CMPT 435
Assignment 1
Sep 23, 2022

Name: *Ahmed Handulle*

This program is divided into 4 Parts:

Part 1: Inside the Main.java file, I have created a Node class to build a Linked List.

Part 2: Inside the StackLink.java file, I have built a Stack Data Structure using the Linked list I have created in Part 1.

Part 3: Inside the QueueLinked.java file, I have created a Queue Data Structure using the linkedlist from Part 1.

Part 4: Inside the Palindrome.java file, I am opening a file "magicitems.txt", reading it line by line and checking where the text in that line is palindrome or not.

Part 1: Linked List Data Structure

Summary: The Linked List data structure is an abstract data type which holds a collection of elements inside of Nodes. The Nodes will then be accessed in a sequential way.

The Linked List DS has Head and Tail as a property, which keeps truck of first Node and the last Node of the list; the Nodes are all connected to one another in one direction

The Linked List DS has other properties like insertion, deletion, search and many more

1. Line 1 creates the main class of the program and that is where we have to define all of our program
2. Lines 6-16, I have created a Node class and stored the properties of a Node in there. A node has a data and a reference to the next node
3. Lines 20 - 33 creates a Linked List class and defines the properties of a Linked List Data Structure; such properties are Head and Tail references
4. Lines 51-61: Inside the Linked List class, I have defined a few functions, like insertion which inserts elements into linked list from the tail
5. Other functions like len() and isEmpty() are also defined
6. Lines 31-33 define the len() method to return the length of the list
7. Lines 39-45 define the isEmpty() method to return the True or False depending of the list if it is empty or not
8. Lines 67-77 define the a display() method which traverses the linkedlist from head to tail and display the data of each node to the screen

---

Part 2: Stack Data Structure

Summary: Stack is a Abstract linear data structure which follows a particular order in which the operations are performed. Stack DS also uses a technique called LIFO OR FILO which basically means the the last element that was inserted into the stack will be the first element to pop out of the stack. To implement the properties of the Stack, we will be using our linked list class and create some functions like push(), pop() and top().

1. Inside the StackLink.java file, Line 3 we have created the class foundation for our program
2. Lines 19-29, we reused our Node class so that we can create linked nodes to store our data
3. Lines 39-41, creates a len() method to compute the size of the stack
4. Lines 44-50, create a isEmpty() method to return a boolean value depending on the state of the function (checks if stack is empty or not)
5. Lines 55-65, creates a push() method to insert elements into the stack. For this method, there are two ways we can insert elements into the stack using linked list. One ways is inserting it at the head of the linked list or at the tail of linked list. Inserting elements into the linked list at the head is efficient since it takes O(1) time in the worst case, while inserting it at the tail takes O(n) in the worst case since we need to traverse the linked list until the last element.
6. Lines 71-83, creates a pop() method which is used to retrieve elements from the stack. pop() also also deletes and the return the element that was just deleted. For the pop() method, we can only retrieve elements from the head since our last element is at the head according to our push() method. Fortunately, retrieving elements at the head also takes a constant time complexity O(1) in the worst case.
7. Lines 90-99, creates the top() methods which operates the same as the pop() method without deleting the element from the stack, which also takes constant time operations.
8. Lines 104-114, creates the display() method which outputs all the elements in the stack.

Part 3: Queue Data Structure

Summary: Queue is an Abstract linear data structure that is open at both ends and the operations are performed in First In First Out (FIFO) order. In other words, the elements are interred in one end of the queue and removed from the other end. To implement Queue DS, we need functions like Enqueue(), Dequeue(), to insert and remove elements from the queue respectively. Queue will also have another function first() which we will return the first or the top element in the queue.

1. Inside the QueueLinked.java file, Line 1 we have created our class foundation to run our program

2. Lines 14-24, we reused our Node class so that we can create linked nodes to store our data
3. Lines 28-30 stores the instance variables of the queue data structure which are a reference nodes to the front and back of the queue.
4. Lines 33-35, creates a len() method to compute the size of the queue
5. Lines 38-44, create a isEmpty() method to return a boolean value depending on the state of the function (checks if queue is empty or not)
6. Lines 48-60 creates the enqueue() method which inserts elements at back of the queue. Inserting elements at the back of the queue takes O(1) time complexity in worst case since we keeping truck of the reference of last element of our queue; the back reference.
7. Lines 64-77 creates the dequeue() methods which removes element from the front of the queue and then returns that value. This method takes O(1) time complexity in worst case since we are removing the element from the front of the queue, which we don't have to visit any other node but the front.
8. Lines 82-87 creates the first(), which only returns the data in the first element of the queue. It also takes O(1) time complexity in worst case.
9. Lines 91-101 creates the display() method, which is used to traverse the elements of the queue and then return the values.

Part 3: Palindrome

Summary: Palindrome is a word or phrase that reads the same backward as forward. For this task, we will be importing some java libraries to use the file utility package, Scanner class as well as ArrayList class to make this program work.

1. Lines 1-3 imports the java packages we need to make this program function
2. line 5 sets the foundation for our program and where we have defined our Palindrome class
3. Lines 9-23 creates a checkPalindrome() method. This method has one parameter and it takes a string as an argument. This method also has two pointers, one starting from the indices of the first character of the string and the other one from the last character.
4. Then this function gets the first and the last character of the string and compares it, if the character are the same, the indices move to the second and the last second character and does the process again until it finds two different characters. At this point the loop breaks and returns false as the string is not Palindrome.
5. Lines 25-65 creates the main method of the Palindrome class.
6. Line 30 uses an object of the File class and takes the file name as an argument.

7. Line 34 uses an object of the Scanner class and takes the file object as an argument
8. Line 37 also uses an object of the ArrayList class and it will be used to store the lines of strings from the our file ”magicitems.txt”
9. Lines 43-46 uses a while loop and traverses all the lines in the file line by line. First it changes it to lower case as well as removing all the white spaces. Then, it finally adds the string of line into the ArrayList
10. Lines 49-54 also uses a for loop to traverse the strings in the ArrayList and checks if a string a palindrome or not.
11. Line 55 prints out all the strings that are Palindrome