Introduction on part 1 :

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) 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.

Steps of part 1 :

1. We implement a simple DBMS that handles data and store it in XML files. The DBMS

will develop should control the management and retrieval of data from data

files. The DBMS accepts requests for data from application programs and

retrieves and transfers the appropriate data from files that are stored physically

on disk.

2. Each XML file will represent a Database Table. A database table is a set of data

elements (values) using a model of vertical columns (identifiable by name) and

horizontal rows, the cell being the unit where a row and column intersect. A table

has a specified number of columns, but can have any number of rows.

Schema files contain data about the tables and columns that should exist. Tables

should be validated across their schema files (XML DTDs can be used for this

purpose). This means each table will have at least two files: one for data (XML

file), and one for its structure (DTD file).

We use (Stax) parser to parse and Dom in Validate the XML database files.

3. The users that will use your simple DBMS will use a language named Structured

Query Language (SQL) to send their commands to it. SQL is case insensitive this depend on the regex at the class which called “parser “.

4. Your DBMS will support only two types: varchar and int. “varchar” used to store

string, and “int” to store numeric values (no floating point will be supported). Do

not support custom type lengths (e.g. varchar(255), or int(11) ), just assume all

types of the same default length.

For statements which contain conditions, support only the simple conditions: =,

>, and <. You are NOT required to support multiple or composite conditions that

use: AND, OR, or NOT.

5. The DBMS can NOT handle SQL queries directly. DBMS delegates this to

another class, say Parser class. The Parser receives the queries, validates them,

parses them, and then calls the appropriate action or reject the bad (invalid)

queries. The files should be created, updated or deleted, whenever any DBMS

function is called from interface

6. we implement the functions which found in the interface “database”

Introduction on part 2 :

in Part 1 of the project, you were asked to write a program that uses

your DBMS. Your program can work with any other DBMS that preserve the same

interface defined at Part 1. However, it would be great if your program can use

any DBMS including MySQL, Oracle, SQLServer, … etc.

In order to achieve that, Java defined another interface (larger than the one

defined in part 1), and asked the DBMS vendors such as MySQL, Oracle,

SQLServer, …. etc. to provide an implementation for this interface, so all Java

developers can change the DBMS easily without rewriting different Java

programs whenever they need to change the DBMS. This interface named JDBC

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 JDBC driver.

Step of part 2 :

The JDBC defines five interfaces. We implement a subset of the JDBC four

interfaces and leave other interface methods empty (or better to through a

runtime exception java.lang.UnsupportedOperationException)

we use interfaces and functions :

java.sql.Driver

 connect(String url, Properties info)

java.sql.Statement

 close()

 execute(String sql)

 executeQuery(String sql)

 executeUpdate(String sql)

 getConnection()

java.sql.Resultset

 close()

 findColumn(String columnLabel)

 first()

 getInt(int columnIndex)

 getInt(String columnLabel)

 getMetaData()

 getObject(int columnIndex)

 getStatement()

 getString(int columnIndex)

 getString(String columnLabel)

 last()

 next()

 previous()

java.sql.ResultSetMetaData

 getColumnCount()

 getColumnLabel(int column)

 getColumnName(int column)

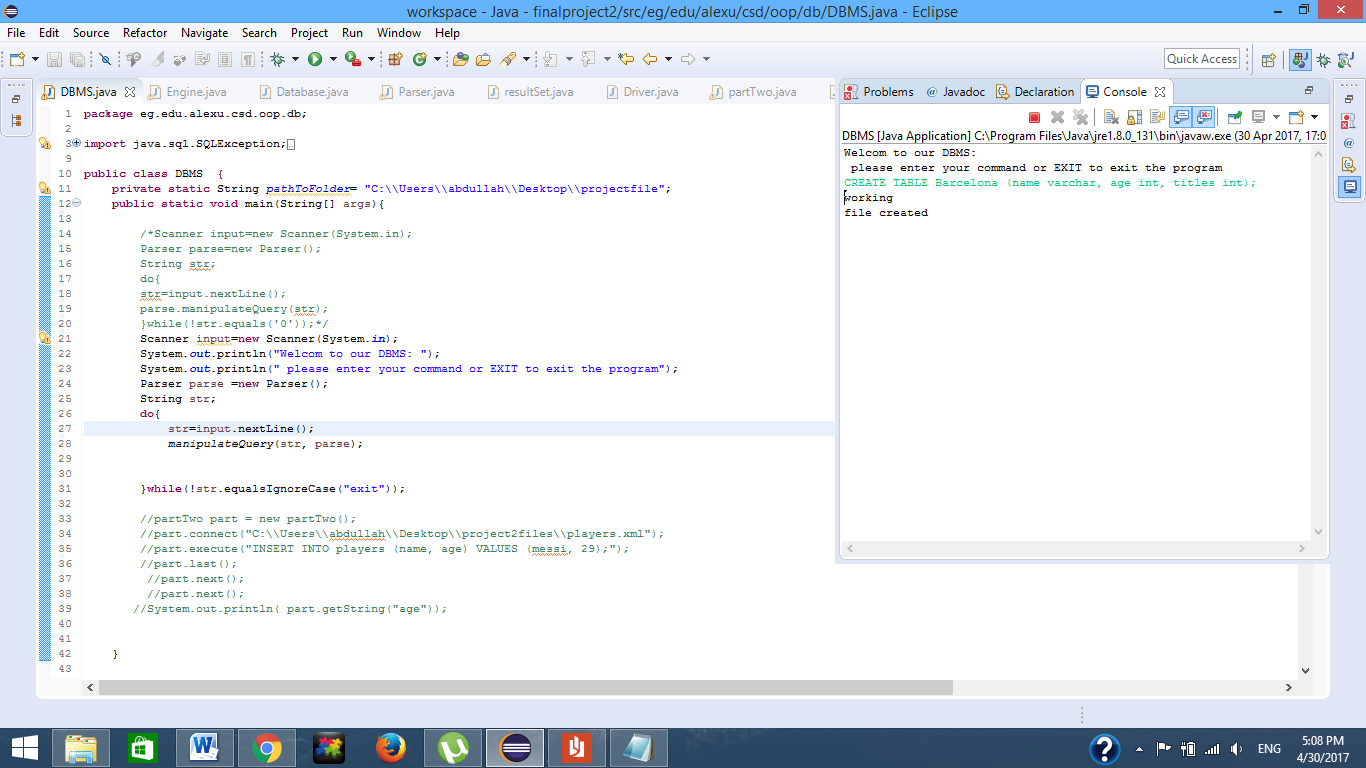
 getColumnType(int column)

