**Client:**

/\*\*

\*

\* @author marie

\*/

public class Client {

static int teamSize = 0;

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

System.out.println("Begin");

dequeTest(new ArrayDeque());

dequeTest(new ListDeque());

}

public static void dequeTest(Deque q){

System.out.println("Testing the following deque: " + q.getClass().getName());

System.out.println("Adding 4 players to the front of deque and 4 players to the end of the deque");

for(int i=0;i<4;i++){

q.addFirst(generatePlayer());

q.addLast(generatePlayer());

}

System.out.println("\nTeam Lineup: " + q);

System.out.println("The following player has been removed from the front of the queue: " + q.removeFirst());

System.out.println("The following player has been removed from the end of the queue: " + q.removeLast());

System.out.println("Adding more players");

try{

q.addLast(generatePlayer());

q.addLast(generatePlayer());

q.addFirst(generatePlayer());

q.addFirst(generatePlayer());

}

catch (IllegalArgumentException g){

System.out.println("\tThe queue is full. No more players may be added");

}

System.out.println("Size of Queue: " + q.size() + "\nchecking if queue is empty: " + q.isEmpty());

System.out.println("The First player on the team: " + q.first()+"\nThe last player on the team: " + q.last());

System.out.println("\nPlayers on the team: " + q);

System.out.println("Checking if array equals its self: " + q.equals(q));

}

//generates a player

public static Player generatePlayer(){

teamSize++;

return new Player("Name" + teamSize, "Position" + teamSize, teamSize);

}

}

**Player:**

/\*\*

\*

\* @author Marie Larson

\* @version 2/26/18

\*/

public class Player {

private String name;

private String position;

private int number;

public Player(){

}

public Player(String name, String position, int number){

this.name = name;

this.position = position;

this.number = number;

}

//return name of player

public String getName(){

return name;

}

//set name of player

public void setName(String name){

this.name = name;

}

//return position of player

public String getPosition(){

return position;

}

//set position of player

public void setPosition(String position){

this.position = position;

}

//get the player number

public int getNumber(){

return number;

}

//set player number

public void setNumber(){

this.number = number;

}

//string method

@Override

public String toString(){

return getClass().getName() + "@" + name + ":" + position + ":" + number;

}

@Override

public boolean equals(Object o){

if(!(o instanceof Player)){

return false;

}

Player p = (Player) o;

return ((p.name.equals(this.name))

&& (p.position.equals(this.position))

&& (p.number == this.number));

}

}

**Deque:**

/\*\*

\*

\* @author Marie Larson

\* @version 2/26/18

\*/

public interface Deque<E> {

//returns number of elements in Deque

int size();

//checks if deque is empty

boolean isEmpty();

//returns but does not remove first element of deque

E first();

//returns but does not remove last element of deque

E last();

//adds an element to beginning of deque

void addFirst(E e);

//adds an element to end of deque

void addLast(E e);

//removes and returns the first element of deque

E removeFirst();

//returns and removes the last element of deque

E removeLast();

}

**ListDeque:**

/\*\*

\*

\* @author marie

\*/

public class ListDeque<E> implements Deque<E> {

private DoublyLinkedList<E> list;

//constructor

public ListDeque(){

list = new DoublyLinkedList<>();

}

//returns size of the double ended queue

@Override

public int size(){

return list.size();

}

//returns true if queue is empty

@Override

public boolean isEmpty(){

return list.isEmpty();

}

//returns the first element in queue

@Override

public E first(){

return list.first();

}

//returns last element of queue

@Override

public E last(){

return list.last();

}

//add an element to the beginning of queue

@Override

public void addFirst(E e){

list.addFirst(e);

}

//adds an element to the end of queue

@Override

public void addLast(E e){

list.addLast(e);

}

//return and remove the last element of queue

@Override

public E removeLast(){

return list.removeLast();

}

//return and remove the first element of queue

@Override

public E removeFirst(){

return list.removeFirst();

}

//equals method

@Override

public boolean equals(Object o){

if(!(o instanceof ListDeque)){

return false;

}

ListDeque test = (ListDeque<E>) o;

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

E temp1 = this.removeFirst();

E temp2 = (E) test.removeFirst();

this.addLast(temp1);

test.addLast(temp2);

if(!temp1.equals(temp2)){

return false;

}

}

return true;

}

//toString method

@Override

public String toString(){

String temp = getClass().getName() + "@";

E hold;

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

hold = this.removeFirst();

temp += " : " + hold;

this.addLast(hold);

}

return temp;

