COMP 1406Z
Fall 2021 - Tutorial #3

Objectives

- Practice working with ArrayLists
- Practice working with HashMaps
- Practice using the Java Collections class

Getting Started:

Download the **Tutorial 3.zip** file from the Brightspace website. This file contains an IntelliJ project with the starting resources for this tutorial.

In this tutorial, you will simulate an on-line music centre called the **MusicExchangeCenter** where **Users** log in and download **Songs** from other users' computers. It is a similar idea to the operation of the original Napster program that operated many years ago where users shared songs. You won't be doing any downloading or internet programming. Instead, you will just simulate what happens in real life. The tutorial will give you familiarity with ArrayLists and HashMaps.

(1) The **Song/ User** Classes

The **Song** and **User** classes represent a song that is available at the Music Exchange Center and a user of the Music Exchange Center that logs in to download music.

Do the following in the **User** class:

Add a **songList** attribute which is an **ArrayList** of **Song** objects. This list will contain all the songs that this user has on his/her hard drive to be made available to the Music Exchange Center community. Adjust the 2nd constructor to ensure that this attribute is initialized properly. Write a get method for the attribute as well.

Adjust the **toString()** method so that the XXX is replaced by the number of songs available in the user's song list.

Create a method called **addSong(Song s)** which adds a given song to the user's song list.

Create a method called **totalSongTime()** that returns an integer indicating the total duration (i.e., amount of time), in seconds, that all of the user's songs would require if played.

(2) The **MusicExchangeCenter** Class

Implement a class called **MusicExchangeCenter** which represents the company website that users log in and out of in order to download music. The class should have this attribute:

> **users** - an **ArrayList** of all registered **Users** (which may be either logged on or not logged on).

Create the following methods:

> A zero-parameter constructor that sets attributes properly.
>
> An **onlineUsers()** method that returns an **ArrayList** of all **Users** that are currently online. Note that this method creates and returns a new **ArrayList** each time it is called.
>
> An **allAvailableSongs()** method that returns a new **ArrayList** of all **Songs** currently available for download (i.e., all songs that are available from all logged-on users). Note that this method creates and returns a new **ArrayList** each time it is called.
>
> A **toString()** method that returns a string representation of the music center showing the number of users currently online as well as the number of songs currently available.
>
> (e.g., **`"Music Exchange Center (3 users online, 15 songs available)"`**).
>
> A **userWithName(String s)** method that finds and returns the user object with the given name if it is in the list of users. If not there, **null** should be returned.
>
> A **registerUser(User x)** method that adds a given **User** to the music center's list of users, provided that there are no other users with the same userName. If there are other users with the same userName, then this method does nothing. Use the **userWithName(String s)** method above.
>
> An **availableSongsByArtist(String artist)** method that returns a new **ArrayList** of all **Songs** currently available for download by the specified artist. Note that this method creates and returns a new **ArrayList** each time it is called.

Go back to the **User** class and add these methods:

> A **register(MusicExchangeCenter m)** that makes use of the **registerUser(User x)** method that you just wrote to register the user into the given **MusicExchangeCenter m**. (Note that this should be a one-line method).
>
> A **logon()** method that simulates a user going online.
>
> A **logoff()** method that simulates a user going offline.

Now you should test your code. To do so, run the **MusicExchangeTestProgram.java** file and compare your results to the expected results below.

