CS221: Programming II

Database Management System

Report

Team Members

Abdelrahman Ibrahim Soliman	(35)
Abdelrahman Sami	(38)
Mahmoud Gamal El-sayed	(61)
Mahmoud Tarek Samir	(62)

Table of Contents

Introduction

Keeping data permanently on a non-volatile memory is an important issue in software development. Saving data in files is the solution thus database management systems appeared.

This is a database management system that allows the user to easily save data in tables and manipulate it by: insert new row, edit existing table, delete element from table, select items from table, drop table and create new tables in different database files.

Design Overview

The design is divided into 4 major parts:

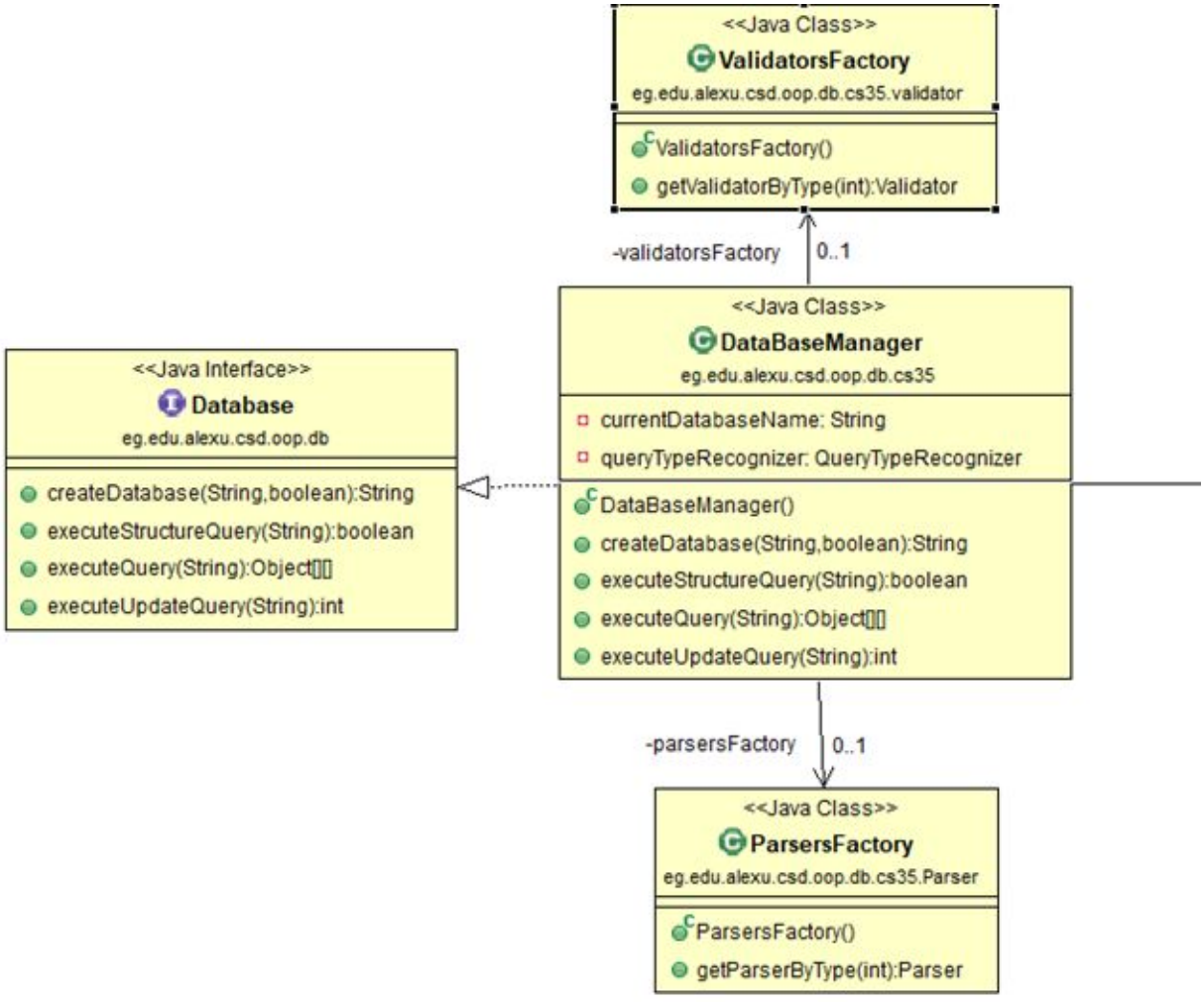
- Validation of the coming query.
- Parsing of the validated query.
- Synthesizing a command to perform the task.
- Executing the command.

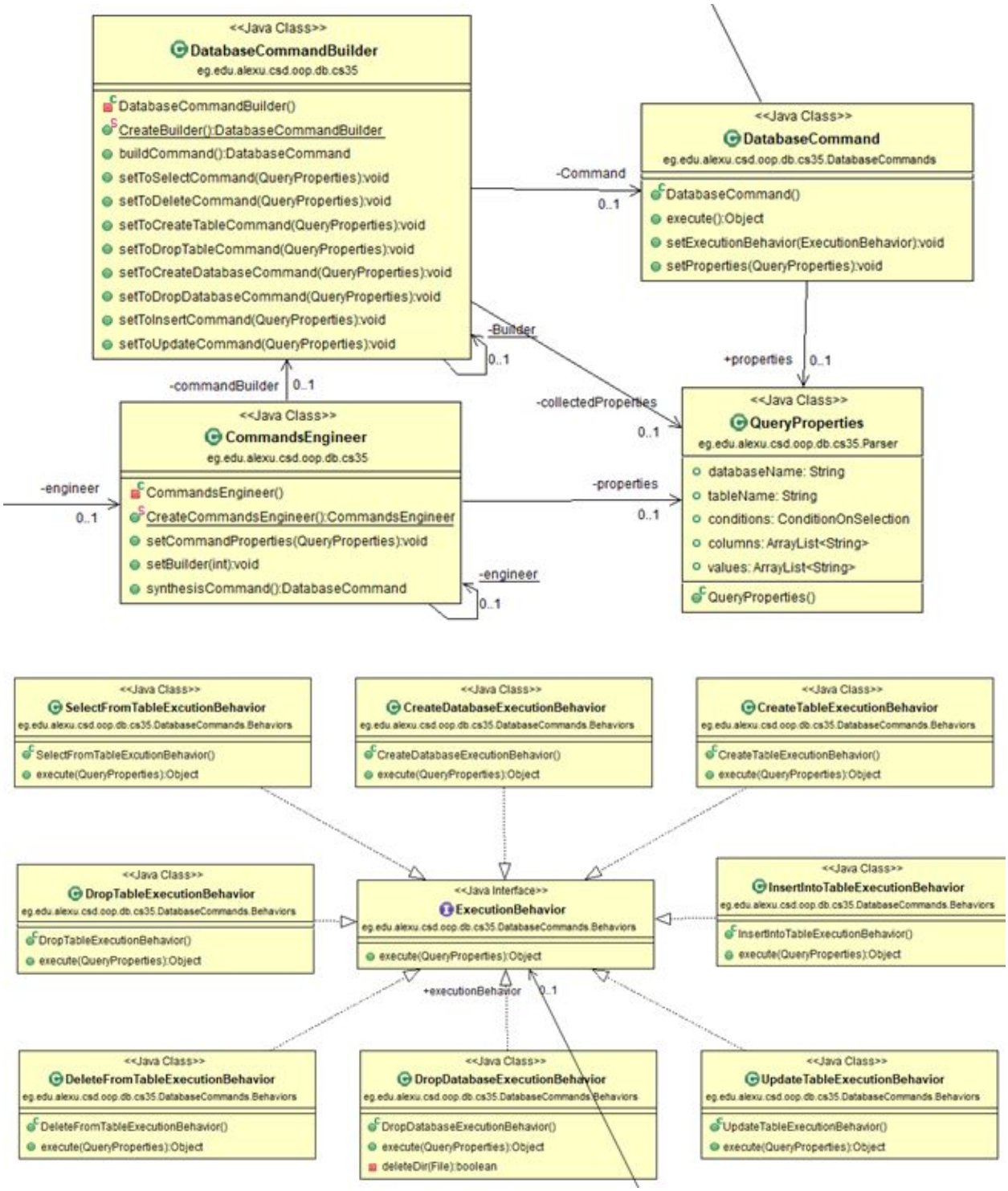
The validation part is made using “Pattern” class which use certain regex to validate each query. The validation of each query is kept in its own class and all the validator classes implements the same interface so by the means of a factory pattern any of this validators can be injected into the main class. There are some utilities common for all the validators so they are encapsulated in their own class. After validation parsing process takes place.
The parsers part is where the query is cut into parts where these parts form the query properties. The query properties is kept in a tailor made class for this purpose only and then is returned to continue and synthesis the command.
An engineer ”director” informs the command builder to build a certain command depending on the type of query sent. The command is built by injecting certain behavior into its structure “Strategy DP” where the behaviors are the lowest level operations in the design where the contact with files takes place.
Executing the command after synthesizing will perform the injected behavior and return certain values.

