CS-141 Introduction to Programming
Creating a Simple Database

In this lab, you will implement a simple database, i.e. a program that consists of an organized collection of data together with a user interface that enables one to read, write, and access the data in multiple ways. One familiar example is The Internet Movie Database (IMDb). There are also many online music databases.

The lab will be divided into several parts:

- Part 1: Pick a subject. Build a database with a simple class structure. Add a menu-style user interface.
- Part 2: Add the ability to read and write the data from files.
- Part 3: Add the ability to store and search more complex data. You will need to add additional classes, modify classes from part 1, and modify the file I/O from part 2 in order to read this more complex data.

Things to Consider:
As your program gets larger, it can be confusing to figure out what goes where. Developing a preliminary design in the form of a **Class Diagram** is a must. Also, always try to place methods in the classes that have access to the needed data.

Part 1

Goals:

1. Practice working ArrayLists.
2. Practice creating and using classes.
3. Develop a menu-style user interface.

Instructions:

- **Pick a topic:** Pick a subject, e.g. movies, music, other
- **Build the Class:** Write a class that stores data about a single representative of this subject. For example, if you pick movies, you would write a Movie class that stores data such as the name of the movie and the year it came out. At a minimum, you should have:
    a. Two Member (instance) variables
    b. Constructors
    c. Mutators & Accessors
    d. toString method
- **Unit Testing your Class:** Test to make sure your class and its methods work properly. Do this by creating another class which contains only the main method. In the main method, create an object (e.g. of type Movie) and call each of the methods you created in your class.

- **Creating your database:** Once you are confident that your class works properly, create another class which contains an ArrayList to hold multiple objects of the type you have created. This class should have member variables and the above methods. It should also have, at a minimum, add() and remove() methods.
- **Unit Testing your Database Class:** Test that your database works by creating a database object in main(). Add and remove several items to your database (using add() and remove()). Print the database(using toString()) after each change. Also test out all of the other methods in your class.
- **Query Menu:** Decide on a menu of the kinds of things you might want to ask your database, e.g.
    1. Add a Movie (name and year) to your database
    2. Remove a Movie from your database
    3. List the entire database
    4. List all movies made in a given year.
    5. Quit

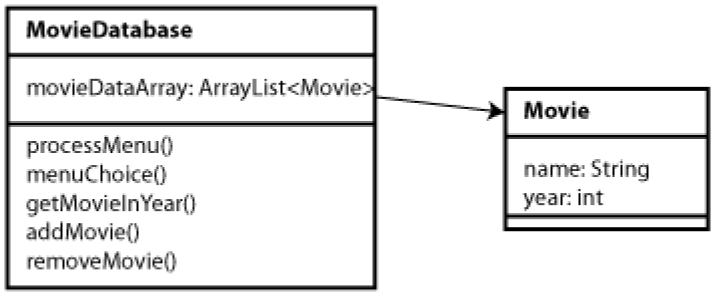
  Write a method that prints the menu and prompts the user for which item they choose. The method should then return their choice. This method could go directly in the database class or you could write an entire separate class. If you write a separate class, this class will need to have a database object as a member variable.
- **Processing Queries:** Add the methods:
    a. A method that calls the menu method and executes the user's choice. This should probably be a do-while loop to prompt for the choice. Inside of the loop there will be a multi-way if-else over the possible choices.
    b. Some of the choices, will require only a few lines of code to process. However, for the choices that are more complex you should write a separate method that gets called from the if-else statement.

Stay tuned for part 2 - Reading and Writing to Files: Create a text data file that contains data for your database. Add a read method that reads from the file, loading the ArrayList as you go. Write a write method that, when exiting the program, will write the data out to a separate output file. More details will follow …

Class Diagram

As your programs get more complex, it is important that you draw the class diagram so that you see the high level structure. This should be done *before* you begin coding.

Class Diagram



Note, in this simplified diagram, not all methods are listed. For example, the obvious methods such as constructors, setters, getters, toString are not listed in order keep the diagram small. However, to be complete, they really ought to be included.

Sample Output

This program is a little more complex than the one above since it reads and writes to a file. However, it should give you an idea of what the interaction should look like.

```
Welcome to the State Population Database.

*************** Menu Items ***************
0. Exit program.
1. Change the output file name from statesModified.txt.
2. List all states.
3. List the state with the largest population.
4. List the state with the smallest population.
5. Compute the average population.
6. List all states that begin with a given letter.
7. Add a state.
8. Remove a state.

Please enter your choice:
3
Largest state is California with a population of 37691912

*************** Menu Items ***************
0. Exit program.
1. Change the output file name from statesModified.txt.
2. List all states.
3. List the state with the largest population.
4. List the state with the smallest population.
5. Compute the average population.
6. List all states that begin with a given letter.
```

```
7. Add a state.
8. Remove a state.

Please enter your choice:
6
Enter the first letter:
W
State:     Washington Population: 6830038
State:     West Virginia   Population: 1855364
State:     Wisconsin  Population: 5711767
State:     Wyoming    Population: 568158

*************** Menu Items ***************
0. Exit program.
1. Change the output file name from statesModified.txt.
2. List all states.
3. List the state with the largest population.
4. List the state with the smallest population.
5. Compute the average population.
6. List all states that begin with a given letter.
7. Add a state.
8. Remove a state.

Please enter your choice:
0
Thank-you for asking questions. Good-bye.
```