Faculty of Engineering,
Alexandria University
Computer and Systems Engineering Department

Real image of the project

JDBC-DBMS

12.2019

Fares Medhat 47
Abobakr Abdelaziz 02
Hazem Ahmed 23
Kareem Ahmed 48

Content

Description of the project

1- DBMS description

A Computer Database is a structured collection of records or data that is stored in a computer system. On the other hand, a Database Management System (DBMS) is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. DBMS are categorized according to their data structures or types. The DBMS accepts requests for data from the application program and instructs the operating system to transfer the appropriate data.

Extensible Markup Language (XML) (encoding: ISO-8859-1) is a set of rules for encoding documents in machine readable form. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all gratis open standards.

2- JDBC description

Java Database Connectivity (JDBC) provides Java developers with a standard API that is used to access

databases, regardless of the driver and database product. JDBC presents a uniform interface to databases

- change vendors and your applications only need to change their driver.

Project features:

- o Create database
- o Create table
- o Insert into table
- o Delete from table
- o Drop database
- o Drop table
- o Select from table
- o Update table
- o Conditions
- o Saving and Loading in XML files

Bonus features:

- o Supports Strings, Integers, Date and Float data types.
- o Flexible to add new data types.
- o Supports assigning time for executing queries.
- o Supports GUI to interact with the user.
- o Supporting select with order by up to any number of columns ascending or descending.

Used Design Pattern

Here is some of the design patterns that we used to make the code more readable and maintainable .

1) Factory Design Pattern

We have used Factory design pattern to create Objects from the text representation:

‘12345’ -> Creates a String Object and stores it in the table

12345 -> Creates an Integer Object and stores it in the table

Also we have used Factory Design pattern to generate the correct Commands for the SQL queries which makes the implementation as abstract as possible

2) Singleton Design Pattern

We have used Factory design in classes such as: All Factory classes, and FilesHandler. Since we only need one instance of each of these classes

3) Facade Design Pattern

Table, DatabaseManager and FilesHandler all use Instance of other classes inside them to do some functionalities

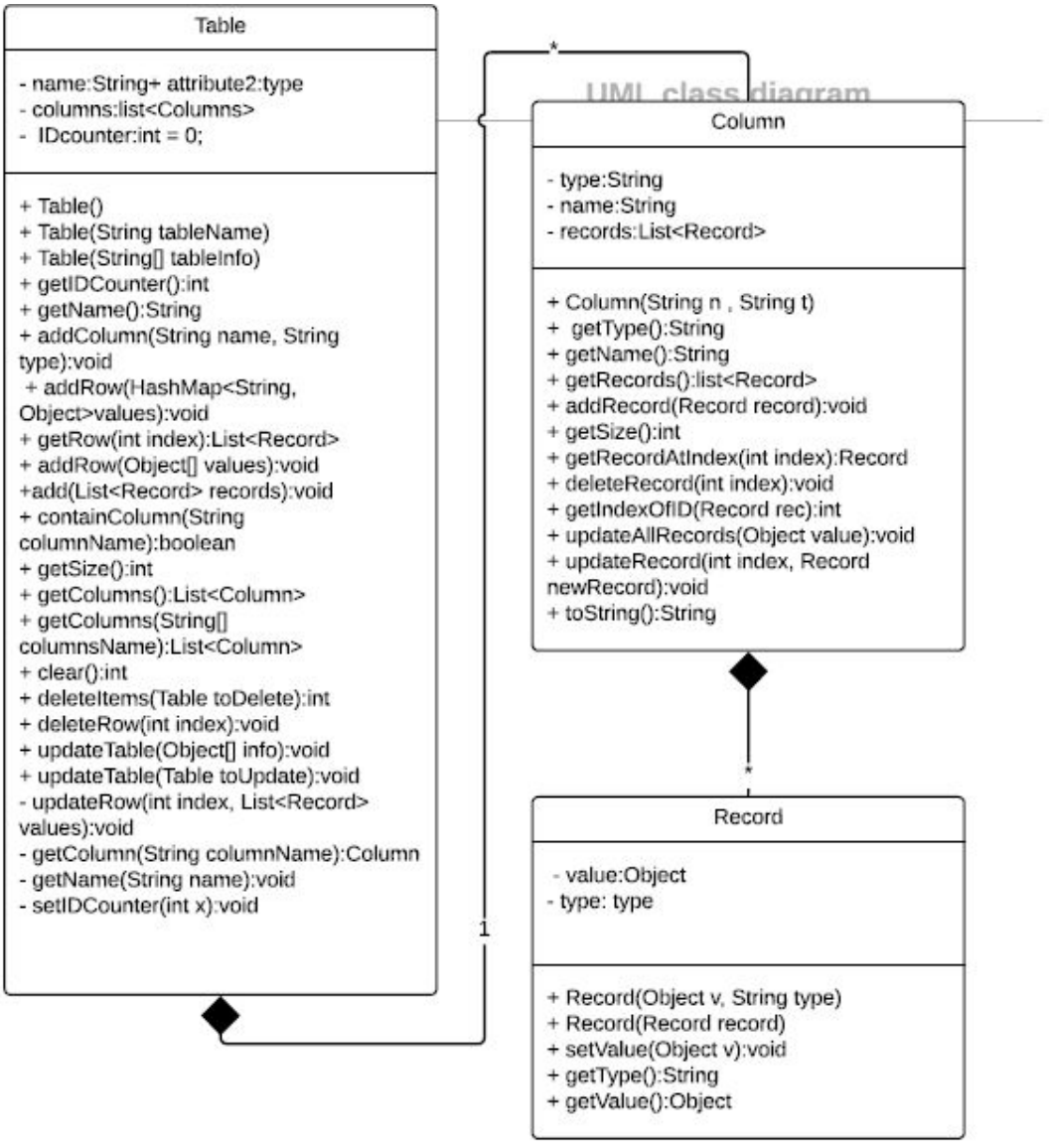
4) Filter Design Pattern

We have used Filter design pattern in Conditions where there is a ConditionFilter interface and all filter must implement that interface. There where 3 conditions in the project : “=”, “>”, “<”

5) Commands Design Pattern

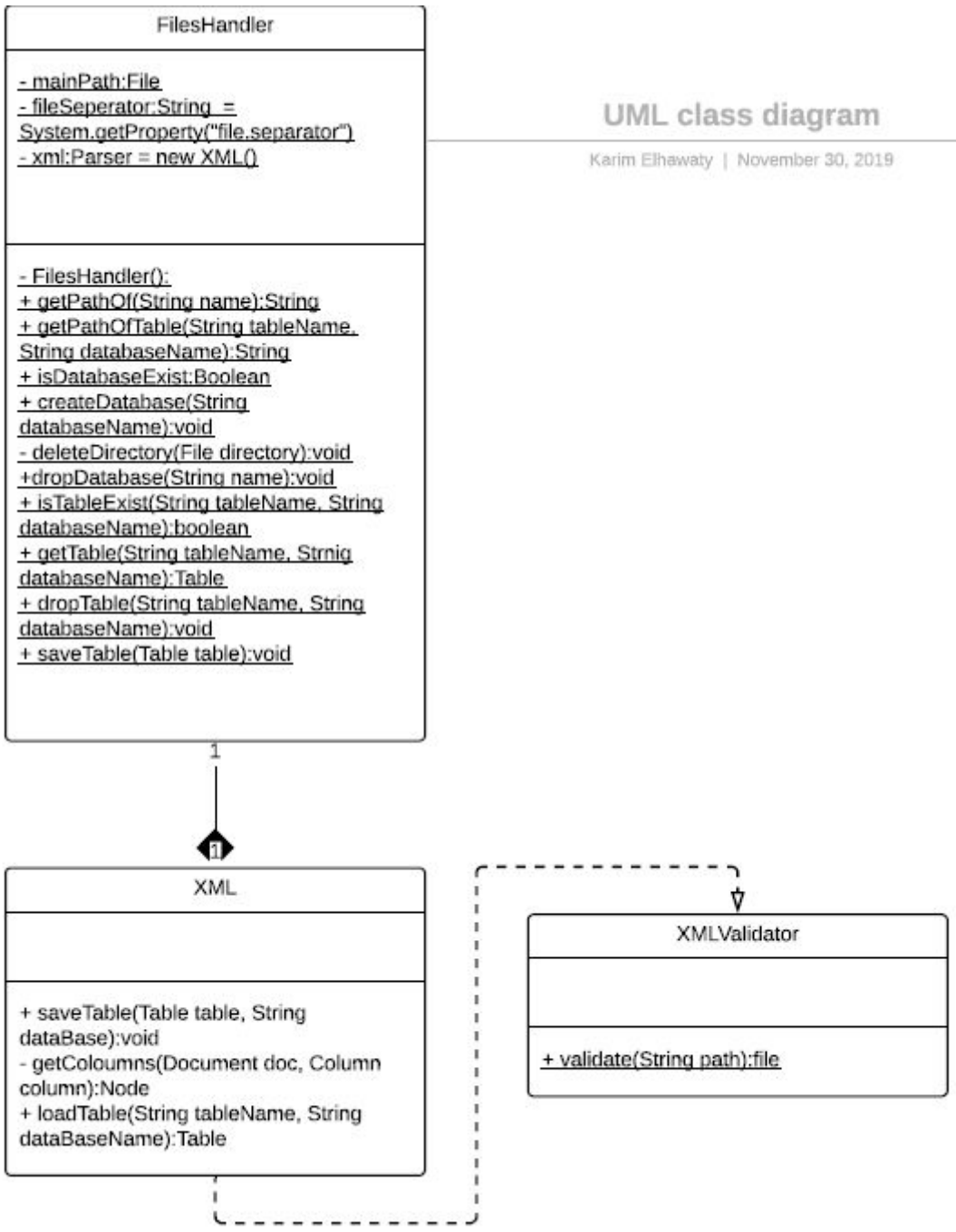
For each Command, There is a class that implements the Command interface such that every command can be simply executes as command.exec(). There factory classes which generate the correct command for a specific query

