CIRCULAR LINKED LIST

import java.util.Scanner;

public class que2 {

//Represents the node of list.

public class Node{

int data;

Node next;

public Node(int data) {

this.data = data;

}

}

//Declaring head and tail pointer as null.

public Node head = null;

public Node tail = null;

//This function will add the new node at the end of the list.

public void add(int data){

//Create new node

Node newNode = new Node(data);

//Checks if the list is empty.

if(head == null) {

//If list is empty, both head and tail would point to new node.

head = newNode;

tail = newNode;

newNode.next = head;

}

else {

//tail will point to new node.

tail.next = newNode;

//New node will become new tail.

tail = newNode;

//Since, it is circular linked list tail will point to head.

tail.next = head;

}

}

//Displays all the nodes in the list

public void display() {

Node current = head;

if(head == null) {

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

}

else {

System.out.println("Nodes of the circular linked list: ");

do{

//Prints each node by incrementing pointer.

System.out.print(" "+ current.data);

current = current.next;

}while(current != head);

System.out.println();

}

}

public static void main(String[] args) {

que2 cl = new que2();

//Adds data to the list

cl.add(1);

cl.add(2);

cl.add(3);

cl.add(4);

//Displays all the nodes present in the list

cl.display();

}

}

DOUBLE LINKED LIST

import java.util.Scanner;

public class que1 {

//A node class for doubly linked list

class Node{

int item;

Node previous;

Node next;

public Node(int item) {

this.item = item;

}

}

//Initially, head and tail is set to null

Node head, tail = null;

//add a node to the list

public void addNode(int item) {

//Create a new node

Node newNode = new Node(item);

//if list is empty, head and tail points to newNode

if(head == null) {

head = tail = newNode;

//head's previous will be null

head.previous = null;

//tail's next will be null

tail.next = null;

}

else {

//add newNode to the end of list. tail->next set to newNode

tail.next = newNode;

//newNode->previous set to tail

newNode.previous = tail;

//newNode becomes new tail

tail = newNode;

//tail's next point to null

tail.next = null;

}

}

//print all the nodes of doubly linked list

public void printNodes() {

//Node current will point to head

Node current = head;

if(head == null) {

System.out.println("Doubly linked list is empty");

return;

}

System.out.println("Nodes of doubly linked list: ");

while(current != null) {

//Print each node and then go to next.

System.out.print(current.item + " ");

current = current.next;

}

}

}

class Main{

public static void main(String[] args) {

//create a DoublyLinkedList object

que1 dl\_List = new que1();

//Add nodes to the list

dl\_List.addNode(10);

dl\_List.addNode(20);

dl\_List.addNode(30);

dl\_List.addNode(40);

dl\_List.addNode(50);

//print the nodes of DoublyLinkedList

dl\_List.printNodes();

}

}