Here is the output that you should see:

```
Status: Music Exchange Center (0 users on-line, 0 songs available)
On-Line Users: []
Available Songs: []

Status: Music Exchange Center (2 users on-line, 8 songs available)
```

```
On-Line Users: [Disco Stew: 4 songs (online), Ronnie Rocker: 4 songs (online)]
Available Songs: ["Hey Jude" by The Beatles 4:35, "Barbie Girl" by Aqua 3:54, "Only You Can Rock
Me" by UFO 4:59, "Paper Soup Cats" by Jaw 4:18, "Rock is Cool" by Yeah 4:17, "My Girl is Mean to
Me" by Can't Stand Up 3:29, "Only You Can Rock Me" by UFO 4:52, "We're Not Gonna Take It" by
Twisted Sister 3:9]

Status: Music Exchange Center (4 users on-line, 16 songs available)
On-Line Users: [Disco Stew: 4 songs (online), Ronnie Rocker: 4 songs (online), Country Candy: 3
songs (online), Peter Punk: 5 songs (online)]
Available Songs: ["Hey Jude" by The Beatles 4:35, "Barbie Girl" by Aqua 3:54, "Only You Can Rock
Me" by UFO 4:59, "Paper Soup Cats" by Jaw 4:18, "Rock is Cool" by Yeah 4:17, "My Girl is Mean to
Me" by Can't Stand Up 3:29, "Only You Can Rock Me" by UFO 4:52, "We're Not Gonna Take It" by
Twisted Sister 3:9, "If I Had a Hammer" by Long Road 4:15, "My Man is a 4x4 Driver" by Ms. Lonely
3:7, "This Song is for Johnny" by Lone Wolf 4:22, "Bite My Arms Off" by Jaw 4:12, "Where's My
Sweater" by The Knitters 3:41, "Is that My Toenail ?" by Clip 4:47, "Anvil Headache" by Clip 4:34,
"My Hair is on Fire" by Jaw 3:55]
Available Songs By Jaw: ["Paper Soup Cats" by Jaw 4:18, "Bite My Arms Off" by Jaw 4:12, "My Hair is
on Fire" by Jaw 3:55]

Status: Music Exchange Center (2 users on-line, 9 songs available)
On-Line Users: [Ronnie Rocker: 4 songs (online), Peter Punk: 5 songs (online)]
Available Songs: ["Rock is Cool" by Yeah 4:17, "My Girl is Mean to Me" by Can't Stand Up 3:29,
"Only You Can Rock Me" by UFO 4:52, "We're Not Gonna Take It" by Twisted Sister 3:9, "Bite My Arms
Off" by Jaw 4:12, "Where's My Sweater" by The Knitters 3:41, "Is that My Toenail ?" by Clip 4:47,
"Anvil Headache" by Clip 4:34, "My Hair is on Fire" by Jaw 3:55]
Available Songs By Jaw: ["Bite My Arms Off" by Jaw 4:12, "My Hair is on Fire" by Jaw 3:55]

Status: Music Exchange Center (0 users on-line, 0 songs available)
On-Line Users: []
Available Songs: []
Available Songs By Jaw: []
```

---

(3) Downloading Music

Go back to the **Song** class and add to it an attribute called **owner** which will be a **User** object representing the user who happens to own this copy of this song. Note that a **Song** object may only be owned by one **User** and that many users can have copies of the same song. Set it to **null** initially. In the **User** class, adjust the **addSong(Song s)** method so that it sets the owner properly.

In the **User** class, add a method called **requestCompleteSonglist(MusicExchangeCenter m)**. This method should gather the list of all available songs from all users that are online at the given music exchange center (i.e., the union of all of their local song lists), and then return an **ArrayList<String>** formatted as follows (note: there should be 1 String element in the ArrayList per line):

```
TITLE                          ARTIST           TIME  OWNER

1. Hey Jude                    The Beatles      4:35  Disco Stew
2. Barbie Girl                 Aqua             3:54  Disco Stew
3. Only You Can Rock Me        UFO              4:59  Disco Stew
4. Paper Soup Cats             Jaw              4:18  Disco Stew
5. Rock is Cool                Yeah             4:17  Ronnie Rocker
6. My Girl is Mean to Me       Can't Stand Up   3:29  Ronnie Rocker
7. Only You Can Rock Me        UFO              4:59  Ronnie Rocker
8. We're Not Gonna Take It     Twisted Sister   3:09  Ronnie Rocker
```

Notice that the songs are numbered and that the title, artist, time and owner are all lined up nicely. You should use the **String.format()** method as described in the notes (Chapter 1). Recall that **%-30s** in the format string will allow you to display a left-justified 30-character string. Also, **%2d** and **%02d** will allow you to display numbers so that they take 2 places, the 0 indicating that a leading zero character is desired.

