per experiment description. And we will also have to define a drivers class that will demonstrate the experiment.

Objectives :

- how to inherit from a class
- how to use inheritance to solve a problem.

Analysis:

After analysing our problem we have found following components of our p solution.

- we have to define a base student class having name, id and name of dept as private members.
- we have to define a derived class having two arrays. In each array at same index we will have to store a course mark and that course's credits. The arrays will be of length 5.
- then we will have to define another derived class having the corresponding CTMA marks of the previously described courses.
- we will have to define another derive class, that will store the letter grades of the corresponding courses.
- and lastly another derived class to calculate the total CGPA.
- and a driver class to demonstrate the classes.

Let us consider the base class name as Student, the 1st derived class name as CourseAndCredits, the 2nd derived class name as Ctma, the 3rd derived class name as LetterGrade, the 4th derived class name as Cgpa and the driver class name as Main. From the analysis the conceptual class diagram is given in figure-1

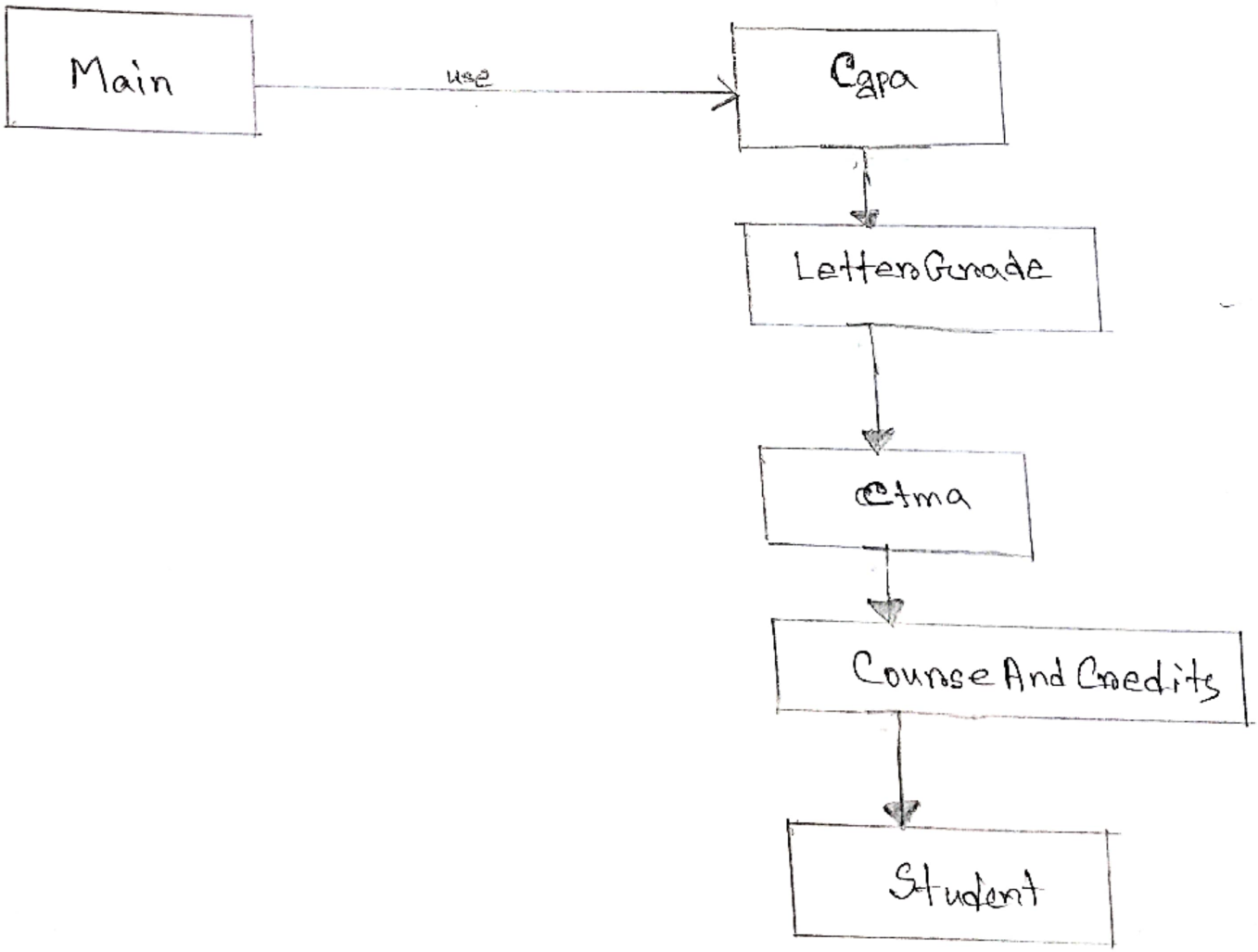

Figure-1: Conceptual Class Diagram

Design:

From the above analysis our class design descriptions are given below,

- class Student : our base class
  - Data Members :
    - name : a string representing the name of a student
    - Id : a string representing the id of a student
    - deptName : a string representing the department name
  - Methods :
    - set : this method will take three strings as parameters and set the name, Id and deptName.
