CPS209 Assignment 1:

MyAudio: Audio App Simulator

Due Date: Wed, Mar. 22 11:59pm

Late Policy: 10% off per day, not accepted more than 3 days late

You are to write parts of a simple audio application program (i.e. an **app**) called ***MyAudio***. The MyAudio app stores and manages audio content of various types (e.g. Songs, Audio Books, Podcasts) in a library. Audio content can be downloaded from a simulated online store. The audio content in the library can be played (i.e. in this app, playing a song means printing its lyrics). Playlists can be created from **library content**. A playlist can store songs or even a mixture of different types of audio content.

This programming assignment will increase your knowledge of array lists, objects and classes, inheritance, and interfaces. **You must do this assignment alone – no groups. Do not attempt to find source code on the web for this assignment. It will not help you and you risk extremely serious consequences. Your program will be checked for plagiarism.** Begin designing and programming early! This assignment is worth 10 percent of your final mark. If you have any questions or clarifications about the assignment, please ask on either Discord or the D2L discussion board. However, **do not post code on these forums**, as they are only intended for clarifications. If you are having issues completing parts of the assignment or otherwise have a personal issue that needs addressing, please reach out to your instructor or ask during office hours.

NOTE: some parts of the assignment are using java concepts we have not yet studied (e.g. inheritance, polymorphism, interface Comparable and interface Comparator, the use of the *super* keyword). These topics will be covered in detail in the two weeks after the break. You may want to wait until these topics are covered before writing the part of the system that uses them.

Skeleton code has been provided for you!! This skeleton code compiles and runs! It defines the Java classes for you and your job is to basically fill in the methods for these classes. In addition, one of the basic system actions has been written for you: STORE. The user interface part of another action has also been written for you: PLAYSONG – see MyAudioUI.java

We suggest you begin by compiling and running the skeleton code. Then examine it and understand how the flow of execution works for the two actions provided. Insert temporary print statements to help you understand. Start in MyAudioUI.java and see how it calls the methods in Library.java and how the MyAudio app uses the support classes (e.g. Song, AudioBook etc.). Then, begin by implementing the simpler actions (see the marking scheme at the end of this document and use it as a guide). Also use the video as a guide. That is, write the code necessary to get a single action completely working (compiled, debugged and tested). Then move on to the next action. This way of "growing" your code piece by piece will minimize bugs and allow you to always have a working system to submit – even if you have not finished some of the more difficult actions. NOTE: if you submit code that does not compile, you are eligible for at most 2 marks out of 10.

Program Functionality Requirements:

Please view the video included with this assignment. Below is a description of the classes for your assignment and a list of methods and variables. **NOTE:** We **have posted skeleton code for you of all the necessary classes. Some of these classes are complete, most are not. You are to fill in code for the methods according to the instructions below**

and according to the comments in the skeleton code java files. This makes the assignment much easier to mark for the TAs and ensures you are adhering to proper OO design.

Look at the constructor method of AudioContentStore.java and you will see that we have created a list of songs and audiobooks. These can be used for testing. Feel free to update this file as you choose to do any additional testing or debugging. The given examples will cover the vast majority of tests we plan to run, but we will replace it as needed for marking. However, for all other java files, only update the files as specified.

Please look closely at the skeleton code and comments within. For this first assignment, you should always strive to use the methods outlined in the skeleton code rather than creating your own. In other words, we are controlling the overall design in this assignment. **You are permitted to add supporting methods and other instance variables if you deem it necessary.** Here is a list of the classes in the app, including descriptions about which you are to modify:

1. **Library:** this class contains the bulk of the logic for the app. It maintains an array list of songs, an array list of audiobooks, an array list of podcasts and an array list of playlists. These lists are initially empty. Some methods have been written for you. Fill in the code for the other methods based on the comments. See the video to help you print output in the correct format.

2. **MyAudioUI:** This class has the **main() method and the user interface (UI). Some UI skeleton code has been provided for you with some comments.** In a ***while loop***, a scanner reads a line of input from the user. The input lines contain words (Strings) which represent actions (e.g. songs, store, download etc). Some actions require parameters to be input by the user. The parameters may be strings or an integer. The code should prompt the user by printing out what parameter should be entered. See the video and look at the example code provided for you for action PLAYSONG in MyAudio.java. Fill in the code based on the comments for all the other actions. **NOTE: do not change the user interface!!** The TAs are marking 450 assignments and may read actions from a file as input to your program.