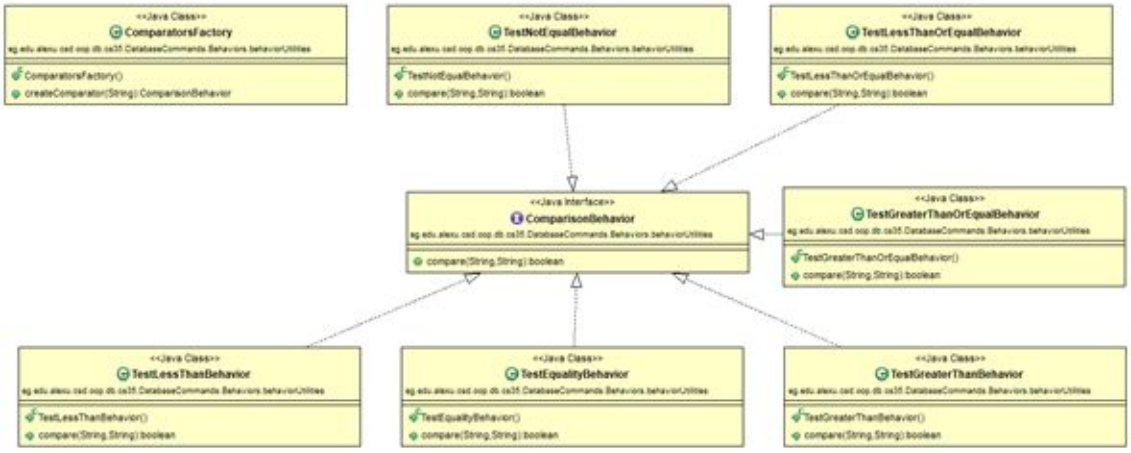
Design Patterns Used

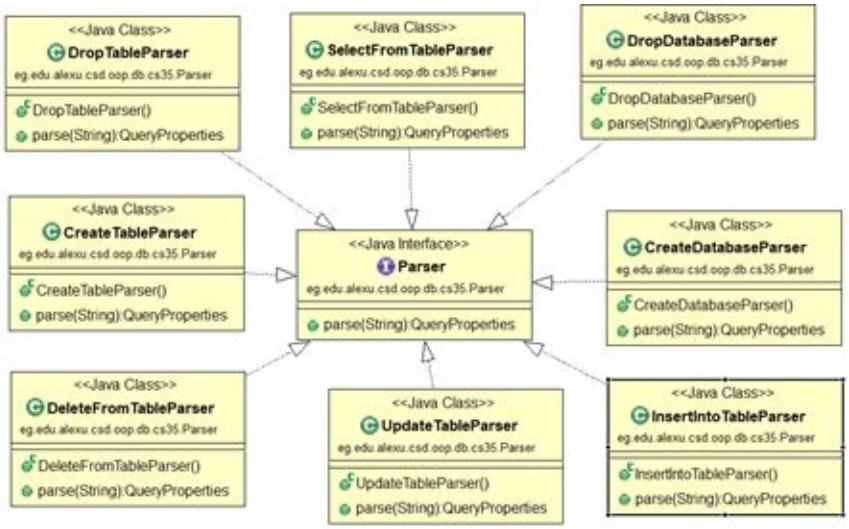
- **Factory DP**: Used three times in the design to create validators and parsers; One with low level to create comparison behaviors depending on a certain condition.
- **Builder DP**: Used in the manufacturing of the command to be executed.
- **Strategy DP**: Used in injecting the execution behavior into the constructed command
- **Singleton DP**: Only one engineer and one builder are allowed at runtime.
- **Command DP**: “not implemented literally but with the same logic”. Common class with “execute ()” method that carries out certain algorithm.

