**CS 116 Final Project Fall 20212**

One submission per group

**Design document:** due in Blackboard: Tuesday, November 15, 2022 at 11:00 PM CST

**Final code:** submission (updated design for re-grading and final code) due in Blackboard: Wednesday, December 7, 2022 at 11:00 PM CST

**Project objectives**

1. [20 points] Design a multi-object application.
2. [80 points] Implement a multi-object application.

**Description**

Your task is to design and implement a Streaming System which allows **multiple users** to **organize recordings (audio or video) into playlists and play them**. This system expands some of your past lab problems. Re-use the code you already have where possible. The Streaming System should:

* Have a method run that runs displays the menu and runs the system,
* Keep a “database” of **UNIQUE** (with unique ID) users that allows to:
  + add new users,
  + remove existing users based on ID or name,
* Have a menu with following options and sub-menus and controlled using keyboard:
  + Add user,
    - Provide username
  + Remove user,
    - Remove user based on ID,
    - Remove user based on username,
  + List all users,
  + User
    - Add recording
      * provide all recording information,
    - Add playlist from file,
      * provide file name
    - Add playlist from another user
      * Provide user ID (cannot be the same)
    - Remove recording from playlist
      * Remove based on index
        + Provide index
      * Remove based on name
        + Provide recording name
    - Play individual recording
      * Based on index
        + Provide index
      * Based on name
        + Provide name
    - Play entire playlist once
    - Shuffle entire playlist once
    - Save playlist to a file
    - Display playlist stats
  + Exit
* Each user has their own **playlist** (initially empty) and can:
  + **Add** an individual recording (either audio or video; use overloading and dedicated methods to handle each type) – duplicates not allowed (**no ARTIST – NAME pair duplicates**; only one version is allowed whether it is an audio or video),
  + **Add** an entire set of recordings **from a file** (file name is an input) to the playlist – duplicates should be ignored (**no ARTIST – NAME pair duplicates**; only one version is allowed whether it is an audio or video).
  + **Add** an entire set of recordings from **another playlist** (for example: from another user) to the playlist – duplicates should be ignored (**no ARTIST – NAME pair duplicates**; only one version is allowed whether it is an audio or video),
  + **Remove** individual recording in a playlist **based on its playlist index / position**,
  + **Remove** individual recording **based on its name** (if more than one recording with the same name – remove each),
  + **Play** individual recording in a playlist **based on its playlist index / position**,
  + **Play** individual recording **based on its name** (if more than one recording with the same name – play each),
  + **Play** entire playlist in order it was populated (from recording 1 to last),
  + **Shuffle** entire playlist once (play entire playlist in random order once),
  + **Save** entire playlist under the following name USERNAME\_PLAYLIST\_MM\_DD\_YYYY\_HH\_MM\_SS.csv, where MM\_DD\_YYYY\_HH\_MM\_SS is current date and time). It should have the same format as the sample file,
  + **Display playlist statistics**: display all playlist songs (ARTIST – NAME – Number of plays),
  + **NOTE**: **every recording playback should increment its (recording) number-of-times-played counter**.

Other requirements:

* You can use sample BillboardHot100.csv file to practice and test your code,
* Use ArrayLists,
* Modify Playlist class: it has to be based on an ArrayList
* Multiple users can share the same name,
* Users cannot change their name,
* Use static attribute to generate unique user IDs,
* Create an interface Playable with one abstract method:
  + public abstract void play()
* User, Playlist and Recording with its descendants should implement Playable,
* Make Recording class abstract and make any other necessary modifications if there is a need for it,
* Create an Unplayable exception class (not covered in class yet). It will be thrown whenever a recording is unplayable (0 duration). Your code should try-catch it somewhere,
* Write an app class (this is not the Streaming System class) that runs everything

**Deliverables**

The following deliverables are required at their corresponding deadlines:

* **Design stage** (due in Blackboard: Friday, Tuesday 15th at 11:00 PM CST):
  + **[20 pts]** Use the design document draft provided below to provide a DETAILED system design specification along with test cases. I will review it and send it back to you with my comments. Add notes where appropriate,
  + **[5 pts]**
* **Coding stage [80 pts]:** Your final code submission along with updated design document

**Project design draft**

Use the following tables (copy, extend as you please) to outline your design for each:

* Class or interface,
* Class interaction,
* Test case (provide at least two test cases for each method: one that leads to some failure and one that represents legitimate behavior).

Class / interface design table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class or interface name  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | Abstract or concrete: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | # of instances [0, 1, many] \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| Attributes | | | | | | | | |
| Name | type | Static [YES/NO] | | Access modifier | Initial value if any | Range of possible values | | Notes / comments |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
| Methods (including constructors) | | | | | | | | |
| Name | Return type | Static [YES/NO] | | Abstract [YES/NO] | Access modifier | List of arguments and their types | | Notes / comments |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |
|  |  |  | |  |  |  | |  |

Class / object interactions design table (A class / object interacts (calls a method of) with class B):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class / Object A | calls method (it can be constructor to create an object) | of class / object B with arguments (if any) | to | Notes: |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Test case design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case name | Tested class | Tested method | Test input | Expected outcome |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |