Printout

Module.java:

```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, StudentRecord[
] records){
        this.year = year;
        this.term = term;
        this.module = module;
        this.records = records;
        this.finalAverageGrade = calcAverageGrade();
    }

    public int getYear(){
        return year;
    }

    public byte getTerm(){
        return term;
    }

    public ModuleDescriptor getModuleDescriptor(){
        return module;
    }

    public StudentRecord[] getRecord(){
        return records;
    }

    public double getFinalAverageGrade(){
        return finalAverageGrade;
    }

    private double calcAverageGrade(){
        double sum = 0;
        for (int i=0;i<records.length;i++){
            sum += records[i].getFinalScore();
        }
        return sum / records.length;
    }

}
```

Line Numbers = 50

ModuleDescriptor.java:

```
public class ModuleDescriptor {

    private String code = "";

    private String name = "";

    private double[] continuousAssignmentWeights;

    public ModuleDescriptor(String code,String name,double[] weights, ModuleDe
scriptor[] moduleDescriptors){
        this.code = code;
        this.name = name;
        continuousAssignmentWeights = weights;
        checkWeights();
        checkCodeUnique(moduleDescriptors);
    }

    public String getCode(){
        return code;
    }

    public String getName(){
        return name;
    }

    public double[] getContinuousAssignmentWeights(){
        return continuousAssignmentWeights;
    }

    private void checkCodeUnique(ModuleDescriptor[] moduleDescriptors){
        final int length = moduleDescriptors.length;
        for (int i=1;i<length;i++){
            if (moduleDescriptors[i].getCode().equals(code)){
                System.out.println("Error - Code entered is not unique.");
                System.exit(0);
            }
        }
    }
    private void checkWeights(){
        double sum = 0;
        final int length = continuousAssignmentWeights.length;
        for (int i=0;i<length;i++){
            if (continuousAssignmentWeights[i] < 0){
                System.out.println("Error - negative weight in controlled asse
ssment.");
                System.exit(0);
            }
            sum += continuousAssignmentWeights[i];
        }
        if (sum != 1.0){
            System.out.println("Error - weights do not sum to 1 in controlled 
assessment.");
            System.exit(0);
        }
    }
}
```

Number of lines = 55

Students.java:

```
import java.lang.System;

public class Student {

    private int id = 0;

    private String name = "";

    private char gender = ' ';

    private double gpa;

    private StudentRecord[] records;

    public Student(int id, String name, Character gender, StudentRecord[] reco
rds){
        char genderUpper = Character.toUpperCase(gender);
        this.id = id;
        this.name = name;
        this.gender = genderUpper;
        this.records = records;
        gpa = calcGpa();

        checkGender();
        checkIdUnique();
        }
```

```java
public int getId(){
    return id;
}

public String getName(){
    return name;
}

public char getGender(){
    return gender;
}

public double getGpa(){
    return gpa;
}

public StudentRecord[] getRecord(){
    return records;
}

private void checkIdUnique(){
    for (int i = 0;i<records.length;i++){
        if (id == records[i].getStudent().getId()){
            System.out.println("Error - Id is not unique.");
            System.exit(0);
        }
    }
}

private void checkGender(){
    if (gender != 'F'&& gender != 'M'&& gender != 'X'&& gender != ' '){
        System.out.println("Error - Gender is not blank, X, M or F.");
        System.exit(0);
    }
}

private double calcGpa(){
    double sum = 0;
    for (int i=0;i<records.length;i++){
        sum += records[i].getFinalScore();
    }
    return sum / records.length;
}

public String printTranscript() {
    String transcript = "\n\nID: "+id+"\nName: "+name+"\nGPA: "+gpa+"\n";
    byte prevTerm = records[0].getModule().getTerm();
```

```
        for (int i=0;i<records.length;i++){
            int year = records[i].getModule().getYear();
            byte term = records[i].getModule().getTerm();
            String code = records[i].getModule().getModuleDescriptor().getCode
();
            double score = records[i].getFinalScore();
            transcript = transcript.concat("|"+year+"|"+term+"|"+code+"|"+scor
e+"\n");
            if (prevTerm != term){
                transcript = transcript.concat("\n");
            }
            prevTerm = term;
        }
        return transcript;
    }


}
```

