KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Duomenų struktūros (P175B014)

Projektas

Atliko:

IF-8/1 gr. studentas

Tomas Odinas

2019 m. lapkričio 26 d.

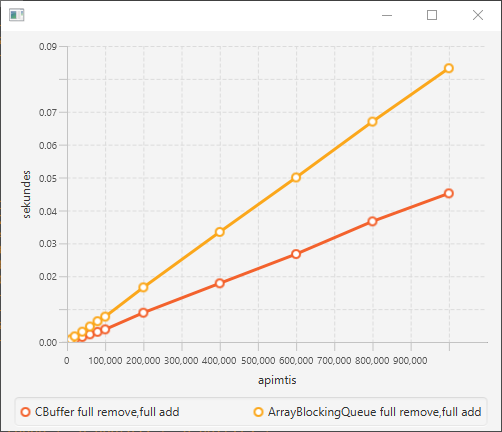
Priėmė:

Doc. Eimutis Karčiauskas

KAUNAS 2019

# Metodų greitaveika

## CircularBuffer ir ArrayBlockingQueue greitaveikos palyginimas (pilnas ištrynimas ir pilnas užpildymas)



# Programinis kodas

## CircularBuffer sąsaja

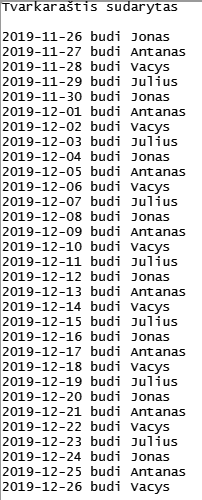
package odinas;  
  
*/\*\*  
 \* FIFO data structure  
 \** ***@param*** <*T*> *Type param  
 \*/*public interface CircularBuffer<T> {  
 */\*\*  
 \** ***@return*** *true if buffer is full.  
 \*/* boolean isFull();  
  
 */\*\*  
 \** ***@return*** *true if buffer is empty  
 \*/* boolean isEmpty();  
  
 */\*\*  
 \* adds element to the end  
 \** ***@param*** *element element to add  
 \*/* void add(T element);  
  
 */\*\*  
 \* Removes element at head position (FIFO)  
 \** ***@return*** *removed element or null if buffer is empty.  
 \*/* T remove();  
  
 */\*\*  
 \* Doesn't remove element, just returns it  
 \** ***@return*** *element at head position or null if buffer is empty.  
 \*/* T peek();  
  
 */\*\*  
 \* Clears buffer to empty  
 \*/* void clear();  
}

## Sąsają realizuojanti klasė

package odinas;  
  
import java.util.Arrays;  
import java.util.Iterator;  
import java.util.NoSuchElementException;  
  
public class CBuffer<T> implements CircularBuffer<T>, Iterable<T> {  
 private Object[] elements;  
 private int head = -1;  
 private int tail = -1;  
 private int size = 0;  
  
 public CBuffer(int maxSize) {  
 elements = new Object[maxSize];  
 }  
  
 public CBuffer(T[] elements) {  
 this.elements = elements;  
 head = tail = 0;  
 size = elements.length;  
 }  
  
 @Override  
 public boolean isFull() {  
 return size == elements.length;  
 }  
  
 @Override  
 public boolean isEmpty() {  
 return size == 0;  
 }  
  
 @Override  
 public void add(T element) throws IndexOutOfBoundsException {  
 if (isFull())  
 throw new IndexOutOfBoundsException("Buffer is full");  
 else if (tail == -1) {  
 tail = head = 0;  
 elements[0] = element;  
 size++;  
 } else {  
 tail = (tail + 1) % elements.length;  
 elements[tail] = element;  
 size++;  
 }  
 }  
  
 @Override  
 public T remove() throws NoSuchElementException {  
 if (isEmpty())  
 throw new NoSuchElementException("Buffer is empty");  
 else {  
 var element = elements[head];  
 head = (head + 1) % elements.length;  
 size--;  
 return (T) element;  
 }  
 }  
  
 @Override  
 public T peek() {  
 if (!isEmpty())  
 return (T) elements[head];  
  
 return null;  
 }  
  
 @Override  
 public void clear() {  
 tail = head = size = 0;  
 }  
  
 @Override  
 public String toString() {  
 return Arrays.*toString*(elements);  
 }  
  
 @Override  
 public Iterator<T> iterator() {  
 return new InfiniteIterator();  
 }  
  
 */\*\*  
 \* Infinite looping iterator (used for calculating timetables or shifts in fabrics, etc.)  
 \*/* private class InfiniteIterator implements Iterator<T> {  
 private int i = -1;  
  
 @Override  
 public boolean hasNext() {  
 return true;  
 }  
  
 @Override  
 public T next() {  
 i = (i + 1) % elements.length;  
 return (T) elements[i];  
 }  
 }  
}

## Panaudojimo pavyzdys

Slenkančio darbo grafiko, pamainų, budėjimų tvarkaraščių sudarymas.

