Computing and Algorithms II: Programming Assignment 1

CS-102 **Winter 2022**

In this programming assignment, you will demonstrate your knowledge of material covered in CS-101 (and re-acquaint yourself with the Java environment) by creating a simple database system for an electronic dictionary.

File Input

Your program will accept a single argument on the command line, which is a non-empty string giving the name of a file in the current directory.

This file will contain definitions of terms to appear in the dictionary. Each line of the file will contain one record of data. Each record will be represented by several strings, separated by slashes, in the following order:

1. The term being defined. A term is an arbitrary string, and may include spaces or other special characters.
2. A sequence number, represented as a positive integer. Each definition for a given term will be given a unique positive integer.
3. The definition for the term. A definition is an arbitrary string, and may include spaces or other special characters.

Note the following:

- A term may have multiple definitions, each with a unique sequence number.
- The definitions will appear in the data file in an arbitrary order, without regard to term or sequence number.
- Terms and definitions may (in general, will) contain blanks and other characters; the only delimiter for fields in the data file is the slash ("/") character.

Your program will begin by reading in this information and storing it internally in an appropriate format. (See *Internal Requirements*.)

Interactive Input

Your program will then interact with the user running the program, offering the user a menu of commands. The following commands should be offered at this time:

- Define a term. If selected, the program should ask the user for a string. Using that string, the program should display all records in the database whose term *matches* the specified input string, ignoring upper and lower case distinctions. If no matching record can be found, an appropriate message should be displayed.

- Search for matching terms. If selected, the program should ask the user for a string. Using that string, the program should display the names of all terms in the database which *begin with* the specified input string, ignoring upper and lower case distinctions. (NOTE: only the terms should be displayed, not the definitions.) If no matching record can be found, an appropriate message should be displayed.

- Print the entire database. If selected, the program should print the entire contents of the database in an appropriate format.

- Exit the program. If selected, the program should terminate.

Internal Requirements

As always, your program should use good style, as defined by the CS-102 Style Requirements handout.

Your program should catch (and handle appropriately) all exceptions generated by any system routines you use, as well as any exceptions you generate yourself. (That is, your main method should not throw any exceptions.)

Your program should be designed with modifiability in mind. You will be revising and extending this program several times throughout the semester; consequently, it is to your benefit to design your program in as modular a fashion as possible. In particular:

- You should define a class `Definition` which contains members corresponding to a given definition, and methods which deal with definitions (*e.g.* print, getTerm).

- You should define a class `Database` which contains methods which allow one to manipulate a collection of `Definition` objects (*e.g.* search, print). The `Database` class should never access the internal members of the `Definition` class directly; instead, it should use the public methods of the `Definition` class for that purpose.

  For this assignment, your `Database` object should use an unordered array of `Definition` objects. You should define the size of this array using a `final` constant and use that constant appropriately.

- You should define a class `Prog1` with a `main` method which interacts with the user and calls the `Database` class to perform the required operations. The `Prog1` class should never access the internal members of the other classes directly.

You will be replacing these classes several times through the semester as we learn about different data structures and algorithms. Thus, it is to your advantage to make your design as modular as possible. A little bit of extra work now will make your work simpler as the term progresses.

A Sample Session

Here is a *sample* data file which could be read by this program:

```
ray/1/A beam of light or radiation
tree/6/A connected graph with no cycles
treat/7/To care for medicinally or surgically; to apply medical care to
tree/4/A device used to hold or stretch a shoe open
treat/1/To negotiate, discuss terms, bargain
ray/7/A tiny amount
ray/8/A drop of golden sun
tree/14/A mass of crystals
tree/1/A large plant
treat/2/To handle a subject in writing or speaking
```

And here is a *sample* interactive session:

```
Welcome to the CS-102 Dictionary Program
Current available commands:
1 --> Define a term
2 --> Search for matching terms
3 --> Print all records
9 --> Exit
Your choice? 3

ray: (1) A beam of light or radiation
tree: (6) A connected graph with no cycles
treat: (7) To care for medicinally or surgically; to apply medical care to
tree: (4) A device used to hold or stretch a shoe open
treat: (1) To negotiate, discuss terms, bargain
ray: (7) A tiny amount
ray: (8) A drop of golden sun
tree: (14) A mass of crystals
tree: (1) A large plant
treat: (2) To handle a subject in writing or speaking

Current available commands:
1 --> Define a term
2 --> Search for matching terms
3 --> Print all records
9 --> Exit
Your choice? 1

Enter a term to define:  tree
```

```
tree: (6) A connected graph with no cycles
tree: (4) A device used to hold or stretch a shoe open
tree: (14) A mass of crystals
tree: (1) A large plant

Current available commands:
1 --> Define a term
2 --> Search for matching terms
3 --> Print all records
9 --> Exit
Your choice? 2

Enter a prefix to search:  tr

Matching terms:
  tree
  treat

Current available commands:
1 --> Define a term
2 --> Search for matching terms
3 --> Print all records
9 --> Exit
Your choice? 9
```

Notice that this is a *sample* run; your run may operate differently (and appear differently) as long as it fulfills the requirements outlined above.

You should create several input files in order to test your program thoroughly; do not assume that simply testing the program on the input file shown above is sufficient to test your program. (What happens if the input file is empty? What if it contains too many entries?)

Submitting Your Program

Before 11:59:59 p.m., Wednesday, 19 January 2022 (2nd Wednesday), you must upload a zip archive to the course Blackboard assignment for Programming Assignment 1. This zip archive must contain all source code files for your program, including a class named `Prog1` with a `main` method. You may optionally submit a description of errors remaining in your program through the "Add Comments" section in your Blackboard submission.

Notes

1. *Start Early!* The intent of this program is to make sure you are familiar with Java and our Java environment. However, there are some new features in the program (*e.g.* file input, exception handling), and some new style requirements, that may catch you if you are not careful.

2. A reminder: please read and use the class style guidelines distributed separately. 25% of your program grade is allocated for style.

3. Input from the console and from files can be conveniently handled using the `Scanner` class. See the Scanner Crib for details.

4. Recall that the `main` method for a Java program begins as follows:

   ```
   public static void main(String[] args) ...
   ```

   `args` is an array of strings; the elements of this array are the command-line arguments given to this program. `args.length` gives the length of the array, *i.e.*, the number of command-line arguments supplied to the program.

5. `String` objects should be not be compared directly with the "`==`" operator. Various methods in the `String` class will assist you in performing your searches; see the Java API for details.