Number of lines = 90

StudentRecord.java:

```
public class StudentRecord {

    private Student student;

    private Module module;

    private double[] marks;

    private double finalScore;

    private boolean isAboveAverage;

    public StudentRecord
    (Student student, Module module, double[] marks){
        this.student = student;
        this.module = module;
        this.marks = marks;
        finalScore = calcFinalScore();
        isAboveAverage = calcAboveAverage();
        checkRange(finalScore);
        for (int i=0;i<marks.length;i++){
            checkRange(marks[i]);
        }
    }

    public Student getStudent(){
```

```java
        return student;
    }

    public Module getModule(){
        return module;
    }

    public double[] getMarks(){
        return marks;
    }

    public double getFinalScore(){
        return finalScore;
    }

    public boolean getIsAboveAverage(){
        return isAboveAverage;
    }

    private void checkRange(double value){
        if (value < 0 || value > 100){
            System.out.println("Error - score or marks out of range (between 0
and 100)");
            System.exit(0);
        }
    }

    private double calcFinalScore(){
        double product = 0.0;
        int length = marks.length;
        for (int i = 0;i<length;i++){
            product += marks[i]*module.getFinalAverageGrade();
        }
        return product/length;
    }

    private boolean calcAboveAverage(){
        double sum = 0.0;
        int numberStudents = student.getRecord().length;
        for (int i=0;i<numberStudents;i++){
            sum += student.getRecord()[i].finalScore;
        }
        double average = sum/numberStudents;
        if (finalScore > average){
            return true;
        }
        return false;
    }
```

```
}
```

