PA4 Spotify

Due: Wednesday 9/30 by 11:30PM
Submission: Submit Library.java, Playlist.java Song.java User.java and UserCollection.java to Gradescope. Do not submit the entire project.

Overview

The goal of this assignment is to write multiple classes that all interact. You will be writing 5 classes in 5 separate files to simulate Spotify.

Assignment

In this assignment you will write Spotify! Okay, not quite, but we are getting there. You have been learning about how to write classes, so that is what this assignment is all about. We have already written one file for you: PA4Main.java. **You should not edit this file**. We will be using the PA4Main.java that we distributed with the project to grade. So if you change it, that could negatively impact your grade. Below follows a description of each class. Note that you must match the names of methods exactly in order for our PA4Main.java to call the right methods. The order described below is also the order you should approach the assignment in, in our opinion anyway.

We also recommend before diving deep into each class, write ”stubs” for each class so that your code compiles. This means you write the skeleton of each class without an implementation. This will get rid of all of the red lines in PA4Main.java.

PA4Main.java and songFiles/*

We have already created the text-based UI (User Interface) for you. PA4Main.java is the ”brains” of the Spotify simulation. But it relies heavily on the different parts (classes) of the spotify application. This is where you come in! So your first task is to read and understand how PA4Main.java works. You will see that it has a TON of compiler errors because it calls a lot of functions that you have to write. You should also look at an example song file in the songFile folder. **It is critical that you understand what PA4Main.java is doing before moving on.** You can assume that all songFiles are well formatted. The artist and song title names are separated with a ”/”. It is safe to split on this ”/” to read the artist and title names.

Song.java

This class collects methods/data for a song.

- public Song(String title, String artist) - construct a new instance of the Song class with the specified song title and artist.
- public String getTitle() - return the title of the song.
- public String getArtist() - return the artist of the song.
- public int getTimesPlayed() - return the number of times this song has been played.
- public void play() - ”play” this song. To play a song you just print out a description of the song.
- public String toString() - return a string description of this song. It should be of the format: title by artist, timesPlayed play(s). So if you have the song God’s Plan by Drake and it has been played 27 times, it should return ”God’s Plan by Drake, 27 play(s)”

Library.java

This class must keep track of the library of songs in your Spotify program. This comes from the file in songFiles. That filename is passed in as a command-line argument.

- public Library() - constructs a new instance of this class.
- public Song getSong(String title) - returns the Song associated with the String title passed in if it exists in the library, or `null` if the song does not exist in the library.
- public List<Song> getAllSongs() - returns a list of all the songs in the library.
- public void addSong(Song song) - adds the passed in song to the library.
- public void removeSong(Song song) - removes the passed in song from the library if it exists in the library.
- public String toString() - returns a string representation of this library. The format is to have the string representation of each song on its own line. So you want to add the string representation of a song, then a newline for each song. **These should be printed in alphabetical order based on the song name.**

Playlist.java

This class will keep track of a playlist.

- public Playlist(String name) - constructs a new instance of the Playlist class with the specified name.

- public Playlist(String name, List<Song> contents) - constructs a new instance of the Playlist class with the specified name and songs.
- public String getName() - returns the name of the playlist.
- public void addSong(Song song) - adds the specified song to the playlist.
- public void play() - plays the playlist. In order to ”play” a song, you just need to print each song out. Remember that a song already knows how to describe itself with its toString() method.
- public void shuffle() - shuffle the songs in the playlist so they play in a different random order next time.
- public void removeSong(Song song) - remove the passed in song from the playlist if it exists in the playlist.

User.java

This class bundles methods/data for a User object.

- public User(String name, String password) - constructs a new instance of the User class with the specified name and password.
- public String getName() - returns the name of the user.
- public boolean attemptLogin(String password) - returns true if the password is valid for this user. False otherwise.
- public void addPlaylist(Playlist newPlaylist) - adds the specified playlist to the user's playlists.
- public List<Playlist> getPlaylists() - returns a list of the playlists for this user.
- public void selectPlaylist(String name) - selects the playlist with the specified name if the user has a playlist by that name and **plays the newly selected playlist**.
- public String toString() - returns a string description of the user. The format is name, numPlaylists playlists. So if I had a user cjenkins who had 12 playlists this method would return ”cjenkins, 12 playlists”.

UserCollection.java

This class tracks all of the users in Spotify.

- public UserCollection() - constructs a new instance of the UserCollection class.
- public boolean userExists(String username) - returns true if a user with the specified username exists.

- public User login(String username, String password) - returns the User associated with the login credentials if it was a valid login or returns `null` if the login was invalid.
- public void addUser(User add) - adds this user to the collection of all users.
- public String toString() - returns a string description of all the users. For example if there were two users (tconklin, cjenkins) both with 0 playlists, toString() should return "{ cjenkins: cjenkins, 0 playlists, tconklin: tconklin, 0 playlists, }".

Hints

- We haven't talked much about `null` but it is commonly used in Java. It is used to indicate nonexistence. One use case that we saw with the Scanner class was when you declare a variable, but haven't initialized it to a proper value yet. In this PA (and in many other cases) it is used as a return value from a function indicating "failure" or that the item you are searching for does not exist. Note `null` is only a valid value for objects, not primitives. Google it!

Grading Criteria

For this PA, we are providing a few testcases, but we will also be testing your code on other testcases, make sure you do the same.

Your grade will consist of similar style and code clarity points we have looked for in the first few assignments.

Write your own code. We will be using a tool that finds overly similar code. Do not look at other students' code. Do not let other students look at your code or talk in detail about how to solve this programming project. Do not use online resources that have solved the same or similar problems. It is okay to look up, "How do I do X in Java", where X is indexing into an array, splitting a string, or something like that. It is **not** okay to look up, "How do I solve this programming assignment from CSc210" and copy someone else's hard work in coming up with a working algorithm.