COP3530 Data Structures
Project#1 (Due on June 2nd by 11:59 pm)
25 points
Summer 2019

Project Description:

In this project, you will be creating a Linked List based song player, see Figure 1. You will be provided with a fully functional Mp3 player that uses arraylists as its underlying data structure. Your task is to completely replace the arraylist and its methods with the Linked List equivalents. The Mp3Player.java file contains all code related to the GUI and playing of Mp3 files, as well as a main class that will test the data structure. You do not modify this file. In fact, you do not even need to understand all of the details of the Mp3Player.java file. However, you will need to completely rewrite the List.java file so that it uses Linked Lists. For your reference, I am using the javazoom mp3 decoder which is imported in the following way,

```
import javazoom.jl.player.*; //  This is the MP3 decoder and player
import java.io.FileInputStream; // This allows us to read files from disk
```

To compile (and run) the jar files associated with the MP3 player, you can modify the classpath, i.e.

```
javac -cp jl1.0.1.jar Mp3Player.java
java -cp ./jl1.0.1.jar:. Mp3Player
```

You can safely ignore the compiler warning that states Mp3Player.java uses or overrides a deprecated API.

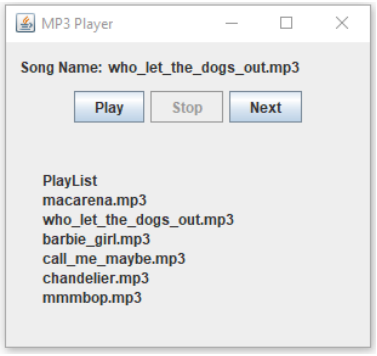


Figure 1: Visualization of the Mp3 player and correct song ordering.

For the List, i.e., List.java, you will be creating a Singly Linked List with the following methods.

Method name	Description
void insertItem (String name)	This method will insert a String item to the end of the linked list.
void insertItem (String name, int pos)	This method will insert a String item to the given position in the linked list. You can assume that a position of 0 means the front of the list. (if position is out of bounds, do not insert, but output out of bounds)
boolean removeItem(String aname)	This method will find the String with the same value as aname and remove the first occurrence of that string from the list. The return of this method is either true (success) or false if the string does not exist in the list. Use the .equals method when comparing strings.
void removeItem(int position)	Remove the node at position (if position is out of bounds, do not remove, but output out of bounds)

COP3530 Data Structures
Project#1 (Due on June 2nd by 11:59 pm)
25 points
Summer 2019

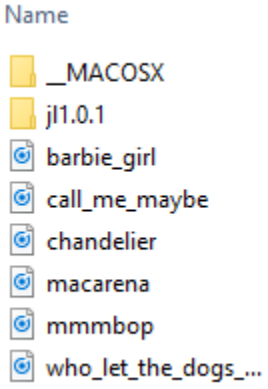
boolean contains(String name)	Return true if the linked list contains the string
void clear()	Remove all of the elements from the linked list
String get(int index)	Return the element at the specified position in the list.
int size()	Returns the number of elements in this list.
String toString()	Override the toString method to print out the names of all the songs in the list
String toHTMLString()	Return a String that starts with < html > ends with < /html > and uses the line breaks, < br/ >, in replacement of new line characters and prints out all the names of all the songs in the list. See the current implementation for details.

Next, create a ListNode class within the List.java file with fields head and next. You will additionally be creating three constructors listed below.

1. public ListNode() – constructor with no parameters, songName should be an empty string
2. public ListNode(String name) – constructor with 1 parameter, set the name of the song file
3. public ListNode(String name, ListNode next) – constructor with 2 parameters, name of the song and the next node in the list

Source Files Provided (on Canvas):

1. Mp3Player.java
2. List.java
3. Other project-related files:



Deliverables:

Make sure your project folder (i.e., Lastname_P1.zip) contains:

1. Source code (i.e., .java files NOT .class files)
2. A readme file (i.e., readme.txt) explaining how to compile and run your project
3. Screenshots of your output.
4. Any input files related to this project

Finally, compress your project folder and submit it on Canvas.