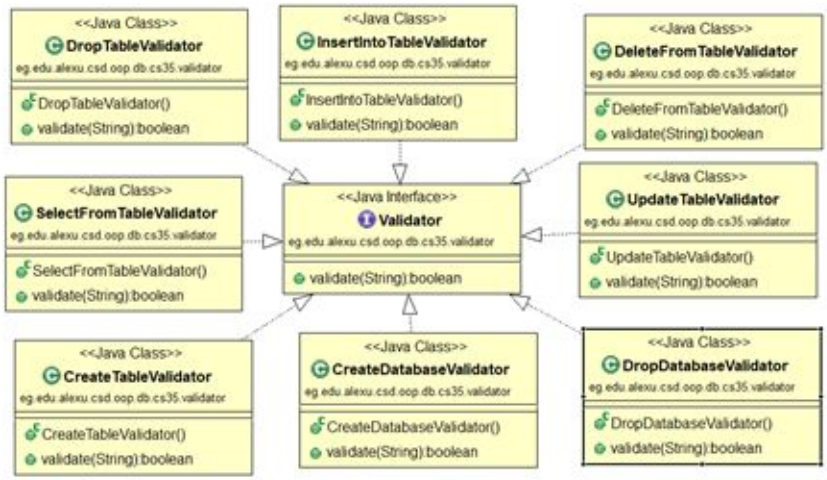
UML Diagrams and Classes Relations

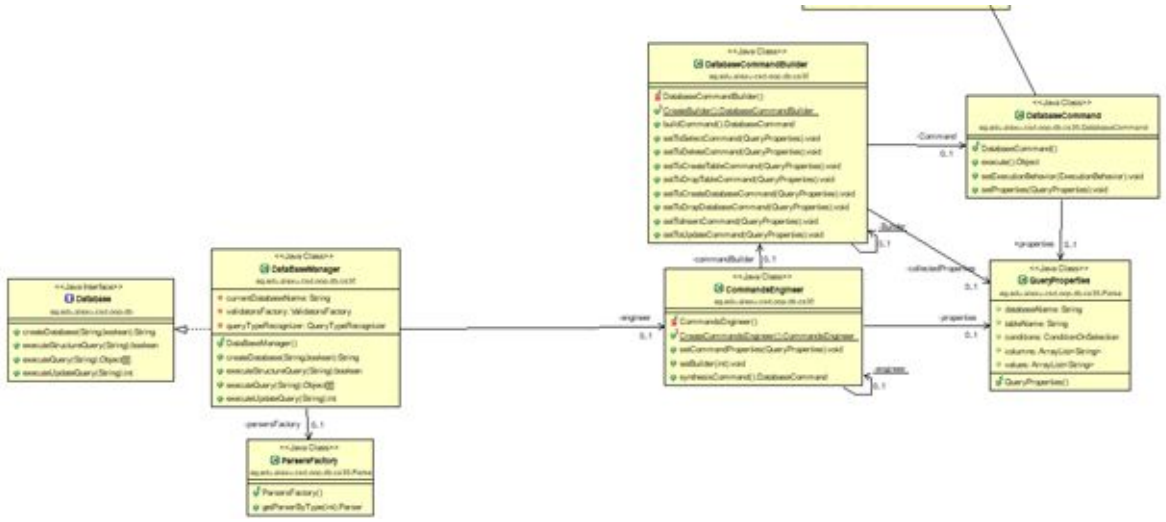


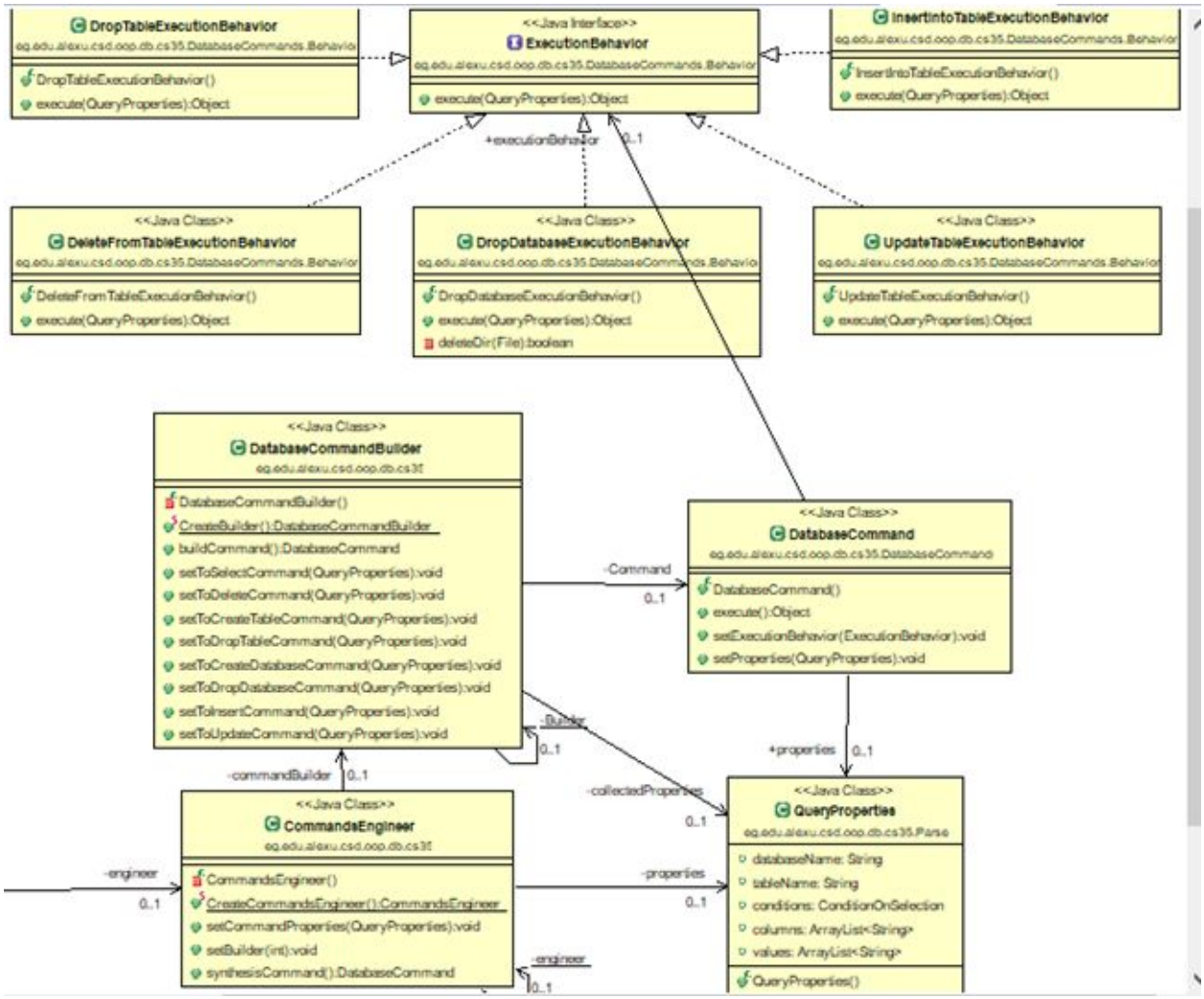

<<Java Class>>
DatabaseCommandBuilder
eg.edu.alexu.csd.oop.db.cs35
DatabaseCommandBuilder()
CreateBuilder():DatabaseCommandBuilder
buildCommand():DatabaseCommand
setToSelectCommand(QueryProperties):void
setToDeleteCommand(QueryProperties):void
setToCreateTableCommand(QueryProperties):void
setToDropTableCommand(QueryProperties):void
setToCreateDatabaseCommand(QueryProperties):void
setToDropDatabaseCommand(QueryProperties):void
setToInsertCommand(QueryProperties):void
setToUpdateCommand(QueryProperties):void
-Command
0..1
<<Java Class>>
DatabaseCommand
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands
DatabaseCommand()
execute():Object
setExecutionBehavior(ExecutionBehavior):void
setProperties(QueryProperties):void
-Builder
0..1
+properties
0..1
-commandBuilder
0..1
-collectedProperties
0..1
<<Java Class>>
CommandsEngineer
eg.edu.alexu.csd.oop.db.cs35
CommandsEngineer()
CreateCommandsEngineer():CommandsEngineer
setCommandProperties(QueryProperties):void
setBuilder(int):void
synthesisCommand():DatabaseCommand
-engineer
0..1
-properties
0..1
<<Java Class>>
QueryProperties
eg.edu.alexu.csd.oop.db.cs35.Parser
databaseName: String
tableName: String
conditions: ConditionOnSelection
columns: ArrayList<String>
values: ArrayList<String>
QueryProperties()
-engineer
0..1
<<Java Class>>
SelectFromTableExcutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behaviors
SelectFromTableExcutionBehavior()
execute(QueryProperties):Object
<<Java Class>>
CreateDatabaseExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behaviors
CreateDatabaseExecutionBehavior()
execute(QueryProperties):Object
<<Java Class>>
CreateTableExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behaviors
CreateTableExecutionBehavior()
execute(QueryProperties):Object
<<Java Class>>
DropTableExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behaviors
DropTableExecutionBehavior()
execute(QueryProperties):Object
<<Java Interface>>
ExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behaviors
execute(QueryProperties):Object
<<Java Class>>
InsertIntoTableExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behaviors
InsertIntoTableExecutionBehavior()
execute(QueryProperties):Object
+executionBehavior
0..1
<<Java Class>>
DeleteFromTableExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behaviors
DeleteFromTableExecutionBehavior()
execute(QueryProperties):Object
<<Java Class>>
DropDatabaseExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behaviors
DropDatabaseExecutionBehavior()
execute(QueryProperties):Object
deleteDir(File):boolean
<<Java Class>>
UpdateTableExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behaviors
UpdateTableExecutionBehavior()
execute(QueryProperties):Object


<<Java Class>>
ComparatorsFactory
ComparatorsFactory()
createComparator(String):ComparisonBehavior
<<Java Class>>
TestNotEqualBehavior
TestNotEqualBehavior()
compare(String,String):boolean
<<Java Class>>
TestLessThanOrEqualBehavior
TestLessThanOrEqualBehavior()
compare(String,String):boolean
<<Java Interface>>
ComparisoeBehavior
compare(String,String):boolean
<<Java Class>>
TestGreaterThanOrEqualBehavior
TestGreaterThanOrEqualBehavior()
compare(String,String):boolean
<<Java Class>>
TestLessThanBehavior
TestLessThanBehavior()
compare(String,String):boolean
<<Java Class>>
TestEqualityBehavior
TestEqualityBehavior()
compare(String,String):boolean
<<Java Class>>
TestGreaterThanBehavior
TestGreaterThanBehavior()
compare(String,String):boolean


<<Java Class>>
DropTableParser
eg.edu.alexu.csd.oop.db.cs35.Parser
DropTableParser()
parse(String):QueryProperties
<<Java Class>>
SelectFromTableParser
eg.edu.alexu.csd.oop.db.cs35.Parser
SelectFromTableParser()
parse(String):QueryProperties
<<Java Class>>
DropDatabaseParser
eg.edu.alexu.csd.oop.db.cs35.Parser
DropDatabaseParser()
parse(String):QueryProperties
<<Java Class>>
CreateTableParser
eg.edu.alexu.csd.oop.db.cs35.Parser
CreateTableParser()
parse(String):QueryProperties
<<Java Interface>>
Parser
eg.edu.alexu.csd.oop.db.cs35.Parser
parse(String):QueryProperties
<<Java Class>>
CreateDatabaseParser
eg.edu.alexu.csd.oop.db.cs35.Parser
CreateDatabaseParser()
parse(String):QueryProperties
<<Java Class>>
DeleteFromTableParser
eg.edu.alexu.csd.oop.db.cs35.Parser
DeleteFromTableParser()
parse(String):QueryProperties
<<Java Class>>
UpdateTableParser
eg.edu.alexu.csd.oop.db.cs35.Parser
UpdateTableParser()
parse(String):QueryProperties
<<Java Class>>
InsertIntoTableParser
eg.edu.alexu.csd.oop.db.cs35.Parser
InsertIntoTableParser()
parse(String):QueryProperties

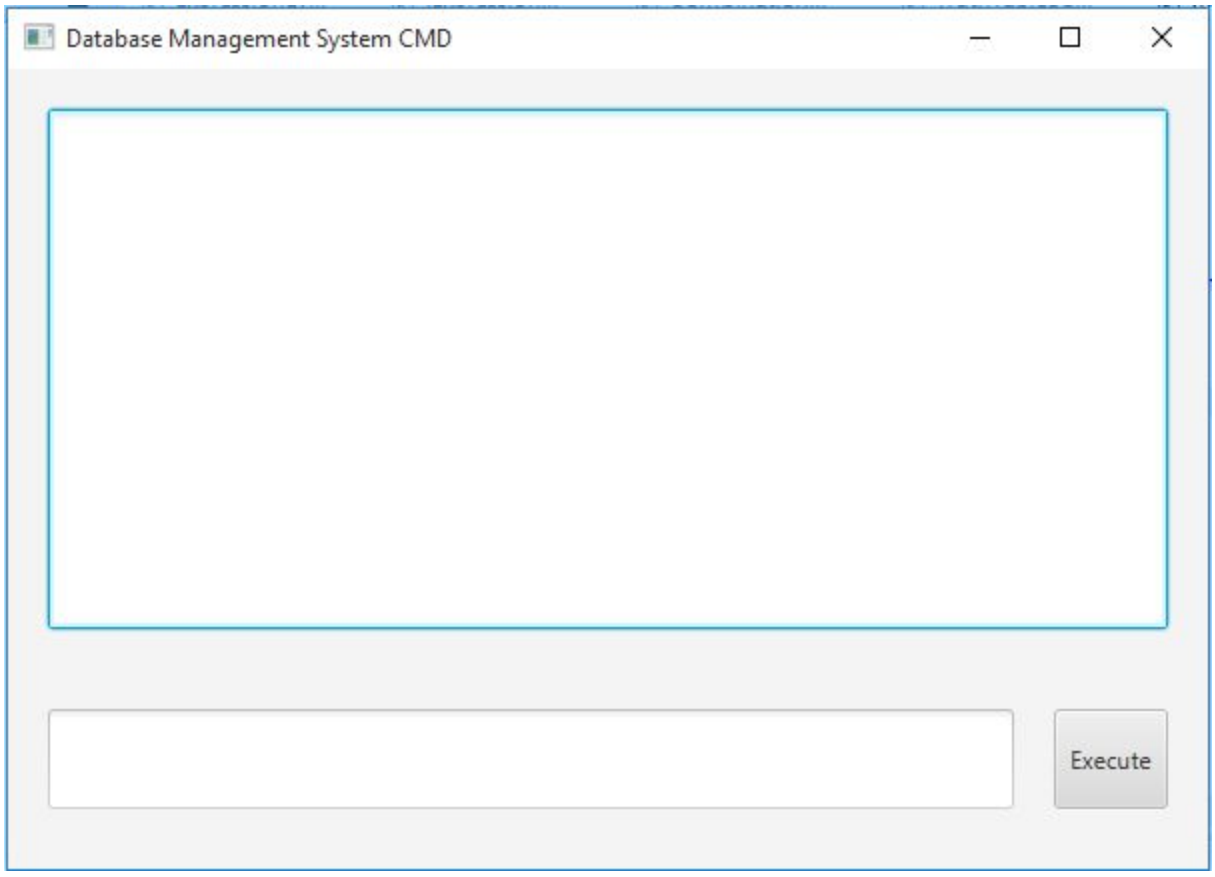

<<Java Class>>
DropTableValidator
eg.edu.alexu.csd.oop.db.cs35.validator
DropTableValidator()
validate(String):boolean
<<Java Class>>
InsertIntoTableValidator
eg.edu.alexu.csd.oop.db.cs35.validator
InsertIntoTableValidator()
validate(String):boolean
<<Java Class>>
DeleteFromTableValidator
eg.edu.alexu.csd.oop.db.cs35.validator
DeleteFromTableValidator()
validate(String):boolean
<<Java Class>>
SelectFromTableValidator
eg.edu.alexu.csd.oop.db.cs35.validator
SelectFromTableValidator()
validate(String):boolean
<<Java Interface>>
Validator
eg.edu.alexu.csd.oop.db.cs35.validator
validate(String):boolean
<<Java Class>>
UpdateTableValidator
eg.edu.alexu.csd.oop.db.cs35.validator
UpdateTableValidator()
validate(String):boolean
<<Java Class>>
CreateTableValidator
eg.edu.alexu.csd.oop.db.cs35.validator
CreateTableValidator()
validate(String):boolean
<<Java Class>>
CreateDatabaseValidator
eg.edu.alexu.csd.oop.db.cs35.validator
CreateDatabaseValidator()
validate(String):boolean
<<Java Class>>
DropDatabaseValidator
eg.edu.alexu.csd.oop.db.cs35.validator
DropDatabaseValidator()
validate(String):boolean



