****

**King Abdulaziz University  
Faculty of Computing and Information Technology  
Fall 2021 – 1st – Term (2021)**

|  |  |
| --- | --- |
| **Course Code: CPCS 203** | **Course Name: Programming II** |

**Assignment #2 (Object Oriented Students Information System)**

|  |  |
| --- | --- |
| **Assigned Date** | **Sunday 10 /10/2021** |
| **Delivery Date and Time** | **Sunday 31/10/2021 at 11:00 PM** |

**WARNING**:

* This program must ONLY be submitted on the Blackboard!
* This project worth 10% of the overall module marks (100%).
* NO assignment will be accepted after 11:59 pm for any reasons.
* Students can submit their assignments between 11 and 11:59 PM but in this case it, will be considered as late submission, and they will lose 2 points from the total mark of the assignment.
* For discussion schedule, check the teacher’s name, date and time on the blackboard. **Further information is provided in the course syllabus.**

**Objectives:**

* Declaring classes and creating objects.
* Performing procedure on objects of different classes.
* Learn how to use and implement class and object concepts.
* Learn to use File I/O (Reading/Writing from/to files).

**Description**

This program is a Student Information System that is developed to manage students’ information at different universities. The program registers students at a particular course and allow instructors to enter student’s final grades into the system. It displays registered students’ information at a particular university (Name, ID, GPA, and course) and perform some simple statistical operations on the students’ GPAs (average GPA, Max GPA, and Min GPA). In addition, it also calculates some statistics on the students grades in a specific course at a specific university. Figure 1 shows a sample data with which the program deals.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| University | Course | Student ID | Student Name | GPA | Grade |
| KAU | CPCS202 | 7908489 | Hassan Ahmad | 4.5 | 97 |
| 7011231 | Faisal Hamza | 3.2 | 73 |
| 7409084 | Nada Jamal | 4.23 | 90 |
| 7500101 | Maryam Ali | 4.9 | 87 |
| KAUST | CS601 | 1866666 | Momtaz Saad | 3.33 | 81 |
| 1625843 | Nasser Faisal | 3.52 | 74 |

Figure 1: Data Sample.

You will need to design different classes to manage all the information in this system. For an illustration of classes in this system, please refer to the UML diagram in the following sections.

**Basic Requirements**

* The program must read the data from a text file called [input.txt] that follows a specific pattern. If the file doesn’t exist, print a message to let the user know what happened– see **Input File Pattern** section for more details.
* The program must generate a text file as the output called [output.txt] that contains the results of the commands written in the input file – see the attached sample output [outpu.txt] – see **Output File Pattern** section for more details.
* The program must load and deal with data dynamically. This means you should never write the data as hard codes. You should use loops to deal with the arrays, and classes.
* The program must follow the format of the output file as possible. So, you have to reformat strings. For example: Hassan Ahmad, you should know how to extract and separately print first name (Hassan) and last name (Ahmad). You can use a method called “split” in the string object.

**UML Diagram**

The Figure 2 shows the functionality of classes in this system. Mainly, there are three classes which are: University, Course, and Student. Mutators (setter) and accessors (getter) are implemented for every data field in every class, see Figure 2.

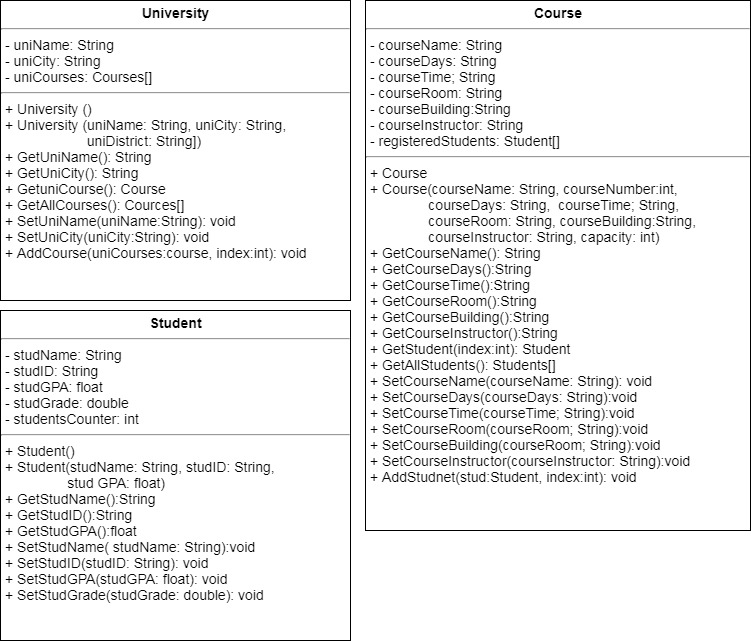


Figure 2: UML Diagram.