 getTableName(int column)

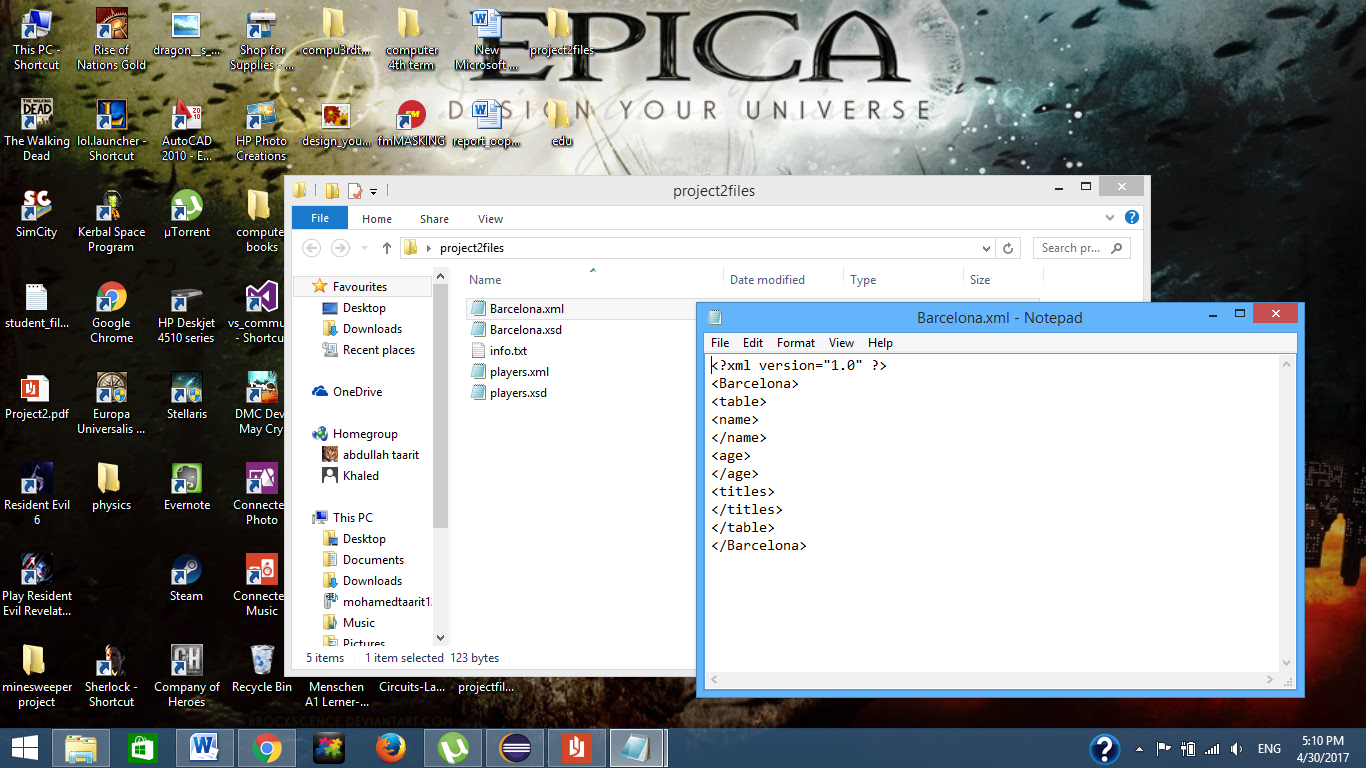
**Part 1:**

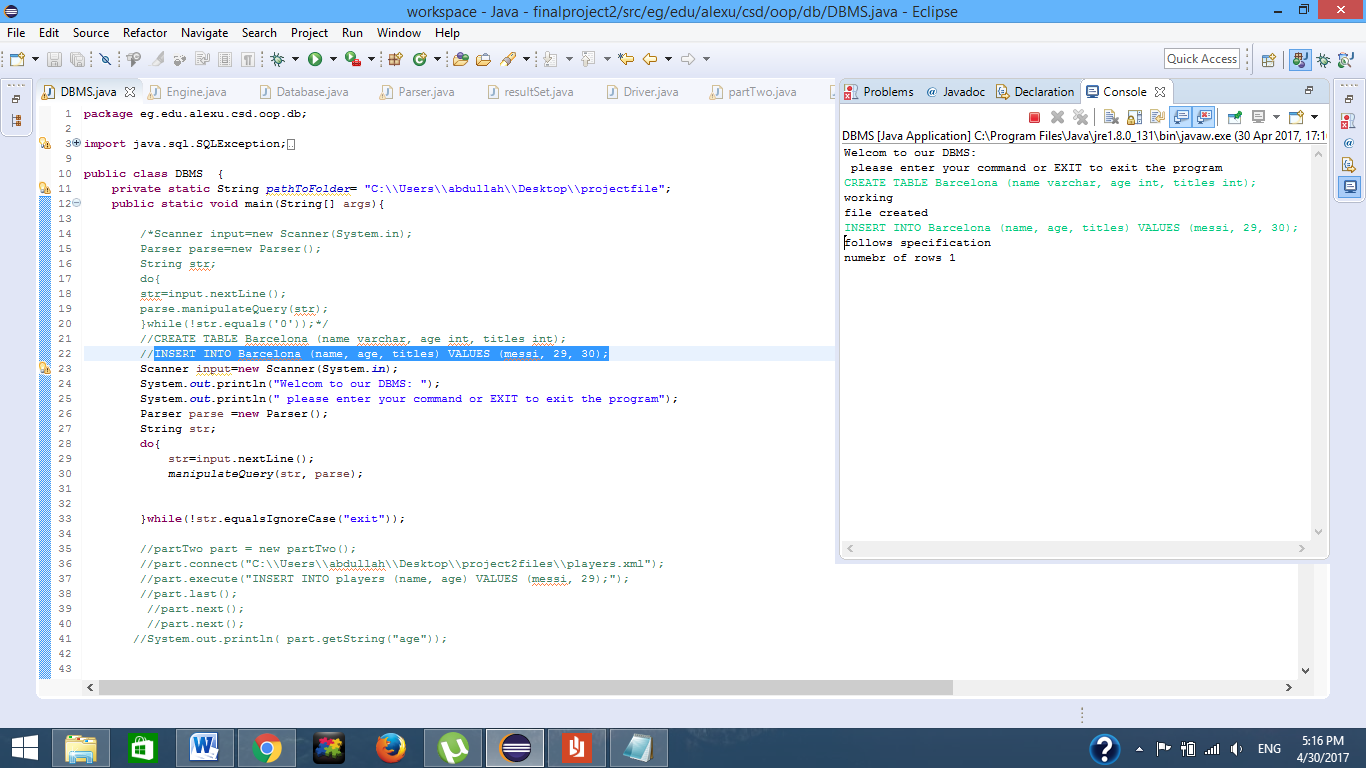
*In part 1 we were asked to Create table, delete table, Insert a row, Delete a row, and select rows that follows a specific condition inputted by the user.*

