Simple DBMS

OBJECT-ORIANTED PROJECT

AHMED ASHRAF ABDEL AZIZ BASHA (2).

KARIM IBRAHIM MUSTAFA (32).

MAHMOUD ABU ELENEIN (48).

Description

- This Program provides you with a service to keep data in tables by creating databases you can keep any amount of data.

- You can use the SQL commands to perform operations on your data.

Features

- In this program, you can create or delete databases and tables.

- You can select any column with any condition.

- You can also update your existent data or remove them.

- You can insert any additional data under any condition.

Design overview

JAVA SWING IS USED TO IMPLEMENT THE CONSOLE.

- **JFrame:** is used to create a window.

- **JTextArea:** is used to show the inputs and outputs.

- **JScrollPane:** is used to make the window scroll down and up.

Data structures

- **Object[][]:** Array of selected data.

- **String:** To keep the path and data of type varchar.

- **int:** Used in iterations and to keep data of type int.

- **File:** To create a new file or directory.

- **Enum:** To keep the type of possible commands.

Description of important function

- **createDatabase:** create a new database.

- **executeQuery:** used to perform selection queries and return array 2-D of the selected data.

- **executeUpdateuery:** used to update, insert or remove data.

User manual

Once the user open the program he will see the shown:
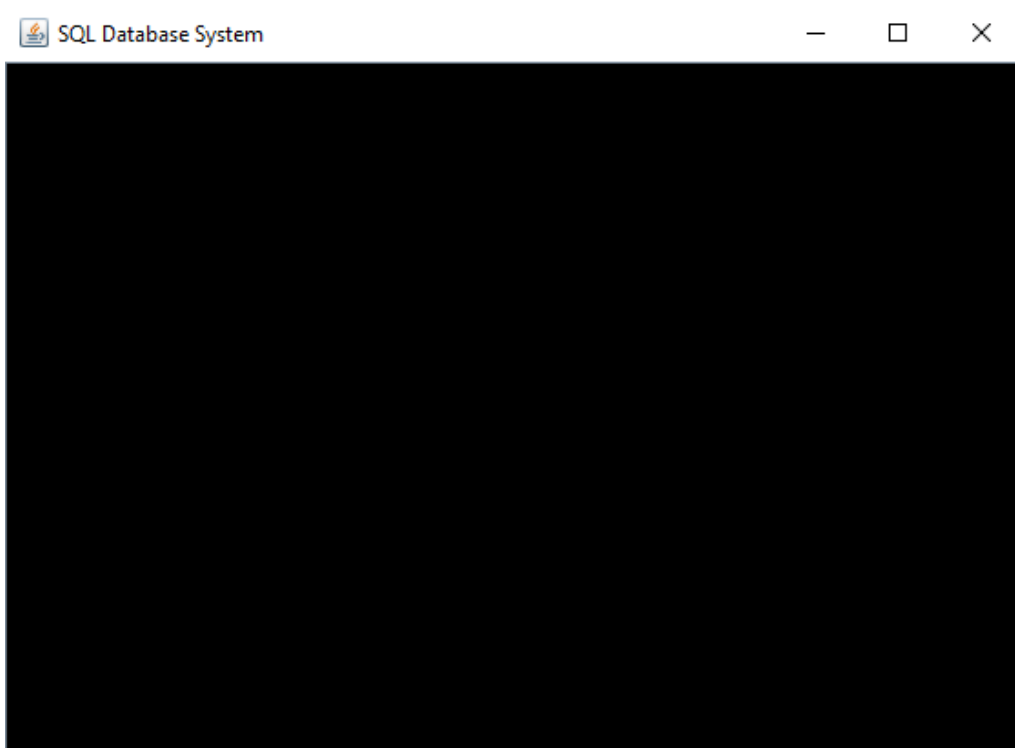
You can start by creating a database with writing the correct query to create.

