Zadanie 5 *17, 19 listopada 2020 r.*

KURS JĘZYKA JAVA

LISTA UPORZĄDKOWANA

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie 1.

W pakiecie `structures` zdefiniuj interfejs generyczny `OrderedSequence<>`, reprezentujący kolekcję uporządkowaną według wartości.

```
public interface OrderedSequence<T extends Comparable<T>> {
    // ...
}
```

Interfejs ten powinien definiować podstawowe operacje, które można będzie wykonywać na uporządkowanym ciągu: dodanie elementu do ciągu `insert(T el)`, usunięcie elementu z ciągu `remove(T el)` o ile taki istnieje, wskazanie elementu najmniejszego `min()` i największego `max()`, pobranie elenentu z określonej pozycji `at(int pos)`, sprawdzenie czy zadany element znajduje się w ciągu `search(T el)`, wskazanie pozycji na której znajduje się element `index(T el)`, itp.

Zadanie 2.

W pakiecie `structures` zdefiniuj klasę `OrderedList<>`, która będzie jednokierunkową listą generyczną implementująca interfejs `OrderedSequence<>`. Na liście tej elementy mają być uporządkowane od najmniejszego do największego i nie mogą się na niej powtarzać takie same wartości. Klasa ta ma być opakowaniem dla homogenicznej struktury tworzonej wewnątrz na węzłach typu `Node`.

```
class OrderedList <T extends Comparable<T>> implements OrderedSequence<T>
{
    private class Node <T extends Comparable<T>>
    {
        private Node<T> next;
        private T data;
        // ...
    }
    private Node<T> start;

    // ... implementacja OrderedSequence<T>

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

Klasę wewnętrzną `Node<>` należy wyposażyć w podobne metody jak w interfejsie `OrderedSequence<>`, wtedy klasa `OrderedList<>` będzie dużo prostsza do zaimplementowania. Zgłaszaj wyjątki zawsze gdy to będzie konieczne, na przykład przy próbie włożenia do listy wartości `null` należy zgłosić wyjątek `NullPointerException`.

Zadanie 3.

Kolekcję `OrderedList<>` usupełnij o implementację interfejsu `Iterable<>`. Umożliwi to przeglądanie kolekcji za pomocą pętli *for–each*. Zdefiniuj własną klasę iteratora implementując interfejs `Iterator<>` z metodami `hasNext()`, `next()` a także `remove()` (metoda ta może usunąć z kolekcji tylko ten element, który był udostępniony ostatnio wywołaną metodą `next()`). Zdefiniowany iterator niech będzie prywatną niestatyczną klasą wewnętrzną w kolekcji `OrderedList<>`.

Zadanie 4.

Uzupełnij swoje zadanie o krótki program testowy napisany poza pakietem `structures`. Program ma rzetelnie sprawdzić działanie listy jednokierunkowej — przetestuj wszystkie metody interfejsu `OrderedSequence<>` w kolekcji typu `OrderedList<Integer>`, `OrderedList<String>` i `OrderedList<Calendar>`. Spawdź też możliwość przeglądania tych kolekcji za pomocą pętli *for–each*.

Uwaga.

Implementując klasę dla listy nie korzystaj z żadnej kolekcji standardowej.