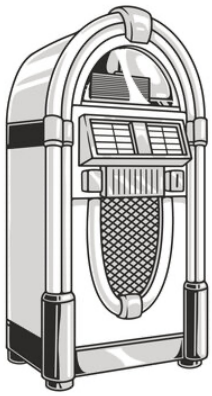
P08 JukeBox

Overview

In this programming assignment, we will be covering stacks and queues. You must be familiar with music streaming services that use a queue to hold songs that you can play. You are going to implement a similar queue in JukeBox! You will use stacks for the album tracklist. You are also going to be implementing a shuffle method, just like Spotify!



Grading Rubric

5 points	**Pre-assignment Quiz**: accessible through Canvas until 11:59PM on **11/17**.
+5%	**Bonus Points**: students whose ***final*** submission to Gradescope is before **5:00 PM Central Time** on **WED 11/20** *and* who pass ALL immediate tests will receive an additional 2.5 points toward this assignment, **up to a maximum total of 50 points**.
15 points	**Immediate Automated Tests**: accessible by submission to Gradescope. You will receive feedback from these tests *before* the submission deadline and may make changes to your code in order to pass these tests. Passing all immediate automated tests does **not** guarantee full credit for the assignment.
20 points	**Additional Automated Tests**: these will also run on submission to Gradescope, but you will not receive feedback from these tests until after the submission deadline.
10 points	**Manual Grading Feedback**: TAs or graders will manually review your code, focusing on algorithms, use of programming constructs, and style/readability.
50 points	**MAXIMUM TOTAL SCORE**

Learning Objectives

After completing this assignment, you should be able to:

- **Describe** the functionality of a Stack and Queue data structure and explain its operational details when implemented using a linked list.
- **Predict** edge case situations for stack and queue usage and verify that your implementation handles them correctly.
- **Implement** a simple Jukebox to play music using your stack and queue data structures.
- **Implement** meaningful unit tests to verify the correctness of a data structure

Additional Assignment Requirements and Notes

Keep in mind:

- Pair programming is **NOT ALLOWED** for this assignment. You must complete and submit P08 individually.
- The ONLY external libraries you may use in your program are:
  `java.util.ArrayList` (in all files),
  `java.util.Collections` (JukeBox.java),
  `java.util.NoSuchElementException` (Album.java, JukeBox.java, JukeBoxTester.java)
- Use of *any* other packages (outside of java.lang) is NOT permitted.
- You are allowed to define any **local** variables you may need to implement the methods in this specification (inside methods). You are NOT allowed to define any additional instance or static variables or constants beyond those specified in the write-up.
- You are allowed to define additional **private** helper methods.
- Only `JukeBoxTester.java` may contain a main method.
- All classes and methods must have their own Javadoc-style method header comments in accordance with the CS 300 Course Style Guide.
- Any source code provided in this specification may be included verbatim in your program without attribution.
- All other sources must be cited explicitly in your program comments, in accordance with the Appropriate Academic Conduct guidelines.
- Any use of ChatGPT or other large language models ***must be cited*** AND your submission MUST include **screenshots** of your interactions with the tool ***clearly showing all prompts and responses in full***. Failure to cite or include your logs is considered academic misconduct and will be handled accordingly.

- **Run your program locally before you submit to Gradescope**. If it doesn't work on your computer, *it will not work on Gradescope*.

Need More Help?

Check out the resources available to CS 300 students here:
https://canvas.wisc.edu/courses/427315/pages/resources

CS 300 Assignment Requirements

You are responsible for following the requirements listed on both of these pages on all CS 300 assignments, whether you've read them recently or not. Take a moment to review them if it's been a while:

- Appropriate Academic Conduct, which addresses such questions as:
  - How much can you talk to your classmates?
  - How much can you look up on the internet?
  - How do I cite my sources?
  - and more!

- Course Style Guide, which addresses such questions as:
  - What should my source code look like?
  - How much should I comment?
  - and more!