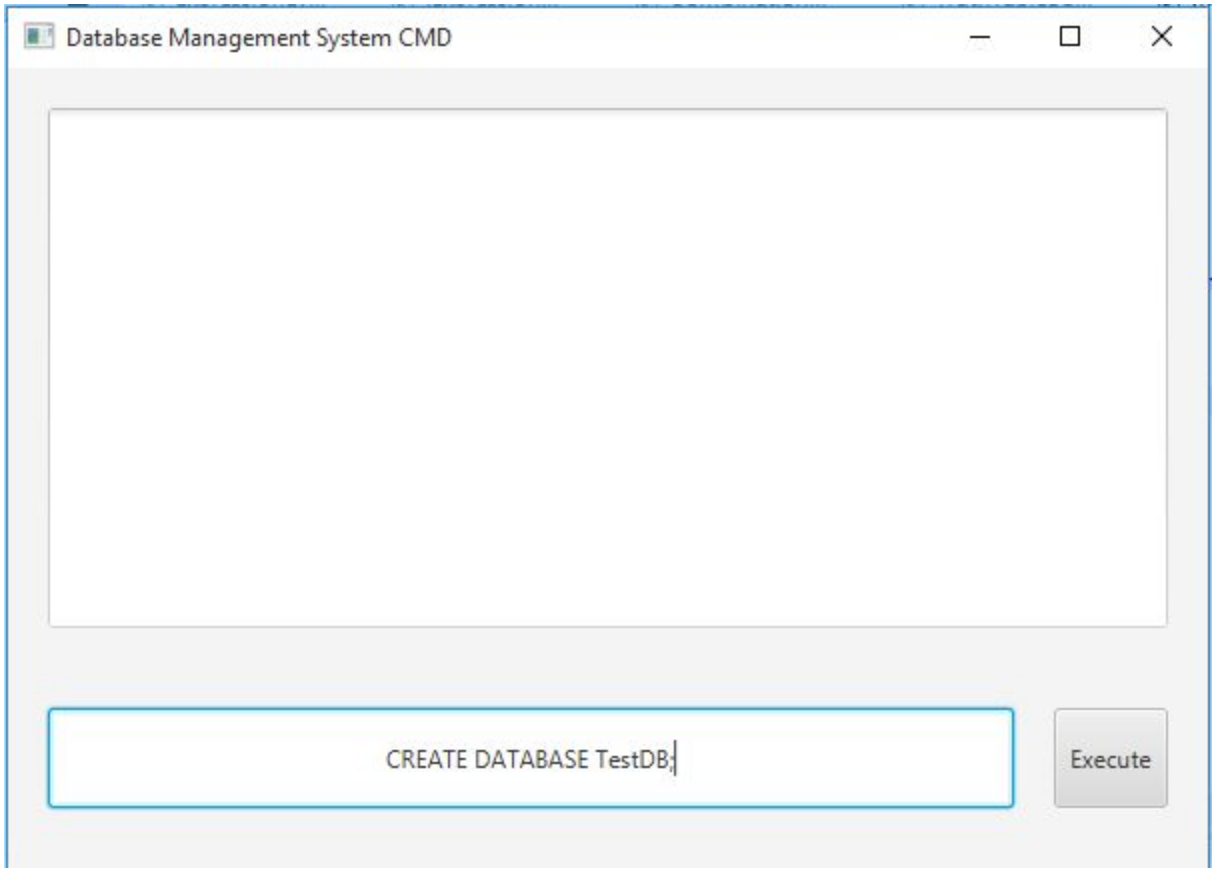

DropTableExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behavior
DropTableExecutionBehavior()
execute(QueryProperties):Object
<<Java Interface>>
ExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behavior
execute(QueryProperties):Object
InsertintoTableExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behavior
InsertIntoTableExecutionBehavior()
execute(QueryProperties):Object
+executionBehavior
0..1
<<Java Class>>
DeleteFromTableExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behavior
DeleteFromTableExecutionBehavior()
execute(QueryProperties):Object
<<Java Class>>
DropDatabaseExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behavior
DropDatabaseExecutionBehavior()
execute(QueryProperties):Object
deleteDir(File):boolean
<<Java Class>>
UpdateTableExecutionBehavior
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommands.Behavior
UpdateTableExecutionBehavior()
execute(QueryProperties):Object
<<Java Class>>
DatabaseCommandBuilder
eg.edu.alexu.csd.oop.db.cs35
DatabaseCommandBuilder()
CreateBuilder():DatabaseCommandBuilder
buildCommand():DatabaseCommand
setToSelectCommand(QueryProperties):void
setToDeleteCommand(QueryProperties):void
setToCreateTableCommand(QueryProperties):void
setToDropTableCommand(QueryProperties):void
setToCreateDatabaseCommand(QueryProperties):void
setToDropDatabaseCommand(QueryProperties):void
setToInsertCommand(QueryProperties):void
setToUpdateCommand(QueryProperties):void
-Command
0..1
<<Java Class>>
DatabaseCommand
eg.edu.alexu.csd.oop.db.cs35.DatabaseCommand
DatabaseCommand()
execute():Object
setExecutionBehavior(ExecutionBehavior):void
setProperties(QueryProperties):void
-Builder
0..1
+properties
0..1
-commandBuilder
0..1
-collectedProperties
0..1
<<Java Class>>
CommandsEngineer
eg.edu.alexu.csd.oop.db.cs35
-engineer
0..1
CommandsEngineer()
CreateCommandsEngineer():CommandsEngineer
setCommandProperties(QueryProperties):void
setBuilder(int):void
synthesisCommand():DatabaseCommand
-properties
0..1
-engineer
0..1
<<Java Class>>
QueryProperties
eg.edu.alexu.csd.oop.db.cs35.Parse
databaseName: String
tableName: String
conditions: ConditionOnSelection
columns: ArrayList<String>
values: ArrayList<String>
QueryProperties()

