JDBC WORKSHOP

1. Create database connection files
2. Create in *src/main/java/ro/teamnet/zth/api/database* an interface named *DBProperties* with following fields: IP, PORT, SID, USER, PASS. The value for this fields are:

String IP = "192.168.99.100";

String PORT = "49161";

String USER = "sys as sysdba"; // user from SQL workshop

String PASS = "oracle"; // pass from SQL workshop

String DRIVER\_CLASS = "oracle.jdbc.driver.OracleDriver";

1. Create in same folder a class *DBManager* with following fields and methods:

* create a private constructor with no params which will throw UnsupportedOperationException();
* create a constant field *CONNECTION\_STRING=* *"jdbc:oracle:thin:@" + DBProperties.IP + ":" + DBProperties.PORT +”:xe”;* where IP and PORT will be taken from *DProperties* file.
* create a private static method *registerDriver()* for registering your driver. (Ex: Class.forName(DBProperties.DRIVER\_CLASS) )
* create a public static method *getConnection()* with the following properties:
* method will return an object of type Connection;
* first you’ll have to register your driver;
* in order to obtain the connection object use DriverManager.getConnection with params: CONNECTION\_STRING, *USER* and *PASS* from DBProperties file;
* create a public static method *checkConnection(Connection connection)* in which you’ll make a simple query to DB (*SELECT 1 FROM DUAL*) using Statement interface.

# Test methods from DBManager.java

Create in *src/test/java/ro/teamnet/zth/api/database*  a public class *DBManagerTest.java* which will test getConnection() and checkConnection() methods from DBManager.java.

# Create EntityManager and EntityManagerImpl files

In src/main/java/ro/teamnet/zth/api/em create an interface which will manage the most important operation on a DB. First create an interface EntityManager.java in which you will define the following methods:

* <T> T findById(Class<T> entityClass, Long id);
* getNextIdVal(String tableName, String columnIdName)
* <T> Object insert(T entity);
* <T> List<T> findAll(Class<T> entityClass);

Next you need to implement EntityManager.java interface in class EntityManagerImpl.java.

Steps:

1. **Method <T> T findById(Class<T> entityClass, Long id)**

* create a connection to DB;
* get table name, columns and fields by annotations using the methods from EntityUtils class;
* create a Condition object in which you have to set column name and the value of the id;
* create a QueryBuilder object in which you have to set the name of the table, columns, query type and conditions;
* call createQuery() method from QueryBuilder.java;
* create a resultSet object using Statement and execute the query obtained above;
* if the resultSet has any value (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;
* return the instance;

1. **Method Long getNextIdVal(String tableName, String columnIdName)**

* create a connection to DB;
* create a statement object and execute a query that returns the column incremented by 1;

1. **Method <T> Object insert(T entity)**

* create a connection to DB;
* get table name, columns using the methods from EntityUtils class;
* iterate through ColumnInfo list
  + if the column is an id, getNextIdVal method;
  + else call getDeclaredField by column name on entity T;
  + make the field accessible;
  + set the value of the columnInfo with the value obtained from the field;
* create a QueryBuilder object in which you have to set the name of the table, columns, query type;
* call createQuery() method from QueryBuilder.java;
* create a Statement object and execute the query;
* return inserted entity using findByIdMethod;

1. **Method <T> List<T> findAll(Class<T> entityClass)**

* create a connection to DB;
* get table name, columns using the methods from EntityUtils class;
* create a QueryBuilder object in which you have to set the name of the table, columns, query type;
* call createQuery() method from QueryBuilder.java;
* create a resultSet object using Statement and execute the query obtained above;
* create an ArrayList of type T;
* 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

Create in *src/test/java/ro/teamnet/zth/api/em*  a public class *EntityManagerImplTest.java* which will test all the methods implemented in *EntityManagerImpl.java*.