    - getName : returns the name
    - getId : returns the Id
    - getDept : returns the deptName.

- class ~~Letter~~ CourseCredits : 1st level derived class
  - Data Members :
    - courses : an array of strings of length 3 storing the course name.
    - credits : an array of ints of length 3 storing the course credits.
    - index : an int representing the index at which the strings will be added

- Methods:
  - SetCourseCredit: a method taking a string: course marks and a double and its credit.
  - getNameM: a method taking an int and returning the corresponding course mark
  - getCredit: a method taking an int and returning the corresponding course credit.

- class Ctma: 2nd level derived class
  - Data members:
    - ctma: an array of ints of length 8, storing course CTMA marks.
  - Methods:
    - set: a method taking two ints, one for the index and one to assign the ctma mark.
    - get: a method taking an index and returning the corresponding ctma mark.

- class LetterGrade: 3rd level derived class
  - Data Members:
    - letterGrade: an array of strings representing the letter grades of each course

- Data Members / Methods:
  - calculateGrade : a method that will calculate the letter grades and store them.
  - getLetterGrade : ~~not~~ accepts int and returns the corresponding letter grade.

- class Cgpa : 4th level derived class.
  - Data Member :
    - cgpa : the total acquired cgpa.
  - Methods:
    - calculateCgpa : calculates the total CGPA.
    - getCgpa : returns the total acquired CGPA.

- class Main : the driver class
  - Methods :
    - main : the main method, from where the program will stat start.

From above design description the architectural of class diagram is given in Figure-2 :

