**School Management System**

**[Strictly adhere to the object oriented programming specifications given in the problem statement. Template code is provided to ease the input output process. Template code will not compile. You need to fill in the missing code.]**

**Business Requirement:**

Your task is to create a basic School management System where students can register to courses, instructor can view the course assigned to them and instructors with admin privileges can views all instructor assigned to a course and can also assign an instructor to any course.

**Work-Flow:**

The school management system starts by asking the user to select whether they are an Instructor or a student or if they want to exit out of the program.

If instructor is selected:

1. Only instructors with right credentials can login. Otherwise, a message is display stating: “Wrong Credentials”.
2. Instructors with non-admin privileges are only able to see the courses they are currently assigned to.
3. Instructors with admin privileges are able to see every instructor that is assigned to a particular course. They can also assign any instructor to any course in the system.

If student is selected:

1. Only students with right credentials can login. Otherwise, a message is display stating: “Wrong Credentials”.
2. Valid students are able to see the courses they are registered.
3. Valid students are able to register to any course in the system, if the student’s GPA is less than the course’s minimum GPA, the program should throw a StudentRegistrationException with a custom message “Did not meet the minimum GPA requirement”. This will terminate execution on the program.

**Requirement 1:**

**Model Class:**

Create a package in the src folder named: **CoreJava.Models**, in this package you will create every model class.

Every Model class must contain the following general two requirements:

1. First constructor takes no parameters and it initializes every members to an initial value.
2. Second constructor must initialize every private member with a parameter provided to the constructor.

Create a class Student with the private member variables specified in **TABLE 1**. These private members must have **GETTERS** and **SETTERS** methods.

The purpose of the Student class is to carry data related to one student.

**TABLE 1:**

|  |  |  |
| --- | --- | --- |
| **Datatype** | **Name** | **Description** |
| int | student\_id | Unique Database identifier |
| String | full\_name | The full name of the student |
| String | email | Student’s current school email |
| double | gpa | Student’s current GPA |
| String | pass | Student’s password in order to login |
| int | Student\_role | Defines the role of a student. Separates any students from being an instructor. This field always takes a value of -1 |

Create a class Instructor with the private member variables specified in **TABLE 2**. These private members must have **GETTERS** and **SETTERS** methods.

The purpose of the Instructor class is to carry data related to one Instructor.

**TABLE 2:**

|  |  |  |
| --- | --- | --- |
| **Datatype** | **Name** | **Description** |
| int | intrcutor\_id | Unique Database identifier |
| String | full\_name | The full name of the instructor |
| String | email | Instructor’s current school email |
| String | speciality | Instructor’s focus of studies |
| int | Student\_role | Defines the role of an Instructor. Separates any Instructor from being a Student. This field takes a value of 0 for regular instructors or 1 for admin instructors |
| String | pass | Instructor’s password. Used t login to the system |

Create a class Course with the private member variables specified in **TABLE 3**. These private members must have **GETTERS** and **SETTERS** methods.

The purpose of the Course class is to carry data related to one Course.

**TABLE 3:**

|  |  |  |
| --- | --- | --- |
| **Datatype** | **Name** | **Description** |
| int | course\_id | Unique Database identifier |
| String | course\_name | Provides the name of the course |
| double | minimum\_gpa | What is the minimum GPA required |

Create a class Teaching with the private member variables specified in **TABLE 4**. These private members must have **GETTERS** and **SETTERS** methods.

The purpose of the Teaching class is to carry data related to any Instructor that is assigned to a course.

**TABLE 4:**

|  |  |  |
| --- | --- | --- |
| **Datatype** | **Name** | **Description** |
| String | course\_name | Provides the name of the course |
| double | minimum\_gpa | What is the minimum GPA required |
| String | full\_name | The full name of the instructor |
| String | email | Instructor’s current school email |

Create a class Attending with the private member variables specified in **TABLE 5**. These private members must have **GETTERS** and **SETTERS** methods.

The purpose of the Attending class is to carry data related to any Student that is registered to a course.

