JDBC WORKSHOP

# Update, delete, findByParams

In EntityManager.java you will define the following methods:

* <T> T update(T entity);
* void delete(Object entity);
* List<T> findByParams(Class<T> entityClass, Map<String, Object> params);

Steps:

1. **Method <T> T update(T entity)**

* create a connection to DB;
* get table name and columns using the methods from EntityUtils class;
* iterate through ColumnInfo list
  + getDeclaredField by column name;
  + make the field accessible;
  + set the value of the columnInfo with the value obtained from the field;
* create a Condition object where you need to set id value which will be updated;
* create a QueryBuilder object where you set the name of table, query type, columns and conditions;
* call createQuery() method from QueryBuilder.java;
* create a Statement object and execute the query;
* return the updated object;

1. **Method void delete(Object entity)**

* create a connection to DB;
* get table name and columns using the methods from EntityUtils class;
* iterate through ColumnInfo list
  + getDeclaredField by column name;
  + make the field accessible;
  + set the value of the columnInfo with the value obtained from the field;
* create a Condition object where you need to set id value which will be updated;
* create a QueryBuilder object where you set the name of table, query type and conditions;
* call createQuery() method from QueryBuilder.java;
* create a Statement object and execute the query;

1. **List<T> findByParams(Class<T> entityClass, Map<String, Object> params)**

* create a connection to DB;
* get table name and columns using the methods from EntityUtils class;
* iterate through ColumnInfo list
  + getDeclaredField by column name;
  + make the field accessible;
  + set the value of the columnInfo with the value obtained from the field;
* create a QueryBuilder object where you set the name of table, query type and conditions;
* call createQuery() method from QueryBuilder.java;
* create a Statement object and execute the query;
* while the resultSet has any values (resultSet.next()) then:
  + you have to create an instance of type T;
  + iterate through ColumnInfo list and obtain the field of the instance using getColumnName().

Ex: instance.getClass().getDeclaredField(column.getColumnName());

* + make the field accessible;
  + set the value of the field with the value obtained from resultSet object;
  + add the instance in ArrayList;
* return the ArrayList;

# Test methods EntityManagerImpl.java

In *EntityManagerImplTest.java* test all the methods implemented in *EntityManagerImpl.java*.

# Create DepartmentDao and LocationDao files

Create in *src/main/java/ro/teamnet/zth/appl/dao*, 2 public classes: *DepartmentDao.java, LocationDao.java*, these will be used to call created methods from *EntityManagerImpl.java.*

1. **DepartmentDao.java**

* create an EntityManager object;
* create 5 methods which will call *EntityManagerImpl.java* methods;

1. **LocationDao.java**

* create an EntityManager object;
* create 5 methods which will call *EntityManagerImpl.java* methods;

# Create tests for DepartmentDao and LocationDao files

Create in *src/test/java/ro/teamnet/zth/appl* 2 public classes *DepartmentDao.java* and *LocationDao.java* which will test all the methods implemented in *DepartmentDao.java* and *LocationDao.java.*