package zad3;  
  
import java.util.EmptyStackException;  
  
public class Main {  
 public static void main(String[] args){  
 System.out.println("------Stack testing------");  
 Stack<Integer> stack = new Stack<>();  
  
 stack.push(15);  
 stack.push(25);  
 stack.push(35);  
 stack.push(45);  
 stack.push(65);  
  
 System.out.println("Stack size: " + stack.size());  
 System.out.println("Stack top: " + stack.top());  
 System.out.println("Stack size: " + stack.size());  
 System.out.println("Stack pop: " + stack.pop());  
 System.out.println("Stack size: " + stack.size());  
  
 while(!stack.isEmpty()){  
 System.out.println("Stack pop: " + stack.pop());  
 }  
  
 try{  
 stack.pop();  
 } catch(EmptyStackException e){  
 System.out.println("Stack is empty!");  
 }  
  
 System.out.println("**\n**------Queue testing------");  
 Queue<Integer> queue = new Queue<>();  
 queue.enqueue(150);  
 queue.enqueue(4995);  
 queue.enqueue(12);  
 queue.enqueue(59);  
 queue.enqueue(458);  
  
 System.out.println("Queue size: " + queue.size());  
 System.out.println("Queue first: " + queue.first());  
 System.out.println("Queue size: " + queue.size());  
 System.out.println("Queue dequeue: " + queue.dequeue());  
 System.out.println("Queue size: " + queue.size());  
  
 while(!queue.isEmpty()){  
 System.out.println("Queue dequeue: " + queue.dequeue());  
 }  
  
 System.out.println("Queue size: " + queue.size());  
  
 try{  
 queue.dequeue();  
 } catch(EmptyQueueException e){  
 System.out.println("Queue is empty!");  
 }  
 }  
}

package zad3;  
  
public class EmptyQueueException extends RuntimeException{  
}

package zad3;  
  
public class LinkedList<E> {  
 public class Element{  
 private E value;  
 private Element next = null;  
  
 public Element(E value){  
 this.value = value;  
 }  
  
 public E getValue(){  
 return value;  
 }  
  
 public void setValue(E value){  
 this.value = value;  
 }  
  
 public Element getNext(){  
 return next;  
 }  
  
 public void setNext(Element next){  
 this.next = next;  
 }  
 }  
  
 private Element head;  
 private Element tail;  
  
 public LinkedList() { head = null; tail = null; }  
  
 boolean isEmpty() { return head == null; }  
  
 public void clear() { head = null; }  
  
 public int size() {  
 if(head == null){  
 return 0;  
 }  
  
 int size = 0;  
 Element element = head;  
 while(element != null){  
 element = element.getNext();  
 size++;  
 }  
  
 return size;  
 }  
  
 public E get(int index){  
 if(head == null){  
 return null;  
 }  
  
 int pos = 0;  
 Element element = head;  
 while(pos != index){  
 element = element.getNext();  
 pos++;  
 }  
  
 return element.getValue();  
 }  
  
 public void addEnd(E value){  
 Element newElement = new Element(value);  
  
 if(head == null){  
 head = new Element(value);  
 tail = head;  
  
 return;  
 }  
  
 tail.setNext(newElement);  
 tail = tail.getNext();  
 }  
  
 public Element deleteIndex(int pos){  
 if(head == null){  
 return null;  
 }  
  
 if(pos == 0){  
 Element element = head;  
 head = head.getNext();  
  
 return element;  
 }  
  
 Element element = head;  
 int currentPos = 0;  
 while(currentPos != pos - 1){  
 element = element.getNext();  
 currentPos++;  
 }  
  
 Element next = element.getNext();  
 element.setNext(next.getNext());  
  
 if(currentPos == size() - 1){  
 tail = next;  
 }  
  
 return next;  
 }  
}

package zad3;  
  
public class Queue<E> {  
 private LinkedList<E> list;  
  
 public Queue() { list = new LinkedList<>(); }  
  
 public boolean isEmpty(){  
 return list.isEmpty();  
 }  
  
 public void enqueue(E element){  
 list.addEnd(element);  
 }  
  
 public E dequeue() throws EmptyQueueException {  
 try{  
 return list.deleteIndex(0).getValue();  
 } catch(NullPointerException e){  
 throw new EmptyQueueException();  
 }  
 }  
  
 public E first(){  
 return list.get(0);  
 }  
  
 public int size(){  
 return list.size();  
 }  
  
  
}

package zad3;  
  
import java.util.EmptyStackException;  
  
public class Stack<E> {  
 private LinkedList<E> list;  
  
 public Stack(){  
 list = new LinkedList<>();  
 }  
  
 public boolean isEmpty(){  
 return list.isEmpty();  
 }  
  
 public E pop() throws EmptyStackException {  
 try{  
 return list.deleteIndex(list.size() - 1).getValue();  
 } catch(NullPointerException e){  
 throw new EmptyStackException();  
 }  
 }  
  
 public void push(E element){  
 list.addEnd(element);  
 }  
  
 public int size(){  
 return list.size();  
 }  
  
 public E top(){  
 return list.get(list.size() - 1);  
 }  
}

Wyniki działania programu:

------Stack testing------

Stack size: 5

Stack top: 65

Stack size: 5

Stack pop: 65

Stack size: 4

Stack pop: 45

Stack pop: 35

Stack pop: 25

Stack pop: 15

Stack is empty!

------Queue testing------

Queue size: 5

Queue first: 150

Queue size: 5

Queue dequeue: 150

Queue size: 4

Queue dequeue: 4995

Queue dequeue: 12

Queue dequeue: 59

Queue dequeue: 458

Queue size: 0

Queue is empty!