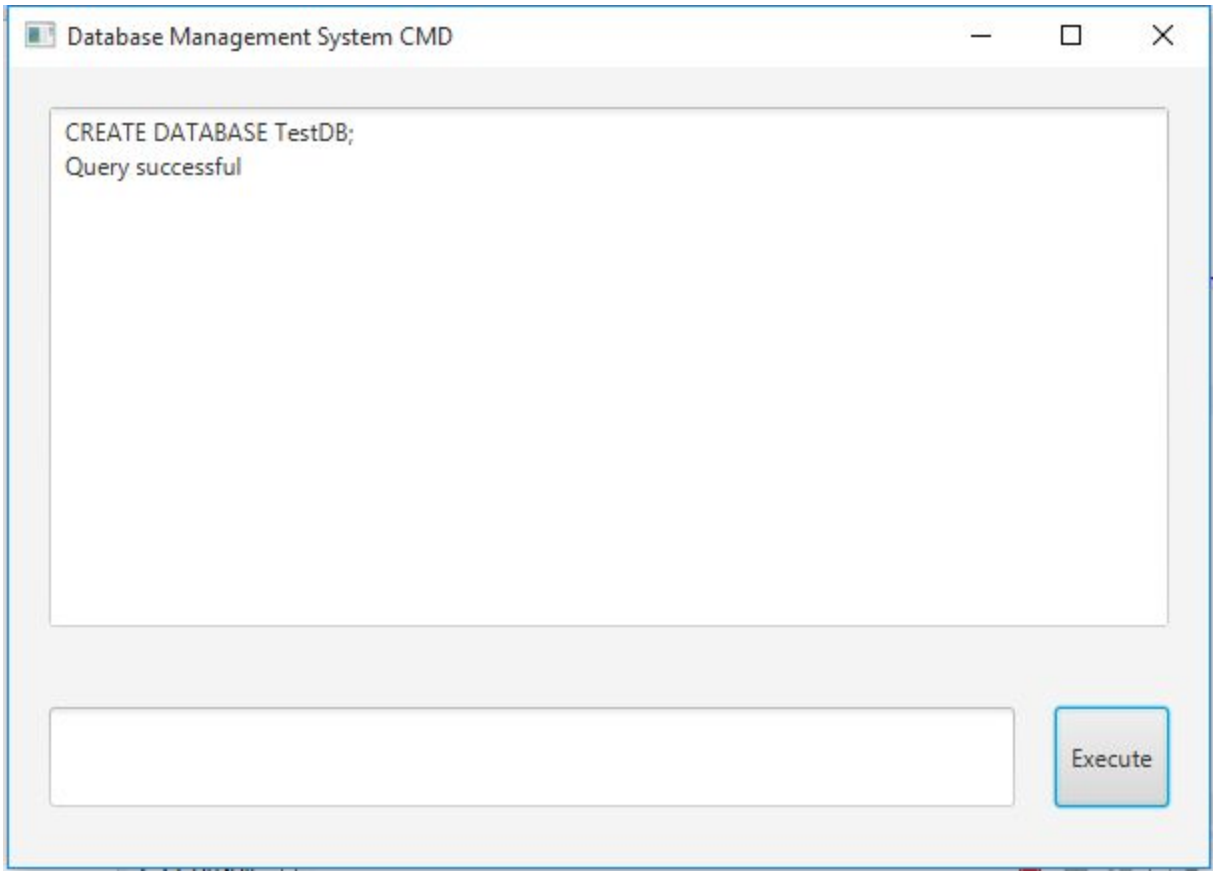
GUI Snapshots

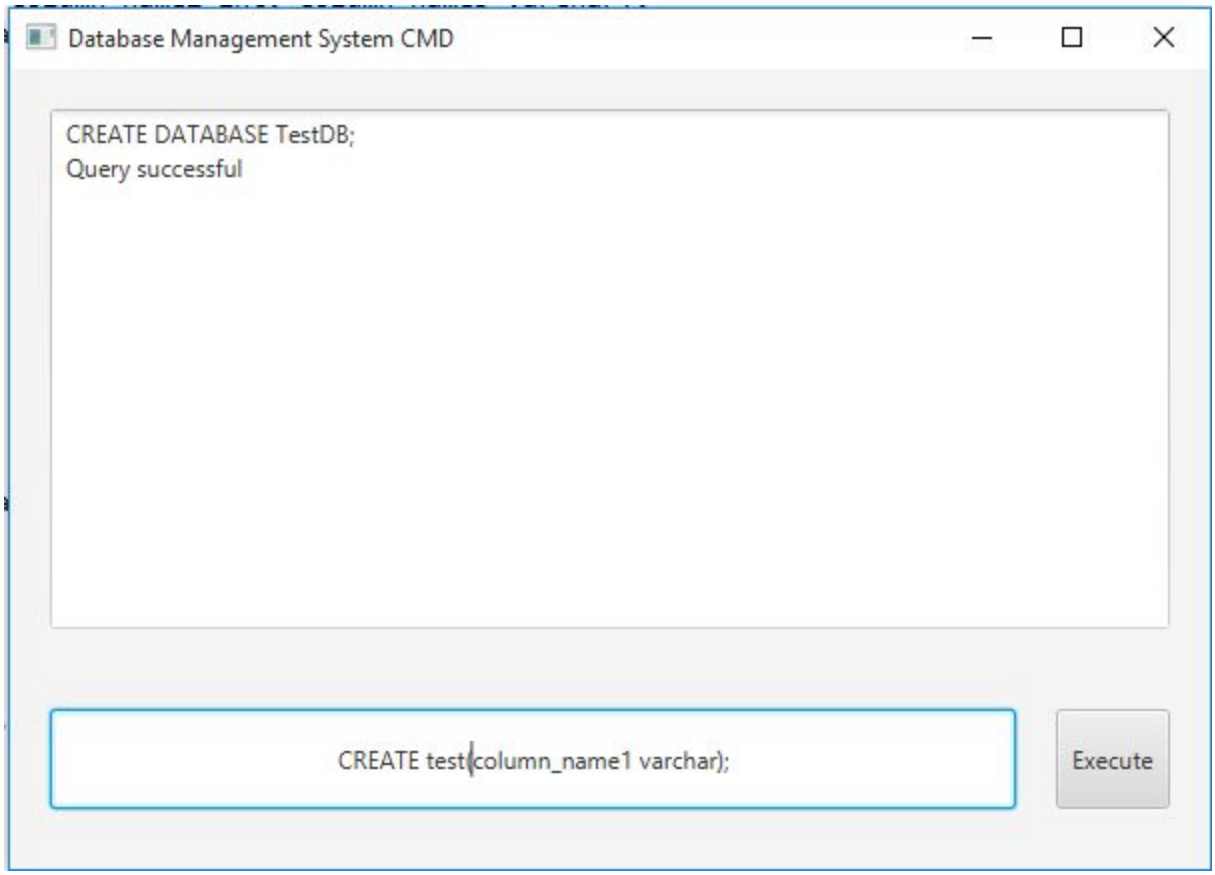
When you open the program, the following window will appear

