Week9Lab – 10 pts

Pre-lab questions

1. Show the contents of a queue after the following operations are performed. Assume the queue is initially empty:  
   enqueue(5);  
   enqueue(10);  
   enqueue(15);  
   dequeue();  
   dequeue();  
   enqueue(20);  
   enqueue(25);  
   dequeue();  
   enqueue(30);  
   enqueue(35);  
   enqueue(50);  
   dequeue();

Inserting 5

Inserting 10

Inserting 15

Removing 5

Removing 10

Inserting 20

Inserting 25

Removing 15

Inserting 30

Inserting 35

Inserting 50

Removing 20

The front element is 25

Removing 25

The front element is 30

The queue size is 3

The queue is not empty

[25, 30, 35, 50]

1. Show the contents of a stack after the following operations are performed. Assume the stack is initially empty:  
   push(5);  
   push(10);  
   push(15);  
   pop();  
   pop();  
   push(20);  
   push(25);  
   pop();  
   push(30);  
   push(35);  
   push(50);  
   pop();

Inserting 5  
Inserting 10

Inserting 15

Removing 15

Removing 10

Inserting 20

Inserting 25

Removing 25

Inserting 30

Inserting 35

Inserting 50

Removing 50

[5, 20, 30, 35]

Choose one of the following to develop into a program that will a linked list. Once chosen, do the following:

Understand the problem (restate in your own words, make any assumptions clear):  
I choose the problem of student in doubly linked list where it add the student first and last name. The linked list should print the sorted list by last name. then I have to print the front and back student name of current student name. I use LinkedList to perform the operations like sort and display the data.

UML diagrams of the classes you will code, including the one with main:  
Graphical user interface, application

Description automatically generated  
Pseudocode of any non-trivial methods in each class (no pseudocode needed for basic setters and getters or no args constructors):  
In class node I Defined string data, node prev and next.

After that node head and tail set to null. Then add node method takes string data. In this method I create new node that takes data. Perform the doubly linked list like check if head is null set head to newnode and tail to newnode. Head.prev and tail.next to null. Else tail.next set to newnode newnode.prev to tail tail to newnode and tail.next to null.

For printing data node current set to head if head is null list is empty while current is not empty it will print current.data and set the current to current.next for printing next data.

Same for frontbackdata print method with some if else loop to check if null or not.

Sort is simple string temp for swapping the data. It will compare current.data to index.data if it is greater than 0 it will swap.

Name of files (.java) submitted:

CreateDoublyLinkedList.java

Rummy Hand

Create a basic playing card class that represents a typical poker deck card with suit and rank (no Jokers). Populate an array or arrayList with the 52 cards (hint, use a nested loop). Simulate a hand by dealing 5 cards randomly from the deck into a hand. Make the hand a linked list. Provide a method to sort the linked list by rank then suit (A of Spades, 3 of Diamonds, 3 of Spades, 8 of Hearts, King of Diamonds) and another method that sorts the list by suit then rank (3 of Diamonds, King of Diamonds, 8 of Hearts, A of Spades, 3 of Spades). Provide a method that will find if 3 or more cards in the sorted list are sequential (like 2, 3 and 4 of the same suit) or matching (3 of the same rank like three ten’s) and pull them out to make a separate list.

**Kindergarteners at Fire Drill**

**One way to keep track of lots of kids in a file drill or other emergency is to have each child know who is before them and behind them in a line. Simulate this behavior with a Student class that includes last name and first name. Create a doubly linked list of students so each student “knows” who is before them and behind them. Sort the list based on last name alphabetical order then have each student object write out the name of the person in front of them and the name of the person behind them.**

Conga Line

If you’ve ever been to an American wedding reception or prom, you probably saw or participated in a Conga Line. This is a line dance where one person is at the head and as they dance around the room, other dancers join up by holding on to the hips of the person in front of them. This is definitely a linked list! Simulate a Conga Line with Dancer objects in a linked list with a list header that knows the first and last dancers in the line. Add dancers to the line, with each being added to the end of the line. Make sure Dancer has a toString() method so you can print the line and see that the order is correct.