**TABLE 5:**

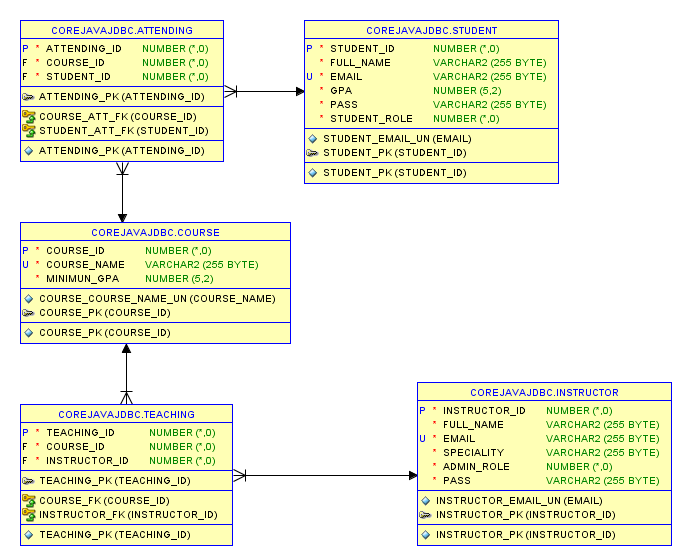
|  |  |  |
| --- | --- | --- |
| **Datatype** | **Name** | **Description** |
| String | course\_name | Provides the name of the course |
| String | full\_name | The full name of the instructor |
| String | email | Instructor’s current school email |

**Requirement 2:**

**Properties Files and Oracle Connection:**

A .properties file is already created for you under the package **CoreJava.Resources.** The name of this file is **db.properties.** You will use the values that this file contains such as the ‘user’ and ‘password’ to create a User/Connection in SQL Developer.

Inside SQL Developer, run the ‘CoreJavaAssessmet.sql’ file that was provided to you. This file contains SQL code to create every table needed with initial data. As you may have already noticed, every primary key in all tables are auto-generated by the database. Don’t forget to commit.



An OracleConnection.java class is provided to you under the package **CoreJava.DAO.** This class is responsible for returning a connection of your database. No changes are required for this class.

**Requirement 3:**

**Custom Exceptions**

Create a package name: **CoreJava.CustomExceptions.** Inside this package, create a custom Exception class named: **StudentRegistrationException.**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Class Name** | **Constructor Name** | **Input Parameters** |
| 1. | StudentRegistrationException extends Exception | public StudentRegistrationException(String) – This constructor accepts a string message parameter and passes it to its parent class using the super keyword | String Massage |

**Requirement 4:**

**Interfaces**

Create a package and call it: **‘CoreJava.systemsInterfaces’**

Create an Interface and call it: StudentDAOI and this interface contains the following methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Return Type** | **Interface Name** | **Method Name** | **Input Parameters** |
| 1 | Student | StudentDAOI | getStudentByGmail | String email |

Create an Interface and call it: InstructorDAOI and this interface contains the following methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Return Type** | **Interface Name** | **Method Name** | **Input Parameters** |
| 1 | List<Instructor> | InstructorDAOI | getAllInstructors | None |
| 2 | Instructor | InstructorDAOI | getInstructoByGmail | String email |

Create an Interface and call it: CourseDAOI and this interface contains the following methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Return Type** | **Interface Name** | **Method Name** | **Input Parameters** |
| 1 | List<Course> | CourseDAOI | getAllCourses | None |
| 2 | List<Course> | CourseDAOI | getCourseByInstructor | int instructor\_id |

Create an Interface and call it: AttendingDAOI and this interface contains the following methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Return Type** | **Interface Name** | **Method Name** | **Input Parameters** |
| 1 | int | AttendingDAOI | registerStudentToCourse | Student student, Course course |
| 2 | List<Attending> | AttendingDAOI | getStudentCourse | int student\_id |

