**TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES**

**QUEZON CITY**

**COLLEGE OF INFORMATION TECHNOLOGY EDUCATION (CITE)**

**CS 201 - Data Structures and Algorithms**

|  |  |
| --- | --- |
| **Name: Aristotle Buenaventura** | **Date: September 27, 2021** |
| **Program/Section: IT21S1** | **Instructor: Ms. Rosmina Joy M. Cabauatan** |
| **Assessment Task: Midterm Examination Part 2** | |

**Code**

**MidtermExamPart2 Class**

package midtermExaminationPart2;

import javax.swing.JOptionPane;

public class MidtermExamPart2 {

public static void main(String[] args) {

int[] arrayElement = new int[10];

Queue q = new Queue();

LinkedList list = new LinkedList();

for(int i=0;i<10;i++) {

int arrElement = Integer.parseInt(JOptionPane.showInputDialog("Enter a number: "));

arrayElement[i]=arrElement;

}

System.out.print("Initial Array list: ");

for(int array:arrayElement) {

System.out.print(array+ " ");

//Each of the read elements of the array must be stored in a queue data structure

q.enqueue(array);

//Each of the read elements of the array must be stored in a LinkedList data structure.

list.addNodeAtTheEnd(array);

}

System.out.println();

System.out.println();

System.out.println("Queue");

//Manipulate Queue using 3 methods

System.out.println("Manipulate the Queue using 3 method");

q.dequeue();

System.out.println("Apply dequeue in the list");

q.dequeue();

System.out.println("Apply dequeue in the list");

q.enqueue(11);

System.out.println("Apply enqueue(11) in the list");

//Display the queue after applying the 3 methods.

System.out.println("Is the queue empty?: "+q.isEmpty());

System.out.println("The front key is: " + q.front());

System.out.println("The rear key is: " + q.back());

q.display();

System.out.println();

System.out.println();

System.out.println("LinkedList");

//Manipulate the linked list using 3 methods.

System.out.println("Manipulate the linked list using 3 methods.");

System.out.println("Add 0 at the beginning of the list");

list.addNodeAtTheBeginning(0);

System.out.println("Add 11 at the end of the list");

list.addNodeAtTheEnd(11);

System.out.println("Add 99 at position 2");

list.add(2, 99);

//Display the LinkedList after applying the 3 methods.

System.out.print("Linkedlist: ");

list.print();

}

}

class Queue {

QNode front, rear;

public Queue()

{

this.front = this.rear = null;

}

// Method to add an key to the queue.

void enqueue(int key)

{

// Create a new LL node

QNode temp = new QNode(key);

// If queue is empty, then new node is front and rear both

if (this.rear == null) {

this.front = this.rear = temp;

return;

}

// Add the new node at the end of queue and change rear

this.rear.next = temp;

this.rear = temp;

}

// Method to remove an key from queue.

void dequeue()

{

// If queue is empty, return NULL.

if (this.front == null)

return;

// Store previous front and move front one node ahead

QNode temp = this.front;

this.front = this.front.next;

// If front becomes NULL, then change rear also as NULL

if (this.front == null)

this.rear = null;

}

int front()

{

return this.front.key;

}

int back()

{

return this.rear.key;

}

boolean isEmpty()

{

return this.front == null;

}

void display()

{

System.out.print("Queue List: ");

while(!this.isEmpty()) {

System.out.print(this.front() + " ");

this.dequeue();

}

}

}

class QNode {

int key;

QNode next;

// constructor to create a new linked list node

public QNode(int key)

{

this.key = key;

this.next = null;

}

}

class LinkedList {

// reference to head / first node of the Singly Linked List

public Node head = null;

// class Node that hold data and a reference/link

// to the next Node in the list

class Node {

private int data;

private Node next;

public Node(int key) {

this.data = key;

this.next = null;

}

}

/\*

\* Method to add a node at the beginning of the list

\*/

public void addNodeAtTheBeginning(int key) {

// Create a new node

Node newNode = new Node(key);

// Check if the list is empty

if (this.head == null) {

// Make the new node as head

this.head = newNode;

} else {

// Point the new node's next to head

newNode.next = this.head;

// Make the new node as head

this.head = newNode;

}

}

/\*

\* Method to add a node at the end of the list

\*/

public void addNodeAtTheEnd(int data) {

// Create a new node

Node newNode = new Node(data);

// Check if the list is empty

if (this.head == null) {

// Make the new node as head

this.head = newNode;

} else {

Node cur = this.head;

// traverse to the end of the list

while (cur.next != null) {

cur = cur.next;

}

cur.next = newNode;

}

}

/\*

\* Method to add a node at the specified position in the list

\*/

public void add(int position, int data) {

// Create a new node

Node newNode = new Node(data);

// Init the cur and prev nodes to the head

Node cur = this.head, prev = this.head;

if (position == 1) {

// Point the new node's next to head

newNode.next = head;

// Make the new node as head

this.head = newNode;

return;

}

// traverse to the end of the list and check positions moved

while (cur.next != null && --position > 0) {

// update the prev and cur references

prev = cur;

cur = cur.next;

}

// update prev to point to new node

prev.next = newNode;

// & new node to point to current node

newNode.next = cur;

}

public void print() {

if (this.head == null) {

System.out.println("The List is empty.");

} else {

Node cur = this.head;

while (cur != null) {

System.out.print(cur.data + " -> ");

cur = cur.next;

}

System.out.println("NULL\n");

}

}

}

**Output**

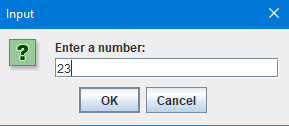


Figure 1. Enter a Number

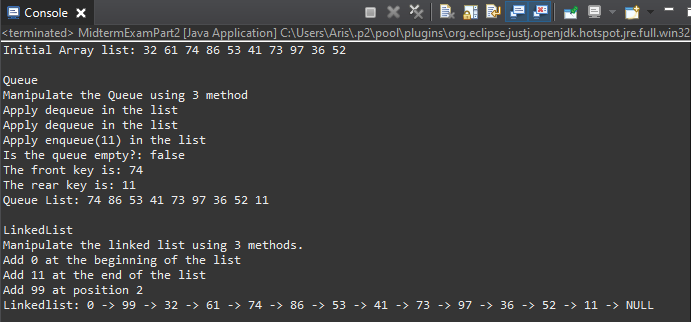


Figure 2. The Final Output