# Exercise 6 - JDBC

- Define the DatabaseConnectionException class extending Exception to model database connection failure.
- Define the DbAccess class that realises access to the database.

## Attributes

*String DRIVER_CLASS_NAME = "com.mysql.cj.jdbc.Driver";* (Check the actual name of the Driver class in the jar you decide to use; To use this Driver download and add the mysql connector to the classpath)

*final String DBMS = 'jdbc:mysql';*

*final String SERVER="localhost":* contains the identifier of the server on which the database resides (e.g. localhost)

*final String DATABASE = "MapDB":* contains the name of the database

*final String PORT=3306*: The port on which the MySQL DBMS accepts connections

*final String USER_ID = "MapUser"*: contains the name of the user to access the database

*final String PASSWORD = 'map'*: contains the authentication password for the user identified by USER_ID

Connection conn: manages a connection

## Methods

*void initConnection() throws DatabaseConnectionException: instructs* the class loader to load the mysql driver, initialise the connection referred by *conn*. The method raises and throws a DatabaseConnectionException if the connection to the database fails.

*Connection getConnection()*: returns *conn*;

*void closeConnection():* closes the connection *conn*;

- Define the Table_Schema class (provided by teacher) that models the schema of a table in the relational database
- Define the class NoValueException extending Exception to model t h e absence of a

value within a resultset

- Define the EmptySetException class extending Exception to model the return of an empty resultset.
- Define the Example class (provided by the teacher) that models a transaction read from the database.
- Define the Table_Data class (partially provided by the teacher) that models the set of transactions collected in a table. The individual transaction is modelled by the Example class.

*public List<Example> getDistinctTransactions(String table) throws SQLException, EmptySetException*

Input: name of the table in the database.

Output: List of distinct transactions stored in the table.

Behaviour: : Obtains the schema of the table named table. It executes a query to extract the *distinct* tuples from that table. For each tuple in the resultset, an object is created, an instance of the Example class, whose reference is included in the list to be returned. Specifically, for the current tuple in the resultset, we extract the values of the individual fields (using getFloat() or getString()), and add them to the instance object of the Example class being constructed.

The method can propagate an exception of type *SQLException* (if there are errors in the query execution) or *EmptySetException* (if the resultset is empty)

*public Set<Object> getDistinctColumnValues (String table, Column column) throws SQLException*

Input: Table name, column name in the table

Output: Set of distinct values ordered in ascending mode that the attribute identified by name *column* takes on in the table identified by name *table*.

Behaviour: Formulates and executes an SQL query to extract ordered *column* values and populate a set to return *(choose appropriately in Set to use)*.

*public Object getAggregateColumnValue(String table, Column column, QUERY_TYPE aggregate) throws SQLException, NoValueException*

Input: Table name, column name , SQL aggregation operator (min,max) Output:

Aggregate searched.

Behaviour: Formulates and executes an SQL query to extract the aggregate value (minimum value or maximum value) searched for in the column named column of the table named *table*. The method raises and propagates a NoValueException if the resultset is empty or the value

calculated is equal to null

N.B. aggregate is of type QUERY_TYPE where QUERY_TYPE is the enumerative class provided by the teacher

```
public enum QUERY_TYPE {

MIN, MAX

}
```

- Remove the Example inner class and use the database.Example class instead. Replace the constructor of the **Data** class with a constructor that takes care of loading training data from a database table. The table name is a constructor parameter.