Computing and Algorithms II: Programming Assignment 1

CS-102 **Fall 2016**

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 maintaining data on named tropical storms.

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 descriptions of tropical storms. 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. A four-digit integer representing the year of the storm. The smallest valid year is 1950; there is no largest valid year.
2. A string representing the name given to the storm (*e.g.* "Frances", "Ivan", "Jeanne").
3. A four-digit integer representing the starting date of the storm. The integers will have the form "*mmdd*", where "*mm*" is a month (01-12), and "*dd*" is a day (01-31).
4. A four-digit integer representing the ending date of the storm (in the same format as the starting date).
5. A single digit representing the strength of the storm at its largest point. Valid digits are 0-6; 0 represents a tropical storm, while 1-5 represent hurricanes of the corresponding category. (*e.g.* "category 3 hurricane").

Note: Any of the last three items may be the empty string; this allows the database to be used for future storms as well as past storms. The first two items are required for all entries. (Note that even if the items are absent, the delimiting slashes are required.)

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:

- Search the database for a storm name. If selected, the program should ask the user for a string. Using that name, the program should display *all* storms in the database whose names *begins with* that string. If no matching storms can be found, an appropriate message should be displayed.

- Search the database for a year. This operates as above, except that the user specifies a four-digit year as the basis of the search, and displays all storms which occurred in that year.
- Print the 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.
- 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 validate input whenever possible. For example, you should check that the user provides valid arguments on the command line, that database entries read from the input file meet the required format, and that the user provides valid options during the interactive phase of the program.

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 `Storm` which contains members corresponding to a given storm, and methods which deal with storms (*e.g.* print, getYear).
- You should define a class `Database` which contains methods which allow one to manipulate a collection of `Storm` objects (*e.g.* search, print). The `Database` class should never access the internal members of the `Storm` class directly; intsead, it should use the public methods of the `Storm` class for that purpose.

  For this assignment, your `Database` object should use an unordered array of `Storm` objects. You should define the sizes of this array using a named constant and use that constant appropriately.
- You should define a class `Prog1` with a `main` method which calls the `Database` class to perform the required operations. The `Prog1` class should never access the 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:

```
2004/Matthew///
2004/Nicole///
2003/Ana/0418/0427/0
2003/Bill/0628/0703/0
2003/Claudette/0707/0717/1
2003/Danny/0716/0727/1
1997/Ana/0630/0705/0
1997/Bill/0711/0713/1
2004/Alex/0731/0806/3
2004/Gaston/0827/0901/0
2004/Lisa/0919//1
```

And here is a *sample* interactive session:

```
Welcome to the CS-102 Storm Tracker Program.

Current available commands:
1 --> Search for a storm name
2 --> Search for a year
3 --> Print all storms
9 --> Exit
Your choice? 3

2004: Matthew (no info): (no start) - (no end)
2004: Nicole (no info): (no start) - (no end)
2003: Ana (tropical storm): 04/18 - 04/27
2003: Bill (tropical storm): 06/28 - 07/03
2003: Claudette (hurricane level 1): 07/07 - 07/17
2003: Danny (hurricane level 1): 07/16 - 07/27
1997: Ana (tropical storm): 06/30 - 07/05
1997: Bill (hurricane level 1): 07/11 - 07/13
2004: Alex (hurricane level 3): 07/31 - 08/06
2004: Gaston (tropical storm): 08/27 - 09/01
2004: Lisa (hurricane level 1): 09/19 - (no end)

Current available commands:
1 --> Search for a storm name
2 --> Search for a year
3 --> Print all storms
```

```
9 --> Exit
Your choice? 1

Enter storm name prefix: Ana

2003: Ana (tropical storm): 04/18 - 04/27
1997: Ana (tropical storm): 06/30 - 07/05

Current available commands:
1 --> Search for a storm name
2 --> Search for a year
3 --> Print all storms
9 --> Exit
Your choice? 1

Enter storm name prefix: Yankee

Could not find any storm named "Yankee".

Current available commands:
1 --> Search for a storm name
2 --> Search for a year
3 --> Print all storms
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; testing the program on the input file shown above is *not* sufficient to test your program. (What happens if the input files are empty? What if the files contain too many entries?)

Submitting Your Program

Before 11:59:59 p.m., Wednesday, 12 October 2016 (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.

In addition, you must deliver to the instructor a printout of your program files at the start of class on Friday, 14 October 2016 (2nd Friday).

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.

6. For those who may be interested ... the data in the sample file above is actual Atlantic storm data. You can find out more about tropical storms and naming at the following websites:

   - `http://weather.unisys.com/hurricane/`
   - `http://en.wikipedia.org/wiki/List_of_historic_tropical_cyclone_names`