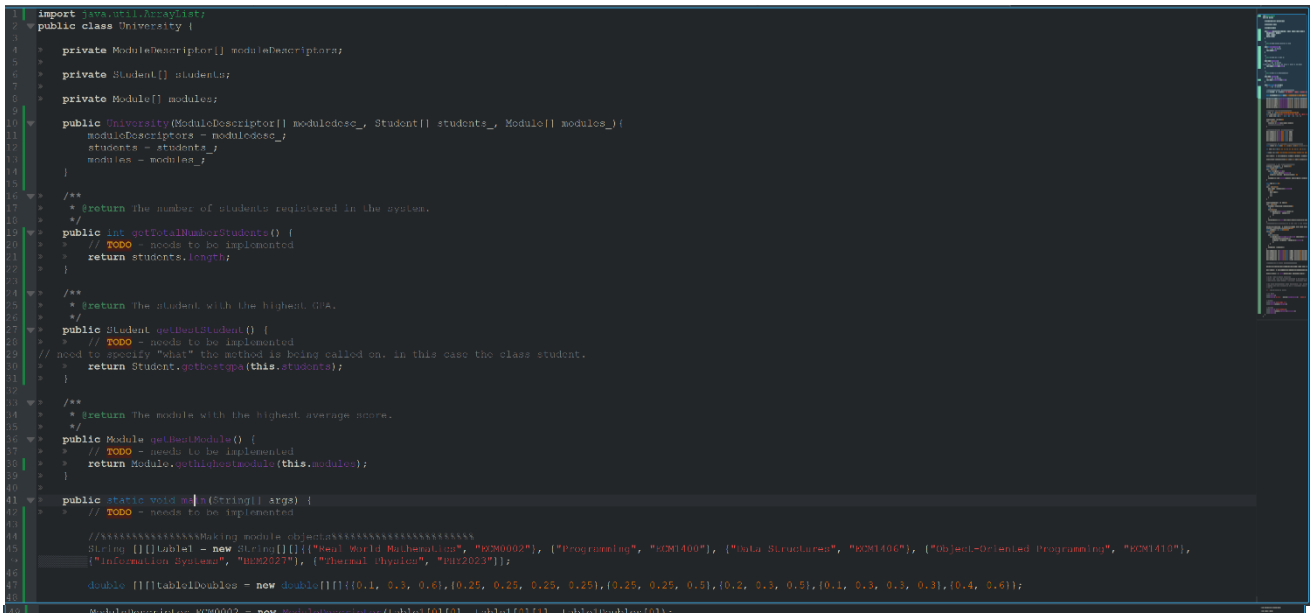
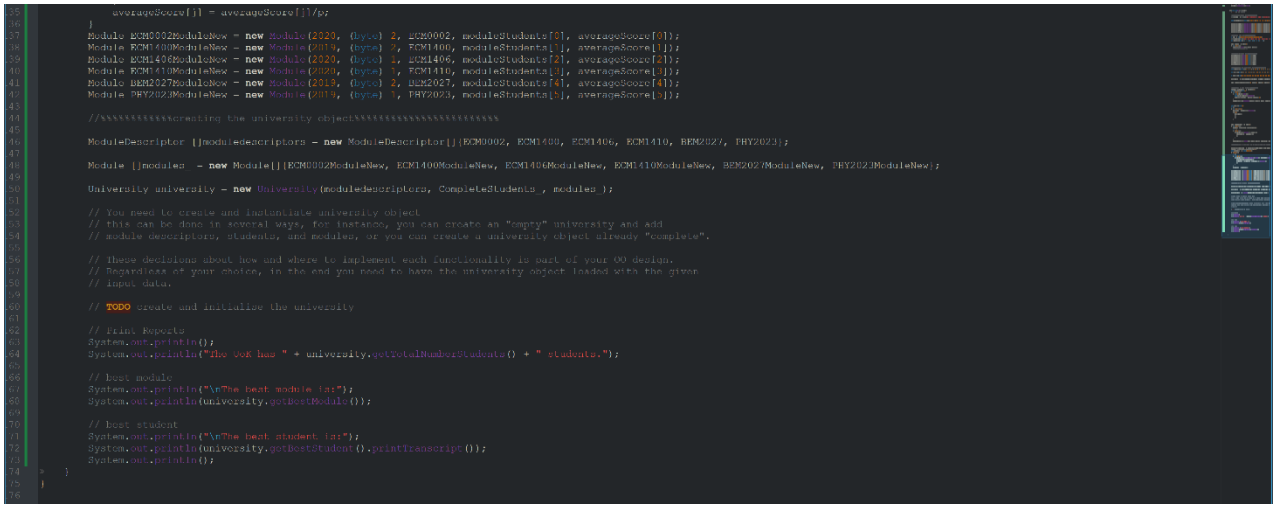
University.java

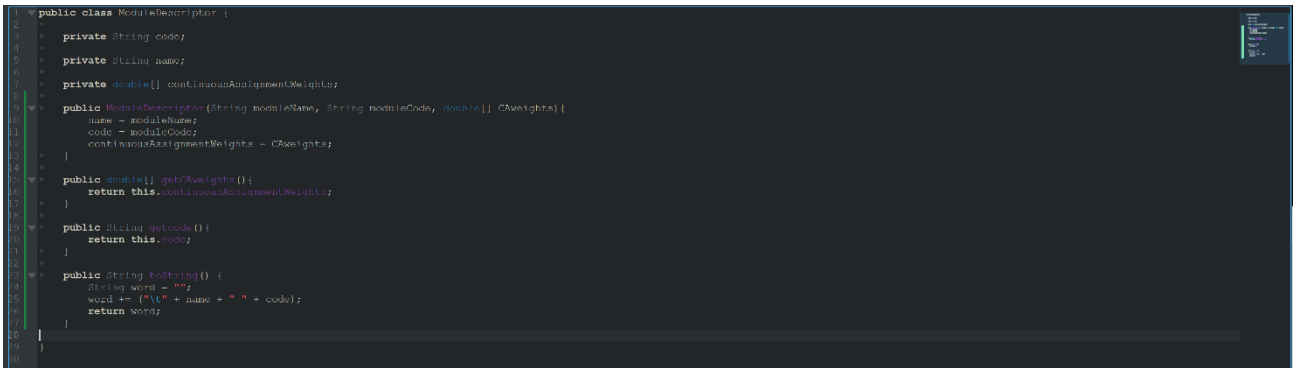


```
import java.util.ArrayList;
public class University {

    private ModuleDescriptor[] moduleDescriptors;

    private Student[] students;

    private Module[] modules;

    public University(ModuleDescriptor[] moduledesc_, Student[] students_, Module[] modules_){
        moduleDescriptors = moduledesc_;
        students = students_;
        modules = modules_;
    }

    /**
     * @return The number of students registered in the system.
     */
    public int getTotalNumberStudents() {
        // TODO - needs to be implemented
        return students.length;
    }

    /**
     * @return The student with the highest GPA.
     */
    public Student getBestStudent() {
        // TODO - needs to be implemented
// need to specify "what" the method is being called on. in this case the class student.
        return Student.getbestgpa(this.students);
    }

    /**
     * @return The module with the highest average score.
     */
    public Module getBestModule() {
        // TODO - needs to be implemented
        return Module.gethighestmodule(this.modules);
    }

    public static void main(String[] args) {
        // TODO - needs to be implemented

        //%%%%%%%%%%%%%%%%%%Making module objects%%%%%%%%%%%%%%%%%%%%%%%%%%
        String [][]table1 = new String[][]{{"Real World Mathematics", "ECM0002"}, {"Programming", "ECM1400"}, {"Data Structures", "ECM1406"}, {"Object-Oriented Programming", "ECM1410"},
        {"Information Systems", "BEM2027"}, {"Thermal Physics", "PHY2023"}};

        double [][][]table1Doubles = new double[][][]{{0.1, 0.3, 0.6},{0.25, 0.25, 0.25, 0.25},{0.25, 0.25, 0.5},{0.2, 0.3, 0.5},{0.1, 0.3, 0.3, 0.3},{0.4, 0.6}};

        ModuleDescriptor ECM0002 = new ModuleDescriptor(table1[0][0], table1[0][1], table1Doubles[0]);