DBMS UML Diagrams



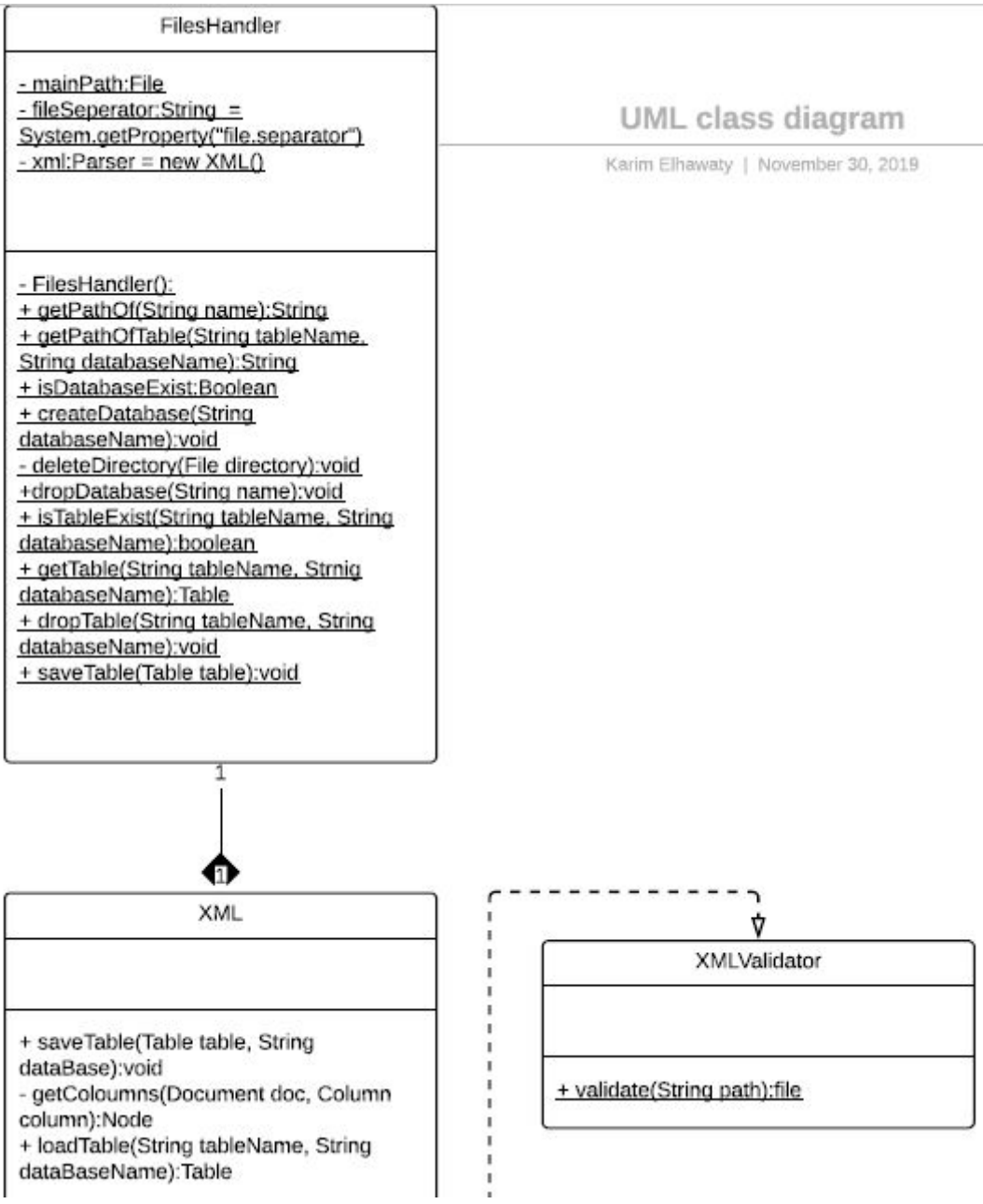
UML class diagram

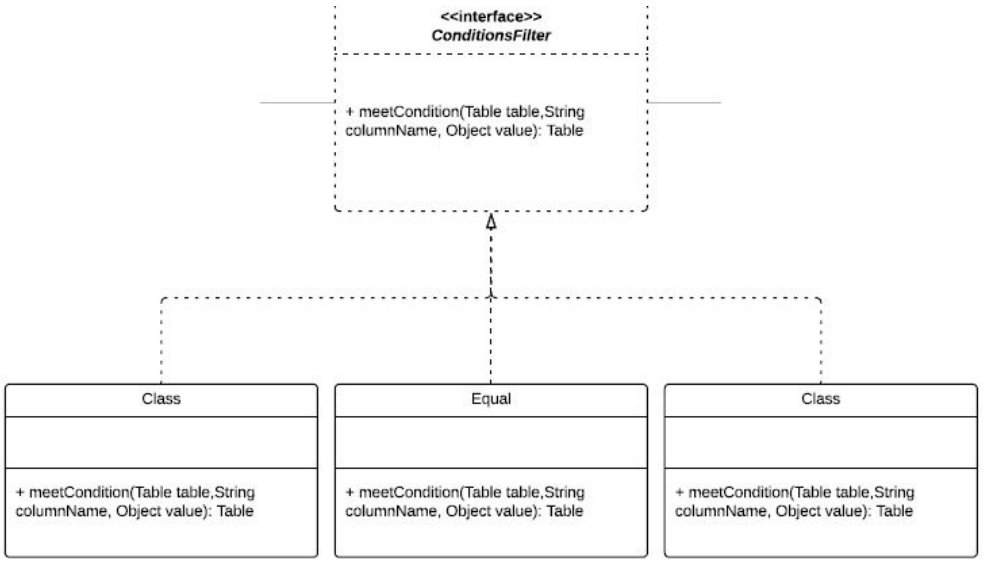
Karim Elhawaty | November 30, 2019

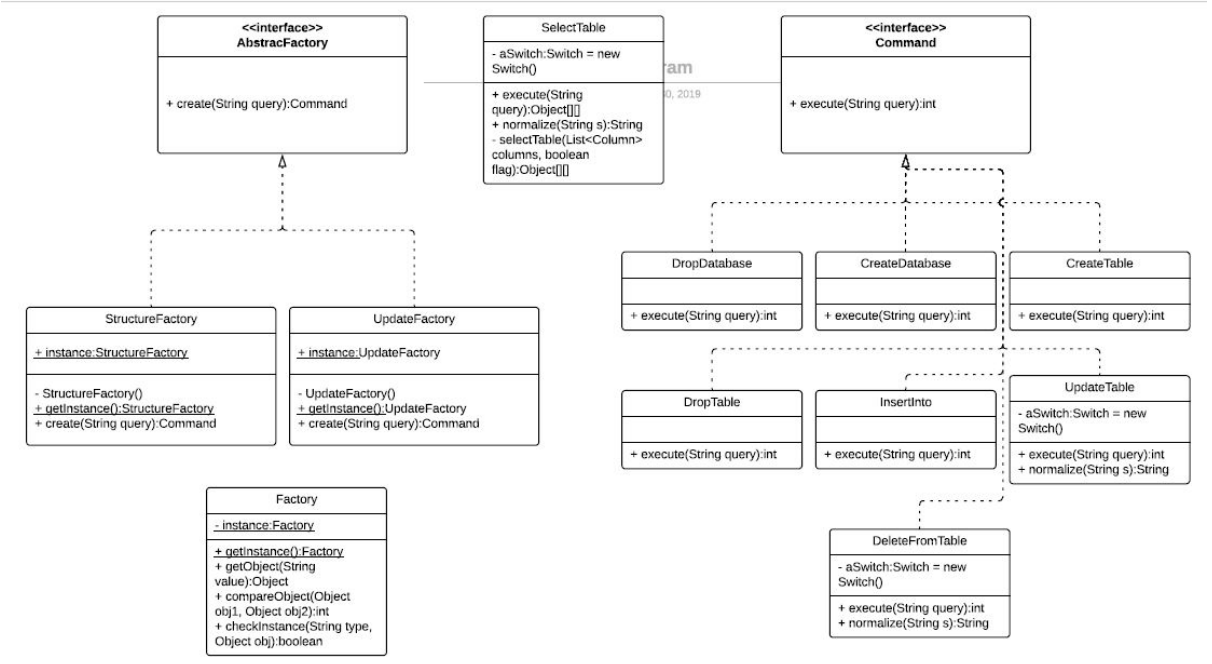


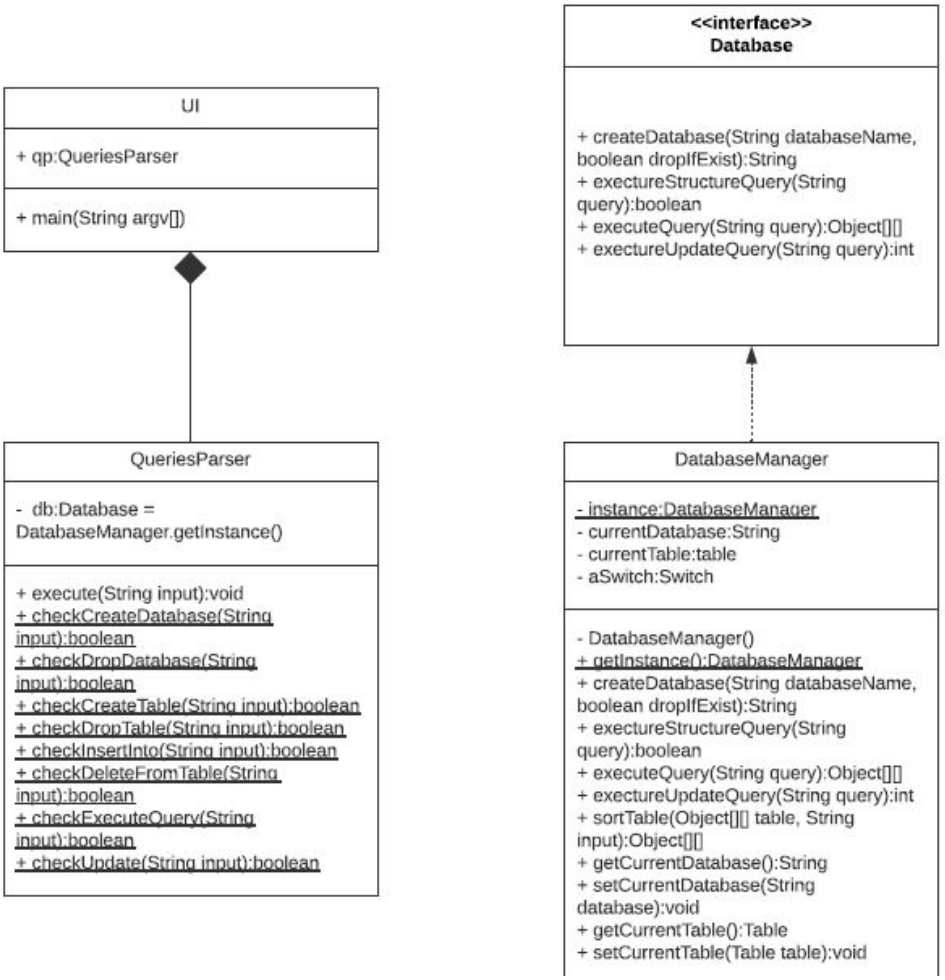
UML class diagram

Karim Elhawaty | November 30, 2019



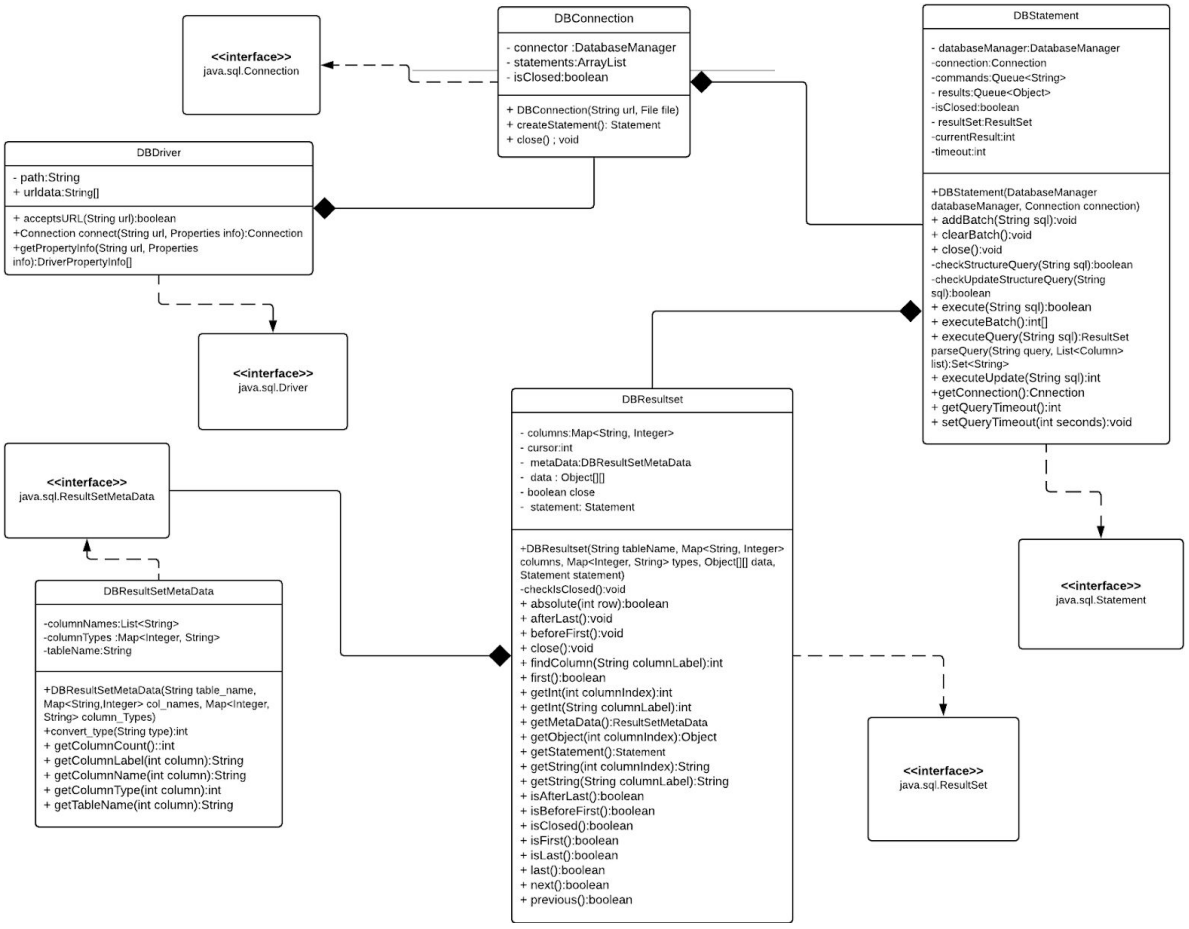

<<interface>>
ConditionsFilter
+ meetCondition(Table table,String columnName, Object value): Table
Class
+ meetCondition(Table table,String columnName, Object value): Table
Equal
+ meetCondition(Table table,String columnName, Object value): Table
Class
+ meetCondition(Table table,String columnName, Object value): Table


