Computing and Algorithms I: Project 6

CS-101 **Summer 2013**

In this programming assignment, you will create a simple database system for a library of books, music, and movies. You will create an hierarchy of classes to represent objects of these items. At the top level you will have a class called Library. Library will have two subclasses: Book, and Playable. Playable will have two subclasses: Music and Movie. Music will have two subclasses: Song and Album. Book will have two subclasses: Fiction and Nonfiction. Movie, Fiction, Nonfiction, Song, and Album will not have any subclasses.

You will also write two classes Date and RunTime, each with a compareTo method similar to that for the String class.

File Input

Your program will accept two arguments on the command line, which are non-empty strings giving the names of files in the current directory. The first file named will be the input file.

This file will contain information about songs, albums, books, or movies. The first character of each line of input will be a single letter:

- 'a': a music album.
- 'f': a fiction book.
- 'm': a movie.
- 'n': a non-fiction book
- 's': a song.

Each line of the file will contain one record of data. Each record will have a number of fields represented by strings, separated by "at signs", @.

Music albums will have the following fields represented in the line of input in this order:

1. The release date of the recording in the format ddmmyyyy where dd is the day (01 to 31), mm is the month (01 to 12), and yyyy is the year. For example, '02012001' represents January 2, 2001. As another example, '29022002' represents February 29, 2002, which is not a valid date.
2. The name of the artist ("McCartney, Paul" for example) or group ("The Beatles").
3. The title of the album ("Magical Mystery Tour").
4. The number of songs on the album.
5. The running time in the format hours:minutes:seconds. These three portions of the running time string are integers with the proviso that the hours portion can be empty or 0 for no hours, or any positive value less than 11. The minutes and seconds portions will each be values from 0 to 59 inclusive.

Music songs will have the following fields represented in the line of input in this order:

1. The release date of the recording in the format ddmmyyyy where dd is the day (01 to 31), mm is the month (01 to 12), and yyyy is the year. For example, '02012001' represents January 2, 2001. As another example, '29022002' represents February 29, 2002, which is not a valid date.
2. The name of the artist ("McCartney, Paul" for example) or group ("The Beatles").
3. The title of the song ("In My Life" for example).
4. The name of the album in which this song is found. (Note, we are not concerned with songs which are found on multiple albums).
5. The running time in the format hours:minutes:seconds. This follows the same format as for albums.

Movies will have the following fields represented in the line of input in this order:

1. The release date of the movie in the format ddmmyyyy where dd is the day (01 to 31), mm is the month (01 to 12), and yyyy is the year. For example, '02012001' represents January 2, 2001. As another example, '29022002' represents February 29, 2002, which is not a valid date.
2. The name of the studio which released the movie ("MGM" for example).
3. The title of the movie ("Star Trek 2, the Wrath of Khan" for example).
4. The name or names of the star or stars of the movie ("Neeson, Liam; Smith, Will" for example).
5. The running time in the format hours:minutes:seconds. This follows the same format as for albums and songs.

Fiction books will have the following fields represented in the line of input in this order:

1. The ISBN for the book.
2. The copyright date of the book in the format ddmmyyyy where dd is the day (01 to 31), mm is the month (01 to 12), and yyyy is the year. For example, '02012001' represents January 2, 2001. As another example, '29022002' represents February 29, 2002, which is not a valid date.
3. The name of the author(s) of the book ("Stout, Rex" for example).
4. The title of the book ("Three for the Chair" for example).
5. The genre of the book ("Mystery" for example).
6. The number of pages in the book.

Non-fiction books will have the following fields represented in the line of input in this order:

1. The ISBN for the book.
2. The copyright date of the book in the format ddmmyyyy where dd is the day (01 to 31), mm is the month (01 to 12), and yyyy is the year. For example, '02012001' represents January 2, 2001. As another example, '29022002' represents February 29, 2002, which is not a valid date.
3. The name of the author(s) of the book ("Dale, Nell; Joyce, Daniel" for example).
4. The title of the book ("Object-Oriented Data Structures Using Java" for example).
5. The book category ("Computer Science" for example).
6. The reading level of the book expressed as a number from 1 (first grade level) to 18 (Graduate student past Master's level).