        ModuleDescriptor ECM1400 = new ModuleDescriptor(table1[1][0], table1[1][1], table1Doubles[1]);
        ModuleDescriptor ECM1406 = new ModuleDescriptor(table1[2][0], table1[2][1], table1Doubles[2]);
        ModuleDescriptor ECM1410 = new ModuleDescriptor(table1[3][0], table1[3][1], table1Doubles[3]);
        ModuleDescriptor BEM2027 = new ModuleDescriptor(table1[4][0], table1[4][1], table1Doubles[4]);
        ModuleDescriptor PHY2023 = new ModuleDescriptor(table1[5][0], table1[5][1], table1Doubles[5]);

        //$$$$$$$$$$making student objects$$$$$$$$$$$$$$$$$$$$$$$$$$$$
        int []IDs = new int[]{1000,1001,1002,1003,1004,1005,1006,1007,1008,1009};
        String []names = new String[]{"Anna", "Oliver", "Mary", "Noah", "Chloe", "Maria", "Mark", "Tia", "Rachel"};
        char []genders = new char[]{'F', 'M', 'F', 'M', 'F', 'M', 'F', 'F', 'F', 'F'};

        Student []students_ = new Student[10];
        for (int i=0;i<10;i++){
            students_[i] = new Student(IDs[i], names[i], genders[i]);
            //System.out.println(students_[i]);
        }

        //$$$$$$$$$$$$making the module object$$$$$$$$$$$$$$$$$$$$$$$$
        Module ECM0002Module = new Module(2020, (byte) 2, ECM0002);
        Module ECM1400Module = new Module(2019, (byte) 2, ECM1400);
        Module ECM1406Module = new Module(2020, (byte) 1, ECM1406);
        Module ECM1410Module = new Module(2020, (byte) 1, ECM1410);
        Module BEM2027Module = new Module(2019, (byte) 2, BEM2027);
        Module PHY2023Module = new Module(2019, (byte) 1, PHY2023);

        //%%%%%%%%%%%%%%%%%%making module student and marks objects%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        double [][]Marks = new double[][]{{9, 10, 10, 10}, {8, 8, 8, 9}, {5, 5, 6, 8}, {6, 4, 7, 9}, {10, 9, 10, 9}, {9, 9}, {6, 9}, {5, 6}, {9, 7}, {8, 8}, {10, 10, 9.5, 10}, {7, 8.5, 8.2, 8},