<<interface>>
AbstracFactory
+ create(String query):Command
SelectTable
- aSwitch:Switch = new Switch()
+ execute(String query):Object[][]
+ normalize(String s):String
- selectTable(List<Column> columns, boolean flag):Object[][]
<<interface>>
Command
+ execute(String query):int
StructureFactory
+ instance:StructureFactory
- StructureFactory()
+ getInstance():StructureFactory
+ create(String query):Command
UpdateFactory
+ instance:UpdateFactory
- UpdateFactory()
+ getInstance():UpdateFactory
+ create(String query):Command
DropDatabase
+ execute(String query):int
CreateDatabase
+ execute(String query):int
CreateTable
+ execute(String query):int
DropTable
+ execute(String query):int
InsertInto
+ execute(String query):int
UpdateTable
- aSwitch:Switch = new Switch()
+ execute(String query):int
+ normalize(String s):String
Factory
- instance:Factory
+ getInstance():Factory
+ getObject(String value):Object
+ compareObject(Object obj1, Object obj2):int
+ checkInstance(String type, Object obj):boolean
DeleteFromTable
- aSwitch:Switch = new Switch()
+ execute(String query):int
+ normalize(String s):String

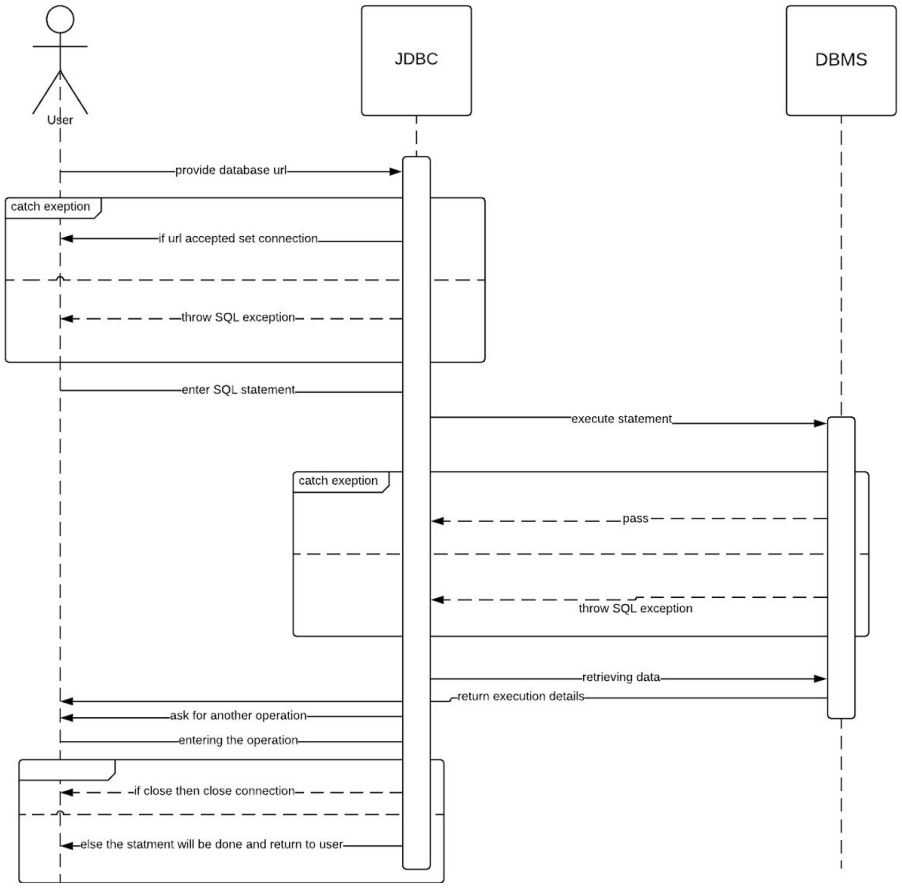

UI
+ qp:QueriesParser
+ main(String argv[])
QueriesParser
- db:Database = DatabaseManager.getInstance()
+ execute(String input):void
+ checkCreateDatabase(String input):boolean
+ checkDropDatabase(String input):boolean
+ checkCreateTable(String input):boolean
+ checkDropTable(String input):boolean
+ checkInsertInto(String input):boolean
+ checkDeleteFromTable(String input):boolean
+ checkExecuteQuery(String input):boolean
+ checkUpdate(String input):boolean
<<interface>>
Database
+ createDatabase(String databaseName, boolean dropIfExist):String
+ exectureStructureQuery(String query):boolean
+ executeQuery(String query):Object[][]
+ exectureUpdateQuery(String query):int
DatabaseManager
- instance:DatabaseManager
- currentDatabase:String
- currentTable:table
- aSwitch:Switch
- DatabaseManager()
+ getInstance():DatabaseManager
+ createDatabase(String databaseName, boolean dropIfExist):String
+ exectureStructureQuery(String query):boolean
+ executeQuery(String query):Object[][]
+ exectureUpdateQuery(String query):int
+ sortTable(Object[][] table, String input):Object[][]
+ getCurrentDatabase():String
+ setCurrentDatabase(String database):void
+ getCurrentTable():Table
+ setCurrentTable(Table table):void