create database data1
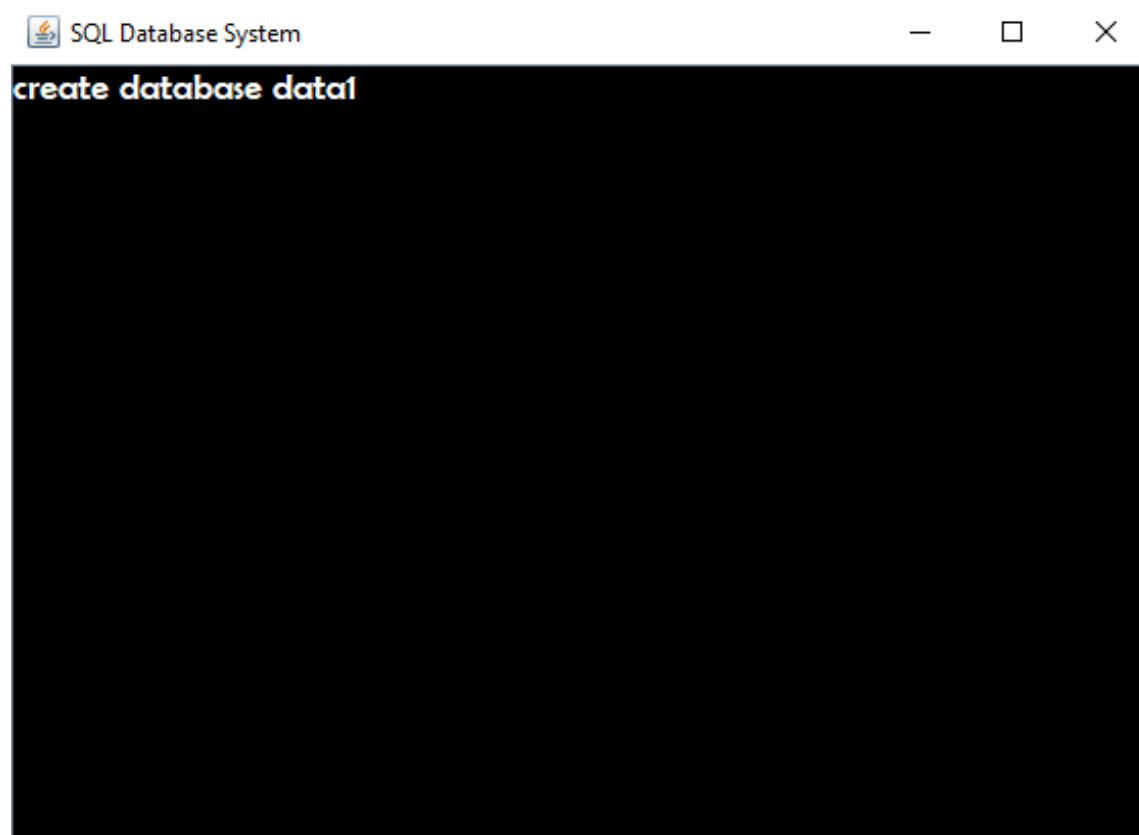
Then you can create tables in your database.

Then you insert the data in your table.

UML

<<Java Class>>
⊖ Facade

□ selected: Object[][]

