A02 - Circular Queue

0/30 Points

9/16/2023

Attempt 1

9/17/2023
NEXT UP: Review Feedback

Attempt 1 Score:
N/A

Add Comment

Unlimited Attempts Allowed
9/2/2023 to 9/23/2023

Details

Fundamentals

Assignment: A02

Learning Objectives

- Implement stacks or queues using arrays and linked lists.
- Implement stacks or queues using generics.
- Pair program with a partner.

Overview

This assignment introduces you to circular queues and pair programming.
You will pair program with a partner as you implement a circular queue that stores elements in an array.

Previous (https://slcc.instructure.com/courses/915963/modules/items/21581547)

New Attempt

Next (https://slcc.instructure.com/courses/915963/modules/items/21581551)

Instruction

Getting Started:

- Read this **introduction to pair programming (https://slcc.instructure.com/courses/915963/pages/introduction-to-pair-programming)** .
- Connect with your assigned partner and schedule the first pair programming session at least 5 days before the due date.
- Read the article **Circular Queue Data Structure (https://www.programiz.com/dsa/circular-queue)** to learn about circular queues.
- Do NOT start coding on your own.
  All the code should be written together with your partner during the pair programming sessions.

What if I can't connect with my partner?

Please contact your instructor if you were unable to schedule a pair programming session with your partner five days before the due date.

Implement Class CircularQueue:

Implement the following class together with your partner during the pair programming sessions.
Class CircularQueue<Item> should store the items in an **array**.

```
public class CircularQueue<Item> implements Iterable<Item> {
   CircularQueueA(int capacity)
   public boolean isFull()
   public boolean isEmpty()
   public int size()
   public void enqueue(Item item)
   public Item dequeue()
   public Item peek()
   public String toString()
}
```

The **article (https://www.programiz.com/dsa/circular-queue)** above includes an array-based implementation of the circular queue. Feel free to use that as a starting point. Note though,

Previous (https://slcc.instructure.com/courses/915963/modules/items/21581547)

New Attempt

Next (https://slcc.instructure.com/courses/915963/modules/items/21581551)

this is different from the size of the array (capacity) of a queue. To avoid confusion, refer to the fixed number of elements that can be stored in the queue as capacity.

Method **toString**:
This method should return a string that lists all elements in the order in which they are removed from the queue. Each element should be followed by a single space. An empty queue should return an empty string.

Iterator:
The iterator should iterate over the queue elements in the order in which they are removed.

Boundary cases:

- If the user calls the method peek or dequeue on an empty queue, a NoSuchElementException should be thrown.
- If there is an attempt to add an element to a full queue, an UnsupportedOperationException should be thrown.
- If the argument of the constructor (capacity) is less than 1, an IllegalArgumentException should be thrown.

Doc Comments
Remember to add doc comments to the class and the methods, and include an @author tag with your names to the doc comment describing the class. For more information on doc comments, review the corresponding section in the Style Guide.

Write Test Code

Write test code to determine whether your implementation meets the requirements.
Test all methods and boundary cases. As always, include labels. Display the method you are testing and the input and output values.

JUnit Tests

This assignment includes JUnit tests to help you provide evidence that your code works as expected.
Run these JUnit tests after you used your own test code to convince yourself your code meets the specifications.
Download the file **CircularQueueTest.java (https://slcc.instructure.com/courses/915963/files**

Submission

1. **Attach the file CircularQueue.java**
2. **Create and embed a Video: (1.5 - 3 min)**
   Follow the **Guidelines for Assignment/CE Recordings (https://slcc.instructure.com/courses/915963/pages/guidelines-and-expectations)** and include the following:

- A title page and a brief introduction
- Show your implementation of class CircularQueue, your test code, and the unchanged JUnit tests.
- Run your test code
- Run the jUnit tests
- Answer each of the following Team/Collaboration questions:
  A. Did you and your partner pair program?
  B. How often did you meet to work on the assignment?
  C. Clear Communication: Did you communicate as you pair programmed?
  D. Switching Roles: Did you regularly switch roles?
  E. Did you and your partner maintain a positive and respectful attitude?
  F. What was the hardest part?
  G. What is your **pebble distribution (https://slcc.instructure.com/courses/915963/pages/pebble-distribution)** ?

ONE team member submits the video and the file CircularQueue.java.
EACH team member submits the team member(s), the pebble distribution, and a reason for the pebble distribution if it is uneven.

Previous (https://slcc.instructure.com/courses/915963/modules/items/21581547)
New Attempt
Next (https://slcc.instructure.com/courses/915963/modules/items/21581551)

2420 - A02				
Criteria	**Ratings**			**Pts**
Class CircularQueue view longer description	**20 pts** **Full Marks** All tests passed.	**19 pts** **Close** All but one test passed	**0 pts** **No Marks** No submission or the implementation differed too much from the instructions.	/ 20 pts
Test Code view longer description	**5 pts** **Full Marks** All or all but one CircularQueueB tests passed.	**4 pts** **Close**	**0 pts** **Insufficient**	/ 5 pts
Style | Best Practices | Video | Pair Programming view longer description	**5 pts** **Full Marks**	**0 pts** **No Marks**		/ 5 pts
				Total Points: 0

Previous (https://slcc.instructure.com/courses/915963/modules/items/21581547)

New Attempt

Next (https://slcc.instructure.com/courses/915963/modules/items/21581551)