Internal Requirements

Your program will use **exactly one array** of type Library to store all information read from the input file. This array will be defined to hold any number of items up to and including 100 items. This maximum number of items should be specified by a constant in your program.

Your program will not validate input. Your program should work correctly if all lines of input are correct, but is allowed to fail if any line contains invalid input. You will test your program with valid data.

You will provide exactly one Constructor for each of the classes Movie, Fiction, Nonfiction, Song, and Album. The signatures for these constructors will be:

- Album(Date, String, String, int, RunTime)
- Song(Date, String, String, String, RunTime)
- Movie(Date, String, String, String, RunTime)
- Fiction(String, Date, String, String, String, int)
- Nonfiction(String, Date, String, String, String, int)

A Sample Data Set

Here is a *sample* data file restricted to music items which could be read by this program:

```
a@15071971@Moody Blues@Every Good Boy Deserves Favour@9@:40:8
a@03101965@The Beatles@Rubber Soul@14@0:37:42
s@03101965@The Beatles@In My Life@Rubber Soul@0:2:38
s@28061971@Jethro Tull@Aqualung@Aqualung@:6:31
a@03011973@Pink Floyd@Dark Side of the Moon@10@:42:29
a@02021963@Setrak A. Setrakian@Setrak Setrakian Performing Chopin@8@1:34:58
```

Design and Coding

You will write a UML design for this project. Do not turn in the UML design generated by jgrasp, write your own design.

Test file

Create a file to test your program. This file will have at least three good input lines of each type: Fiction, Nonfiction, Album, Song, and Movie. Your file will have a minimum of 20 lines of input data, thus at least one of Fiction, Nonfiction, Album, Song, or Movie will have more than three lines of data.

Output

Your output will be written to a file. The name of the output file will be given to your program as the second command line argument (args[1]).

Your output will consist of the following items in this order:

1. A title for the output, your name, and your course and section number.
2. A label "Echo Print of Input" followed by each line of input exactly as read.
3. A label "Library Sorted by Title" followed by all the library items in Unicode (or ASCII) order sorted by title. The title is of course the title of the album, the title of the song, the title of the movie, or the title of the book. Use the String compareTo method for comparing these Strings. Each object will be printed with a label stating the type of object (Song, Album, Nonfiction Book, etc.), and each field of the object labelled (for example: Title: Three for the Chair Author: Rex Stout).
4. A label "Books Sorted by ISBN" followed by Books sorted by ISBN. Again use the String compareTo method for comparison in this sort.
5. A label "Songs sorted by Album" in which just the songs are printed in the order of the name of the album they are on. If two songs are from the same album, they will be sorted in order of name of song.
6. A label "Albums Sorted By Number of Songs" followed by the Albums listed in order by the number of songs on the albums. If two albums have the same number of songs, these albums will be in order of name of the album.
7. A label "Movies Sorted by Date" followed by all the movies sorted by date. If two Movies have the same date, these movies will be in order of movie title.
8. A label "Music Sorted by Run Time" followed by songs and albums in order of run time.

Note that you will have only one array to perform all the storage of Library objects. There will be no arrays of Music, Albums, Books, etc.

Note that you must write the code for sorting. You may not use classes such as Arrays in the API.

Submitting Your Program

This is your final project. It is due at the latest by 3:30 p.m. on Thursday September 26. You will turn in a printout consisting of your design (data tables, algorithms, UML class diagrams, UML class interaction diagram), your Java code for each class (include data tables and algorithms in comments in code), input file, and output file.

You will make an appointment to demonstrate your program to your instructor prior to 3:30 p.m. on Thursday September 26. You will demonstrate that your program works correctly during your appointment. Before the appointment you will submit using Blackboard all your Java code for the project zipped into one file.

Notes

1. *Start Early!* It is best if you complete this project before the actual start of the final exam period, then you will have those days free for exams in other courses.

2. The class `java.util.StringTokenizer` will take a `String` object and break it up into components, depending on a user-specified delimiter. Alternatively, the `Scanner` class can be used for this purpose. Alternatively, String objects have a split method which accomplishes the same task.

3. 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.