Assignment 5: Database [40%]

This assignment has a weighting of 40% (it **WILL** contribute to your unit mark)

Overview

The aim of this assignment is to build a "homebrew" database system (from the ground up !)
This will require you to implement:

- A mechanism to store Entity-Relationship data as a collection of formatted text files
- A Socket Server to listen for incoming connections and manipulate the stored data
- A Query language that allows operations to be performed on the stored data

Your database should consist of a number of tables, each contain a set of records (aka entities or rows)
The first column in each table should contain a primary key (called 'id')
You should allow relationships between entities to be expressed using foreign keys (in the usual way !)
You may assume single element primary keys are to be used (i.e. you do not need to cope with composite keys)

The query language you should implement is a simplified version of SQL
(A BNF grammar that defines this language is provided in a separate document)

It is not necessary to implement a type system within the database - you can just store everything as Strings.
The comparison operators will however need to bit fairly intelligent to deal with different kinds of data.

It is up to you to pick a suitable scheme to store data in files (so that data persists between server invocations) You should however only use the core Java classes to implement your solution (do not use any fancy data storage libraries !)

Note: It is NOT your responsibility to normalise the database - this is a job for developers who will use of your database

Note: For the sake of simplicity, you may assume only a single client exists (i.e. there is no need to handle parallel queries or issues of contention)

Communication Protocol

You should base your server on the StagServer class from the previous assignment (in terms of Socket communication).
Your database server should listen on port 8888 and receive incoming queries in the specified query language.
It should interrogate the stored data and return the result of the query to the client via the network connection.

NOTE: The exact structural formatting that you use for the response data is not important.
However, you should attempt to make it as human-reader-friendly as possible.
During marking, we will only analyse the data content of the responses and not formatting aspects (such as tabulation).

IMPORTANT: To allow the client application to detect the end of a response, you should terminate the entire message with an "End of Transmission" (EOT) sequence. This consists of the EOT character (ASCII value 4) on a line on its own after the content of the response.

Error handling

Your query interpreter should identify any errors in the construction of queries
(for example queries not conforming to the BNF or queries that include unknown identifiers).
When an error is identified, your sever should return an error message that provides the user
with information about the nature of the issue.

Testing

Your main class MUST be called DBServer (so that we can run and test it automatically)

An example set of basic queries and expected responses is provided in a separate file to aid you in testing your server.
This set of test cases does not cover the entire query language - it is up to you to include additional test cases
for some of the more complex features of that language.

A simple DBClient class has also been provided to help you manually test your server.
You may wish to convert this into an automated test script that can be run to confirm correct operation of your server.

Marking Criteria

Your submission will be assessed against the following criteria:

- Functionality testing based on a set of test cases that includes those provided as examples
- An assessment of the operational performance (speed) of your implementation
- Code Quality metrics as used on previous assignments

Implementing a "good" project will involve trading off the above in order to achieve an effective solution

Submission

This assignment **IS** assessed: go to "Assessment, submission and feedback" to submit it.
It is **essential** that you ensure your code compiles and runs before you submit it
(otherwise we will not be able to run it to mark it !)

You should submit all of the files you have written to complete the task.
Scripts will be used to automatically test your code to make sure it operates correctly.
Don't change the name of the class or the signatures of any of the methods that already exist
(or we won't be able to test your code !).

Plagiarism

You are encouraged to discuss assignments and possible solutions with other students.
HOWEVER it is **essential** that you only submit your own work.
This may feel like a grey area, however if you adhere to the following advice, you should be fine:

- Never exchange code with other students (via IM/email, USB stick, GIT, printouts or photos !)
- Although pair programming is encouraged in some circumstances, on this unit you must type your own work !
- It's OK to seek help from online sources (e.g. Stack Overflow) but don't just cut-and-paste chunks of code...
- If you don't understand what a line of code actually does, you shouldn't be submitting it !
- Don't submit anything you couldn't re-implement under exam conditions (with a good textbook !)

An automated checker will be used to flag any incidences of possible plagiarism.
If the markers feel that intentional plagiarism has actually taken place, marks may be deducted.
In serious or extensive cases, the incident may be reported to the faculty plagiarism panel.
This may result in a mark of zero for the assignment, or perhaps even the entire unit
(if it is a repeat offence).
Don't panic - if you stick to the above list of advice, you should remain safe !