CPS112: Homework 4

Overview

You have been asked to help a small buy / sell music store better manage their inventory. They have frequent phone calls asking if they have a certain album by a certain artist, so they need a program that tracks inventory changes and gives a list of the current albums in their inventory. In other words, a simple database. Your task is to write this program for them.

There are two ways in which the music store will use your program. First, the store has a transactions file that records the history of past transactions. When your program starts up, it needs to initialize the database based on these past transactions. Second, the store needs a simple way to list the contents of the database to see what they currently have in inventory, add items to it when they buy an album, and remove items when they sell an album. Since you know dynamic arrays (aka Array List) would be able to handle any size inventory, you figure a dynamic array will be a good data structure for this database.

The goals of this assignment are the following:

- Gain experience in the details of an Array List by implementing one from scratch.
- Learning to use an Array List by integrating one into an application.
- Gain more practice in processing input files.
- Learn to develop a simple user interaction loop.

Part 1 – Implement and Test an Array List

Your first task is to create an Array List to store a list of albums. The Array List must adhere to the constraints of the List ADT while storing its contents in an array. It is also important that the Array List operate efficiently, even for very large lists. Thus, be sure to not expand the size of the array too often.

You must implement your Array List as the class `AlbumArrayList` in `AlbumArrayList.java`. Your class needs to also extend the provided `AlbumList` abstract class. Do not modify `AlbumList`. In implementing your `AlbumArrayList`, **do not use the standard Java `ArrayList`**. Since the list contains albums, you need an `Album` object to represent the albums. You should use the `Album` class you created in Lab 3.

As you implement each method in `AlbumList`, write a test for it. Create a `AlbumListTest.java` that has at least one test for every method in `AlbumList`. To really ensure that your code is complete, you might want to create three or more tests (see the Linked List lab for inspiration for what more tests you may want).

Part 2 – File Processing

Implement the database program `ArrayListDB.java` using the Array List you just implemented to store the list of albums currently in stock. Open and process the transaction history file and generate the list of albums that have been added, and not yet removed. These are the albums currently in stock. A few details for you to consider.

- The same album may be added and/or removed more than once. Adding more than one of the same album should result in multiple instances of that album in the database.
- The filename should be specified on the command line (i.e., `java ArrayListDB history1.txt`)
- The file provided to you (`history1.txt`) has only a few lines, but when your program is graded other files will be used with various lengths.
- The transaction lines in the file are always well formed, you can assume it will always be of the form `ADD:foo - bar` or `REMOVE:biz - baz`
- The `ADD` command in the file adds the album to the end of the database.
- The `REMOVE` command in the file removes the first instance of the matching album in the database, if any match.
- The transaction file may contain some "incoherent" operations. For example, removing an album that is actually not present.
- Note: You can use `string.split("\\:")` to separate a string by the delimiter "`:`"

Part 3 – User Interaction

Implement a user interaction (UI) for the program. After processing the file, the user should be given a prompt that allows them to input four different commands.

ADD – The user can add another album to the list. The user is prompted to enter the artist name and album name.
REMOVE – The user can remove an album from the list. The user is prompted to enter the number of the album to be removed.
LIST – The program prints out a numbered list of programs. The numbers of the list start with one.
QUIT – The user can choose to end the program.

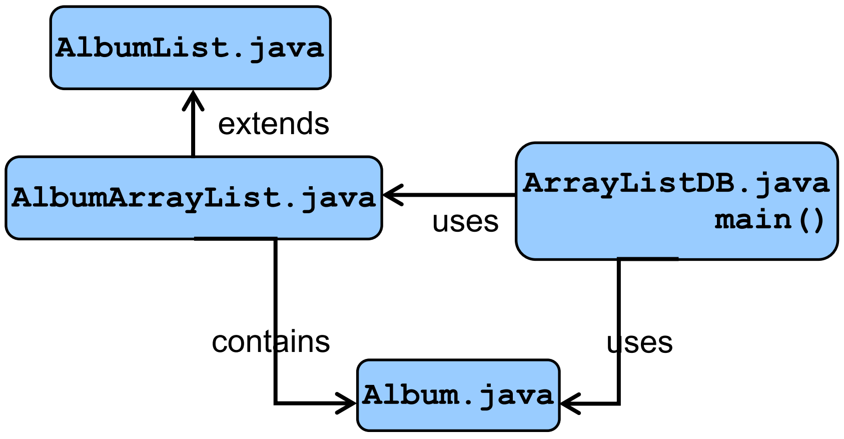
A few implementation details for part 2

- It is NOT safe to assume the user types sane / rational things. Carefully validate their input. Under no circumstances should your program crash.
- The additional transactions the user inputs do not have to be kept after the program is ended.
- On the final page of these instructions is an example interaction for further clarification.

Example Run / Interaction (user input shown in bold)

```
java ArrayListDB history1.txt
Finished processing 5 transactions.
Please provide instructions: ADD, REMOVE, LIST, QUIT
:LIST
--3 albums--
1. Chicago - Chicago II
2. The Chills - Submarine Bells
3. Fugees - The Score
:ADD
Artist - Album:Norah Jones - Come Away With Me
:LIST
--4 albums--
1. Chicago - Chicago II
2. The Chills - Submarine Bells
3. Fugees - The Score
4. Norah Jones - Come Away With Me
:REMOVE
Number:2
:LIST
--3 albums--
1. Chicago - Chicago II
2. Fugees - The Score
3. Norah Jones - Come Away With Me
:REMOVE
Number:-15
:LIST
--3 albums--
1. Chicago - Chicago II
2. Fugees - The Score
3. Norah Jones - Come Away With Me
:QUIT
Bye bye!
```

When you're done, the program will have the following structure.


AlbumList.java
extends
AlbumArrayList.java
uses
ArrayListDB.java
main()
contains
uses
Album.java