3. **AudioContent:** class AudioContent ***is a general superclass*** containing several instance variables (see the skeleton code) that model audio content. This class can be extended to model more specific types of audio content (e.g. song, podcast, audiobook). It has instance variables: title (String), year (int), id (String), type (String), audioFile (String), length (int). For example a title might be "Yesterday". The ID (a string) is generated by the system (e.g. "913"). **This class has been done for you**!!

4. **AudioContentStore:** class AudioContentStore simulates an online store. The MyAudio app can download content (e.g. a song, an audiobook) from this store to your library. When the user types the action STORE then a method is called to list the store contents. The contents are numbered (1, 2, 3 etc). Each content has a type (Song, AudioBook, Podcast). The user can type the DOWNLOAD action and then specify the number of the content they want stored in their library. **This class has been done for you**!! It contains some predefined songs and audiobooks (see the constructor method) which you will use to test your code.

5. **Song:** class Song is a subclass of Audio Content (i.e. a song **is a** type of audio content). In addition to the AudioContent information, a song has fields to store the artist name, the composer name, the genre and the lyrics. These fields have been created for you. See the skeleton code and write code for the methods (including the constructor method) by following the comments.

6. **AudioBook:** This class is also a subclass of AudioContent that contains extra information. An AudioBook contains the book's author and AudioBook's narrator. It also contains two array lists of Strings: one that contains the chapter titles and one that contains the chapter contents. The class has a method to select a specific chapter to play. Fill in the methods by following the comments in the AudioBook.java file.

7. **Playlist:** class Playlist stores a list of audio content (e.g. songs or audiobooks or podcasts or even a mixture of these). A Playlist is given a title (e.g. "Workout" "Dinner Music" "Dance Mix"). There are a series of user actions to make a playlist, add content to a playlist etc. All the content can be played (i.e. playAll()) or a specific audio content can be played (e.g. song 5)

8. ***BONUS*: Podcast:** Design a class which is also a subclass of AudioContent that contains extra information. A podcast is essentially a talk radio series on demand. Podcasts tend to be focused on a theme or topic. Podcasts should have a host (String) as well as a list of Seasons. Each Season should consist of a list of episodes (strings representing the "audiofiles") a list of episode titles (strings) and a list of episode lengths (in minutes). See the video of an example of podcast content. Provide the ability to play a specific episode of a season. You may want to create a class Season to hold the episode information for a season. Add code to MyAudioUI.java for actions PODCASTS, PLAYPOD. Add methods to class Library.

Grading (Out of 10 marks)

Highest Grade Possible: 11

Basic: actions DOWNLOAD, SONGS, BOOKS work correctly	3 marks
Actions PLAYSONG, PLAYBOOK	2 marks
Action ARTISTS, DELSONG	1 mark
Actions MAKEPL, PRINTPL, ADDTOPL, DELFROMPL, PLAYALLPL, PLAYPL,	3 marks
Actions SORTBYLENGTH, SORTBYYEAR	0.5 marks
Action SORTBYNAME	0.5 marks
BONUS: Class Podcast (actions PODCASTS, PLAYPOD, PODTOC)	1 mark
Penalty for insufficient comments or bad program structure or bad use of inheritance and interfaces	Up to -1 marks

Submitting Your Assignment: READ CAREFULLY!!

- Inside each of your ".java" files have your name and student id as comments at the top of the file. Make sure your files can be compiled as is without any further modification!! **Try compiling on a command line using javac.**
- Include a README.txt file that describes what aspects of your program works and what does not work (or partially works). If your program does not compile, state this!! You may still get 1 or 2 (out of 10) part marks. The TA will use this README file as a guide and make fewer marking mistakes.
- Place all your ".java" file(s) and README.txt file into an archive (zip or rar archiving format only).
- Open your archive and make sure all your files are there.
- **Do not include ".class" files!!!!**
- **Remove any Package keywords in your java files!!!**
- Once everything is ready, submit it to D2L.