Homework 6 – DJ Stasko

Authors: Allan Nguyen, Lucas Morais Henrique, David Cornell, Sumit Choudhury

Problem Description

This assignment will test your knowledge on File I/O and exceptions.

You are a 1331 student who deeply enjoys the groovy songs that DJ Stasko plays every time before lecture. You decide to organize all of DJ Stasko's songs into a song database!

Make sure to download **`database_songs.zip`** from Canvas, place it in the same directory as your code, and unzip it.

Song Database Folder Structure

Within the `database_songs` directory, there are directories that represent genres. Within each genre directory, there are artist directories. Within the artist directories, there are song txt files. Please make sure to look inside the song txt files!

```
database_songs
|___ genre1
    |___ artist1
        |___ song1
|___ genre2
    |___artist2
        |___ song2
        |___ song3
```

Solution Description

You will create a database represented through an ArrayList of Song objects by iteratively traversing through the folder structure to create Song objects. You will then create various static methods that will interact with the ArrayList and folder structure.

Song.java

Instance Data :

- String name – the song name
- String artist – the song's artist
- String genre – the song's genre
- File file – the song txt file

Constructor:

- 4-param constructor that takes in arguments in the following order: name, artist, genre, file.

Methods:

- Getter methods for name, artist, genre, and file
- toString method that returns in the format:
  "<name> by <artist>
  Genre: <genre>
  File: <file name>"

NotAnArtistException.java

An unchecked exception that has a no-args constructor and uses the super class's constructor to have the message "That is not an artist we know!". This exception occurs when someone tries to list an artist's songs, but the artist is not in the database.

SongNotFoundException.java

A checked exception that has a 1-arg constructor that takes in a String message and prints out the message when the exception is printed (Hint: use the super constructor that takes in a message). This exception occurs when someone tries to read a song's lyrics, but the song is not found in the database.

SongDB.java

This class will hold a various number of static methods that will help you work with your database. You may create your own helpful methods to keep your code clean and succinct! Remember to use encapsulation – will any of these helper methods be called outside of the class?

There is no constructor or data fields, though you are free to create them if needed.

Static Method `load`:

- Takes in a `File` object as the parameter (this will be the outermost directory)
- The method should throw a `FileNotFoundException` if the passed-in file is null, or the file does not correspond to a valid file in the directory.
- Implement an iterative method that will iterate through the nested folders and create a `Song` object for every song text file and initialize their instance variables accordingly
    - The song name is the first line of the file. **Note: the song name is not necessarily the name of the text file.**
    - The artist is the name of the folder the song is in
    - The genre is the name of the folder the artist folder is in
    - **Hint**: refer to the File API to find useful methods! listFiles() will be important here.
- Add the song object to the ArrayList
- Returns an ArrayList of `Song` objects

Static Method `outputData`:

- Takes in the ArrayList of `Song` objects
- This method should create and print to a file called `SongDB.txt` using `PrintWriter`
- Assume that there is no existing `SongDB.txt` file
- Iterates through the songs in the ArrayList printing the songs' data: name, artist, genre.
- See example output

Static Method `readSong`:

- Takes in a Song object and ArrayList of `Song` objects
- If the song object isn't in the ArrayList, throw a `SongNotFoundException` with the message, "Song (song.name) was not found in the database!"
- Reads the object's song file line-by-line, analyzing the lyrics.
    - You should find the total word-count and the longest line according to string length
    - Print out the lyrics first 5 lines, then print out the word-count, then print out the longest line (by string length)
    - See example output
- The method should throw a `FileNotFoundException` if the file name is null, or the file does not correspond to a valid file in the directory.

Static Method `listArtistSongs`:

- Takes in a string that represents the artist's name and the ArrayList of `Song` objects that represents the database
- This method should iterate through the songs in the ArrayList and if the artist's name matches to a song(s), print the song name(s) on a new line.
- If the artist does not exist / have any songs in the ArrayList, throw a `NotAnArtistException`

Main Method:

- Create a File object for the database_songs directory. This directory should be in the same directory that `SongDB.java` is in.
- Test all the static methods with the provided database
- Be sure to catch a `NotAnArtistException`, `SongNotFoundException`, or `FileNotFoundException` from any methods you call in the main method and print out its message.

Example Output

Static Method `outputData` with only two songs in ArrayList:
SongDB.txt

```
So Not Over You
Simply Red
Pop

Hollywood Nights
Bob Seger & The Silver Bullet Band
Rock
```

Static Method `readSong`: (for Kenny Loggins' Celebrate me Home)

```
Home for the holidays,
I believe I've missed each and every face,
Come on and play my music,
Let's turn on the love light in the place
It's time I found myself,

Word Count: 270
Longest Line: Traveling where the Westerly winds can fly,
```

Hints

The assignment may seem more complex than it actually is. Again, we suggest you begin incrementally and create the Song class. In the SongDB class, there are just four methods to be implemented and none of them is very long. Make sure you have a good handle on what needs to be done in each method before proceeding with it.

Submitting

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- `Song.java`
- `NotAnArtistException.java`
- `SongNotFoundException.java`
- `SongDB.java`

Make sure you see the message stating, "Homework 6 submitted successfully." We will only grade your last submission be sure to **submit *every file* each time you resubmit.**

Import Restrictions

To prevent trivialization of the assignment, you may only import:

- `java.util.Scanner`
- `java.util.ArrayList`
- `java.io.File`
- `java.io.PrintWriter`
- `java.io.IOException`

- `java.io.FileNotFoundException`

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- `var` (the reserved keyword)
- `System.exit`

Checkstyle and Javadoc

You must run Checkstyle on your submission. (To learn more about Checkstyle, examine our course's external website Java Resources page under "CS 1331 Style Guide".) The Checkstyle cap for this assignment is **20 points**. If you don't have Checkstyle yet, download it from Canvas → files → checkstyle-8.28.jar. Place it in the same folder as the files you want to run Checkstyle on. Run checkstyle on your code like so:

```
$java -jar checkstyle-8.28.jar yourFileName.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up above this message, and the number at the end would be the points we would take off (limited by the Checkstyle cap mentioned above). The Java source files we provide contain no Checkstyle errors. In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early! Additionally, you must Javadoc your code.

Run the following to only check your Javadocs:

```
$ java -jar checkstyle-8.28.jar -j yourFileName.java
```

Run the following to check both Javadocs and Checkstyle:

```
$ java -jar checkstyle-8.28.jar -a yourFileName.java
```

Collaboration

No collaboration is allowed on this assignment. See syllabus for more details.

In addition, note that it is not allowed to upload your code to any sort of public repository. This could be considered an Honor Code violation, even if it is after the homework is due. Only post code on Piazza in a **private** post.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Take a look at the File API if you are stuck
- Do not submit `.class` files.
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- Check on Piazza for all official clarifications
- Make sure to test your program manually based on the assignment instructions and expected outputs. The Gradescope autograder visible to you is **NOT** comprehensive.