*Sample run:*

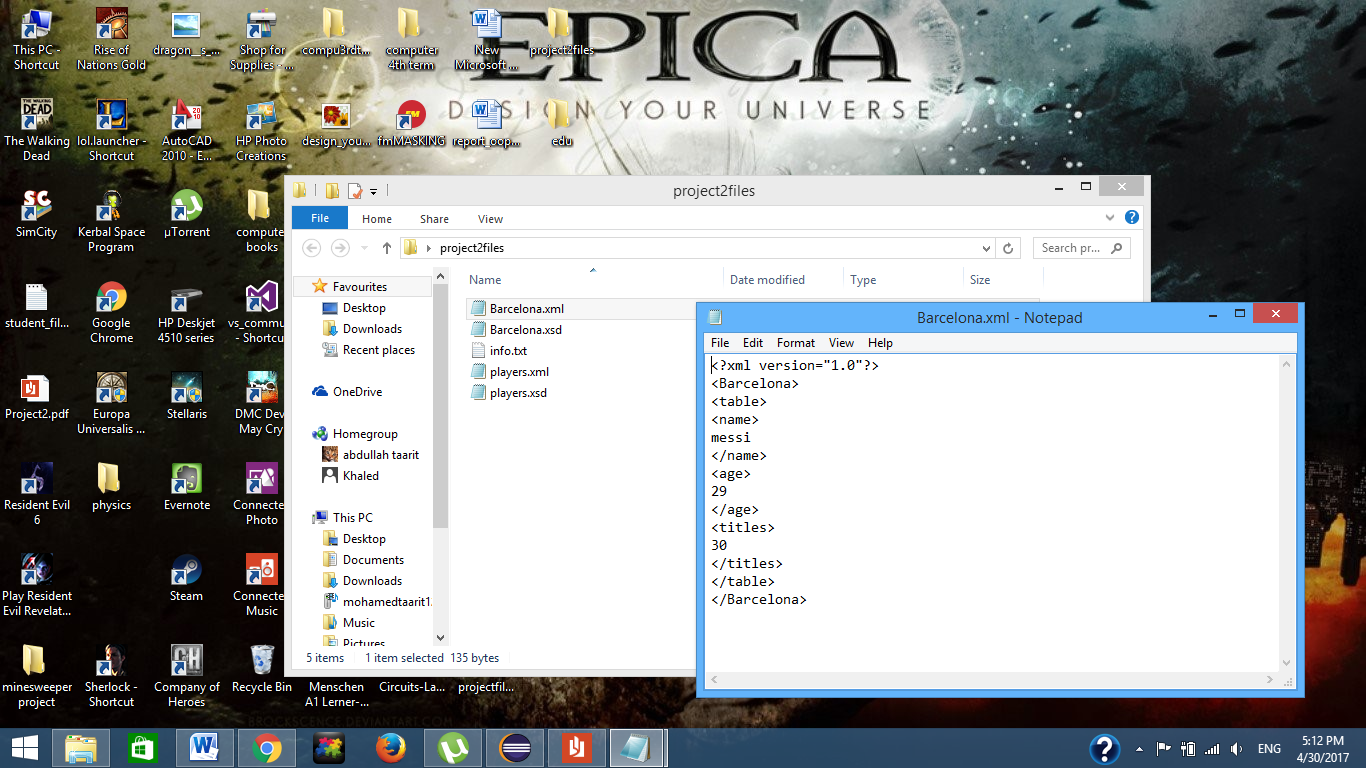


*Table:*





*Table:*