Number of lines = 74

University.java

```
public class University {

    private ModuleDescriptor[] moduleDescriptors;

    private Student[] students;

    private Module[] modules;

    public University(){
        moduleDescriptors = createModuleDescriptors();
        StudentRecord[] records = createStudentsModules();
        Module[] moduleArray = new Module[7];
        Student[] studentArray = new Student[10];
        for (int i=0;i<7;i++){
            moduleArray[i] = records[i].getModule();
        }
        for (int i=0;i<10;i++){
            studentArray[i] = records[i].getStudent();
        }
        modules = moduleArray;
        students = studentArray;
    }

    private ModuleDescriptor[] createModuleDescriptors(){
        ModuleDescriptor[] array = new ModuleDescriptor[6];
        double[] weights1 = {0.1,0.3,0.6};
        array[0] = new ModuleDescriptor("ECM0002","Real World Mathematics",wei
ghts1,array);
        double[] weights2 = {0.25,0.25,0.25,0.25};
        array[1] = new ModuleDescriptor("ECM1400","Programming",weights2,array
);
        double[] weights3 = {0.25,0.25,0.5};
        array[2] = new ModuleDescriptor("ECM1406","Data Structures",weights3,a
rray);
        double[] weights4 = {0.2,0.3,0.5};
        array[3] = new ModuleDescriptor("ECM1410","Object-
Oriented Programming",weights4,array);
        double[] weights5 = {0.1,0.3,0.3,0.3};
        array[4] = new ModuleDescriptor("BEM2027","Information Systems",weight
s5,array);
        double[] weights6 = {0.4,0.6};
        array[5] = new ModuleDescriptor("PHY2023","Thermal Physics",weights6,a
rray);
        return array;
```

```java
    }

    private StudentRecord[] createStudentsModules(){
        StudentRecord[] records = new StudentRecord[40];
        Module[] moduleArray = new Module[7];
        Student[] studentArray = new Student[10];

        studentArray[0] = new Student(1000, "Ana", 'F', records);
        studentArray[1] = new Student(1001, "Oliver", 'M', records);
        studentArray[2] = new Student(1002, "Mary", 'F', records);
        studentArray[3] = new Student(1003, "John", 'M', records);
        studentArray[4] = new Student(1004, "Noah", 'M', records);
        studentArray[5] = new Student(1005, "Chico", 'M', records);
        studentArray[6] = new Student(1006, "Maria", 'F', records);
        studentArray[7] = new Student(1007, "Mark", 'X', records);
        studentArray[8] = new Student(1008, "Lia", 'F', records);
        studentArray[9] = new Student(1009, "Rachel", 'F', records);

        byte term = 1;
        moduleArray[0] = new Module(2019, term, moduleDescriptors[1], records)
;
        moduleArray[1] = new Module(2019, term, moduleDescriptors[5], records)
;
        moduleArray[2] = new Module(2019, ++term, moduleDescriptors[4], record
s);
        moduleArray[3] = new Module(2019, ++term, moduleDescriptors[1], record
s);
        moduleArray[4] = new Module(2020, term, moduleDescriptors[2], records)
;
        moduleArray[5] = new Module(2020, term, moduleDescriptors[3], records)
;
        moduleArray[6] = new Module(2020, ++term, moduleDescriptors[0], record
s);

        double[] marks1 = {9,10,10,10};
        records[0] = new StudentRecord(studentArray[0], moduleArray[0], marks1
);
        double[] marks2 = {8, 8, 8, 9};
        records[1] = new StudentRecord(studentArray[1], moduleArray[0], marks2
);
        double[] marks3 = {5,5,6,5};
        records[2] = new StudentRecord(studentArray[2], moduleArray[0], marks3
);
        double[] marks4 = {6,4,7,9};
        records[3] = new StudentRecord(studentArray[3], moduleArray[0], marks4
);
        double[] marks5 = {10,9,10,9};
        records[4] = new StudentRecord(studentArray[4], moduleArray[0], marks5
);

        double[] marks6 = {9 ,9};
        records[5] = new StudentRecord(studentArray[5], moduleArray[1], marks6
);
        double[] marks7 = {6, 9};
        records[6] = new StudentRecord(studentArray[6], moduleArray[1], marks7
);
        double[] marks8 = {5, 6};
        records[7] = new StudentRecord(studentArray[7], moduleArray[1], marks8
);
        double[] marks9 = {9, 7};
        records[8] = new StudentRecord(studentArray[8], moduleArray[1], marks9
);
        double[] marks10 = {8, 5};
        records[9] = new StudentRecord(studentArray[9], moduleArray[1], marks1
0);

        double[] marks11 = {10 ,10, 9.5, 10};
        records[10] = new StudentRecord(studentArray[0], moduleArray[2], marks
11);
        double[] marks12 = {7, 8.5, 8.2, 8};
        records[11] = new StudentRecord(studentArray[1], moduleArray[2], marks
12);
        double[] marks13 = {6.5,7.0,5.5,8.5};
        records[12] = new StudentRecord(studentArray[2], moduleArray[2], marks
13);
        double[] marks14 = {5.5,5,6.5,7};