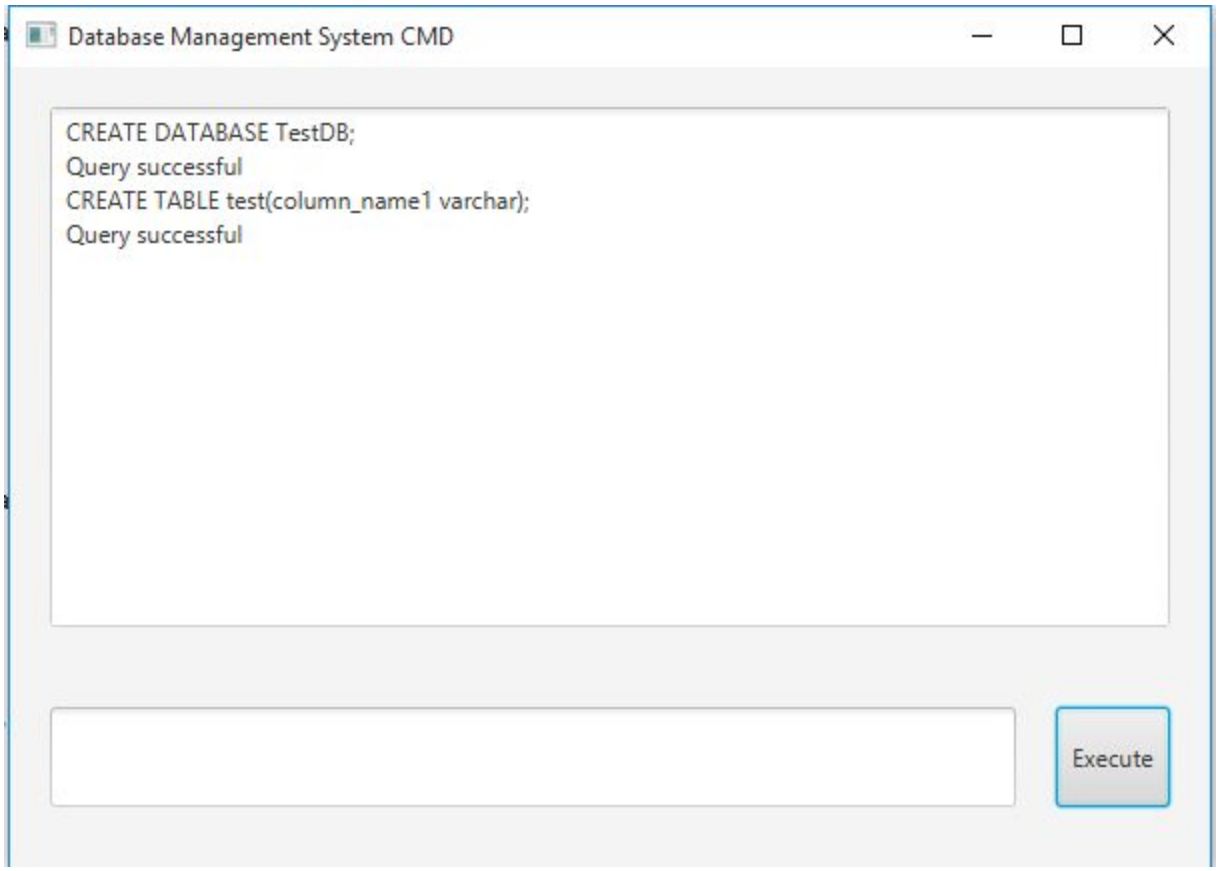


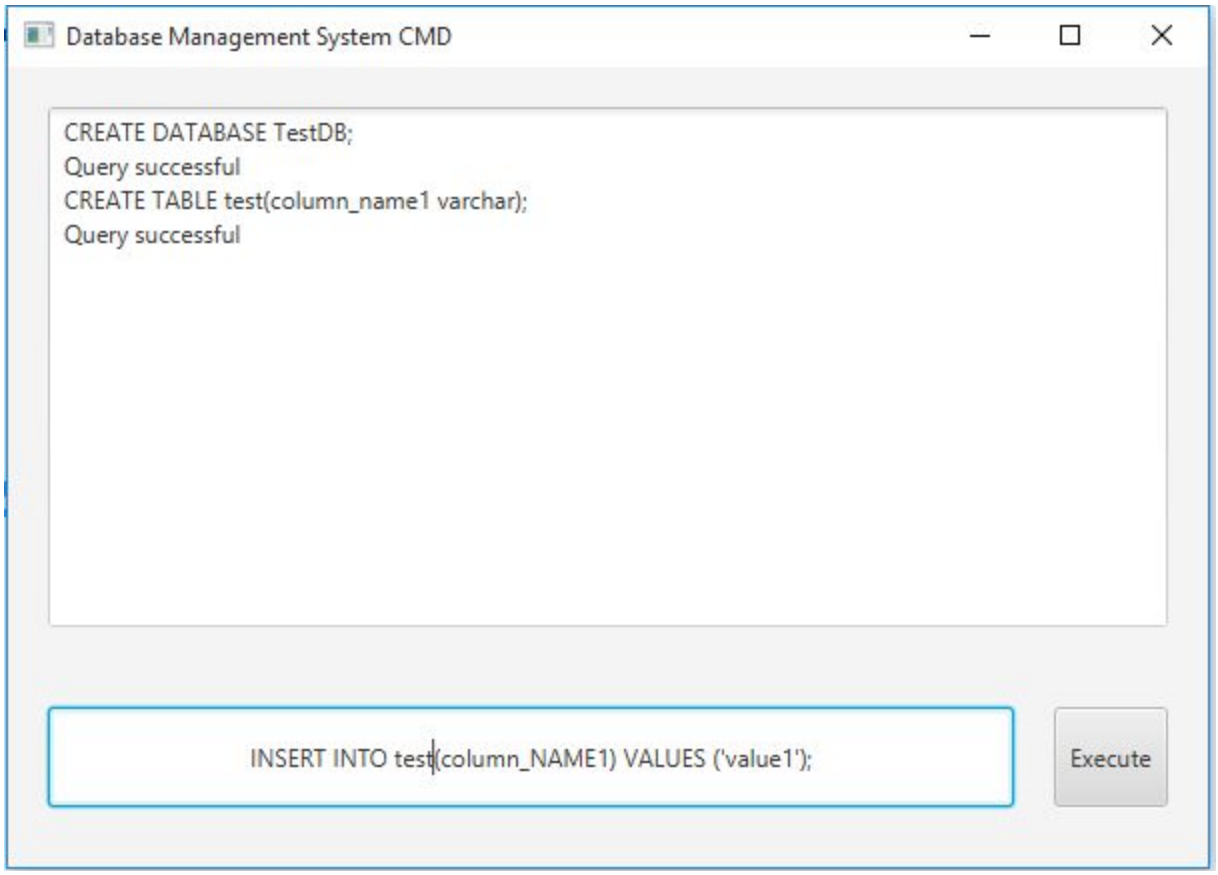
You can write the query in the text field at the bottom and the query status and result will be shown in the above panel, the following figures illustrates this

