COP3530 – Assignment 2

Objective

Students will be able to create skills in the use of linked lists, the stack, and the queue abstract data types, by implementing solutions to fundamental data structures and associated problems.

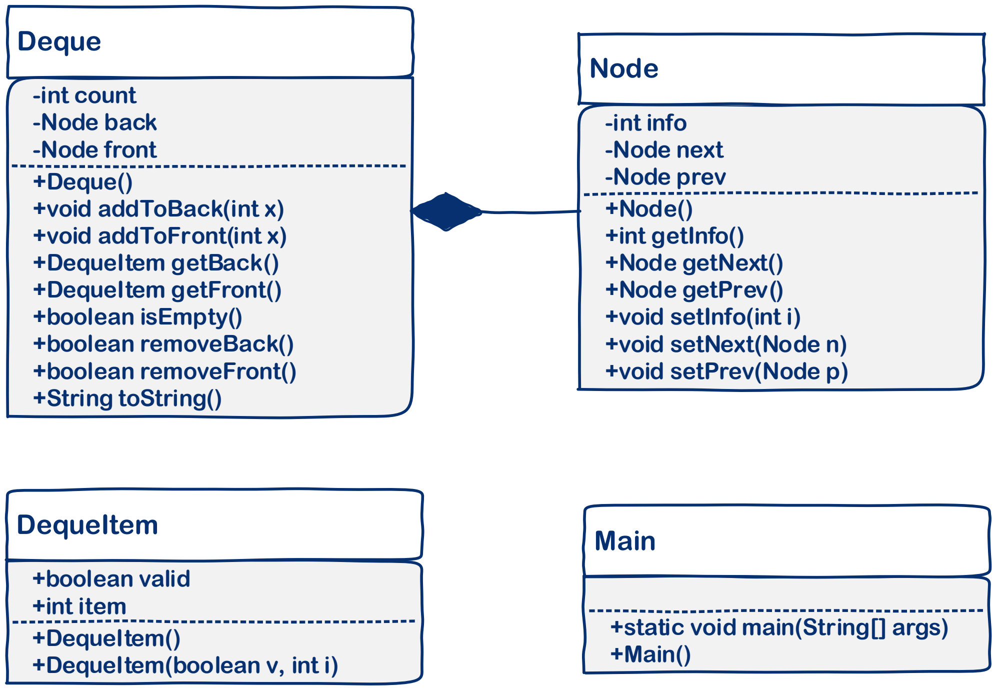
Assignment Questions

1. A double-ended queue, or *deque*, is a data structure consisting of a list of items on which the following operations are defined:

addToBack(x): insert item x on the back end of the queue
addToFront(x): insert item x on the front end of the queue
getBack(): returns the element on the back end of the queue
getFront(): returns the element on the front end of the queue
removeBack(): remove the back item from the queue
removeFront(): remove the front item from the queue

Write routines to support the deque that take O(1) time per operation. Use a doubly linked list implementation.

UML class diagram:



Guidelines

- The assignment is to be completed individually or in teams of two students. Questions are based on the content studied in class on linked lists, stacks, queues, and their variations.

- You are allowed to use all of the code given in the lectures. In those cases, make sure you properly credit its source.

- Classes from the Java Collection Framework, e.g. java.util.LinkedList, are not allowed for use in your implementation.

- Students are required to structure the code as indicated in the UML class diagrams.

- Several files accompany the exercise:
    - *Deque.java*: the framework of the Deque class; note that one or more methods are already implemented,
    - *DequeItem.java*: implementation of the DequeItem class
    - *Main.java*: a complete implementation of the Main class (tests your code using the given test file),
    - *Node.java*: the framework of the Node class.
    - *assignment 2 test set.txt*: test file,
    - *assignment 2 output.pdf*: a file with the output generated by the Main class provided.

- Students are required to use the given class frameworks included, completing all the methods as indicated in the Javadoc comment section of each method.

- The toString method provided is to be used in all output.

- The output of your program is expected to be the same as the output example provided.

Deliverables:

• A compressed folder, *PID Assignment 2 (e.g. 1234567 Assignment 2)*, containing the files:

- files *Deque.java*,
- *Node.java*,
- A screenshot of the running program. Note: a screenshot is an image that shows 1) the IDE environment with code and 2) the output window. A partial view of the IDE with code is fine, the output window is to be displayed in its entirety. More than one image file might be required to capture everything.

• Include **only** the .java files mentioned above; do not include other files or folders generated by the IDE.

• Make sure you write your Panther ID(s), class section(s), and your full name(s) in the first lines of the source code, given as comments.

Grading Rubric

The assignment is worth 100 points (out of 1000 total course points). Grade components:

Component	Points	Description
Submission	3	The student has submitted the project solution using the requirements for deliverables specified in the *Deliverables* section.
Organization	3	Code is expected to be neat, organized, and readable.
Content	94	

Element	Points
Deque.java	76 pts
Node.java	14 pts
screenshots	4