        records[13] = new StudentRecord(studentArray[3], moduleArray[2], marks
14);
        double[] marks15 = {7,5,8,6};
        records[14] = new StudentRecord(studentArray[4], moduleArray[2], marks
15);

        double[] marks16 = {9,10,10,10};
        records[15] = new StudentRecord(studentArray[5], moduleArray[3], marks
16);
        double[] marks17 = {8, 8, 8, 9};
        records[16] = new StudentRecord(studentArray[6], moduleArray[3], marks
17);
        double[] marks18 = {5,5,6,5};
        records[17] = new StudentRecord(studentArray[7], moduleArray[3], marks
18);
        double[] marks19 = {6,4,7,9};
        records[18] = new StudentRecord(studentArray[8], moduleArray[3], marks
19);
        double[] marks20 = {10,9,8,9};
        records[19] = new StudentRecord(studentArray[9], moduleArray[3], marks
20);

        double[] marks21 = {10 ,10, 10};
        records[20] = new StudentRecord(studentArray[0], moduleArray[4], marks
21);
        double[] marks22 = {8, 7.5, 7.5};
        records[21] = new StudentRecord(studentArray[1], moduleArray[4], marks
22);
        double[] marks23 = {9,7,7};
        records[22] = new StudentRecord(studentArray[2], moduleArray[4], marks
23);
        double[] marks24 = {9,8,7};
        records[23] = new StudentRecord(studentArray[3], moduleArray[4], marks
24);
        double[] marks25 = {2,7,7};
        records[24] = new StudentRecord(studentArray[4], moduleArray[4], marks
25);

        double[] marks26 = {10 ,10, 10};
        records[25] = new StudentRecord(studentArray[5], moduleArray[4], marks
26);
        double[] marks27 = {8, 7.5, 7.5};
        records[26] = new StudentRecord(studentArray[6], moduleArray[4], marks
27);
        double[] marks28 = {10,10,10};
        records[27] = new StudentRecord(studentArray[7], moduleArray[4], marks
28);
        double[] marks29 = {9,8,7};
        records[28] = new StudentRecord(studentArray[8], moduleArray[4], marks
29);
        double[] marks30 = {8,9,10};
        records[29] = new StudentRecord(studentArray[9], moduleArray[4], marks
30);

        double[] marks31 = {10 ,9, 10};
        records[30] = new StudentRecord(studentArray[0], moduleArray[5], marks
31);
        double[] marks32 = {8.5,9,7.5};
        records[31] = new StudentRecord(studentArray[1], moduleArray[5], marks
32);
        double[] marks33 = {10,10,5.5};
        records[32] = new StudentRecord(studentArray[2], moduleArray[5], marks
33);
        double[] marks34 = {7,7,7};
        records[33] = new StudentRecord(studentArray[3], moduleArray[5], marks
34);
        double[] marks35 = {5,6,10};
        records[34] = new StudentRecord(studentArray[4], moduleArray[5], marks
35);

        double[] marks36 = {8 ,9, 8};
        records[35] = new StudentRecord(studentArray[5], moduleArray[6], marks
36);
        double[] marks37 = {6.5,9,9.5};
        records[36] = new StudentRecord(studentArray[6], moduleArray[6], marks
37);
        double[] marks38 = {8.5,10,8.5};
        records[37] = new StudentRecord(studentArray[7], moduleArray[6], marks
38);
        double[] marks39 = {7.5,8,10};
        records[38] = new StudentRecord(studentArray[8], moduleArray[6], marks
39);
        double[] marks40 = {10, 6, 10};
        records[39] = new StudentRecord(studentArray[9], moduleArray[6], marks
40);
        return records;
    }

    /**
     * @return The number of students registered in the system.
     */
    public int getTotalNumberStudents() {
        return students.length;
    }

    /**
     * @return The student with the highest GPA.
     */
    public Student getBestStudent() {
        double highest = students[0].getGpa();
        int studentNumber = 0;
        for (int i=1;i<getTotalNumberStudents();i++){
            if (students[i].getGpa() > highest){
                highest = students[i].getGpa();
                studentNumber = i;
            }
        }
        return students[studentNumber];
    }

    /**
     * @return The module with the highest average score.
     */
    public Module getBestModule() {
        double highest = modules[0].getFinalAverageGrade();
```

```java
        int moduleNumber = 0;
        for (int i=1;i<modules.length;i++){
            if (modules[i].getFinalAverageGrade() > highest){
                highest = modules[i].getFinalAverageGrade();
                moduleNumber = i;
            }
        }
        return modules[moduleNumber];
    }

    public static void main(String[] args) {
        University uok = new University();
        System.out.println("The number of students are: "+uok.getTotalNumberStudents());
        System.out.println("The best module score is: "+uok.getBestModule().getFinalAverageGrade());
        System.out.println("The best student is: "+uok.getBestStudent().printTranscript());
    }
}
```
Numbers of lines = 200