JDBC UML Diagrams

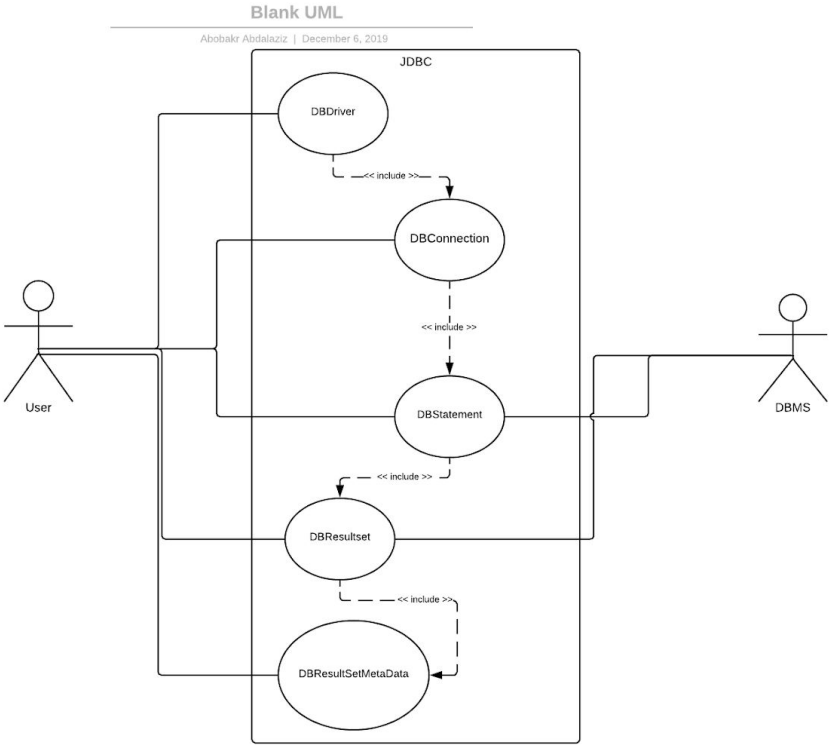
Class Diagram



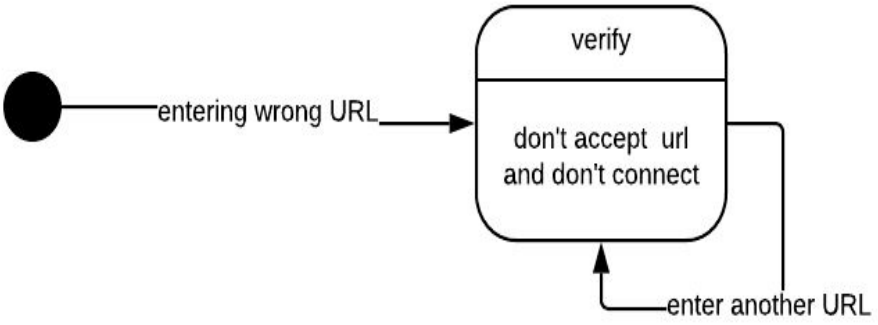
Sequence diagram

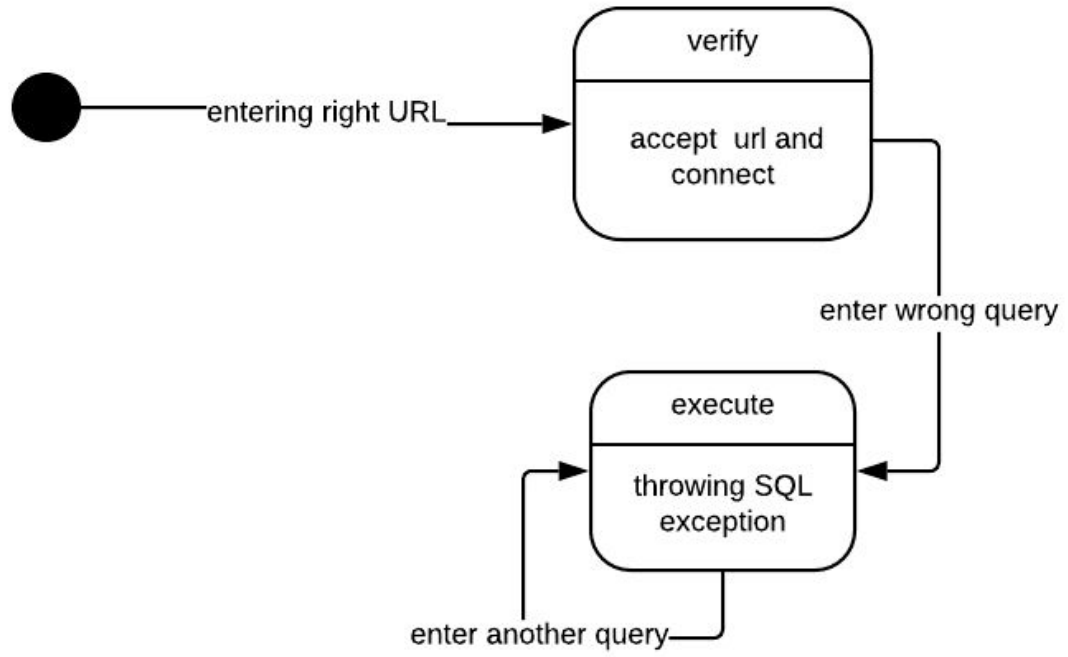


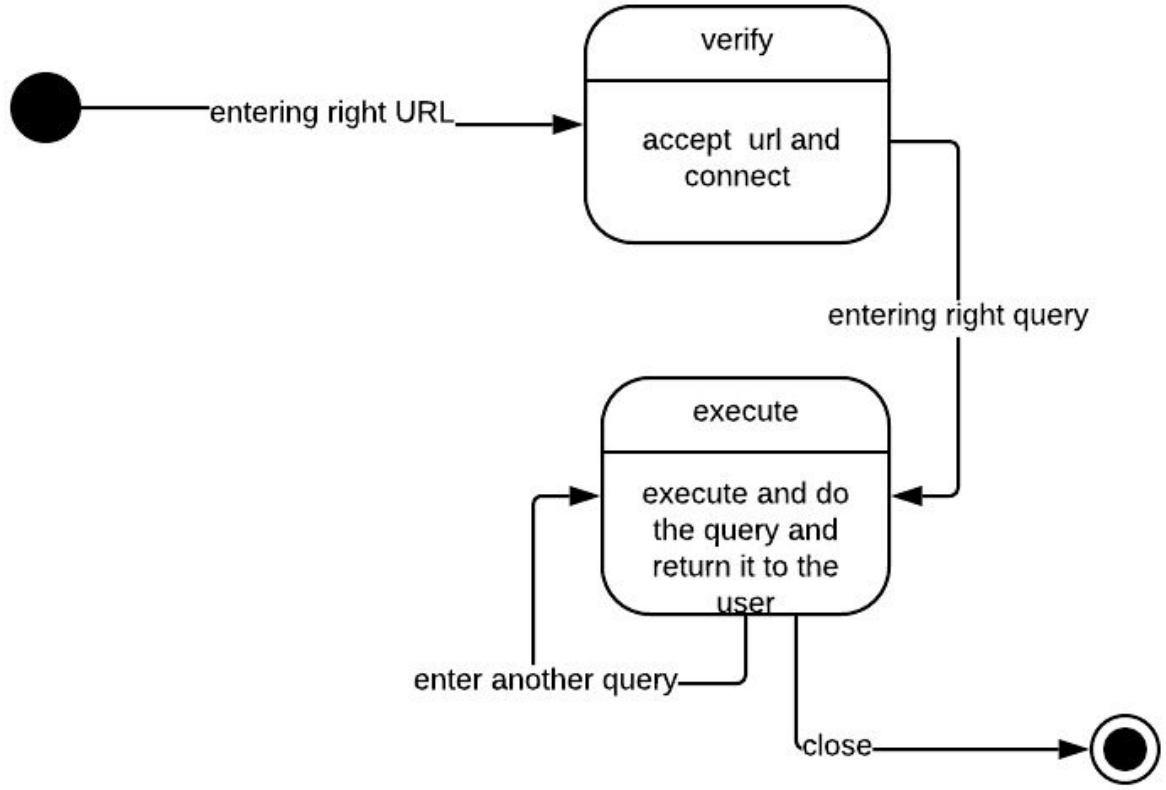
Use case



State digram for 3 scenarios

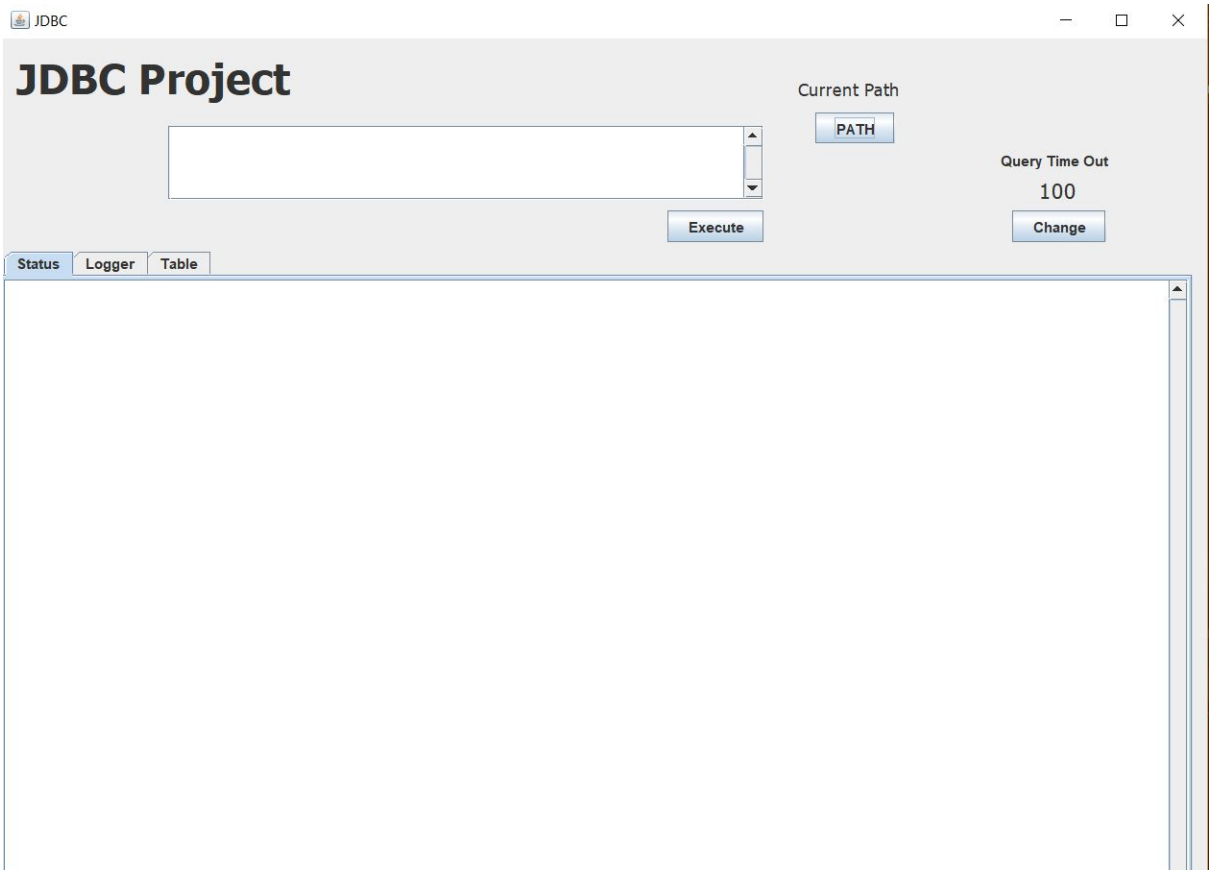



verify
entering right URL
accept url and connect
enter wrong query
execute
throwing SQL exception
enter another query


verify
entering right URL
accept url and connect
entering right query
execute
execute and do the query and return it to the user
enter another query
close

