Data Structures and Algorithms, Spring '23 section1 Dr. Hristescu
Lab 6

Assigned: Thursday, February 23
Prelab(P0 and P1): Due beginning of lab Thursday, February 23
Lab: 10pm on Monday February 27

Note:

- For Problems 0 and 1 keeping track of the number of items in the collection is application-specific and not part of the Queue/DEQ functionality and therefore should NOT be part of the ADT implementation, but part of the driver.
- All Drivers for Problems 2 and 3 will be used with 4 command line arguments: args[0] for List, args[1] for Stack, args[2] for Queue and args[3] for DEQ
- The provided input data for P2 and P3 is not guaranteed to be enough to thoroughly test your solution. It is your responsibility to test it thoroughly on additional input data.

Problem 0 (Prelab): Using QueueException implement the QueueInterface with the 2 approaches discussed and developed in class: a circular simply linked structure approach (QueueCSLS) and a resizable array-based approach.(QueueCRAB) that uses the array in a circular fashion. Both QueueCSLS and QueueCRAB should be generic classes. The QueueCSLS implementation was discussed in class only has a **back** reference (make sure that **CNode** is a generic class). For QueueCRAB use **these 4 data fields** [use these exact names] **items**, **front**, **back, numItems**. **front** holds the index of the first (existing) item in the queue collection; **back** holds the index of the first available array entry – to be used by enqueue. Therefore both **front** and **back** should both be initialized to 0. Note that there is NO size method in the QueueInterface and there should NOT be any size method implemented. Use **protected *data fields*** and ***resize*** method.
As discussed in class, for the **resize** and **toString** methods you CANNOT use the same code that you had for List or Stack; they need to be redesigned to copy/collect from the first item in the queue to the last item.
Design **your own test data** and test your solution thoroughly using a menu driven test driver called **DriverL6P0.java** that takes uses only the generic QueueInterface and takes the implementation name as a command line argument (args[0]) with the following options:

0. Exit Program
1. Insert item at back of FIFO collection
2. Remove item from front of FIFO collection
3. Display front item of FIFO collection
4. Clear FIFO collection
5. Display number of items and content of FIFO collection

Problem1(Prelab): Design a **subclass** of QueueCRAB called **DEQ** (Doubly-Ended Queue) that supports operations at both ends of the collection. Design DEQ as a **generic class**. Therefore, declare: **public class DEQ<T> extends QueueCRAB<T> implements ExtendedQueueInterface<T>**
Use EQE. As emphasized in class, I strongly suggest solving decrementing the indices in a formulaic manner rather than using conditionals. You also need to have the QueueCRAB **data fields** as well as internal **resize** method *protected* to ensure direct access and avoid code duplication in DEQ.
Design **your own test data** and thoroughly test your solution thoroughly using a menu driven driver called **DriverL6P1.java java** that takes uses only the generic ExtendedQueueInterface and takes the implementation name as a command line argument (args[0]) with the following options:

0. Exit Program
1. Insert item at back
2. Insert item at front
3. Remove item from front
4. Remove item from back
5. Display front item
6. Display last item
7. Clear collection
8. Display number of items and content of collection

For the next 2 problems you will have to select your favorite implementation for each of the following **generic** interfaces: ListInterface, StackInterface, QueueInterface and **ExtendedQueueInterface** and name your implementations **List, Stack, Queue, DEQ** (make sure they compile).
For Problems 2 and 3, the drivers will be named **DriverL6P2** and **DriverL6P3**, will use only generic interfaces and 4 command line arguments for the 4 implementations. The drivers will be run using the following command, ex for Problem 2:

```
java DriverL6P2 List Stack Queue DEQ <input_file >output_file
```

You can **specialize** the Bags to be collection type-specific, but keep them item type-generic**.** Call them **BagFIFO, BagLIFO and BagDEQ.** Use the appropriate 2 for P2 and P3.
The Bags' toString should display the bag-specific info and then delegate the display of the content to the collection's toString.

Problem2:
Consider Lab 5's Problem 2, but this time, the delivery person has a delivery bag large enough to process the orders received in fair first-in-first-out (FIFO) order. The sample bag is still organized in last-in-first-out (LIFO) order. Use a specialized bag for each collection: BagLIFO and BagFIFO.

Testing your solution on this data should look like this.
Note:

- **DO NOT enhance the functionality of the stack or the queue.** Everything that is specific to the application should be handled in the driver.
- Since the Stack and Queue are generic classes, make sure you **use the appropriate data type** for each collection that you are using. Do not use more memory than necessary (i.e. don't use types whose data fields are not used).

Problem3: Consider Problem 2 from this lab with an additional menu option 7 that allows senders to request express delivery. The last picked up express order will have priority to be delivered over all other already picked up orders (including other picked up express orders) when option 2 is chosen. All regular(non-express) packages will be delivered in FIFO order. The sample bag is still organized in LIFO order. Choose the appropriate ADT for the delivery bag and test is thoroughly. Use a specialized bag for each collection: BagLIFO and BagDEQ. Testing your solution on this data should look like this.

0. Exit Program
1. Pick up a regular order
2. Drop off an order
3. Display number of packages, the weight and content of the delivery bag
4. Display number of items, the weight and content of the bag of samples
5. Enjoy a sample from the bag of samples
6. Enjoy all the samples in the samples bag
7. **Pick up an express order**

Problem4: **Lab6Conclusions**: Summarize in a few sentences what you have learned from working on this lab.

Submit: Follow the instructions to turn in the lab electronically using **Lab6** for the name of your directory.
All the code developed for this lab and all sample runs **including the ones that you designed** (for P0,P1) as well as on the provided input data (P2,P3) and your conclusions.