Koç University COMP202 Data Structures and Algorithms

Assignment 2

Instructor: Gözde Gül Şahin

TA: Burak Can Biner| Email: bbiner21@ku.edu.tr

Due Date: April 09 2023, 23:59

Submission Through: **BOTH** Blackboard and Github Classroom

This programming assignment will test your knowledge and your implementation abilities of what you have learned in the Lists, Stacks, and Queues parts of the class. This homework must be completed individually. Discussion about algorithms and data structures are allowed but group work is not. Coming up with the same algorithm and talking about the ways of implementation leads to very similar code which is treated as plagiarism! Discuss carefully. Any academic dishonesty, which includes taking someone else's code or having another person doing the assignment for you will not be tolerated. By submitting your assignment, you agree to abide by the Koç University codes of conduct.

Description

This assignment is designed to assess your understanding of Arrays, Linked Lists, Stacks, and Queues and your implementation abilities. This assignment requires you to implement the Double Ended Queue - Deque data structure. Deque, which is pronounced as "deck", is a generalization of the queue data structure that supports inserting and removing elements from either the front or the back of the data structure. The assignment will have two phases; (1) implementation of the data structures and (2) application of the data structures. You code will be test via autograder. So it is important to have same output formatting the compare outputs.

Part I

In the first part of the assignment, you will implement the deque ADT both using Arrays and Doubly Linked Lists. You will be given a deque interface and two starter files. These files have comments for your convenience. You are also given another interface for a simple container. This interface specifies three operations; (1) push, (2) pop and (3) peek. The details are in the comments but you can figure out that these methods resemble stack and queue operations. You are asked develop a stack class and a queue class which implement this interface. However, you are required to use a deque for this; by wrapping the dequeue operations to achieve the desired input-output behavior. The type of deque will be given as a generic. See the relevant files and the main file for how they are used.

The code has comments about the assignment. Majority of what you need to do is actually in these comments, so make sure you read them carefully. The descriptions for the provided files are below. The bold ones are the ones you need to modify, and they are placed under the code folder, with the rest under given.

- iDeque.java: This is the interface that shows you the operations of the Deque ADT. You do not need to modify this file. The ArrayDeque and LLDeque classes implements this interface. Make sure you pay attention to the comments.
- **ArrayDeque.java**: You need to code the ArrayDeque class that implements the iDeque interface using a Dynamic Array in this file. Follow the instructions given in the code. This must be a circular implementation.
- **LLDeque.java:** You need to code the LLDeque class that implements the iDeque interface using a Doubly Linked Lists in this file. Follow the instructions given in the code.
- iSimpleContainer.java: This interface has three simple methods of a container that allow manipulation from a "single point". You do not need to modify this file but make sure to read its comments.
- **Stack.java:** You need to code the Stack class that implements the iSimpleContainer interface using a generic deque in this file. Look at the code to see how this is setup. This must have the last-in-first-out behavior.
- **Queue.java:** You need to code the Queue class that implements the iSimpleContainer interface using a generic deque in this file. Look at the code to see how this is setup. This must have the first-in first-out behavior.
- Main.java: Runs the autograder on your code and gives printouts to help you.
- Util.java: This file includes utility functions. Do not worry about it, you will not need to use anything from this file in this assignment.

Part II

For the application of the data structures that you implemented, you are going to design a playlist system. You are going to keep your songs in an LLDeque data structure which is named dequeu in the code. You are expected to make effective use of Stacks and Queues here in different ways. Methods and classes that you need to implement are as follows:

- Song Class: Take a look at the code of this class, you do not need to modify it. It basically creates song instances with song name, artist, and year of release values.

- loadPlaylist(String fileName): This function loads a playlist from a txt file. This file will be given to you as songs.txt inside the “given” folder. The name, artist, and year of the songs are separated with “;” in this text file.
- insertSong(String n, String a, int y): This function inserts a new Song to the end of our dequeu. Name, artist, and year are given to you as input arguments.
- deleteSong(String n): Delete a song with a given name from the playlist. You do not need to consider the case where multiple songs have the same name. Do not do anything if the song does not exist in the playlist. Return the deleted song. After deleting a Song rest of the order MUST remain the same as before.
- reverse(): Reverse the order of Songs in the playlist. For instance, if the initial order is A-B-C-D new order has to be D-C-B-A.
- shuffle(int[][] input_ints): Perform shuffle operation with the algorithm given below.

```
p <- playlist
n <- playlist len
m <- a given number of steps to shuffle

for i from 1 to m do
    x, y <- random integers such tat x<y<n
    exchange p[x] and p [y]
```

Here “input_ints“ is an double integer array. It has random integers generated with the fixed seed to perform autograding later on. From the given algorithm, they will correspond to x and y pairs. The shape of the “input_ints” is (number_of_step, 2). You will be given two integers, and as shown in the algorithm above, you are going to change the place of two songs in those indices.

For instance, assume our shuffle step is two, and input_ints input is given as 2,3 and 4,2. For a playlist with songs A, B, C, D, E first the order becomes A, C, B, D, E. Then the order becomes A, D, B, C, E.

- play(): There will be a play aspect of our playlist. When we call play method, a PlayQueue class is initialized for you.
- PlayQueue(): This is a class to put songs into a play queue. This class has Queue property which is called “playqueue”. Change only the constructor of this class.
- played_song(String n): This function is called when a song is played from the queue. If a song with the given name does not exist in the play queue do not do anything.
  If the loop flag is true, this song needs to go to the end of the queue. If there were songs before this song in the queue, they must stay in the play queue in the same order. For instance, A-B-C-D-E-F with C played will yield the output D-E-F-A-B-C. If loop flag is false, it should be removed from the queue. If there were songs before this song in the queue, they would also be removed. For instance, A-B-C-D-E-F with C played will yield the output D-E-F.

Grading

Your assignment will be graded through an autograder. Make sure to implement the code as instructed, and use the same variable and method names. We will not only check the outputs, but your whole implementation will be assessed, such as access modifiers of methods and variables, input/output parameters of methods, and class constructors. A version of the autograder is also released to you. Its files are under the autograder folder. Our version will more or less be similar, potentially including more tests. Run the main program to get the autograder output and your grade. In case the autograder fails or gives you 0 when you think you should get more credit, do not panic. Let us know.

Submission

You are going to submit a compressed archive through the blackboard site along with a github classroom submission. The file you submit to blackboard will be simply the archive of your github code saved as a zip file. The file should have a name with your student ID without the leading zeros. Important: Make sure to submit to both blackboard and github, you can be graded by either of them. The github invitation link is the following: . Follow the link and choose your ID from the list. The readme and the details in the comments will give you all the further

information you need. This assignment should work fine with almost any Java version, we have tested it with JDK 17, but even older versions should work fine.