Zadanie 9 *17–19 grudnia 2014 r.*

KURS JĘZYKA JAVA

LISTA CYKLICZNA

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie 1.

Zdefiniuj klasę sparametryzowaną do pamiętania zbioru dynamicznego w postaci jednokierunkowej listy cyklicznej `CyclicList<T>`. Klasa ta ma być opakowaniem dla homogenicznej struktury listowej tworzonej wewnątrz na obiektach typu `Node`.

Twoja klasa powinna implementować operacje stosowe (dodawać element na początku listy `push` i usuwać element z początku listy `pop`) i kolejkowe (dodawać element na końcu listy `append` i usuwać element z początku listy `pop`).

```
public class CyclicList<T> implements MyStack<T>, MyQueue<t>
{
    private class Node { /*...*/ }

    private Node tail;

    public CyclicList () { /*...*/ }

    // ... metody push, pop, append i inne

    public String toString () { /*...*/ }
}
```

Przy próbie włożenia do listy wartości `null` należy zgłosić wyjątek `IllegalArgumentException`. Dopisz też metody podające długość listy `size()` i usuwającej wszystkie elementy z listy `clear()`.

Zadanie 2.

Do klasy reprezentującej listę cylkiczną dopisz w nagłówku interfejs `Iterable<T>` oraz zaimplementuj wewnątrz metodę `iterator()`, która będzie tworzyła i zwracała iterator związany z daną listą. Zdefiniuj też klasę iteratora dedykowaną dla Twojej listy i implementującą interfejs `Iterator<T>`. Metodę `remove()` zaimplementuj tak, że będzie ona zawsze zgłaszała wyjątek `UnsupportedOperationException`.

Twój iterator powinien być wrażliwy na wszelkie zmiany w liście, po której iteruje — jeśli w trakcie iteracji po liście zostanie na niej dokonana jakakolwiek zmiana, to próba użycia nieaktualnego już iteratora powinna skutkować zgłoszeniem wyjątku `IllegalStateException`.

W klasie listy zdefiniuj metodę `toString()`, która będzie zwracać łańcuch znakowy zawierający zawatość całej listy. Metoda ta powinna korzystać z iteratora korzysta z iteratora listy.

Zadanie 3.

Napisz program, który będzie testować działanie zdefiniowanej listy cyklicznej. Jako testu użyj swojej listy do zaimplementowania algorytmu Dijkstry, nazywanego stacją rozrządową, do zamiany wyrażenia z postaci infiksowej do postaci postfiksowej — w algorytmie tym jest używana zarówno kolejka jak i stos.

Twój program testujący powinien być aplikacją okienkową w technologii *Swing*, którą będzie przyjmować od użytkownika wyrażenie infiksowe i wyświetlać to samo wyrażenie w postaci postfiksowej.

Uwaga.

Program należy napisać, skompilować i uruchomić w zintegrowanym środowisku programistycznym *NetBeans*.