In the **User** class, add a method called **requestSonglistByArtist(MusicExchangeCenter m, String artist)**. This method should gather the list of all available songs **by the given artist** from all users that are online at the given music exchange center (i.e., the union of all of their local song lists), and then return an **ArrayList<String>** formatted similar to that shown above.

In the **MusicExchangeCenter** class, add a method called **getSong(String title, String ownerName)** which returns the **Song** object with the given **title** owned by the user with the given **ownerName**, provided that the user is currently online and that the song exists. Return **null** otherwise. (Hint: you will need to search through the center's **users** to find **User** with the matching **ownerName** and then search through that user's songs to find the **Song** with the given **title**. It may be a good idea to write an extra helper method in the **User** class called **songWithTitle(String title)** that you can make use of).

In the **User** class, add a **downloadSong(MusicExchangeCenter m, String title, String ownerName)** method that simulates the downloading of one of the songs in the catalog. It should use the **getSong(String title, String ownerName)** method that you just wrote, and add the downloaded song to the user's **songList** (if not **null**).

Download and execute the **MusicExchangeTestProgram2.java** file to test your new code. Here is the expected output:

```
Status: Music Exchange Center (5 users on-line, 18 songs available) Complete Song
List:
    TITLE                          ARTIST           TIME   OWNER
```

```
 1. Hey Jude                    The Beatles      4:35   Disco Stew
 2. Barbie Girl                 Aqua             3:54   Disco Stew
 3. Only You Can Rock Me        UFO              4:59   Disco Stew
 4. Paper Soup Cats             Jaw              4:18   Disco Stew
 5. Rock is Cool                Yeah             4:17   Ronnie Rocker
 6. My Girl is Mean to Me       Can't Stand Up   3:29   Ronnie Rocker
 7. Only You Can Rock Me        UFO              4:52   Ronnie Rocker
 8. We're Not Gonna Take It     Twisted Sister   3:09   Ronnie Rocker
 9. Meadows                     Sleepfest        7:15   Sleeping Sam
10. Calm is Good                Waterfall        6:22   Sleeping Sam
11. If I Had a Hammer           Long Road        4:15   Country Candy
12. My Man is a 4x4 Driver      Ms. Lonely       3:07   Country Candy
13. This Song is for Johnny     Lone Wolf        4:22   Country Candy
14. Bite My Arms Off            Jaw              4:12   Peter Punk
15. Where's My Sweater          The Knitters     3:41   Peter Punk
16. Is that My Toenail ?        Clip             4:47   Peter Punk
17. Anvil Headache              Clip             4:34   Peter Punk
18. My Hair is on Fire          Jaw              3:55   Peter Punk

Disco Stew before downloading: Disco Stew: 4 songs (online)
Disco Stew after downloading: Disco Stew: 7 songs (online)
Disco Stew after downloading Ronnie's: Disco Stew: 8 songs (online)

Song's by Jaw:
            TITLE                       ARTIST            TIME   OWNER

         1. Paper Soup Cats             Jaw              4:18   Disco Stew
         2. Bite My Arms Off            Jaw              4:12   Peter Punk
         3. Bite My Arms Off            Jaw              4:12   Peter Punk
         4. My Hair is on Fire          Jaw              3:55   Peter Punk
```

(4) Paying the Price

Now, in order to make all of this legal, we must have a way to compensate the musical artists for their hard work and wonderful music. An artist should receive 25 cents each time one of their songs is downloaded. Hence, for each artist, we will keep track of how much money in royalties they have. Add the following attributes to the **MusicExchangeCenter** class:

> **royalties** - a **HashMap** with the artists' names as the keys and the values are floats representing the total amount of royalties for that artist so far. It should only contain artists who have had songs downloaded. You will update this HashMap later.
>
> **downloadedSongs** – an **ArrayList<Song>** containing all of the songs that have been downloaded. This list will, in general, contain duplicate **Song** objects.