        {6.5, 7.0, 5.5, 8.5}, {5.5, 5, 6.5, 7}, {7, 5, 8, 6}, {8, 10, 10, 10}, {8, 8, 6, 8}, {5, 5, 6, 5}, {6, 4, 7, 8}, {10, 8, 6, 9}, {10, 10, 10}, {8, 7.5, 7.5}, {9, 7, 7}, {9, 9, 7}, {2, 7,
        7}, {10, 10, 10}, {9, 7.5, 7.5}, {10, 10, 10}, {9, 6, 7}, {6, 9, 10}, {10, 9, 10}, {8.5, 9, 7.5}, {10, 10, 5.5}, {7, 7, 7}, {9, 6, 10}, {8, 9, 8}, {6.5, 9, 9.5}, {9.5, 10, 9.5}, {7.5, 8,
        10}, {10, 6, 10}};

        byte []term  = new byte[]{1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2};

        int []years_ = new int[]{2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2019, 2020, 2020, 2020, 2020, 2020, 2020, 2020,
        2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020, 2020};

        Module []Modules_ = new Module[]{ECM1400Module, ECM1400Module, ECM1400Module, ECM1400Module, ECM1400Module, PHY2023Module, PHY2023Module, PHY2023Module, PHY2023Module, PHY2023Module,
        BEM2027Module, BEM2027Module, BEM2027Module, BEM2027Module, BEM2027Module, ECM1400Module, ECM1400Module, ECM1400Module, ECM1406Module, ECM1406Module,
        ECM1406Module, ECM1406Module, ECM1406Module, ECM1406Module, ECM1406Module, ECM1406Module, ECM1406Module, ECM1406Module, ECM1410Module, ECM1410Module, ECM1410Module,
        ECM1410Module, ECM0002Module, ECM0002Module, ECM0002Module, ECM0002Module, ECM0002Module};

        Student []Students = new Student[]{students_[0], students_[1], students_[2], students_[3], students_[4], students_[5], students_[6], students_[7], students_[8], students_[9],
        students_[0], students_[1], students_[2], students_[3], students_[4], students_[5], students_[6], students_[7], students_[8], students_[9], students_[0], students_[1], students_[2],
        students_[3], students_[4], students_[5], students_[6], students_[7], students_[8], students_[9], students_[0], students_[1], students_[2], students_[3], students_[4], students_[5],
        students_[6], students_[7], students_[8], students_[9]};