Getting Started

1. Create a new project in Eclipse, called something like **P08 Jukebox**.
   a. Ensure this project uses Java 17. Select "JavaSE-17" under "Use an execution environment JRE" in the New Java Project dialog box.
   b. Do **not** create a project-specific package; use the default package.

2. Download five (5) Java source file(s) from the assignment page on Canvas:
   a. **StackADT.java**
   b. **QueueADT.java**
   c. **LinkedNode.java**
   d. **Song.java**
   e. **JukeBoxTester.java** (includes a main method)

3. Create four (4) Java source file(s) within that project's src folder:
   a. **LinkedStack.java** (implements StackADT)
   b. **LinkedQueue.java** (implements QueueADT)
   c. **Album.java**
   d. **JukeBox.java**

1. Stack and Queue Interfaces

These files define a simple interface for a stack and a queue implemented using a linked list, outlining operations such as adding an item, removing an item, checking the top/front item, verifying if the data structure is empty, and seeing if it contains a specific item.

2. Testing Your Code

The provided **JukeBoxTester** skeleton file contains stub tester methods for all the various classes you implement in this assignment.

- Test as you go, and you can add private tester methods to your tester class. If your initial classes are not implemented correctly, the later ones will not work correctly either.
- Recommend drawing diagrams to understand the workings of the data structures so you can figure out their behavior before you start implementing them.
- If you haven't already, consider spending some time learning how to use your IDE's debugger tool. It is very helpful for following the execution of your code, especially as we get into more complex programs. There is a tutorial for Eclipse's debugger linked on our resources page.

3. LinkedQueue and LinkedStack

You will need to implement both a stack and a queue using a linked list data structure constructed from LinkedNodes. Follow the details in the interfaces to get a better idea of how to approach doing this. It would be a good idea to write the tester methods for these data structures and ensure they are working as expected. Further classes will be built using them, getting the functionalities correct here will make debugging easier later.

Both data structures make use of the **LinkedNode** class which provides a basic structure for each element in the linked list, including its data and link to the next element.

NOTE: **LinkedStack** and **LinkedQueue** contain methods NOT defined in StackADT and QueueADT. ***Don't forget to implement them!***

4. Song and Album

The Song class holds metadata for the songs, this class has been provided to you. The **Album** class allows you to add songs to an album, remove them, or preview the first song of the album. The Album class utilizes the stack class you implemented in the previous section to hold songs. Once done, complete the corresponding tester methods to ensure they are working properly.

5. Jukebox

Once you have the Album class ready, move on to the **JukeBox** class that puts everything together. It contains a queue of songs. Adding an Album empties the stack into the queue. You can add individual songs to the queue as well. You can call shuffle on the queue, and play the songs.

Remember to protect against exceptions! No exceptions except those EXPLICITLY stated in the Javadocs should be thrown at any time.

6. Generating Your Own Javadocs (Optional)

You can generate your own Javadocs! Generating Javadocs is an important part of learning how to document your code, and can be a useful skill to know when creating your projects. If you wish, you can use the `javadoc` command to generate HTML files for your commented code. Simply run the command from a terminal and open the generated HTML file in a browser to view your Javadocs.

Assignment Submission

Hooray, you've finished this CS 300 programming assignment!

Once you're satisfied with your work, both in terms of adherence to this specification and the academic conduct and style guide requirements, make a final submission of your source code to Gradescope.

For full credit, please submit the following files (**source code**, *not* .class files):

- `LinkedStack.java`
- `LinkedQueue.java`
- `Album.java`
- `JukeBox.java`
- `JukeBoxTester.java`

Additionally, if you used generative AI at any point during your development, ***you must include screenshots*** showing your FULL interaction with the tool(s).

Your score for this assignment will be based on the submission marked "**active**" prior to the deadline. You may select which submission to mark active at any time, but by default, this will be your most recent submission.

Students whose final submission (which must pass ALL immediate tests) is made before 5PM on the Wednesday before the due date will receive an additional 5% bonus toward this assignment. Submissions made after this time are NOT eligible for this bonus, but you may continue to make submissions until 10:00PM Central Time on the due date with no penalty.

Copyright notice