**Input File Pattern**

***Note: when reading this section please refer to the input file [input.txt] and the output file [output.txt] to help you better understand what the program does!***

To set the **initial parameters**, your program will **first** read the following parameters from [input.txt]:

**1. Total number of universities:** The first number in the first line of [input.txt] determines the **Total number of universities** used in testing our system, which is 2.

**2.** **Total number of courses in each university:** The number of courses taught at a particular university is determined in subsequent lines for adding universities. For example, the command (**add\_universities KAU Jeddah 3)** adds KAU to universities and also determines the number of courses taught at KAU which is 3.

**3. Total number of students in each course:** When the courses information is added, the capacity of this course section is determined. For example, the command (**add\_courses KAU CPCS202 MW 9-10:20 G125 061 Dr.Ahmad\_Alghamedy 20)** adds CPCS202 course information with a capacity of 20 students.

**Arrays and their Description**

You will have to create the following arrays:

* **University [] universities**, which is an array of **University** objects (created in the **main** program)
* **Course [] courses**, which is an array of **Course** objects (created in **University** objects).
* **Students [] students**, which is an array of **Student** objects (created in **Course** objects).

**Commands and their Description**

In the following, the commands that your program will read from [input.txt] are explained in more details, along with their associated methods that implement them:

1. **add\_university (e.g., add\_university KAU Jeddah 3)**

This command adds a university information, which includes the university name, city, and total number of courses.

1. **add\_course (e.g., add\_course KAU CPCS202 MW 9-10:20 G125 061 Dr.Ahmad\_Alghamedy 20)**

This command adds a course information to a particular university as it takes the university name as the first argument. The rest of the course information include: course name, Days, Time, room, building number, Instructor who teaches the course, and the capacity of students for that course.

1. **add\_students (e.g., add\_students KAU CPCS202 3)**

This command registers students at a particular course. It takes the following information as arguments: university name, course name, and the total number of students who will register in that course. The subsequent lines to this command will include the details of students’ information (ID, first name\_last name, GPA, and grade) as shown in the following example:

**add\_students KAU CPCS202 3**

**7908489 Hassan\_Ahmad 4.5 97**

**7011231 Faisal\_Hamza 3.2 73**

**7409084 Nada\_Jamal 4.23 90**

Note that the total number of students who are added using this command should not exceed the maximum capacity of this course. If the maximum capacity of that course has been reached (e.g. CPCS 202 maximum capacity is 20), an appropriate message should be printed that no more students can be added to that course.

1. **print\_result (e.g., print\_result KAU)**

This command will print all the information about students, who are registered in any of the courses taught at a particular university. It takes the name of the university as an argument.

1. **find\_average\_grade (e.g.,** **find\_average\_grade KAU CPCS202 )**

This command will calculate and print the average grades of students who are registered in a particular course in a specific university. It takes as arguments the university name, and the course name. In the above example, the university name is KAU and the course name is CPCS202.

1. **find\_max\_grade (e.g., find\_max\_grade KAU CPCS202)**
2. **find\_min\_grade (e.g., find\_min\_grade KAU CPCS202)**

These commands will calculate and print the maximum and minimum grades of students who are registered in a particular course at a specific university, respectively. They take as arguments the university name, and the course name. In the above examples, the university name is KAU and the course name is CPCS202.

1. **find\_average\_gpa**

This command will calculate and print the average GPA of all students in all universities. Note that these students are registered in a specific course that is taught in any of the universities in the system.

1. **find\_max\_gpa**
2. **find\_min\_gpa**

These commands will calculate the maximum and minimum GPAs of all students in all the universities in our system, respectively. Then the information of the students who have the maximum and minimum GPAs is printed.

1. **about\_app**

This command will print information about the author or the developer of the program (You should be proud! It’s YOU!)

1. **exit**

This command will end the program. The execution of all the commands listed above (1 -12) must be saved in the same directory of the program. Also, the date and time of generating the output should be printed in the output file [output.txt].

**Output File Pattern**

The output file should include all the results of the commands that have been read from the input file [input.txt]. Your program must generate output in a similar format to the given sample output file [output.txt].

**Deliverable**

You have to submit only the java file of your code. The file and the class names should be “**Project\_2\_YourFirstName\_YourUID**”. Where “YourFirstName“ is your first name, “YourUID“ is your university ID.

**NOTE:** your name, ID, and section number must be included as comments in the file!

**Important Notes:**

* Your Code, output, results etc. must be in a readable form.
* Organize your code in separate methods.
* Repeat the program until Quit command is read by your program.
* Use comments in your code.
* Use meaningful variables.
* Use dash lines separator between each method.

**Good Luck and Start Early!**