package zad1;  
  
public class Main {  
 public static void main(String[] args){  
 Queue<Integer> queue = new Queue<>(10);  
  
 //Dodawanie elementów do kolejki  
 System.*out*.println("Dodawanie elementów to kolejki aż do osiągnięcia maksymalnej wielkości.");  
 queue.enqueue(15);  
 queue.enqueue(3);  
 queue.enqueue(57);  
 queue.enqueue(113);  
 queue.enqueue(432);  
 queue.enqueue(523);  
 queue.enqueue(326);  
 queue.enqueue(116);  
 queue.enqueue(990);  
 queue.enqueue(511);  
 System.*out*.println("Elementow w kolejce: " + queue.size());  
  
 //Próba dodania elementu do pełnej kolejki  
 try{  
 queue.enqueue(96423);  
 } catch (FullQueueException e){  
 System.*out*.println("Kolejka jest pełna!");  
 }  
  
 System.*out*.println("Wyciąganie elementów z kolejki:");  
 System.*out*.println(queue.dequeue());  
 System.*out*.println("Rozmiar kolejki: " + queue.size());  
 System.*out*.println(queue.first());  
 System.*out*.println("Rozmiar kolejki: " + queue.size());  
 System.*out*.println(queue.dequeue());  
 System.*out*.println(queue.dequeue());  
 System.*out*.println(queue.dequeue());  
 System.*out*.println(queue.dequeue());  
 System.*out*.println(queue.dequeue());  
 System.*out*.println(queue.dequeue());  
 System.*out*.println(queue.dequeue());  
 System.*out*.println(queue.dequeue());  
 System.*out*.println(queue.dequeue());  
  
 //Próba wyciągnięcia elementu z pustej kolejki  
 try{  
 queue.dequeue();  
 } catch (EmptyQueueException e){  
 System.*out*.println("Kolejka jest pusta!");  
 }  
 }  
}

package zad1;  
  
public interface IQueue<T> {  
 boolean isEmpty();  
 boolean isFull();  
 T dequeue() throws EmptyQueueException;  
 void enqueue(T element) throws FullQueueException;  
 int size();  
 T first() throws EmptyQueueException;  
 void clear() throws EmptyQueueException;  
}

package zad1;  
  
public class EmptyQueueException extends RuntimeException{  
}

package zad1;  
  
public class FullQueueException extends RuntimeException{  
}

package zad1;  
  
public class Queue<T> implements IQueue<T>{  
 private int topIndex;  
 private T[] array;  
  
 public Queue(int size){  
 topIndex = 0;  
 array = (T[])(new Object[size]);  
 }  
  
 @Override  
 public boolean isEmpty() {  
 return topIndex == 0;  
 }  
  
 @Override  
 public boolean isFull() {  
 return topIndex == array.length;  
 }  
  
 @Override  
 public T dequeue() throws EmptyQueueException {  
 T value = first();  
  
 moveArrayToLeft();  
  
 return value;  
 }  
  
 @Override  
 public void enqueue(T element) throws FullQueueException {  
 if(!isFull())  
 array[topIndex++] = element;  
 else  
 throw new FullQueueException();  
 }  
  
 @Override  
 public int size() {  
 return topIndex;  
 }  
  
 @Override  
 public T first() throws EmptyQueueException {  
 if(!isEmpty())  
 return array[0];  
 else  
 throw new EmptyQueueException();  
 }  
  
 @Override  
 public void clear() throws EmptyQueueException {  
 for(int i = 0; i < size(); i++){  
 array[i] = null;  
 }  
  
 topIndex = 0;  
 }  
  
 private void moveArrayToLeft(){  
 for(int i = 0; i < size() - 1; i++){  
 array[i] = array[i + 1];  
 }  
  
 array[size() - 1] = null;  
 topIndex--;  
 }  
}

--------------------------------------------------------------------------------------------------------------------------------------

Wyniki programu:

Dodawanie elementów to kolejki aż do osiągnięcia maksymalnej wielkości.

Elementow w kolejce: 10

Kolejka jest pełna!

Wyciąganie elementów z kolejki:

15

Rozmiar kolejki: 9

3

Rozmiar kolejki: 9

3

57

113

432

523

326

116

990

511

Kolejka jest pusta!