**CS 116 Final Project Fall 2022**

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 ^(Kinda)
    - 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  **StreamingSystemApp** | | | Abstract or concrete: Concrete | | | | # of instances [0, 1, many] = 0 | |
| 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 |
| main | void | Yes | | No | public | String[] args | | Runs entire application |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class or interface name  **User** | | | Abstract or concrete: Concrete | | | | # of instances [0, 1, many] = many | |
| Attributes | | | | | | | | |
| Name | type | Static [YES/NO] | | Access modifier | Initial value if any | Range of possible values | | Notes / comments |
| userPlaylist | Playlist | No | | private |  | Recording subclasses | |  |
| userName | String | No | | private | “Default” | Any String | |  |
| userID | int | No | | private |  |  | | Also serves as index in userDB |
|  |  |  | |  |  |  | |  |
| Methods (including constructors) | | | | | | | | |
| Name | Return type | Static [YES/NO] | | Abstract [YES/NO] | Access modifier | List of arguments and their types | | Notes / comments |
| getPlaylist | Playlist | No | | No | Public | N/A | | Returns user’s playlist |
| getUsername | String | No | | No | Public | N/A | | Returns the user’s name |
| toString | String | No | | No | Public | N/A | | Override toString to show username instead |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class or interface name  **Playlist** | | | Abstract or concrete: Concrete | | | | # of instances [0, 1, many] = 1 | |
| Attributes | | | | | | | | |
| Name | type | Static [YES/NO] | | Access modifier | Initial value if any | Range of possible values | | Notes / comments |
| recordingsList | ArrayList<Recordings> | No | | Private | N/A | Recording Objects | | Initialized in constructor |
| Name | String | No | | Private | N/A | User’s name + “playlist” | | Think that it is necessary for the toString method |
| Duration | Int | No | | PPrivate | 0 | 0 to MAX\_INT | | Length of playlist in seconds, also for toString |
| Methods (including constructors) | | | | | | | | |
| Name | Return type | Static [YES/NO] | | Abstract [YES/NO] | Access modifier | List of arguments and their types | | Notes / comments |
| Playlist | constructor | No | | No | Public | N/A | | Default constructor |
| Playlist | constructor | No | | No | Public | String name | | Parameterized constructor |
| setName | Void | No | | No | Public | String name | | Setter for name, checks if null |
| getName | String | No | | No | Public | N/A | | Getter |
| getDuration | int | No | | No | Public | N/A | | Getter |
| add | void | No | | No | public | Recording newRecording | | Adds recording object to ArrayList |
| play | void | No | | No | public | N/A | | Displays ArrayList in such a way that it “plays” it |
| shuffle | void | No | | No | public | Int numberOfRecordings | | “Plays” the specified amount of recordings from the playlist in a random order |
| load | void | No | | No | public | String fileName | | Takes the contents of a csv file and then determines if each line makes a valid recording object. Afterwards replacing the contents of the ArrayList with the parsed values |
| toString | String | No | | No | public | N/A | | Overrides toString |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class or interface name  **Recording** | | | Abstract or concrete: Concrete | | | | # of instances [0, 1, many] = 0 | |
| Attributes | | | | | | | | |
| Name | type | Static [YES/NO] | | Access modifier | Initial value if any | Range of possible values | | Notes / comments |
| artist | String | no | | protected |  | All Strings | | Artist name |
| recordingName | String | no | | protected |  | All Strings | | Content name |
| duration | int | no | | protected |  | Positive Ints | | Length of content |
|  |  |  | |  |  |  | |  |
| Methods (including constructors) | | | | | | | | |
| Name | Return type | Static [YES/NO] | | Abstract [YES/NO] | Access modifier | List of arguments and their types | | Notes / comments |
| Recording | N/A | No | | No | public | N/A | | Default constructor |
| Recording | N/A | No | | No | public | String artist, String recordName, int duration | | Parameterized constructor |
| getArtist | String | No | | No | public | N/A | | Getter |
| getName | String | No | | No | public | N/A | | Getter |
| getDuration | int | No | | No | public | N/A | | Getter |
| play | void | No | | No | public | N/A | | “Plays” the specified recording by printing a string to console |
| toString | String | No | | No | public | N/A | | Returns string of relevant data to the song like the attributes of the object in string form and formatted |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class or interface name  **AudioRecording** | | | Abstract or concrete: Concrete | | | | # of instances [0, 1, many] = many | |
| Attributes | | | | | | | | |
| Name | type | Static [YES/NO] | | Access modifier | Initial value if any | Range of possible values | | Notes / comments |
| bitrate | double | No | | Private | 0 | Positive double | |  |
| artist | String | no | | protected |  | All Strings | | Inherited from Recording |
| recordingName | String | no | | protected |  | All Strings | | Inherited from Recording |
| duration | int | no | | protected |  | Positive Ints | | Inherited from Recording |
| Methods (including constructors) | | | | | | | | |
| Name | Return type | Static [YES/NO] | | Abstract [YES/NO] | Access modifier | List of arguments and their types | | Notes / comments |
| AudioRecording | N/A | no | | no | public | N/A | | Default constructor |
| AudioRecording | N/A | no | | no | public | String artist, String recordName, int duration,  Double bitrate | | Parameterized constructor |
| getBitrate | double | no | | no | public | N/A | | Getter |
| toString | String | no | | no | public | N/A | | toString overwrite |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class or interface name  **VideoRecording** | | | Abstract or concrete: Concrete | | | | # of instances [0, 1, many] = 0 | |
| Attributes | | | | | | | | |
| Name | type | Static [YES/NO] | | Access modifier | Initial value if any | Range of possible values | | Notes / comments |
| framerate | double | No | | Private | 0 | Positive double | |  |
| artist | String | no | | protected |  | All Strings | | Inherited from Recording |
| recordingName | String | no | | protected |  | All Strings | | Inherited from Recording |
| duration | int | no | | protected |  | Positive Ints | | Inherited from Recording |
| Methods (including constructors) | | | | | | | | |
| Name | Return type | Static [YES/NO] | | Abstract [YES/NO] | Access modifier | List of arguments and their types | | Notes / comments |
| VideoRecording | N/A | no | | no | public | N/A | | Default constructor |
| VideoRecording | N/A | no | | no | public | String artist, String recordName, int duration,  Double framerate | | Parameterized constructor |
| getFramerate | double | no | | no | public | N/A | | Getter |
| toString | String | no | | no | public | N/A | | toString overwrite |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class or interface name  **StreamingSystem** | | | Abstract or concrete: Concrete | | | | # of instances [0, 1, many] = many | |
| Attributes | | | | | | | | |
| Name | type | Static [YES/NO] | | Access modifier | Initial value if any | Range of possible values | | Notes / comments |
| userList | ArrayList<User> | no | | private |  | User objects | |  |
| inputNumber | int | no | | private | 0 |  | |  |
| currUser | User | no | | private | null | User objects/null | | Current user that is “logged in” (allows access to “User” submenu) |
| userNumber | int | no | | private | 0 | 0-10 | | Choice in the menu that the user has made |
| Methods (including constructors) | | | | | | | | |
| Name | Return type | Static [YES/NO] | | Abstract [YES/NO] | Access modifier | List of arguments and their types | | Notes / comments |
| StreamingSystem | void | no | | no | private | N/A | | Default constructor |
| menuChoice | int | no | | no | private | N/A | | Gets the inputted number for a menu choice and checks if it is valid |
| loadDB | void | yes | | no | private | N/A | | Loads the file called “UserDB.csv” and loads any valid user objects from the parsed file |
| displayMenu | void | yes | | no | private | N/A | | Displays a string to console for the user to pick a choice based on the current state of the program |
| startChoices | void | no | | no | public | N/A | | Loop of displaying choices and then taking a choice from the user |
| actions | void | no | | no | private | N/A | | Set of actions to take depending on the user’s input |

