zlepek.java

```
import java.util.ArrayList;

public class FullException extends Exception{
    public FullException(){}

    public FullException(String message){
        super(message);
    }

}

public class EmptyException extends Exception{
    public EmptyException(){}

    public EmptyException(String message){
        super(message);
    }

}

public interface MyQueue<E> {
    public void enqueue(E x) throws FullException;
    public void dequeue();
    public E first() throws EmptyException;
    public boolean isEmpty();
    public boolean isFull();

}


import java.lang.reflect.Array;
        import java.util.ArrayList;

public class CAQueue<E> implements MyQueue<E> {

    ArrayList<E> ARR;
    int SIZE,f,r;

    public CAQueue(int SIZE){
        this.SIZE = SIZE+1;
        this.f = 0;
        this. r = 0;
        this.ARR = new ArrayList<>(this.SIZE);
    }

    public CAQueue(){
        this.SIZE = 10; //default size
        this.f = 0;
        this. r = 0;
        this.ARR = new ArrayList<>(this.SIZE);
    }

    private int mod(int VAL){
        return VAL % this.SIZE;
    }

    @Override
    public void enqueue(E x) throws FullException {

        if(this.isFull()) throw (new FullException("full queue"));

        else{
```

```
            this.ARR.add(this.r,x);
            this.r = this.mod(this.r +1);
        }

    }

    @Override
    public void dequeue() {

        if (this.isEmpty() ) return;

        else {
            this.f = this.mod(this.f +1);
        }
    }

    @Override
    public E first() throws EmptyException {
        if(this.isEmpty()) throw (new EmptyException("empty queue"));

        else{
            return this.ARR.get(this.f);
        }
    }

    @Override
    public boolean isEmpty() {
        return this.r == this.f;
    }

    @Override
    public boolean isFull() {
        return mod(this.r +1) == this.f;
    }
}




/** ---------------------------------- **/

import java.util.ArrayList;

public class test {

    public static void main(String args[]) throws FullException,EmptyException{

        CAQueue<Integer> caq= new CAQueue<>(4);

        System.out.println(caq.isEmpty());
        System.out.println(caq.isFull());

        caq.enqueue(new Integer(1));
        caq.enqueue(new Integer(2));
        caq.enqueue(new Integer(3));
        caq.enqueue(new Integer(4));

        System.out.println(caq.isEmpty());
        System.out.println(caq.isFull());

        System.out.println(caq.first());
        caq.dequeue();

        System.out.println(caq.first());
        caq.dequeue();
```

```
        System.out.println(caq.first());
        caq.dequeue();

        System.out.println(caq.first());
        caq.dequeue();

        try {
            System.out.println(caq.first());
        }
        catch (Exception e){
            System.err.println( e.getMessage() + '\n');
            e.printStackTrace();
        }

        System.out.println(caq.isEmpty());

        caq.enqueue(new Integer(1));
        caq.enqueue(new Integer(2));
        caq.enqueue(new Integer(3));
        caq.enqueue(new Integer(4));

        System.out.println(caq.isFull());

        try {
            caq.enqueue(new Integer(5));
        }
        catch (Exception e){
            System.err.println( e.getMessage() + '\n');
            e.printStackTrace();
        }

        /*

        true
false
false
true
1
2
3
4
true
empty queue

EmptyException: empty queue
    at CAQueue.first(CAQueue.java:51)
    at test.main(test.java:33)
true
full queue

FullException: full queue
    at CAQueue.enqueue(CAQueue.java:30)
    at test.main(test.java:50)

Process finished with exit code 0



         */


    }
}
```

```java
/** -------- ZAD2 ----------- **/

public class Test {

    int zawartosc = 0;



    static void argNiemodyfikowalny(final Test zmienna){
        zmienna.zawartosc = 1;
        //zmienna = null;    // nie mozna zmienic zawartosci argumentu przekazanego jako final
    }

    static void argModyfikoiwalny(Test zmienna){
        zmienna.zawartosc = 1;
        zmienna = null;
    }

    // nie mozemy zmienic referencji na ktora wskazuje zmienna 'niemodyfikowalna' ale
    // mozemy zmienic zawartosc obiektu na ktory zmienna final wskazuje

    public static void main(String[] args){
        Test modyfikowalna = new Test();
        final Test niemodyfikowalna = new Test();

        // a)
        argNiemodyfikowalny(modyfikowalna);
        System.out.println(modyfikowalna.zawartosc);  //1

        // b)
        argNiemodyfikowalny(niemodyfikowalna);
        System.out.println(niemodyfikowalna.zawartosc); //1

        // c)
        argModyfikoiwalny(modyfikowalna);
        System.out.println(modyfikowalna.zawartosc);  //1

        // d)
        argModyfikoiwalny(niemodyfikowalna);
        System.out.println(niemodyfikowalna.zawartosc); //1
    }
}
```