import java.util.Iterator;

import java.util.NoSuchElementException;

public class Deque<Item> implements Iterable<Item> {

private Node first;

// 8 bytes

private Node last;

// 8 bytes

private int size;

// 4 bytes

private class Node {

// 16字节对象开销+8字节内部类额外开销+8+8+8=48 bytes, n个节点就是48n bytes 10 private Node preNode; // 前一个节点的引用 11 private Item item;

private Node nextNode; // 后一个节点的引用

}

private class ListIterator implements Iterator<Item> {

// 16字节对象开销+8字节内部类额外开销+8=32 bytes

private Node curr = first;

@Override

public boolean hasNext() {

21 // TODO Auto-generated method stub

return curr != null;

}

@Override

public Item next() {

// TODO Auto-generated method stub

if (curr == null) throw new NoSuchElementException("there are no more items!");

Item item = curr.item;

curr = curr.nextNode;

return item;

}

//remove不用设计,父类Iterator的remove方法就抛出UnsupportedOperationException("remove");

}

public Deque() {

// construct an empty deque

size = 0;

first = null;

last = null;

}

public boolean isEmpty() {

// is the deque empty?

return (size == 0);

}

public int size() {

// return the number of items on the deque 51 return size;

}

public void addFirst(Item item) {

// add the item to the front 56 valivate(item);

Node newNode = new Node();

newNode.item = item;

if (size == 0) {

// 空队列的情况

newNode.preNode = null;

newNode.nextNode = null;

first = newNode;

last = newNode;

} else {

newNode.preNode = null;

newNode.nextNode = first;

first.preNode = newNode;

first = newNode;

}

size++;

}

public void addLast(Item item) {

// add the item to the end 75 valivate(item);

Node newNode = new Node();

newNode.item = item;

if (size == 0) {

// 空队列的情况

newNode.preNode = null;

newNode.nextNode = null;

first = newNode;

last = newNode;

} else {

last.nextNode = newNode;

newNode.preNode = last;

newNode.nextNode = null;

last = newNode;

}

size++;

}

public Item removeFirst() {

// remove and return the item from the front

if (size == 0) throw new NoSuchElementException("the deque is empty!");

Item returnItem = null;

if (size == 1) {

returnItem = first.item;

first = null;

last = null;

} else {

Node oldfirst = first;

returnItem = oldfirst.item;  
first = oldfirst.nextNode;  
first.preNode = null;  
oldfirst.nextNode = null;  
oldfirst.item = null;

}  
size--;  
return returnItem;  
}

public Item removeLast() {

// remove and return the item from the end

if (size == 0) throw new NoSuchElementException("the deque is empty!");

Item returnItem = null;

if (size == 1) {

returnItem = first.item;  
first = null;  
last = null;  
} else {

Node oldlast = last;  
returnItem = oldlast.item;  
last = oldlast.preNode;

last.nextNode = null;

oldlast.preNode = null;

oldlast.item = null;

}

size--;  
return returnItem;

}

public Iterator<Item> iterator() {

// return an iterator over items in order from front to end

return new ListIterator();

}

private void valivate(Item item) {

if (item == null)141 throw new IllegalArgumentException("the item is null!");142 }143 144 public static void main(String[] args) {145 // unit testing (optional)146 Deque<String> queue = new Deque<String>();147 System.out.println(queue.size);148 queue.addFirst("a");149 queue.addFirst("b");150 queue.addLast("c");151 queue.addFirst("d");152 queue.addLast("e");153 System.out.println(queue.size);154 Iterator<String> iter = queue.iterator();155 while (iter.hasNext()) {156 System.out.println(iter.next());157 }158 }159 }