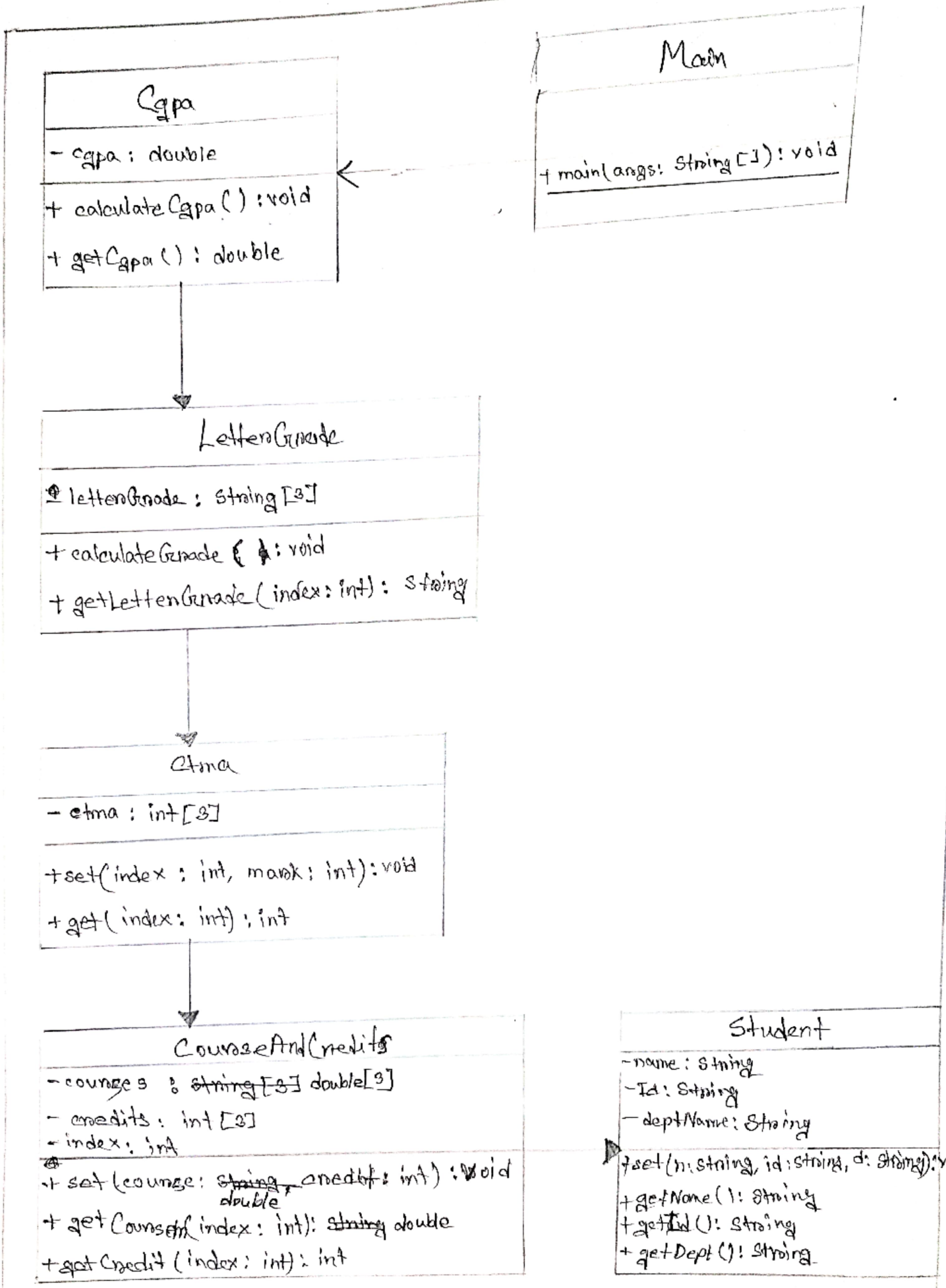

Figure-2: architectural Class diagram

From above design description, the pseudo codes of the methods are given below.

Student::set(name, id, dept): set the name, id and dept of a Student object

set name
set id
set dept

Student::getName(): returns the name of the object

return the name

Student::getId(): returns the Id

return the Id

Student::getDept(): returns the dept

return the dept

CourseAndCredits::set(course, credit): sets course and credit in the appropriate index

assign course and credit at the right index.