Create an Interface and call it: TeachingDAOIand this interface contains the following methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Return Type** | **Interface Name** | **Method Name** | **Input Parameters** |
| 1 | int | TeachingDAOIand | registerStudentToCourse | int course\_id, int instructor\_id |
| 2 | List<Teaching> | TeachingDAOIand | getIntructorsCourses | none |

**Requirement 5:**

**Data Access Objects**

Under the package named: **CoreJava.DAO**, create a class and call it **StudentDAO**. This class is going to be use to query the database for student’s information only and it must implement the StudentDAOI interface. If this class throws any Exception, update the implemented interface.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Return Type | Class Name | Method Name | Input Parameters |
| 1 | Student | StudentDAO | getStudentByGmail – This method takes a String as a parameter and queries the database for an Student with such an email and returns a Student Object. | String email |
| 2 | Boolean | StudentDAO | validateUser – This method takes two parameters: the first one is the password from the database and the second one is the password from the user input. If both passwords are the same return true otherwise return false. | String passToValidate, String comparablePas |

Under the package named: **CoreJava.DAO**, create a class and call it **InstructorDAO**. This class is going to be use to query the database for instrcutor’s information only and it must implement the InstructorDAOI interface. If this class throws any Exception, update the implemented interface.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Return Type | Class Name | Method Name | Input Parameters |
| 1 | List<Instructor> | InstructorDAO | getAllInstructors – This method takes no parameter and returns every Instructor in the database. | None |
| 2 | Instructor | InstructorDAO | getInstructoByGmail – This method takes a String as a parameter and queries the database for an Instructor with such an email and returns an Instructor Object. | String email |
| 3 | String | InstructorDAO | validateUser – This method takes two arguments: an instructor object with all its information and a String which represent the password entered by the user trying to login as an instructor. This returns “Wrong Credentials”, “Admin” or “Instructor”. | Instructor ins, String comparablePas |

Under the package named: **CoreJava.DAO**, create a class and call it **CourseDAO**. This class is going to be use to query the database for course’s information only and it must implement the CourseDAOI interface. If this class throws any Exception, update the implemented interface.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Return Type | Class Name | Method Name | Input Parameters |
| 1 | List<Course> | CourseDAO | getAllCourses – This method takes no parameter and returns every Course in the database. | None |
| 2 | List<Course> | CourseDAO | getCourseByInstructor – This method takes an int as a parameter and queries the database for all the courses one instructor is assigned to. | int instructor\_id |

Under the package named: **CoreJava.DAO**, create a class and call it **AttendingDAO**. This class is going to be use to query the database for Attending’s information only and it must implement the AttendingDAOI interface. The Attending table in the database represents the students registered to one course. If this class throws any Exception, update the implemented interface.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Return Type | Class Name | Method Name | Input Parameters |
| 1 | int | AttendingDAO | registerStudentToCourse – This method takes as a parameter a Student and a Course object. If the student’s GPA id greater or equal to the minimum GPA of the course then the student is allow to register to the course. If not, then throw the StudentRegistrationException with a custom massage such as “\nDid not meet the minimum GPA requirement\nRegistration Denied”. Since you are creating a new record in the database, return the primary key auto-generated by the database. | Student student,  Course course |
| 2 | List<Attending> | AttendingDAO | getStudentCourse – This method takes as a parameter a int student\_id and would query the database for all the courses a student is register base on the Id | int student\_id |

Under the package named: **CoreJava.DAO**, create a class and call it **TeachingDAO**. This class is going to be use to query the database for Teaching’s information only and it must implement the TeachingDAOI interface. The Teaching table in the database represents the instructors assigned to one course. If this class throws any Exception, update the implemented interface.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Return Type | Class Name | Method Name | Input Parameters |
| 1 | int | TeachingDAO | assignInstructorToCourse – This method takes as a parameter a course\_id and a instructor\_id int and perform an INSERT query into the TEACHING table to assign an instructor to a course. | int course\_d,  int instructor\_id |
| 2 | List<Teaching> | TeachingDAO | getIntructorsCourses – This method takes no parameters and queries the database for every instructor assigned to a course. | None |

**Requirement 6:**

**Main Entry**

Inside the package named: **CoreJava.MainEntryPoint**, there is a class named: **MainEntryClass**. Inside this class there is two session separated by a commented line. In each session there is a main method which works as the entry point of the School Management System. Only one session can be comment out at a time. The purpose of session one is to provide you with a testing environment. Make sure to test all the work you have done by using them in any form. I would like to see all the work you have done, so do not erase any work you do in session one. Just comment it out. Session two contains the set of instruction that puts everything together. The following sample outputs will give you an idea of what to expect when you execute session two as a Java Application.