Write a method in the **MusicExchangeCenter** class called **displayRoyalties()** that displays the royalties for all artists who have had at least one of their songs downloaded. It should display a two-line header and then one line per artist showing the royalty amount as well as the artist name as follows:

```
Amount  Artist
--------------
$0.75   Sleepfest
$1.50   Clip
$0.25   Jaw
$0.50   Long Road
$0.25   Yeah
$0.25   UFO
```

In the **getSong()** method in the **MusicExchangeCenter**, adjust the code so that if the song is found (i.e., able to be downloaded), then it is added to the **downloadedSongs** list. Additionally, the royalties for the artist of the song should be updated within this method.

Write a method in the **MusicExchangeCenter** class called **uniqueDownloads()** that returns (i.e., not displays) a **TreeSet** of all downloaded **Song** objects such that the set is sorted alphabetically by song title. There should be no duplicate songs in this set. To add the Song objects to the TreeSet, the Song class will need to **implement Comparable<Song>** and you will need to write the corresponding **compareTo(Song s)** method in the Song class. There is a short example in the readings that uses a TreeSet (see Chapter 8, page 299).

Write a method in the **MusicExchangeCenter** class called **songsByPopularity()** that returns (i.e., not displays) an **ArrayList** of **Pair<Integer,Song>** objects where the key of the pair is an **Integer** representing the number of downloads and the value is the **Song** object. The integer key should represent the number of times that the song was downloaded. The list returned should be sorted in decreasing order according to the number of times the song was downloaded. To sort a list of Pair objects, you can use the following code:

```
Collections.sort(yourListOfPairs, new Comparator<Pair<Integer, Song>>() {
    public int compare(Pair<Integer, Song> p1, Pair<Integer, Song> p2) {
        // PUT YOUR CODE IN HERE
    }
});
```

Just insert the missing code so that it returns an appropriate integer indicating whether pair **p1** comes before or after pair **p2** in the sort order (see notes, Chapter 8, Page 281).

Run the **MusicExchangeTestProgram3.java** test file to make sure it works with your code.

Here is the expected output:

```
Status: Music Exchange Center (5 users on-line, 18 songs available)

Here are the downloaded songs:
"Bite My Arms Off" by Jaw 4:12
"Meadows" by Sleepfest 7:15
"If I Had a Hammer" by Long Road 4:15
"Is that My Toenail ?" by Clip 4:47
"Anvil Headache" by Clip 4:34
"Is that My Toenail ?" by Clip 4:47
"If I Had a Hammer" by Long Road 4:15
"Anvil Headache" by Clip 4:34
"Meadows" by Sleepfest 7:15
"Only You Can Rock Me" by UFO 4:52
"Is that My Toenail ?" by Clip 4:47
"Is that My Toenail ?" by Clip 4:47
"Rock is Cool" by Yeah 4:17
"Meadows" by Sleepfest 7:15

Here are the unique downloaded songs alphabetically:
"Anvil Headache" by Clip 4:34
"Bite My Arms Off" by Jaw 4:12
"If I Had a Hammer" by Long Road 4:15
"Is that My Toenail ?" by Clip 4:47
"Meadows" by Sleepfest 7:15
"Only You Can Rock Me" by UFO 4:52
"Rock is Cool" by Yeah 4:17

Here are the downloaded songs by populariry:
(4 downloads) "Is that My Toenail ?" by Clip 4:47
(3 downloads) "Meadows" by Sleepfest 7:15
(2 downloads) "Anvil Headache" by Clip 4:34
(2 downloads) "If I Had a Hammer" by Long Road 4:15
(1 downloads) "Bite My Arms Off" by Jaw 4:12
(1 downloads) "Only You Can Rock Me" by UFO 4:52
(1 downloads) "Rock is Cool" by Yeah 4:17

Here are the royalties:

Amount  Artist
---------------
$0.75   Sleepfest
$1.50   Clip
$0.25   Jaw
$0.50   Long Road
$0.25   Yeah
$0.25   UFO
```

Submission

Zip your completed tutorial #3 project and submit your **tutorial3.zip** file to the Tutorial #3 submission on Brightspace. Make sure you download and test your submission after you have

submitted. Submitting a corrupt zip or a zip file that does not have the correct files will result in a loss of marks.