        //$$$$$$$$$$$$making the student record object$$$$$$$$$$$$$$$$$$$$
        StudentRecord [][]studentRecords_ = new StudentRecord[40];
        double [][]finalmark = new double[40];
        for (int i=0;i<40;i++){
            for (int j=0;j<Marks[i].length;j++){
                double []weights = Modules_[i].getCAweightsm();
                finalmark[i] = finalmark[i] + (Marks[i][j] * weights[j] * 10);
            }
            studentRecords_[i] = new StudentRecord(Students[i], Modules_[i], Marks[i], finalmark[i]);
        }

        double []gpa = new double[10];
        int j=0;
        for (int i=0;i<10;j=j+10){
            gpa[i] = gpa[i] + studentRecords_[j].getfinalScore();
            if (j>=30){
                gpa[i] = gpa[i]/4;
                i++;
                j=j;
            }
        }

        Student []CompleteStudents_ = new Student[10];
        for (int i=0;i<10;i++){
            StudentRecord []studentsrecords_ = new StudentRecord[4];
            int p = 0;
            for (j=0;j<40;j++){
                if (studentRecords_[j].getStudent() == students_[i]){
                    studentsrecords_[p] = studentRecords_[j];
                    p++;
                }
            }
            CompleteStudents_[i] = new Student(IDs[i], names[i], genders[i], studentsrecords_, gpa[i]);
        }
        //%%%%%%%%%%%%%%%%Creating new module objects with the student records and average scores%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        ModuleDescriptor []modulestoadd = new ModuleDescriptor[]{ECM0002, ECM1400, ECM1406, ECM1410, BEM2027, PHY2023};
        StudentRecord [][]moduleStudents = new StudentRecord[6][10];
        double []averageScore = new double[]{0,0,0,0,0,0};
        for (j=0;j<6;j++){
            int p=0;
            for (int i=0;i<40;i++){
                if (studentRecords_[i].getModule().getmodule().getcode() == modulestoadd[j].getcode()){
                    moduleStudents[j][p] = studentRecords_[i];
                    averageScore[j] = averageScore[j] + studentRecords_[i].getfinalScore();
                    p++;
                }
            }
```

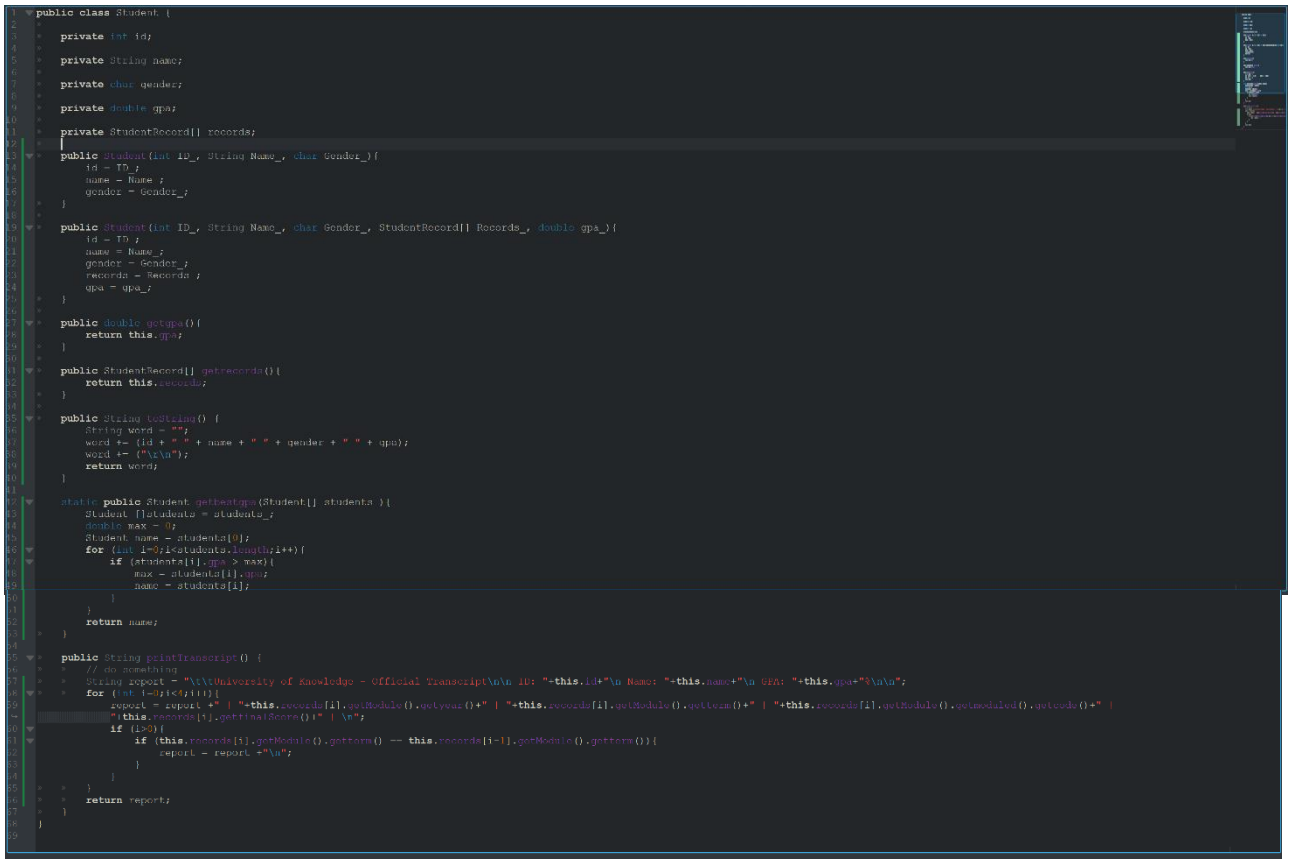


StudentRecord.java

ModuleDescriptor.java

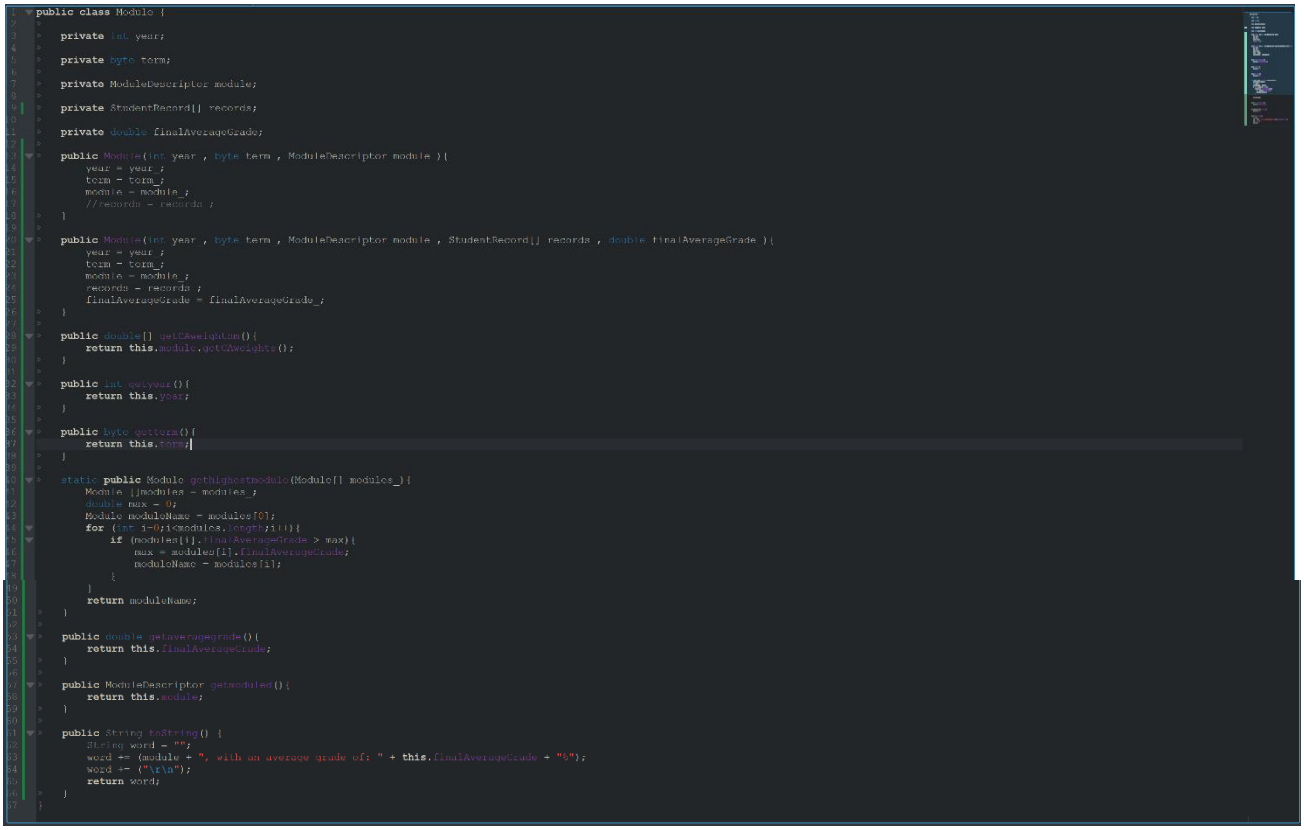


Student.java



```
public class Student {

    private int id;

    private String name;

    private char gender;

    private double gpa;

    private StudentRecord[] records;

    public Student(int ID_, String Name_, char Gender_){
        id = ID_;
        name = Name_;
        gender = Gender_;
    }

    public Student(int ID_, String Name_, char Gender_, StudentRecord[] Records_, double gpa_){
        id = ID_;
        name = Name_;
        gender = Gender_;
        records = Records_;
        gpa = gpa_;
    }

    public double getgpa(){
        return this.gpa;
    }

    public StudentRecord[] getrecords(){
        return this.records;
    }

    public String toString() {
        String word = "";
        word += (id + " " + name + " " + gender + " " + gpa);
        word += ("\r\n");
        return word;
    }

    static public Student getbestgpa(Student[] students ){
        Student []students = students_;
        double max = 0;
        Student name = students[0];
        for (int i=0;i<students.length;i++){
            if (students[i].gpa > max){
                max = students[i].gpa;
                name = students[i];
            }
        }
        return name;
    }

    public String printTranscript() {
        // do something
        String report = "\t\tUniversity of Knowledge - Official Transcript\n\n ID: "+this.id+"\n Name: "+this.name+"\n GPA: "+this.gpa+"%\n\n";
        for (int i=0;i<4;i++){
            report = report +" | "+this.records[i].getModule().getyear()+" | "+this.records[i].getModule().getterm()+" | "+this.records[i].getModule().getmodule().getcode()+" |