♦ Facade(Database)
● getSelected():Object[][]
● executeQuery(String):void

-database
0..1

<<Java Interface>>
① Database

● createDatabase(String,boolean):String
● executeStructureQuery(String):boolean
● executeQuery(String):Object[][]
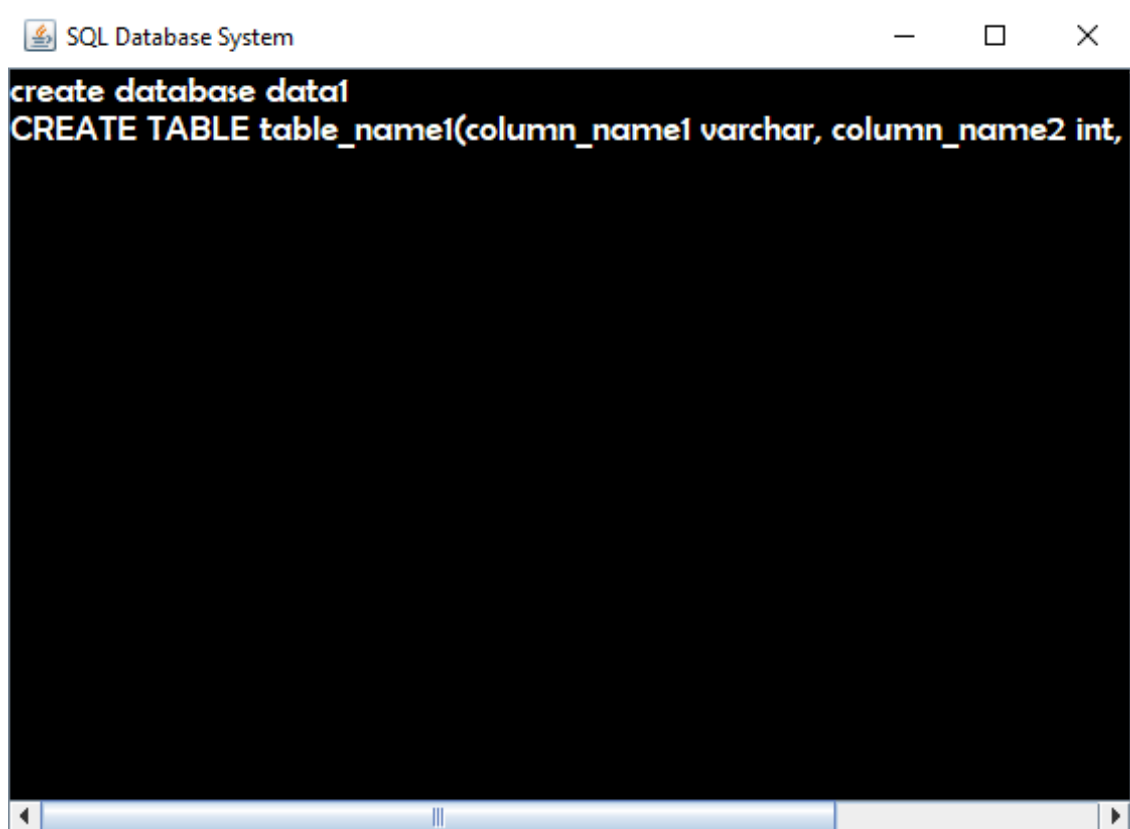● executeUpdateQuery(String):int

<<Java Class>>
⊖ Parser

△ map: HashMap<returnType,Object>

♦ Parser()
● executeStructureQuery(String):boolean
● executeQuery(String):boolean
● executeUpdateQuery(String):boolean
■ regexUpdateQuery(String,ArrayList<Object>):void
■ conditionFinder(String):boolean
■ updateSetter(String,ArrayList<Object>,ArrayList<Object>):void

-singleTable
0..1

<<Java Class>>
⊖ Table

□ tableFullPathXML: String
□ tableFullPathDTD: String
□ colsNames: String[]
□ colsDataTypes: String[]

♦ Table()
♦ loadNewTable(String):Table
● getTableData():ArrayList<Record>
● setTableData(ArrayList<Record>):void
● getColsNames():String[]
● getColsDataTypes():String[]
● insert(HashMap<returnType,Object>):int
● update(HashMap<returnType,Object>):int
● delete(HashMap<returnType,Object>):int
■ saveDataInXML():void
■ readDTD():boolean
■ loadData():void
■ getDTDPath():String

<<Java Class>>
⊖ Record

□ record: String[]

♦ Record(String[])
● length():int
● getItem(int):String
● setItem(int,String):void

-tableData
0..*

<<Java Class>>
⊖ SQLDatabase

□ currentDatabase: String

♦ SQLDatabase()
● getCurrentDataBase():String
● createDatabase(String,boolean):String
● executeStructureQuery(String):boolean
● executeQuery(String):Object[][]
● executeUpdateQuery(String):int

<<Java Class>>
⊖ FoldersAndFilesHandler

♦ FoldersAndFilesHandler()
● createDatabase(String):void
● deleteDatabase(String):void
● deleteTable(String):void
● createTable(String,String,String[],String[]):boolean

<<Java Enumeration>>
⊖ returnType

♦ NAME: returnType
♦ ISCREATE: returnType
♦ ISDATABASE: returnType
♦ COLNAME: returnType
♦ COLTYPE: returnType
♦ COLVALUES: returnType
♦ ISINSERT: returnType
♦ ISUPDATE: returnType
♦ ISDELETE: returnType
♦ CONDITIONOPERATOR: returnType
♦ CONDITIONOPERANDS: returnType

♦ returnType()