User manual

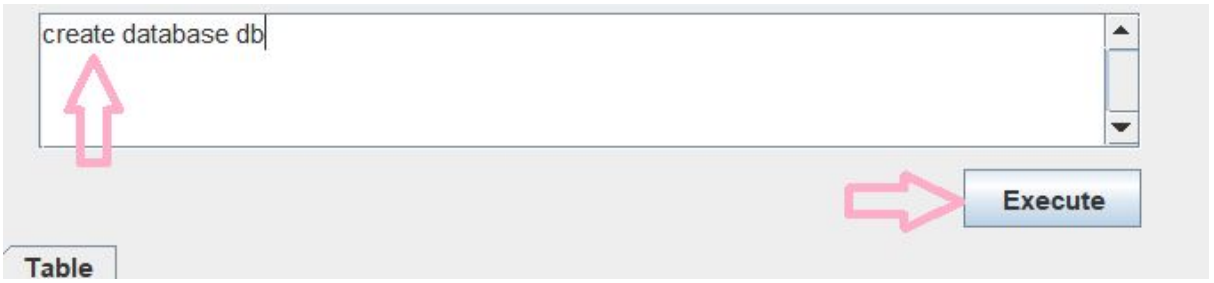
Snapshot from the GUI



The program interface is divided into four main parts.

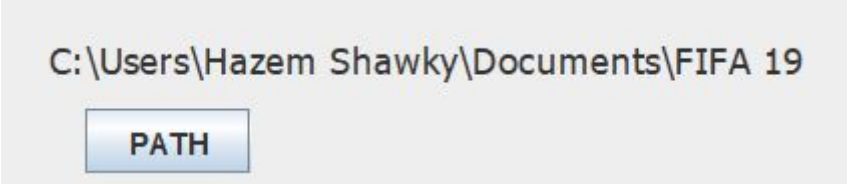
To use the program perform the next steps

1. Type your SQL statements



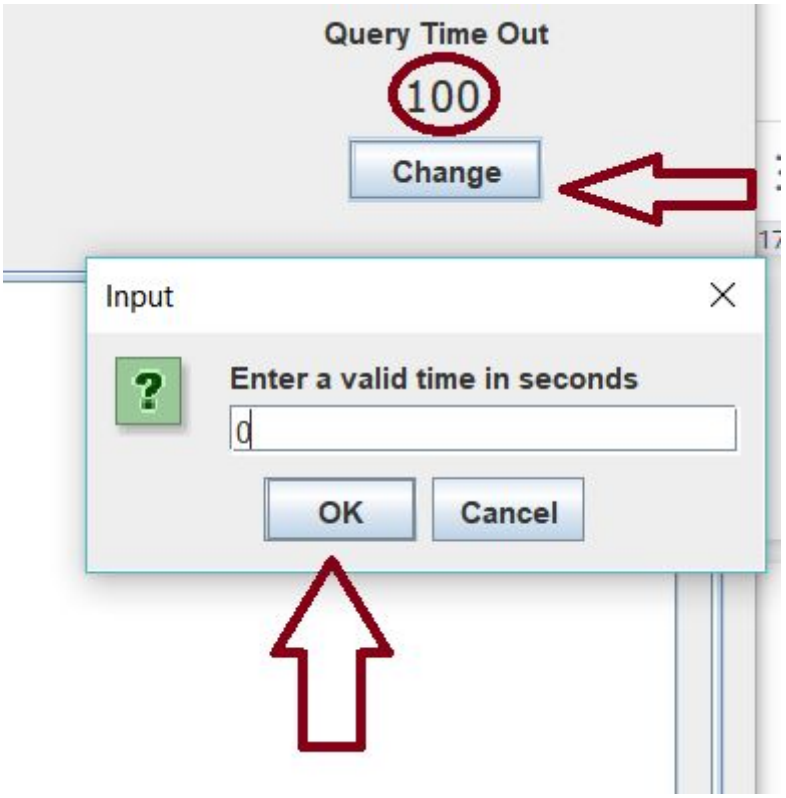
As the previous image shaw you can type your SQL statement and then click on the execute button(you can also type several queries and execute them in one click)

2. Choose your path



You can click on PATH button and change the path of your database from the default path(the path of the program in your computer) to wherever you want.

3. Specify max execution time

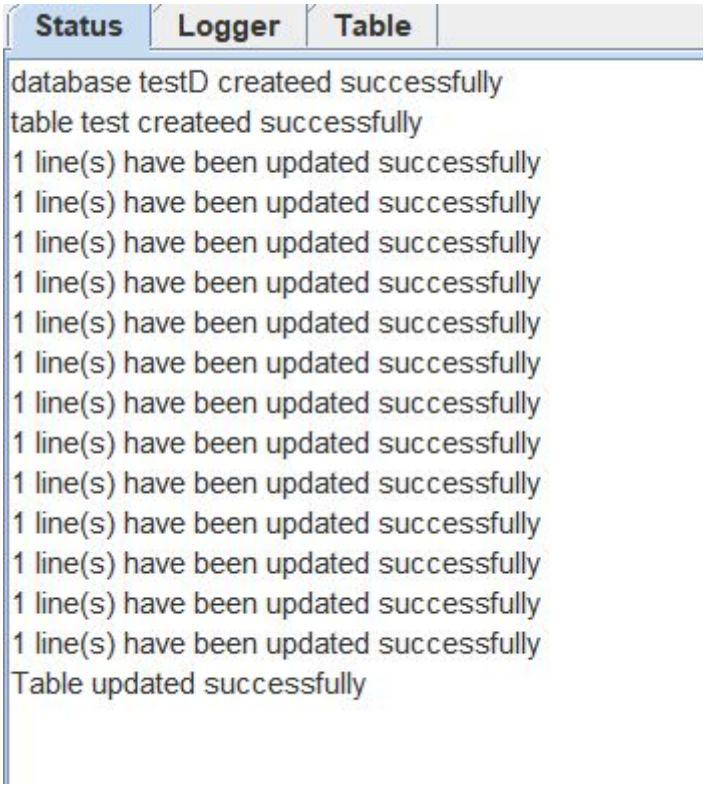


You can click on the change button and type your time(a positive integer) note that time is in seconds.

