Manju Muralidharan Priya

CS310 Data Structures Spring 2020

Programming Assignment 1

Author original Java code to do the following:

Implementing a Queue using two Stacks (which are in turn implemented as a Single Linked List)

A project template is already provided for you.

Class Descriptions:

Interfaces:

1. QueueSpecs<E>
   This interface describes the methods that need to be implemented in the StackQ.java container class.
2. StackSpecs<E>
   This interface describes the methods that need to be implemented in the LLStack.java container class.

Containers:

1. LLStack<E>
   This class IMPLEMENTS StackSpecs interface. All methods must be implemented using a linked list.
   Mandatory variables for this class are:
   a. private Node<E> top;
   b. private int stackSize;
   This class must have a constructor with arguments to initialize all variables.
   This class must also override the toString() method for displaying the stack.
2. Node<E>
   This class is the node for the linked list.
   Mandatory variables for this class are:
   a. private E data;
   b. private Node<E> nextNode;
   This class must have a constructor with arguments to initialize all variables.
3. StackQ<E>
   This class IMPLEMENTS the QueueSpecs interface. All methods must be implemented using two Stacks.
   Mandatory variables for this class are:
   a. private LLStack<E> enQStack
   b. private LLStack<E> deQStack
   This class must have a constructor with arguments to initialize all variables.
   This class must also override the toString() method for displaying the queue.

As you can notice, there are two Stacks (enQStack and deQStack) declared as private member variables in the StackQ class. You need to use these two Stacks for implementing the functionalities of a queue.

How to perform deQ()

- pop the topmost element from the deQStack.
- If the deQStack is empty, pop all the elements from the enQStack and push them one by one into the deQStack then pop from deQStack to perfrom deQ().

How to perform enQ()

- Add the new node to the top of the enQStack.
- Empty condition for the enQStack is only to maintain the top reference.

Data:

1. DataClass
   This is the class that must be used as the type while creating an object of the StackQ class and for demonstrating the Queue operations from the driver class.
   DO NOT USE PRIMITIVE WRAPPER CLASSES.
   This class is already provided for you.

Driver:

1. Driver
   This class must contain the main method.
   Use this class to create an object of the StackQ class to demonstrate all the functions of the Queue implemented as two stacks.
   Display a Menu (similar to Homework 1) to perform the following options:
   a. Enqueue in the queue
   b. Dequeue from the queue
   c. Peek from the Queue
   d. Display the queue
   e. Display enQStack and deQStack
   f. Display size of the queue
   g. Exit
   The output of this program follows a similar pattern to HomeWork 1.

Error Handling:

Handle the following errors:

1. If user enters an option from the menu that is not valid, display error to the User and repeat the menu.
2. Handle null responsibly
3. You may write your own exceptions if you wish to for extra credit (2 Points)

Constraints:

1. All variables must have public Getters and Setters
2. Use Scanner for user Input.
3. All Classes must contain Constructors with arguments to initialize all variables
4. Only use the toString() method for display purposes
5. All code must be generic and cannot be datatype dependent. Reusability is the goal.
6. Usage of getters and setters are important, where ever possible opt to use get and set instead of modifying the variable directly.
7. Write constructors for all the classes

Files to be submitted:

Submit a ZIP file (ZIP file must have your First and Last Name) containing the following:

1. JAR file containing the project structure and all the classes (downloaded from the template)
2. Run.bat – Executable batch file to run the program.
3. ReadMe.pdf
    a. List and briefly explain the project structure (folder structure and what each package contains)
    b. For each class in the project, list and describe the function / usage of every variable, constructor and method.
    For the StackQ class, draw (pictorial representation of stacks) and explain with an example how enQ() and deQ() functions work with two stacks.
    c. If you have written your own exceptions describe them.

COMMENTS IN THE JAVA FILES ARE NOT NECESSARY, FOCUS ON PROPER DESCRIPTIONS IN THE README INSTEAD.

Comments are useful for understanding specific parts of the code, a deep dive. ReadMe helps with understanding overall design.