**Assessment-1**

**Coding question no:1**

**Scenario-Based Question: CRUD Application Using Collections in Java**

**Scenario:**

You are tasked with developing a simple CRUD application for managing a list of Student objects. Each Student has an id, name, and age. The application should allow the following functionalities:

1. **Create**: Add a new student to the list.
2. **Read**: Display all students or find a student by their id.
3. **Update**: Update the details of a student by their id.
4. **Delete**: Remove a student from the list by their id.

**Requirements:**

* Use ArrayList or HashMap to store the Student objects.
* Ensure that the Student class has appropriate constructors, getters, and setters.

**Instructions:**

Implement the following methods in a Java class named StudentManager:

1. **addStudent(Student student)**: Adds a new student to the collection.
2. **getAllStudents()**: Returns a list of all students.
3. **getStudentById(int id)**: Returns a student by their id, or null if not found.
4. **updateStudent(int id, String name, int age)**: Updates the name and age of the student with the specified id.
5. **deleteStudent(int id)**: Removes the student with the specified id from the collection.

**Coding question:2**

### **Scenario-Based Question: CRUD Application Using JDBC with Oracle**

**Scenario:**

You are tasked with developing a simple CRUD application for managing a list of Student objects stored in an Oracle database. Each Student has an id, name, and age. The application should allow the following functionalities:

1. **Create**: Add a new student to the database.
2. **Read**: Retrieve all students or find a student by their id.
3. **Update**: Update the details of a student by their id.
4. **Delete**: Remove a student from the database by their id.

**Database Table Schema:**

sql

Copy code

CREATE TABLE Students (

id NUMBER PRIMARY KEY,

name VARCHAR2(50),

age NUMBER

);

**Requirements:**

* Use JDBC to interact with the Oracle database.
* Ensure that the Student class has appropriate constructors, getters, and setters.

**Assesment-2**

**Coding question-1:**

### Scenario-Based Question: CRUD Application Using Spring Core, Spring JDBC, and Maven

**Scenario:**

You are tasked with developing a CRUD application to manage a list of Employee objects stored in a MySQL database. Each Employee has an id, name, and department. The application should allow the following functionalities:

1. **Create**: Add a new employee to the database.
2. **Read**: Retrieve all employees or find an employee by their id.
3. **Update**: Update the details of an employee by their id.
4. **Delete**: Remove an employee from the database by their id.

**Requirements:**

* Use Spring Core for dependency injection.
* Use Spring JDBC for database operations.
* Use Maven for project management and dependencies.
* Ensure that the Employee class has appropriate constructors, getters, and setters.
* Configure the database connection using Spring.

**Database Table Schema:**

sql

Copy code

CREATE TABLE Employees (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50),

department VARCHAR(50)

);

**Instructions:**

Implement the following methods in a Java class named EmployeeManager:

1. **addEmployee(Employee employee)**: Adds a new employee to the database.
2. **getAllEmployees()**: Returns a list of all employees.
3. **getEmployeeById(int id)**: Returns an employee by their id, or null if not found.
4. **updateEmployee(Employee employee)**: Updates the details of the employee.
5. **deleteEmployee(int id)**: Removes the employee with the specified id from the database.

**Coding question-2:**

### Scenario-Based Question: CRUD Application Using Spring Boot and Spring Data JPA

**Scenario:**

You are tasked with developing a CRUD application to manage a list of Employee objects stored in a MySQL database. Each Employee has an id, name, and department. The application should allow the following functionalities:

1. **Create**: Add a new employee to the database.
2. **Read**: Retrieve all employees or find an employee by their id.
3. **Update**: Update the details of an employee by their id.
4. **Delete**: Remove an employee from the database by their id.

**Requirements:**

* Use Spring Boot for rapid application development.
* Use Spring Data JPA for database operations.
* Ensure that the Employee class has appropriate annotations for JPA.
* Configure the database connection using Spring Boot properties.