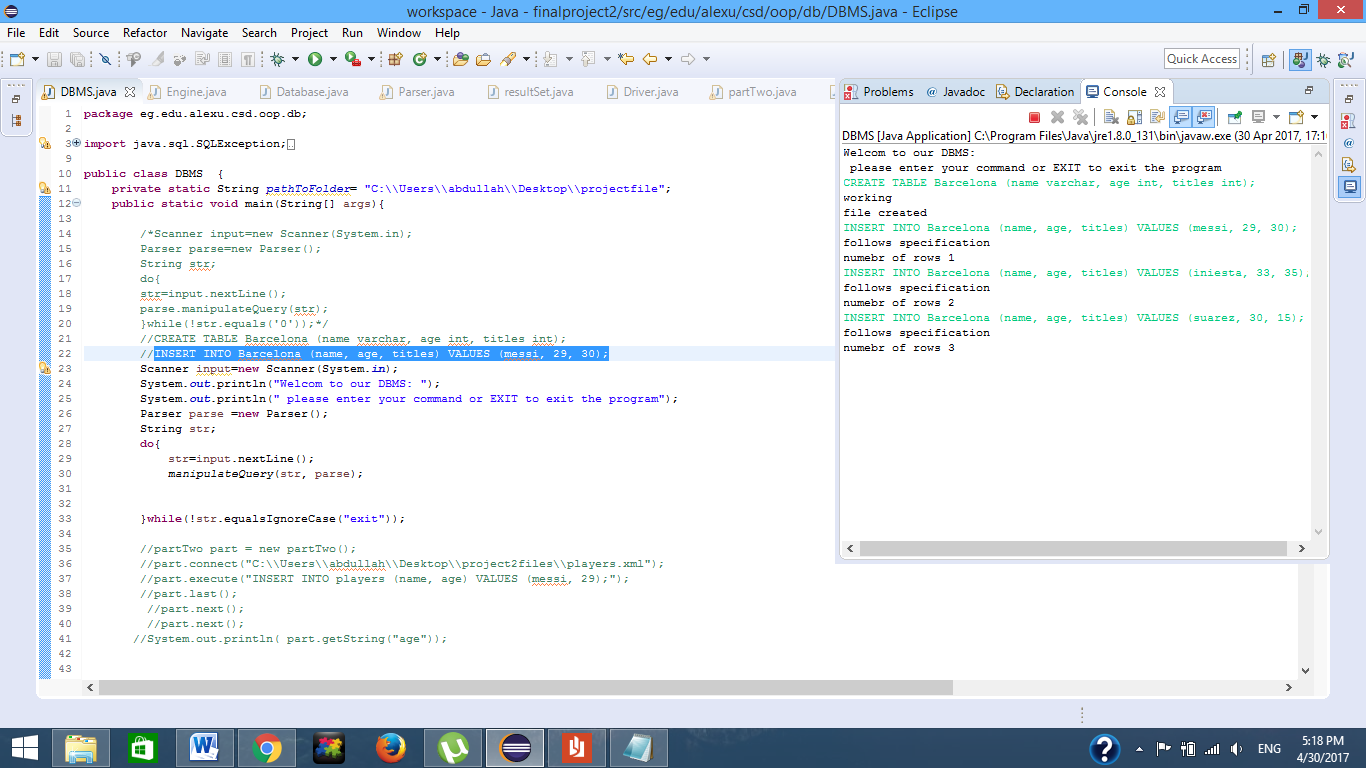
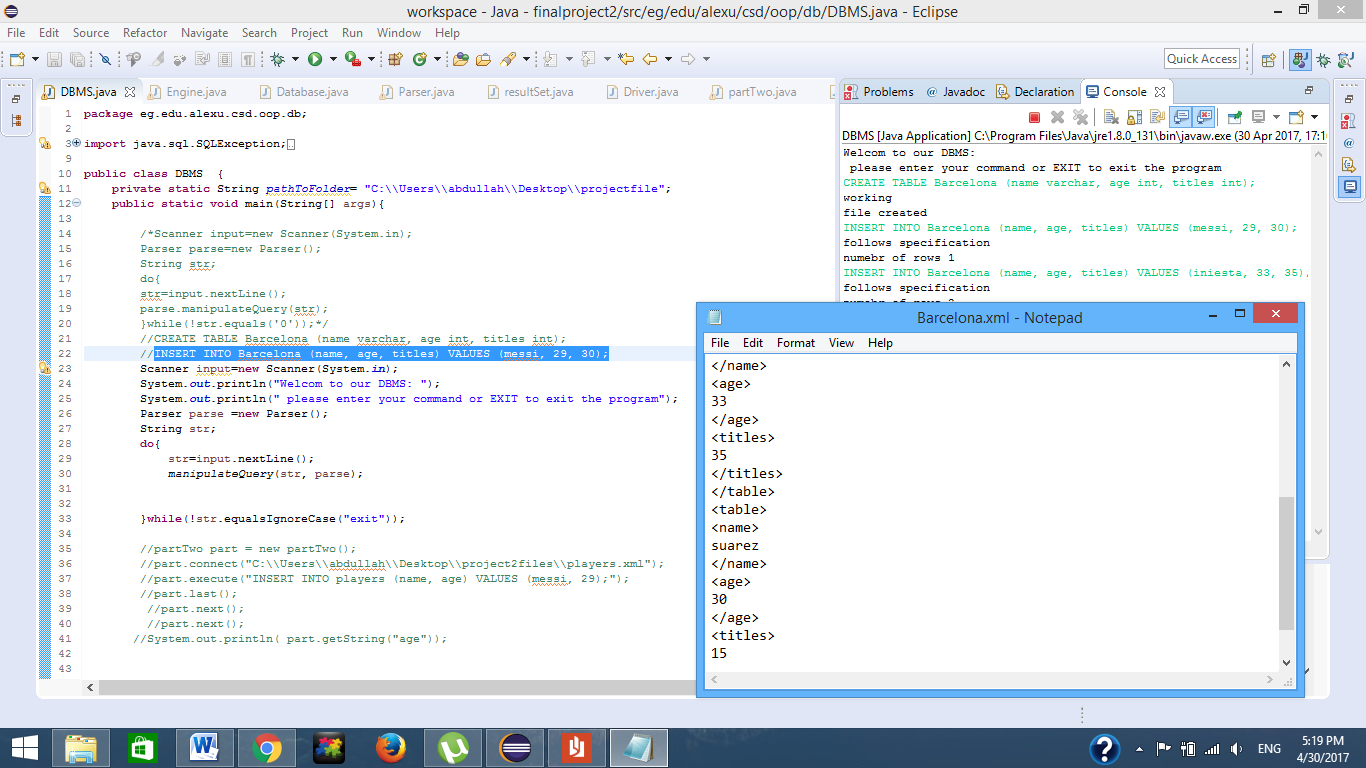