}

}

**ArrayDeque:**

/\*\*

\*

\* @author Marie Larson

\* @version 2/26/18

\*/

public class ArrayDeque<E> implements Deque<E> {

private E[] queue;

private int size;

private int start;

//constructor

public ArrayDeque(){

this.queue = (E[]) new Object[16];

this.size = 0;

this.start = 0;

}

//returns number of elements in queue

@Override

public int size(){

return size;

}

//checks if queue is empty

@Override

public boolean isEmpty(){

return this.size == 0;

}

//returns the first element in queue

@Override

public E first(){

return queue[start];

}

//returns the last element in queue

@Override

public E last(){

return queue[(start + size-1)%queue.length];

}

//adds an item to the front of queue

@Override

public void addFirst(E e){

if(queue.length == size()){

throw new IllegalArgumentException("ERROR: Queue is full");

}

if(start-- <= 0){

start = queue.length -1;

}

if(size==0){

start = 0;

}

queue[start] = e;

size++;

}

//add an element to the end of queue

@Override

public void addLast(E e){

if(queue.length == size()){

throw new IllegalArgumentException("ERROR: Queue is full");

}

queue[(start + size++)%queue.length] = e;

}

//removes first element of queue

@Override

public E removeFirst(){

if(isEmpty()){

return null;

}

E temp = first();

queue[start++] = null;

return temp;

}

//removes last element of queue

@Override

public E removeLast(){

if(isEmpty()){

return null;

}

E temp = last();

queue[(start + --size)%queue.length] = null;

return temp;

}

//equals method

public boolean equal(Object o){

if(!(o instanceof ArrayDeque)){

return false;

}

ArrayDeque test = (ArrayDeque) o;

if((this.size != test.size) || (this.start != test.start)){

return false;

}

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

if(!queue[this.start+i].equals(test.queue[test.start+i])){

return false;

}

}

return true;

}

//string method

@Override

public String toString(){

String temp = getClass().getName() + "@ size:" + size + " Start:" + start;

for(int i=0;i<queue.length;i++){

if(queue[i] != null){

temp += ":" + queue[i];

}

}

return temp;

}

}

**Output:**

run:

Begin

Testing the following deque: ArrayDeque

Adding 4 players to the front of deque and 4 players to the end of the deque

Team Lineup: ArrayDeque@ size:8 Start:13:Player@Name1:Position1:1:Player@Name2:Position2:2:Player@Name4:Position4:4:Player@Name6:Position6:6:Player@Name8:Position8:8:Player@Name7:Position7:7:Player@Name5:Position5:5:Player@Name3:Position3:3

Exception in thread "main" java.lang.NullPointerException

The following player has been removed from the front of the queue: Player@Name7:Position7:7

The following player has been removed from the end of the queue: null

Adding more players

Size of Queue: 11

checking if queue is empty: false

at ListDeque.equals(ListDeque.java:75)

The First player on the team: Player@Name12:Position12:12

at Client.dequeTest(Client.java:48)

The last player on the team: Player@Name10:Position10:10

at Client.main(Client.java:15)

Players on the team: ArrayDeque@ size:11 Start:12:Player@Name1:Position1:1:Player@Name2:Position2:2:Player@Name4:Position4:4:Player@Name6:Position6:6:Player@Name8:Position8:8:Player@Name9:Position9:9:Player@Name10:Position10:10:Player@Name12:Position12:12:Player@Name11:Position11:11:Player@Name5:Position5:5:Player@Name3:Position3:3

Checking if array equals its self: true

Testing the following deque: ListDeque

Adding 4 players to the front of deque and 4 players to the end of the deque

Team Lineup: ListDeque@ : Player@Name19:Position19:19 : Player@Name17:Position17:17 : Player@Name15:Position15:15 : Player@Name13:Position13:13 : Player@Name13:Position13:13 : Player@Name13:Position13:13 : Player@Name13:Position13:13 : Player@Name13:Position13:13

The following player has been removed from the front of the queue: Player@Name13:Position13:13

The following player has been removed from the end of the queue: Player@Name15:Position15:15

Adding more players

Size of Queue: 10

checking if queue is empty: false

The First player on the team: Player@Name24:Position24:24

The last player on the team: Player@Name13:Position13:13

Players on the team: ListDeque@ : Player@Name24:Position24:24 : Player@Name23:Position23:23 : Player@Name13:Position13:13 : null : Player@Name13:Position13:13 : null : null : null : null : null

C:\Users\marie\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned: 1

BUILD FAILED (total time: 0 seconds)