

# AD - SBA - Core Java/Hibernate/JUnit

[Start Assignment](#)


---

**Due** No Due Date      **Points** 900      **Submitting** a file upload

---

## School Management System

**[Strictly adhere to the object-oriented programming specifications given in the problem statement.]**

**Configuration file** (hibernate.cfg.xml) can be found here, change accordingly: [hibernate.cfg.xml](https://platform.instructure.com/courses/402/files/132106/download?download_frd=1)   
([https://platform.instructure.com/courses/402/files/132106/download?download\\_frd=1](https://platform.instructure.com/courses/402/files/132106/download?download_frd=1))

More useful code can be found here [SMS SBA Mr. S hints.zip](https://platform.instructure.com/courses/402/files/132103/download?download_frd=1)   
([https://platform.instructure.com/courses/402/files/132103/download?download\\_frd=1](https://platform.instructure.com/courses/402/files/132103/download?download_frd=1))

### Business Requirement:

Your task is to create a basic School Management System where students can register for courses, and view the course assigned to them.

### Work-Flow:

Only students with the right credentials can log in. Otherwise, a message is displayed stating: "Wrong Credentials".

1. Valid students are able to see the courses they are registered.
2. Valid students are able to register to any course in the system as long as they are not already registered.

### Requirement 1:

#### Tables

Use your RDBMS to store the following tables. The tables should contain the columns from the specification below. You can generate the required tables (without rows) from your entities(model classes) by using Hibernate. The tables will be in the following format:

#### Format:

Datatype	Name	Description
The type of data contained in this column	The name of the column	The description of what this column will contain

**Table 1 – Student table:**

Datatype	Name	Description
varchar(50) not null (PK)	email	Student's current school email, unique student identifier
varchar(50) not null	name	The full name of the student
varchar(50) not null	password	Student's password in order to log in

**Table 2 – Course table:**

Datatype	Name	Description
int not null (PK)	id	Unique Course Identifier
varchar(50) not null	name	Provides the name of the course
varchar(50) not null	Instructor	Provides the name of the instructor

Now, insert test/dummy rows in Table 1 and Table 2 using SQL statements. Necessary SQL statements will be found here: [Student.sql](https://platform.instructure.com/courses/402/files/132105/download?download_frd=1) ↓ ([https://platform.instructure.com/courses/402/files/132105/download?download\\_frd=1](https://platform.instructure.com/courses/402/files/132105/download?download_frd=1)) and [Course.sql](https://platform.instructure.com/courses/402/files/132104/download?download_frd=1) ↓ ([https://platform.instructure.com/courses/402/files/132104/download?download\\_frd=1](https://platform.instructure.com/courses/402/files/132104/download?download_frd=1))

**Requirement 2:** Your project name must be **SMS**.

### Entity Model Class:

Create a package in the **src/main/java** folder named: **jpa.entitymodels**, in this package you will create every **entity** model class for this project.

Use the appropriate annotation to indicate that your models are to be used as an Entities, the name of the table each entity is based on, the variable that is used as a primary key, relationship, and the name

of the variable() is based on each entity.

Every Model class must contain the following general two requirements:

1. The first constructor takes no parameters and it initializes every member to an initial value.
2. The 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.

#### Student:

Datatype	Name	Description
String	sEmail	Student's current school email, unique student identifier
String	sName	The full name of the student
String	sPass	Student's password in order to log in
List	sCourses	All the courses that a student's registered for

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

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

#### Course:

Datatype	Name	Description
int	cId	Unique course Identifier
String	cName	Provides the name of the course

String	clnstructorName	Provides the name of the instructor
--------	-----------------	-------------------------------------

### Requirement 3:

#### Data Access Object (DAO)

You can **NOT** add more methods in the interfaces than the definition below.

Create a package in the **src/main/java** folder named **jpa.dao**, under this package, create an **interface** and call it **StudentDAO**. This interface is going to have the following method declarations. Please include the proper method signature based on the **Service** table:

*getAllStudents();*

*getStudentByEmail();*

*validateStudent();*

*registerStudentToCourse();*

*getStudentCourses();*

Create another interface and call it **CourseDAO** under the same package. This interface is going to have the following method declarations. Please include the proper method signature based on the **Service** table:

*getAllCourses();*

### Requirement 4:

#### Services (Implementation)

You might have some helper methods of your own in the services if necessary.