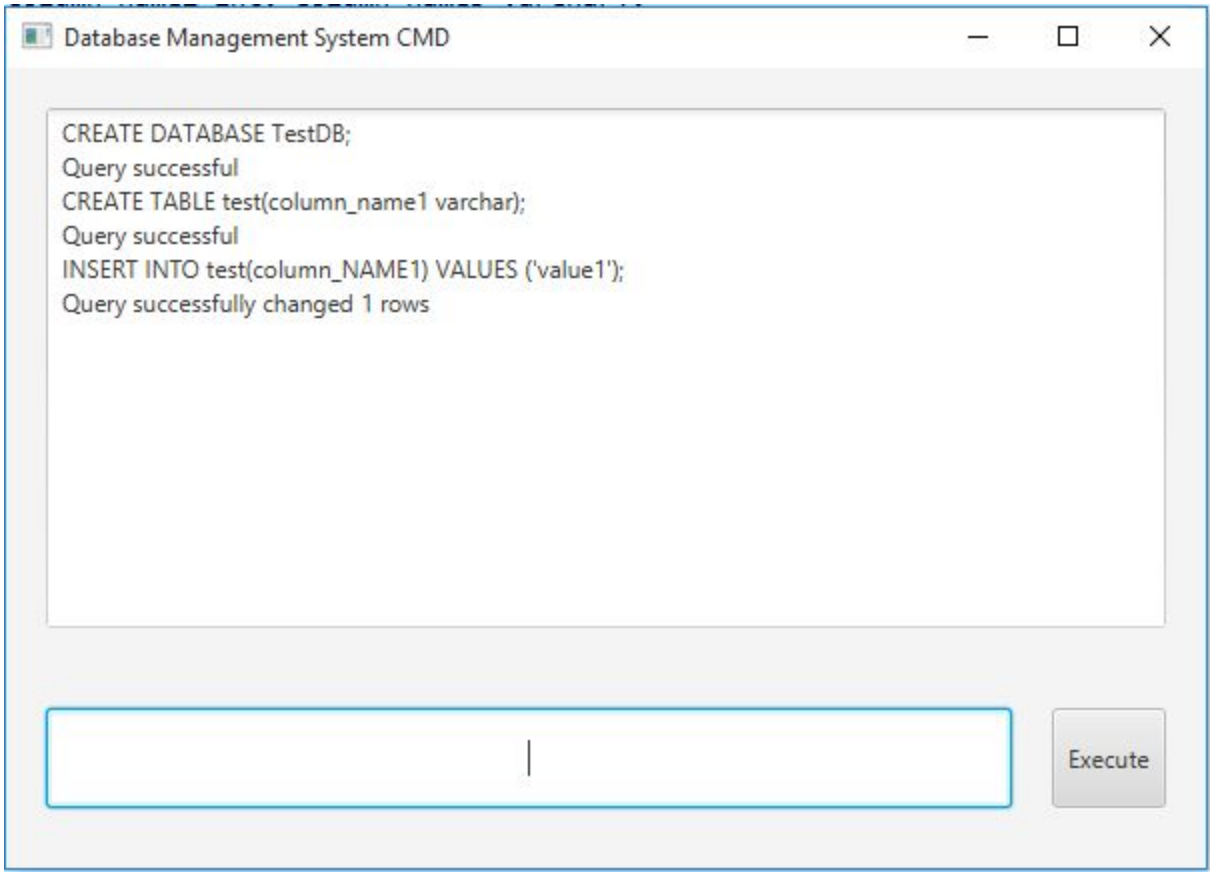


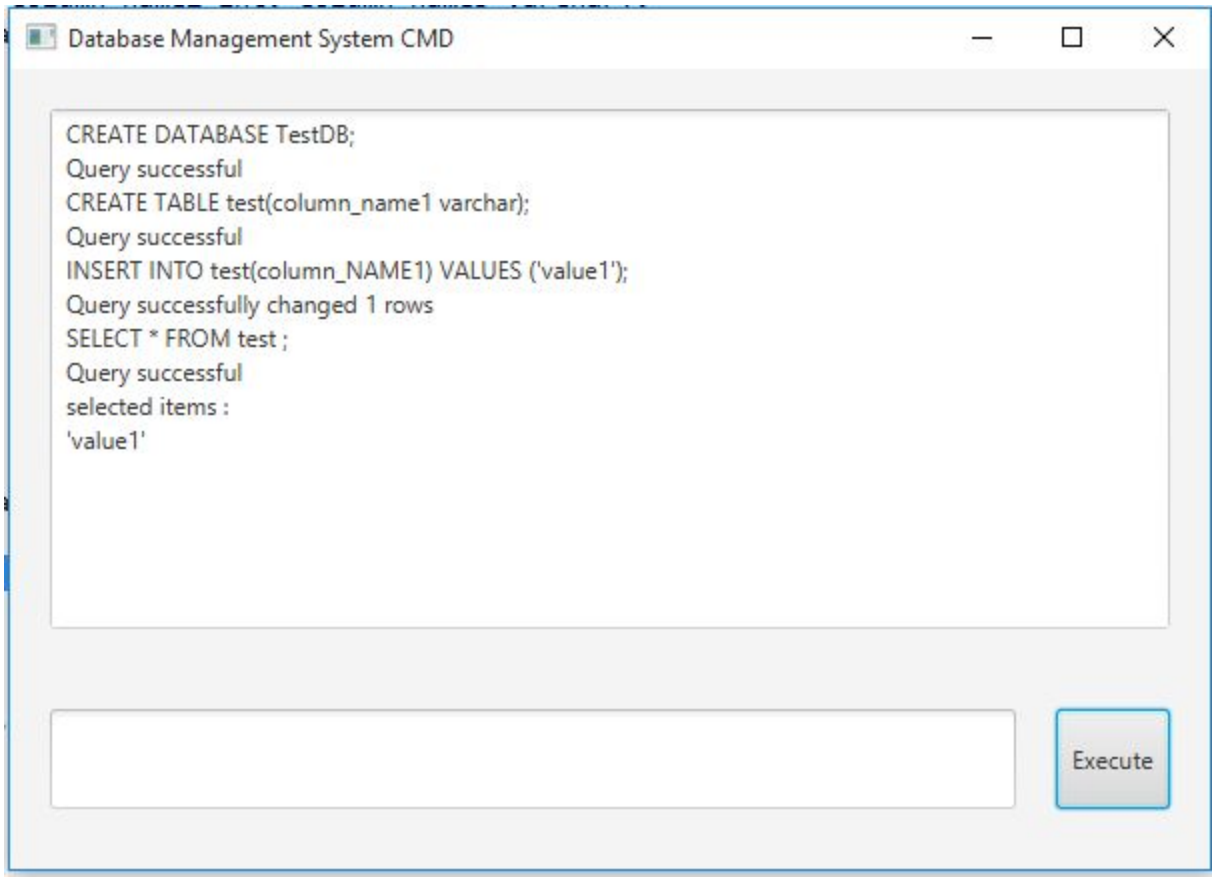

Database Management System CMD
CREATE DATABASE TestDB;
Query successful
Execute


Database Management System CMD
CREATE DATABASE TestDB;
Query successful
CREATE test(column_name1 varchar);
Execute


Database Management System CMD
CREATE DATABASE TestDB;
Query successful
CREATE TABLE test(column_name1 varchar);
Query successful
Execute


Database Management System CMD
CREATE DATABASE TestDB;
Query successful
CREATE TABLE test(column_name1 varchar);
Query successful
INSERT INTO test(column_NAME1) VALUES ('value1');
Execute


Database Management System CMD
CREATE DATABASE TestDB;
Query successful
CREATE TABLE test(column_name1 varchar);
Query successful
INSERT INTO test(column_NAME1) VALUES ('value1');
Query successfully changed 1 rows
Execute



References

- Head First Design Patterns. [Book]
- Design Patterns: Elements of Reusable Object-Oriented Software. [Book]
- https://www.w3schools.com/sql/default.asp
- http://www.vogella.com/tutorials/JavaRegularExpressions/article.html