**Sample 1:** Instructor with admin privileges. Once logged in, the courses that the instructor is assigned to, are displayed. The only option is to Logout. Once logged out, the main menu appears. This is true in any sample.

Are you a(n)

1. Instructor

2. Student

3. quit

Please, enter 1, 2 or 3

1

Enter Your Email:

lance@gmail.com

Enter Your Password:

555

COURSE NAME COURSE MINIMUN GPA

English 3.1

1. Logout

1

Are you a(n)

1. Instructor

2. Student

3. quit

Please, enter 1, 2 or 3

**Sample 2:** Instructor with admin privileges. Once logged in, all Instructors are displayed alone with the courses that the instructor is assigned to. There are two options available: 1. Assign Instructor to Course and 2. Logout. If option 1 is selected, then the user is given the option to choose an instructor and a course. After selecting an instructor and a course, all Instructors are displayed alone with the courses that the instructor is assigned to. Logout is the same as before.

Are you a(n)

1. Instructor

2. Student

3. quit

Please, enter 1, 2 or 3

1

Enter Your Email:

mark@gmail.com

Enter Your Password:

666

COURSE NAME COURSE MINIMUN GPA INTRUCTOR NAME INSTRUCTOR EMAIL

GYM 2.8 mark mark@gmail.com

Math 3.1 Luke luke@gmail.com

Science 3.2 mark mark@gmail.com

English 3.1 lance lance@gmail.com

1. Assign Instructor to Course

2. Logout

1

Instructors:

======================================================================

ID INTRUCTOR NAME INSTRUCTOR EMAIL INSTRUCTOR SPECIALITY

1 Luke luke@gmail.com Mathemathician

2 lance lance@gmail.com scientis

3 mark mark@gmail.com important

What Instructor?

1

Courses:

===============================================

ID COURSE NAME MINIMUN GPA

41 GYM 2.8

1 Math 3.1

2 Science 3.2

3 English 3.1

Which Course?

2

-->Instructor Assigned<--

New Record Id: 81

COURSE NAME COURSE MINIMUN GPA INTRUCTOR NAME INSTRUCTOR EMAIL

GYM 2.8 mark mark@gmail.com

Math 3.1 Luke luke@gmail.com

Science 3.2 Luke luke@gmail.com

Science 3.2 mark mark@gmail.com

English 3.1 lance lance@gmail.com

1. Assign Instructor to Course

2. Logout

**Sample 3:** Students. Once a student is logged in, the student is able to see all the courses she/he is registered to. Two options are available 1. Register to Class and 2. Logout. If option 1 is selected, then the student is able to see all the courses and register to any of them.

Are you a(n)

1. Instructor

2. Student

3. quit

Please, enter 1, 2 or 3

2

Enter Your Email:

J@gmail.com

Enter Your Password:

333

My Classes:

# COURSE NAME INTRUCTOR NAME INSTRUCTOR EMAIL

1 GYM mark mark@gmail.com

2 Math Luke luke@gmail.com

1. Register to Class

2. Logout

1

All Courses:

ID COURSE NAME MINIMUN GPA

1 GYM 2.8

2 Math 3.1

3 Science 3.2

4 English 3.1

Which Course?

3

My Classes:

# COURSE NAME INTRUCTOR NAME INSTRUCTOR EMAIL

1 GYM mark mark@gmail.com

2 Math Luke luke@gmail.com

3 Science Luke luke@gmail.com

4 Science mark mark@gmail.com

1. Register to Class

2. Logout