package zad3;  
  
import zad1.EmptyQueueException;  
import zad2.Stack;  
  
public class Main {  
 public static void main(String[] args){  
 QueueS<Integer> q = new QueueS<>();  
  
 //próba usunięcia elementu z pustej kolejki  
 try{  
 q.dequeue();  
 } catch (EmptyQueueException e){  
 System.*out*.println("Queue is empty!");  
 }  
  
 //dodawanie elementów do kolejki  
 q.enqueue(12);  
 q.enqueue(910000);  
 q.enqueue(1823);  
 q.enqueue(9813);  
 q.enqueue(34);  
  
 //usuwania lub tylko wypisywanie pierwszego elementu  
 System.*out*.println("Elements:");  
 System.*out*.println(q.first());  
 System.*out*.println(q.dequeue());  
 System.*out*.println(q.dequeue());  
 System.*out*.println(q.first());  
 System.*out*.println(q.first());  
  
 //Czyszczenie kolejki  
 q.clear();  
  
 System.*out*.println("Size: " + q.size());  
  
 //Ponowne dodanie do kolejki  
 q.enqueue(143);  
 q.enqueue(2553);  
 q.enqueue(6134);  
  
 System.*out*.println("Size: " + q.size());  
 }  
}

package zad3;  
  
import zad1.EmptyQueueException;  
  
public interface IQueueS<T> {  
 boolean isEmpty();  
 T dequeue() throws EmptyQueueException;  
 void enqueue(T element);  
 int size();  
 T first() throws EmptyQueueException;  
 void clear() throws EmptyQueueException;  
}

package zad3;  
  
import zad1.EmptyQueueException;  
import zad1.FullQueueException;  
import zad2.Stack;  
  
public class QueueS<T> implements IQueueS<T>, Cloneable {  
 private Stack<T> main;  
 private Stack<T> helper;  
 private int topIndex;  
  
 public QueueS()  
 {  
 main = new Stack<>();  
 helper = new Stack<>();  
 topIndex = 0;  
 }  
  
 @Override  
 public boolean isEmpty() {  
 return main.isEmpty();  
 }  
  
 @Override  
 public T dequeue() throws EmptyQueueException {  
 if(isEmpty())  
 throw new EmptyQueueException();  
  
 topIndex--;  
 return removeFirstItem();  
 }  
  
 @Override  
 public void enqueue(T element) throws FullQueueException {  
 main.push(element);  
  
 topIndex++;  
 }  
  
 @Override  
 public int size() {  
 return topIndex;  
 }  
  
 @Override  
 public T first() throws EmptyQueueException {  
 return getFirstItem();  
 }  
  
 @Override  
 public void clear() throws EmptyQueueException {  
 main.clear();  
 topIndex = 0;  
 }  
  
 @Override  
 public QueueS<T> clone() throws CloneNotSupportedException {  
 try{  
 return (QueueS<T>)super.clone();  
 } catch(CloneNotSupportedException e){  
 System.*out*.println("Clone not available");  
 return null;  
 }  
 }  
  
 private T getFirstItem(){  
 helper.clear();  
  
 int currentSize = main.size() - 1;  
 for(int i = 0; i < currentSize; i++){  
 helper.push(main.pop());  
 }  
  
 T element = main.pop();  
 helper.push(element);  
 main = reverseStack(helper);  
  
 return element;  
 }  
  
 private T removeFirstItem()  
 {  
 helper.clear();  
  
 int currentSize = main.size() - 1;  
 for(int i = 0; i < currentSize; i++){  
 helper.push(main.pop());  
 }  
  
 T element = main.pop();  
 main = reverseStack(helper);  
  
 return element;  
 }  
  
 public Stack<T> reverseStack(Stack<T> stack)  
 {  
 Stack<T> second = new Stack<>();  
  
 while(!stack.isEmpty()){  
 second.push(stack.pop());  
 }  
  
 return second;  
 }  
}

Wyjście programu:

Queue is empty!

Elements:

12

12

910000

1823

1823

Size: 0

Size: 3