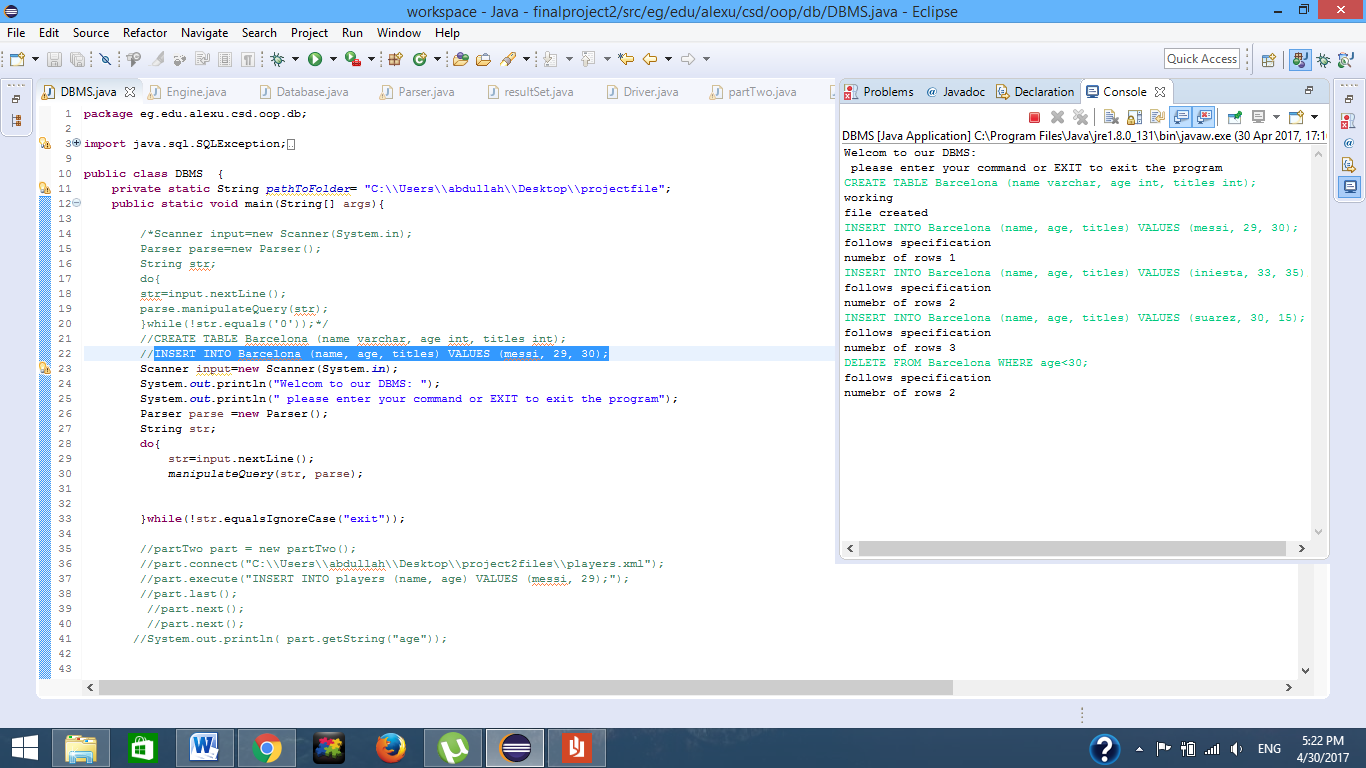
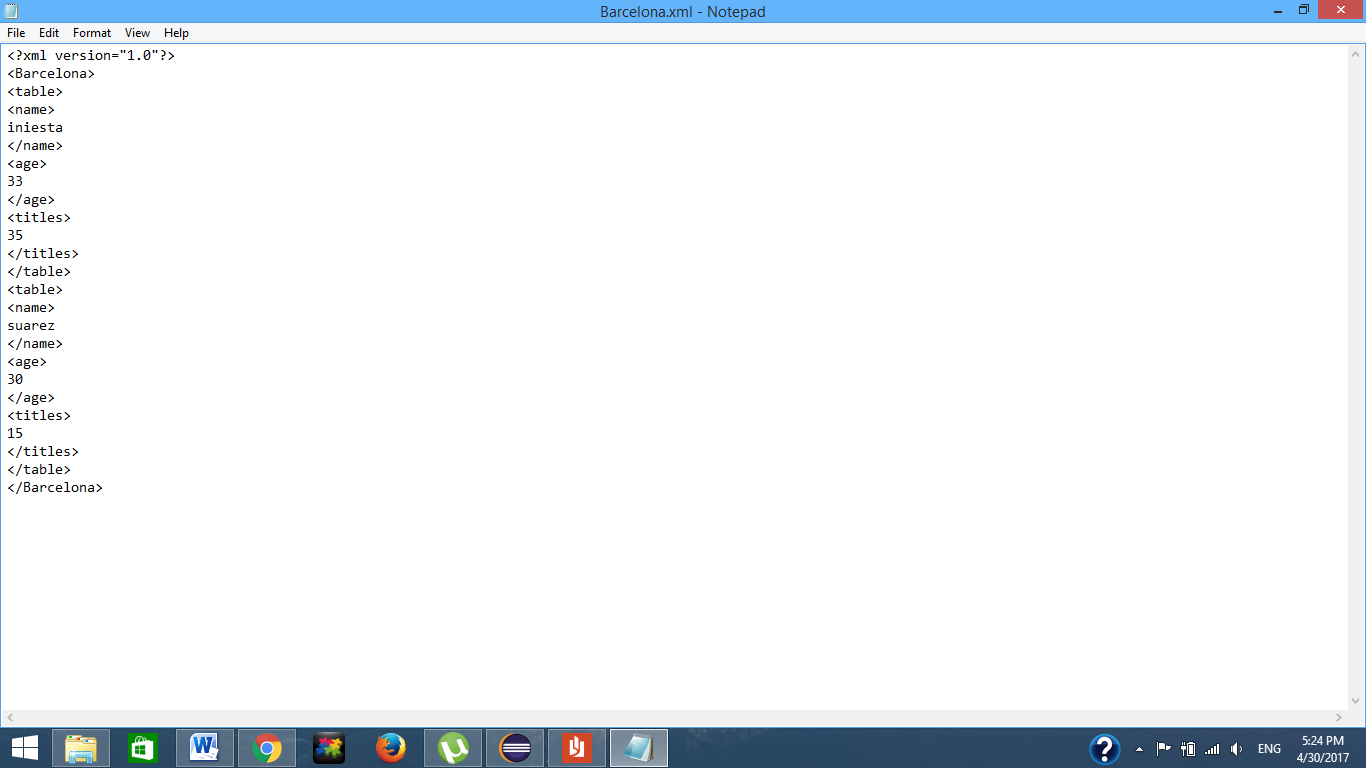


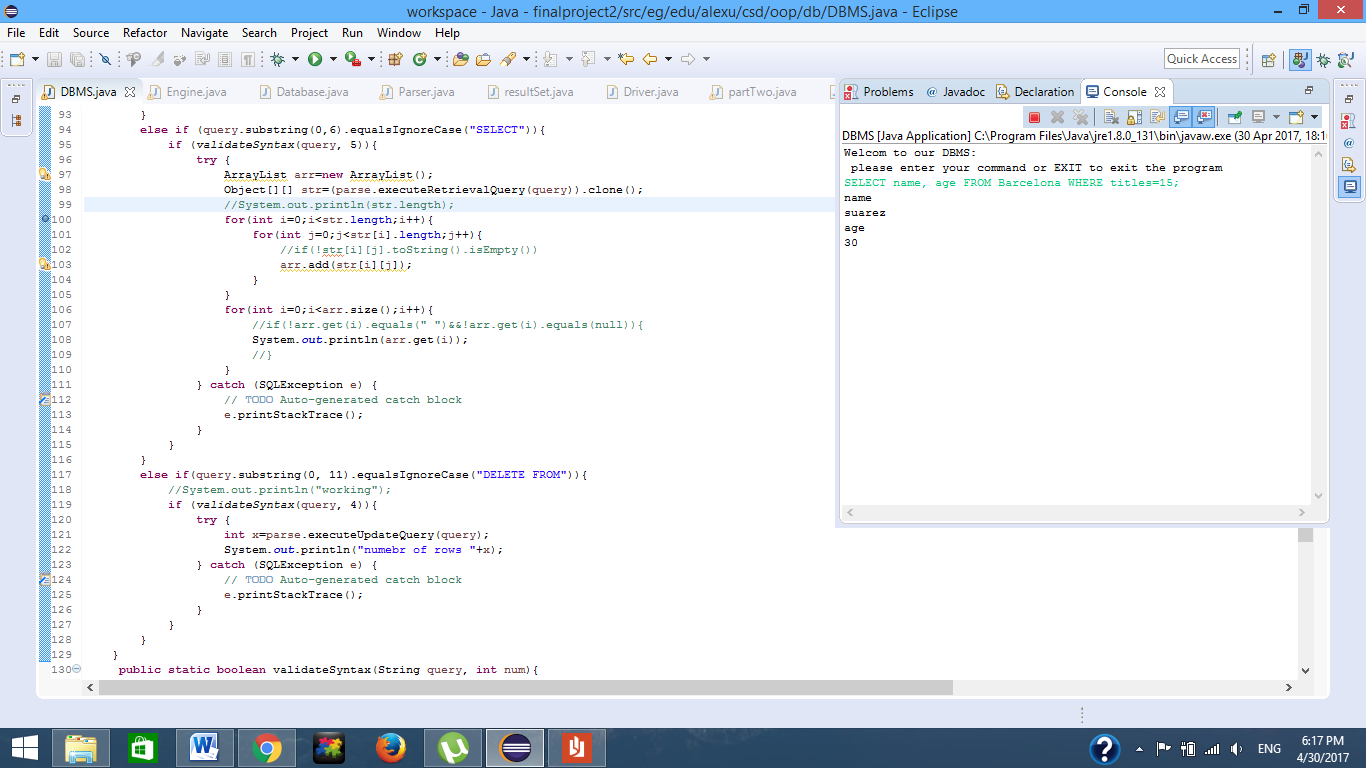
Table:

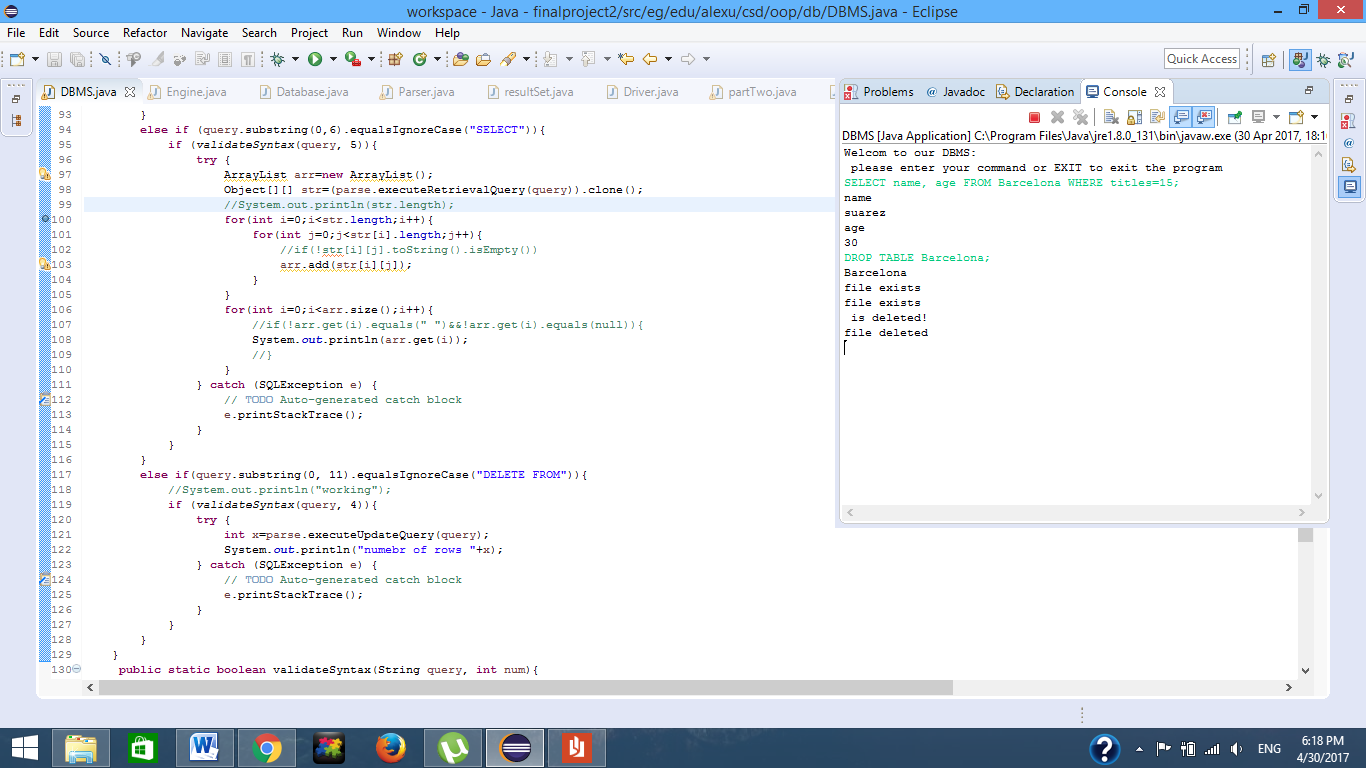




*Table:*







*Design Decisions:*

*We were stuck in the decision where we had 2 choices when inserting a row:*

*1-create a new file with new row added or*

*2-read whole file and the save the information, then write on the same file but with new information present.*

*So we went with the 2nd option…*

*In deleting a file we had to put the pointer of every file to nul for the dele process to be possible.*

*ROLE DIVISION:*

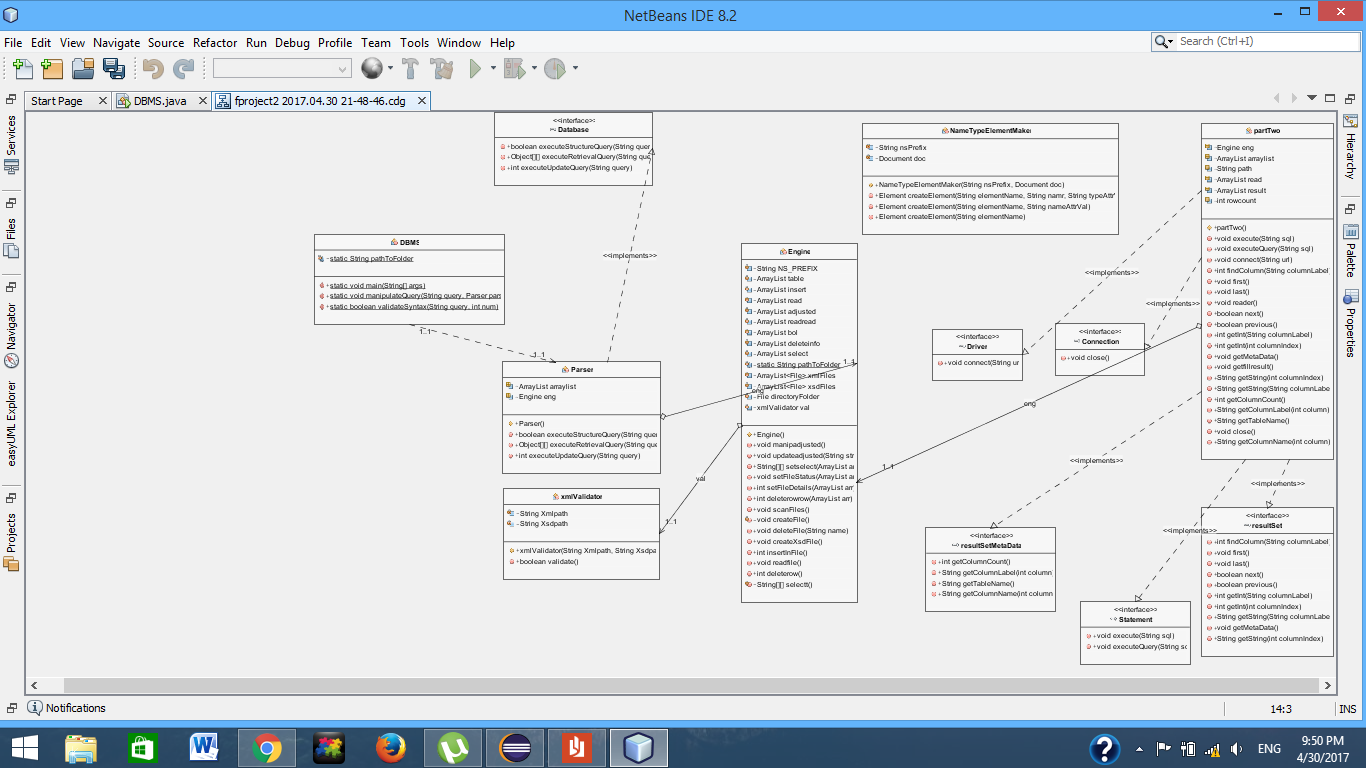
*Mohamed hesham behery : responsible for the regex expressions and their use in the project.*

*Eyad sherif : responsible for part2 in the project.*

*Abdalla Mohamed : Responsible for deleting row and selecting from table and deleting table.*

*Aliaa Mohamed Abbas: Responsible for creating table and inserting row.*

**UMLDiagram:**



|  |  |
| --- | --- |
| Name | id |
| Eyad Sherif Nawar  Aliaa Mohamed Abbass  Mohamed Hesham Behery  Abdullah Mohamed | 4292  3747  4369  4308 |