CourseAndCredits::getCourse(index): returns the course ~~name~~ mark at index

return the course ~~name~~ mark at index

CourseAndCredits::getCredit(index): returns the credit of a course at index

return the course credit at index

Ctma::set(index, mark): sets the ctma mark at index

set ctma marks at index

Ctma::get(index): returns the ctma marks at index

return the ctma mark at index

LetterGrade::calculateGrade(): calculates the grade of courses

calculate the letter grade of all the courses and store them

LetterGrade::getLetterGrade(index): returns the letter grade at index

return the letter grade at index

Cgpa::calculateCgpa(): calculates the total Cgpa

calculate the total CGPA

Cgpa::getCgpa(): returns the total CGPA

return the CGPA

Main::main(args): main function of the program

demonstrate the classes

Implementation:

```
class Student {
    private String name;
    private String Id;
    private String deptName;

    public void set(String n, String id, String dept) {
        name = n;
        Id = id;
        deptName = dept;
    }

    public String getName() {
        return name;
    }

    public String getId() {
        return Id;
    }

    public String getDept() {
        return deptName;
    }
}

class CourseAndCredits extends Student {
    private String[] courses = new String[3]; double[3]
    private int[3] credits = new int[3];
    private int index;

    CourseAndCredits() {
        index = 0;
    }
```

```
public void set(String course, int credit) {
                setCourseCredit   double
    courses[index] = course;
    credits[index] = credit;
    ++index;
}

public String getCourse(int index) {
    return courses[index];
}

public int getCredit(int index) {
    return credits[index];
}
}

class Ctma extends CourseAndCredits {
    private int[3] ctmas = new int[3];

    public void set(index, int index, int mark) {
                setCtma
        ctma[index] = mark;
    }

    public void int get(int index) {
                getCtma
        return ctma[index];
    }
}

class LetterGrade extends ctma {
    private String[3] letterGrades = new String[3];

    public void calculateGrade() {
        for(int i=0; i<3; ++i) {
```

```
int totalMarks = getCredit(i) * 25; // 1 credit = 25 marks
double accuiredMarks = getCtma(i) + getCourseM(i);
int percentage = (int) ( accuiredMarks / totalMarks * 100);
String letter;
if (percentage >= 80 && percentage <= 100)
    letter = "A+";
else if( percentage >= 75 && percentage <= 79)
    letter = "A";
else if (percentage >= 70 && percentage <=74)
    letter = "A-";
else if (percentage >= 65 && percentage <= 69)
    letter = "B+";
else if (percentage >=60 && percentage <= 64)
    letter = "B";
else if (percentage >= 55 && percentage <=59)
    letter = "B-";
else if ( percentage >= 50 && percentage <= 54)
    letter = "C+";
else if ( percentage >= 45 && percentage <=49)
    letter = "C";
else if (percentage >= 40 && percentage <=44)
    letter = "D";
else
    letter = "F";
letterGrade[i] = letter;
}
}
```

```
    public int getLetterGrade(int index) {
        return letterGrade[index];
    }
}

class Cgpa extends LetterGrade {
    private double cgpa;

    public void calculateCgpa() {
        int totalCredit = 0;
        for (int i = 0; i < 3; ++i) {
        int totalCredit = 0;
        double totalCreditPoints = 0;
        for (int i = 0; i < 3; ++i) {
            totalCredit += getCredit(i);
            String grade = getLetterGrade(i);
            if (grade == "A+")
                totalCreditPoints += 4 * getCredit(i);
            else if (grade == "A")
                totalCreditPoints += 3.75 * getCredit(i);
            else if (grade == "A-")
                totalCreditPoints += 3.50 * getCredit(i);
            else if (grade == "B+")
                totalCreditPoints += 3.25 * getCredit(i);
            else if (grade == "B")
                totalCreditPoints += 3.00 * getCredit(i);
            else if (grade == "B-")
                totalCreditPoints += 2.75 * getCredit(i);
            else if (grade == "C+")
                totalCreditPoints += 2.50 * getCredit(i);
            else if (grade == "C")
                totalCreditPoints += 2.25 * getCredit(i);
```

```
        else if (grade == 'D')
            totalCreditPoints += 2.00 * getCredit(i);
        else if (grade == 'F')
            totalCreditPoints += 0;
    }
    cgpa = totalCreditPoints / totalCredit;
}

public double getCgpa() {
    return cgpa;
}
}

class Main {
public static void main (String[] args) {
    Cgpa cgpa = new Cgpa();
    cgpa.set("Raynold", "20701OAA", "CSE");
    cgpa.setCourseCredit("CSE331", 3);
    cgpa.setCourseCredit("MAT331", 3);
    cgpa.setCourseCredit("PHY331", 3);
    cgpa.setCtma(0, 0);
    cgpa.setCtma(1, 1);
    cgpa.setCtma(2, 2);
    cgpa.calculateGrade();
    cgpa.calculateCgpa();
    System.out.println("Acquired CGPA: " + cgpa.getCgpa());
}
}
```

Conclusion:

We have defined a base class Student and four derived classes using multilevel inheritence. And we also defined a driven class Main. in which we defined a main method and showed our defined classes abilities.