Create a package in the **src/main/java** folder named **jpa.service**, create two classes as **StudentService** and **CourseService** which implements the respective DAOs. These classes are going to be used to interact with the respective tables in your database instance.

No.	Return Type	Class Name	Method Name	Input Parameters
-----	-------------	------------	-------------	------------------

1	List<Student>	StudentService	<p>getAllStudents</p> <p>-This method reads the student table in your database and returns the data as a List&lt;Student&gt;</p>	None
2	Student	StudentService	<p>getStudentByEmail</p> <p>–This method takes a Student's email as a String and parses the student list for a Student with that email and returns a Student Object.</p>	String sEmail
3	boolean	StudentService	<p>validateStudent</p> <p>–This method takes two parameters: the first one is the user email and the second one is the password from the user input. Return whether or not a student was found.</p>	String sEmail, String sPassword
4	void	StudentService	<p>registerStudentToCourse –After a successful student validation, this method takes a Student's email and a Course ID. It checks in the join table (i.e. Student_Course) generated by JPA to find if a Student with that Email is currently attending a Course with that ID.</p> <p>If the Student is not attending that Course, register the student to that course; otherwise not.</p>	String sEmail, int cId
5	List<Course>	StudentService	<p>getStudentCourses</p> <p>–This method takes a Student's Email as a parameter and would find all the courses a student is registered.</p>	String sEmail
6	List<Course>	CourseService	<p>getAllCourses</p> <p>–This method takes no parameter and returns every Course in the table.</p>	None

**Requirement 5:****Main Entry**

Create a package in the **src/main/java** folder named: **jpa.mainrunner**, in this package you will create a class as **SMSRunner**. This class will be used to run the School Management System.

No.	Return Type	Class Name	Method Name	Input Parameters
1	void	SMSRunner	<p>main -</p> <p>-This method displays and prompts the user to select one of the following with the option:</p> <p><b>1. Student:</b> which allows the user to enter his/her email and password and check whether or not those credentials are valid, in order to log in. If the credentials are invalid the program should end with an appropriate message to the student.</p> <p>If the credentials are valid, the student is logged in and all the classes the Student is registered to should be displayed. Displays and prompt the student to select one of the following two additional numeric (1 or 2) options that are available:</p> <ul style="list-style-type: none"> <li>· 1) Register to Class: Which displays all the courses in the database and allows the student to select a course in which the student wished to be registered. If the Student is already registered in that course, display the message "You are already registered in that course!", otherwise, register the student to that course and save this result in your database. Also, show the updated registered courses list for that student. After that end the program with an appropriate message.</li> <li>· 2) Logout: Which ends the program with an appropriate message.</li> </ul>	String[] args

			<b>2. quit: which ends the program with an appropriate message.</b>	
--	--	--	---	--

**Example Workflow: The minimum required workflow, you can always enhance it.**

Are you a(n)

1. Student
2. quit

Please, enter 1 or 2.

1

Enter Your Email:

J@gmail.com

Enter Your Password:

333

My Classes:

# COURSE NAME INSTRUCTOR NAME

1 GYM Mark

2 Math Luke

1. Register to Class
2. Logout

1

All Courses:

ID COURSE NAME INSTRUCTOR NAME

1 GYM Mark

2 Math Luke

3 Science Stephanie

4 English Lisa

Which Course?

3

My Classes:

COURSE ID	COURSE NAME	INSTRUCTOR NAME
-----------	-------------	-----------------

1	GYM	Mark
---	-----	------

2	Math	Luke
---	------	------

3	Science	Stephanie
---	---------	-----------

You have been signed out.

### Requirement 6:

Handle all possible exceptions and include appropriate commenting. **Test** at least one of your methods using **Junit**.

### Submission:

Please zip your complete project and upload this in Canvas. Thank you.