            "+this.records[i].getfinalScore()+" | \n";
            if (i>0){
                if (this.records[i].getModule().getterm() == this.records[i-1].getModule().getterm()){
                    report = report +"\n";
                }
            }
        }
        return report;
    }
}
```

Module.java



```
public class Module {

    private int year;

    private byte term;

    private ModuleDescriptor module;

    private StudentRecord[] records;

    private double finalAverageGrade;

    public Module(int year , byte term , ModuleDescriptor module ){
        year = year ;
        term = term_;
        module = module_;
        //records = records ;
    }

    public Module(int year , byte term , ModuleDescriptor module , StudentRecord[] records , double finalAverageGrade ){
        year = year ;
        term = term_;
        module = module_;
        records = records ;
        finalAverageGrade = finalAverageGrade_;
    }

    public double[] getCAweightsm(){
        return this.module.getCAweights();
    }

    public int getyear(){
        return this.year;
    }

    public byte getterm(){
        return this.term;
    }

    static public Module gethighestmodule(Module[] modules_){
        Module []modules = modules_;
        double max = 0;
        Module moduleName = modules[0];
        for (int i=0;i<modules.length;i++){
            if (modules[i].finalAverageGrade > max){
                max = modules[i].finalAverageGrade;
                moduleName = modules[i];
            }
        }
        return moduleName;
    }

    public double getaveragegrade(){
        return this.finalAverageGrade;
    }

    public ModuleDescriptor getmodule(){
        return this.module;
    }

    public String toString() {
        String word = "";
        word += (module + ", with an average grade of: " + this.finalAverageGrade + "%");
        word += ("\r\n");
        return word;
    }
}
```