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: |
| StreamingSystem | constructor | User | Make new users and add them to userList |  |
| StreamingSystem | getPlaylist | User | Use the methods of the users playlist |  |
| StreamingSystem | getUsername | User | Get the username of the user for printing and saving to file |  |
| StreamingSystemApp | constructor | StreamingSystem | Make StreamingSystem |  |
| StreamingSystemApp | startChoices | StreamingSystem | Start main logic loop |  |
| User | constructor | Playlist | Make playlist for user |  |
| User | addRecording | Playlist | Add new recording | Whether the recording is Video or Audio is dependent on further prompts |
| User | removeRecording | Playlist |  |  |
| User | play | Playlist | “Play” the contents of the users Playlist | Could be overloaded to also do one recording |
| User | replacePlaylist | Playlist | Replace the current playlist from a specified source | Will be overloaded to have a user or a file |
| User | showPlaylist | Playlist | Display contents of playlist to console |  |
| Playlist | constructor | Recording/VideoRecording/AudioRecording | Make a Recording object when adding to the playlist |  |
| User | shufflePlaylist | Playlist | “Play” the playlist in a random order |  |

Test case design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case name | Tested class | Tested method | Test input | Expected outcome |
| Invalid input number | StreamingSystem | menuChoice | A number that is not listed as a choice | To repeatedly ask for a valid input |
| “Login” as a user that doesn’t exist | StreamingSystem | actions | A username that does not exist | To inform that the user does not exist and ask for a new valid input |
| Invalid recording values | Playlist | addRecording | Either the wrong data for making a recording or a invalid recording | Inform the user that the data is wrong and then prompt for an attempt to make another recording |
| Invalid/missing file | Playlist | load | File that does not exist or is the wrong type | Inform the user that the files is incompatible and take another input |
| Wrong user refrence | Playlist | load | Takes a user that does not exist and tries to copy the playlist that does not exist | Inform the user that the User object does not exist right now, prompt for either trying a different user or stopping |