4. See the result

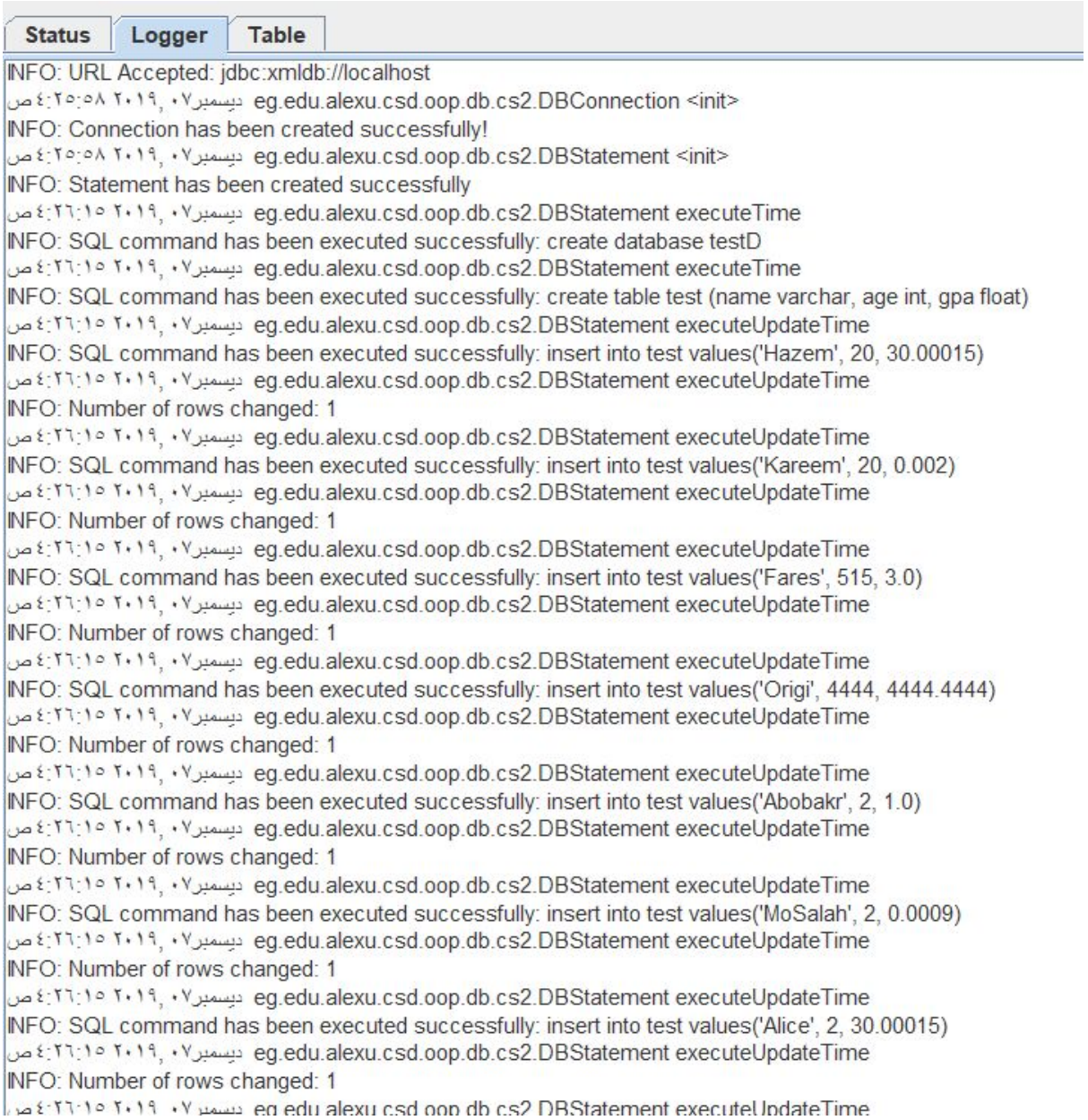
Here you can see the result of the execution of the program

4.1. Status tab



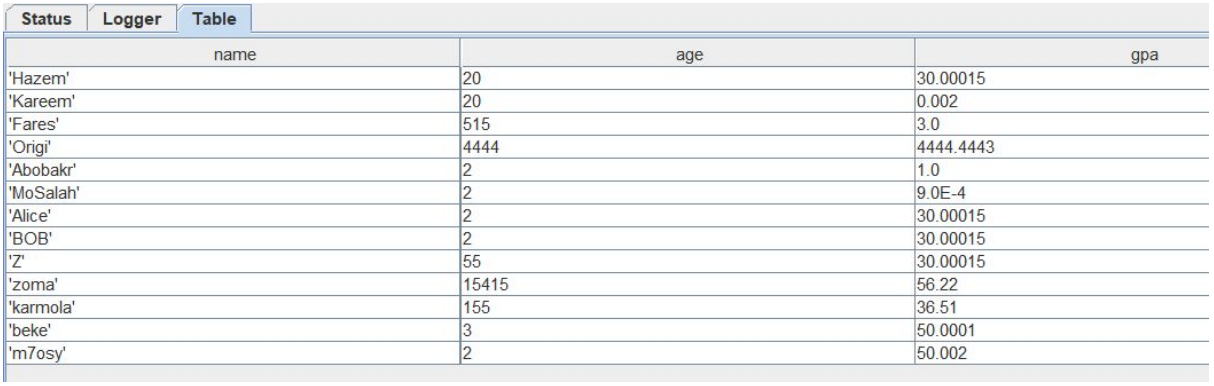
Here is a short statement describe whether a statement executed successfully or not how many lines have been updated and so on..

4.2. Logger tab



It is a more detailed. You can see what happens beyond the front end and keep in touch with any unexpected error and fix it.

4.3. Table tab



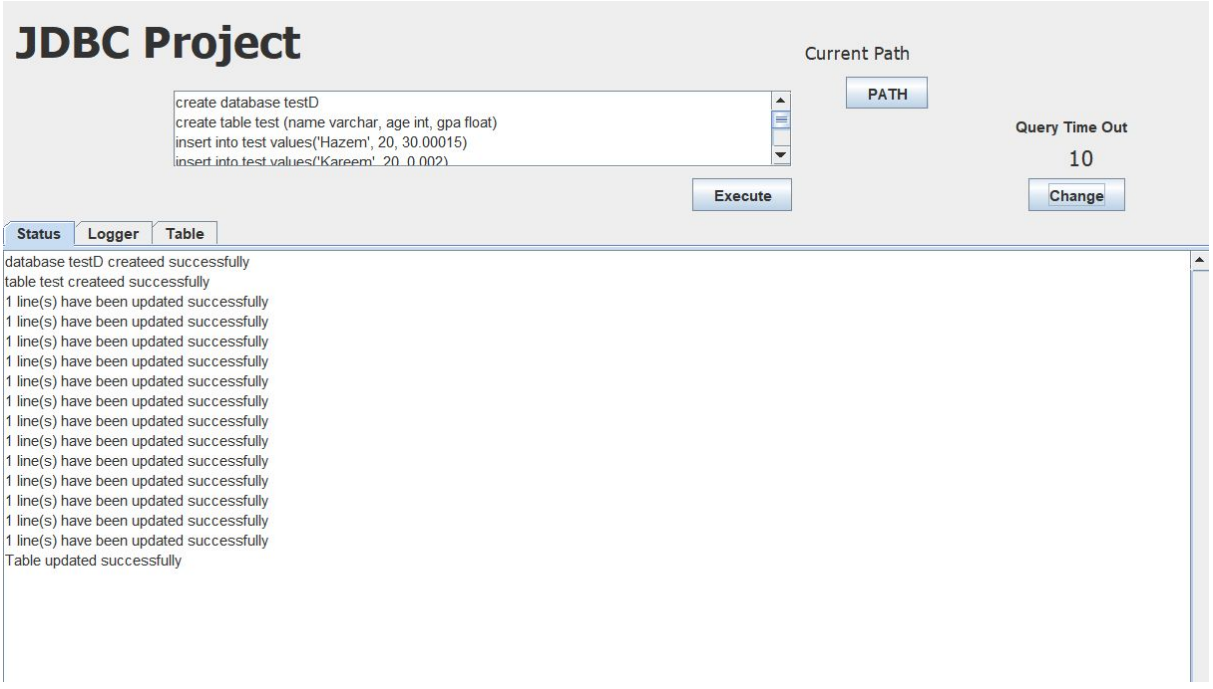
name	age	gpa
'Hazem'	20	30.00015
'Kareem'	20	0.002
'Fares'	515	3.0
'Origi'	4444	4444.4443
'Abobakr'	2	1.0
'MoSalah'	2	9.0E-4
'Alice'	2	30.00015
'BOB'	2	30.00015
'Z'	55	30.00015
'zoma'	15415	56.22
'karmola'	155	36.51
'beke'	3	50.0001
'm7osy'	2	50.002

The most important part kept to the end in this tab you can see the selected table and scroll down with the scroll bar on the right of the screen in case you select too many rows.

Sample run

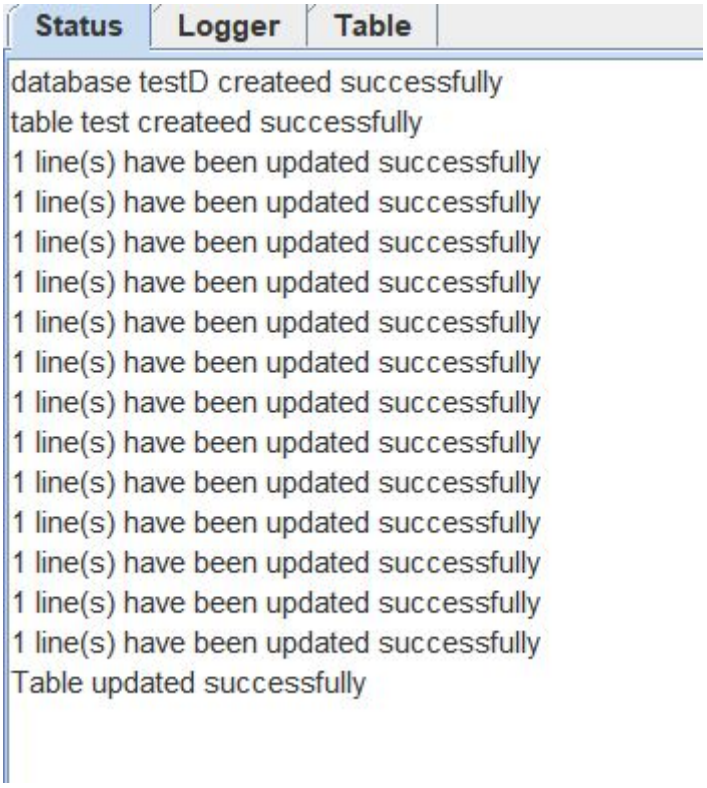
Queries to execute

```
create database testD
create table test (name varchar, age int, gpa float)
insert into test values('Hazem', 20, 30.00015)
insert into test values('Kareem', 20, 0.002)
insert into test values('Fares', 515, 3.0)
insert into test values('Origi', 4444, 4444.4444)
insert into test values('Abobakr', 2, 1.0)
insert into test values('MoSalah', 2, 0.0009)
insert into test values('Alice', 2, 30.00015)
insert into test values('BOB', 2, 30.00015)
insert into test values('Z', 55, 30.00015)
insert into test values('zoma', 15415, 56.22)
insert into test values('karmola', 155, 36.51)
insert into test values('beke', 3, 50.0001)
insert into test values('m7osy', 2, 50.002)
select * from test
```



Results

Status tab



Logger tab

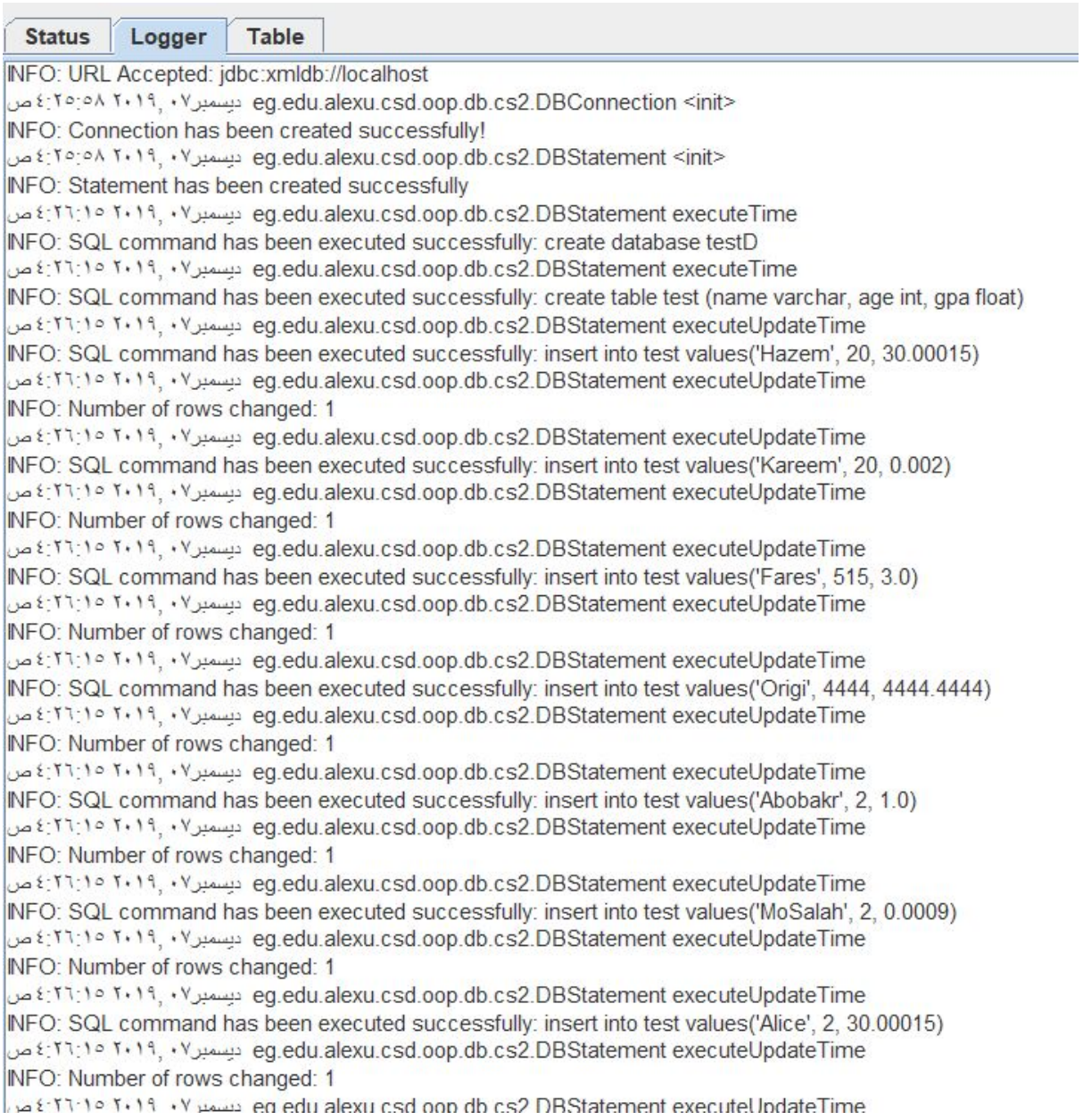

Status
Logger
Table
INFO: URL Accepted: jdbc:xmldb://localhost
٠٧ ديسمبر ٢٠١٩, ٢:٢٥:٥٨ ص eg.edu.alexu.csd.oop.db.cs2.DBConnection <init>
INFO: Connection has been created successfully!
٠٧ ديسمبر ٢٠١٩, ٢:٢٥:٥٨ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement <init>
INFO: Statement has been created successfully
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeTime
INFO: SQL command has been executed successfully: create database testD
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeTime
INFO: SQL command has been executed successfully: create table test (name varchar, age int, gpa float)
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: SQL command has been executed successfully: insert into test values('Hazem', 20, 30.00015)
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: Number of rows changed: 1
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: SQL command has been executed successfully: insert into test values('Kareem', 20, 0.002)
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: Number of rows changed: 1
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: SQL command has been executed successfully: insert into test values('Fares', 515, 3.0)
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: Number of rows changed: 1
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: SQL command has been executed successfully: insert into test values('Origi', 4444, 4444.4444)
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: Number of rows changed: 1
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: SQL command has been executed successfully: insert into test values('Abobakr', 2, 1.0)
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: Number of rows changed: 1
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: SQL command has been executed successfully: insert into test values('MoSalah', 2, 0.0009)
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: Number of rows changed: 1
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: SQL command has been executed successfully: insert into test values('Alice', 2, 30.00015)
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime
INFO: Number of rows changed: 1
٠٧ ديسمبر ٢٠١٩, ٢:٢٦:١٥ ص eg.edu.alexu.csd.oop.db.cs2.DBStatement executeUpdateTime

Table tab

Status | Logger | Table

name	age	gpa
'Hazem'	20	30.00015
'Kareem'	20	0.002
'Fares'	515	3.0
'Origi'	4444	4444.4443
'Abobakr'	2	1.0
'MoSalah'	2	9.0E-4
'Alice'	2	30.00015
'BOB'	2	30.00015
'Z'	55	30.00015
'zoma'	15415	56.22
'karmola'	155	36.51
'beke'	3	50.0001
'm7osy'	2	50.002

when select * from test order by name

name	age	
'Abobakr'	2	1.0
'Alice'	2	30.00015
'beke'	3	50.0001
'BOB'	2	30.00015
'Fares'	515	3.0
'Hazem'	20	30.00015
'Kareem'	20	0.002
'karmola'	155	36.51
'm7osy'	2	50.002
'MoSalah'	2	9.0E-4
'Origi'	4444	4444.4443
'Z'	55	30.00015
'zoma'	15415	56.22

when select * from test order by age asc & gpa desc

name	age	gpa
'm7osy'	2	50.002
'Alice'	2	30.00015
'BOB'	2	30.00015
'Abobakr'	2	1.0
'MoSalah'	2	9.0E-4
'beke'	3	50.0001
'Hazem'	20	30.00015
'Kareem'	20	0.002
'Z'	55	30.00015
'karmola'	155	36.51
'Fares'	515	3.0
'Origi'	4444	4444.4443
'zoma'	15415	56.22