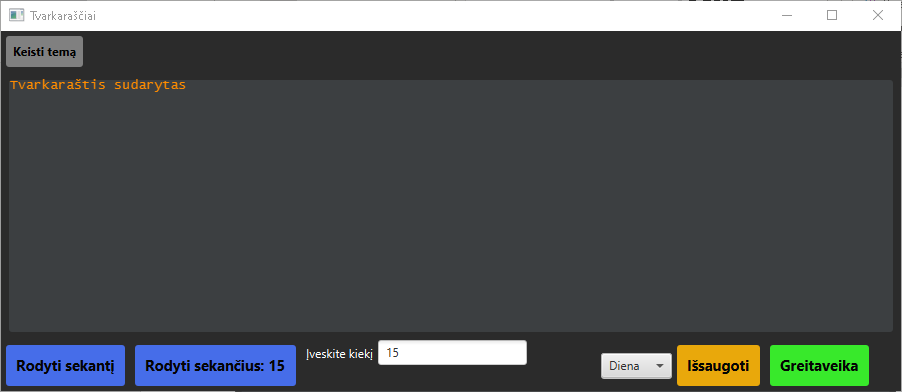


# Papildomai

## Testavimas su JUnit

package tests;  
  
import odinas.CBuffer;  
import org.junit.jupiter.api.Test;  
  
import java.lang.reflect.Field;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class CBufferTest {  
  
 @org.junit.jupiter.api.Test  
 void isFull() {  
 var a = new CBuffer<Integer>(2);  
 *assertFalse*(a.isFull());  
 a.add(1);  
 *assertFalse*(a.isFull());  
 a.add(2);  
 *assertTrue*(a.isFull());  
  
 for (int i = 2; i < 30; i++) {  
 a = new CBuffer<Integer>(i);  
 for (int j = 0; j < i - 1; j++) {  
 a.add(j);  
 *assertFalse*(a.isFull());  
 }  
 a.add(5);  
 *assertTrue*(a.isFull());  
 }  
 }  
  
 @org.junit.jupiter.api.Test  
 void isEmpty() {  
 var a = new CBuffer<Integer>(3);  
 *assertTrue*(a.isEmpty());  
 a.add(1);  
 *assertFalse*(a.isEmpty());  
 a.remove();  
 *assertTrue*(a.isEmpty());  
 }  
  
 @org.junit.jupiter.api.Test  
 void add() throws NoSuchFieldException, IllegalAccessException {  
 for (int i = 1; i < 100; i++) {  
 var a = new CBuffer<Integer>(i);  
 Field f = a.getClass().getDeclaredField("elements");  
 Field f2 = a.getClass().getDeclaredField("tail");  
 f.setAccessible(true);  
 f2.setAccessible(true);  
  
 for (int j = 1; j < i; j++) {  
 Integer element = j;  
 a.add(element);  
 var elements = (Object[]) f.get(a);  
 var tail = (int) f2.get(a);  
 *assertTrue*(elements[tail].equals(element));  
 }  
 }  
 }  
  
  
 @Test  
 void clear() {  
 var a = new CBuffer<String>(3);  
 *assertTrue*(a.isEmpty());  
 a.add("asd");  
 *assertFalse*(a.isEmpty());  
 a.clear();  
 *assertTrue*(a.isEmpty());  
 }  
  
 @Test  
 void peek() throws NoSuchFieldException, IllegalAccessException {  
 var a = new CBuffer<Integer>(3);  
 *assertNull*(a.peek());  
 a.add(5);  
 a.add(6);  
 var field = a.getClass().getDeclaredField("size");  
 field.setAccessible(true);  
 int size1 = (int) field.get(a);  
  
 *assertEquals*(5, a.peek());  
 var field2 = a.getClass().getDeclaredField("size");  
 field2.setAccessible(true);  
 int size2 = (int) field2.get(a);  
 *assertEquals*(size1, size2);  
 }  
}

## Grafinė sąsaja



# Išvados

Atlikus nedidelę modifikaciją programą galima panaudoti slaptažodžių generavimui.

Viena naudingiausių „circular buffer“ duomenų struktūros ypatybių yra tai, jog nereikia perstumti elementų, kai prieinamas buferio galas. Todėl buferis tinka dažnoms įrašymo/trynimo operacijoms.

Visos operacijos buferyje atliekamos per laikotarpį O(1).

LZ77 duomenų suspaudimo algoritmų šeima naudoja „circular buffer“ duomenims saugoti.

Jeigu reikalinga fiksuoto dydžio eilė (FIFO), tai „circular buffer“ implementacija tam yra ideali.

Laboratorinio darbo kodas: <https://github.com/d0ubletr0uble/Projektas_IF-8-1_Odinas_Tomas>