Zadanie 7 1, 4 grudnia 2025 r.

kurs języka Java

lista dwukierunkowa

Instytut Informatyki
Uniwersytetu Wrocławskiego Paweł Rzechonek

Część 1.

W pakiecie `structures` zdefiniuj interfejs generyczny `SimpleSequence<>`, reprezentujący nieuporządkowany ciąg danych (wartości w tym ciągu można jednak porównywać).

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

Interfejs `SimpleSequence<>` powinien definiować podstawowe operacje, które można będzie wykonywać na ciągu:

- dodanie do ciągu elementu na zadaną pozycję `insert(T el, int pos)`,
- usunięcie z ciągu pierwszego elementu o zadanej wartości `remove(T el)`,
- usunięcie z ciągu elementu ze wskazanej pozycji `remove(int pos)`,
- wskazanie elementu najmniejszego `min()` i największego `max()`,
- sprawdzenie czy element o zadanej wartości znajduje się w ciągu `search(T el)`,
- pobranie elementu z określonej pozycji `at(int pos)`,
- wskazanie pozycji na której znajduje się element o zadanej wartości `index(T el)`,
- sprawdzenie ile jest wszystkich elementów w ciągu `size()`,
- sprawdzenie czy ciąg jest pusty `empty()`.

Część 2.

W pakiecie `structures` zdefiniuj klasę `SimpleList<>`, która będzie dwukierunkową listą generyczną implementująca interfejs `SimpleSequence<>`. Na liście tej elementy nie mają być uporządkowane względem wartości i mogą się na niej powtarzać takie same wartości. Klasa ta ma być opakowaniem dla homogenicznej struktury tworzonej wewnątrz na węzłach typu `SimpleNode<>`.

```
class SimpleList<T extends Comparable<T>>
implements SimpleSequence<T> {
    private class SimpleNode <T> {
        private Node<T> prev, next;
        T data;
        // ... implementacja wspierająca SimpleSequence<T>
    }
    private SimpleNode<T> head;
```

```
    // ... implementacja SimpleSequence<T>
    @Override
    public String toString() { /*...*/ }
}
```

Klasę wewnętrzną `SimpleNode<>` należy wyposażyć w podobne metody jak w interfejsie `SimpleSequence<>`, wtedy klasa `SimpleList<>` będzie tylko opakowaniem dla homogenicznej struktury listowej zbudowanej na węzłach i będzie tym samym dużo prostsza do zaimplementowania. Zgłaszaj wyjątki zawsze, gdy to będzie konieczne, na przykład przy próbie włożenia do listy wartości `null` należy zgłosić wyjątek `NullPointerException`.

Część 3.

Kolekcję `SimpleList<>` uzupełnij o implementację interfejsu `Iterable<>`. Umożliwi to przeglądanie kolekcji za pomocą pętli *for-each*. Zdefiniuj własną klasę iteratora implementującą interfejs `Iterator<>` z metodami `hasNext()` i `next()`. Gdy podczas przeglądania kolekcji iteratorem kolekcja ta zostanie zmodyfikowana, to iterator traci ważność (każda próba jego użycia ma powodować zgłoszenie wyjątku `IllegalStateException`. Zdefiniowany iterator niech będzie prywatną niestatyczną klasą wewnętrzną w kolekcji `SimpleList<>`.

Część 4.

Uzupełnij swoje zadanie o krótki program testowy napisany poza pakietem `structures`. Program ma rzetelnie sprawdzić działanie listy dwukierunkowej `SimpleList<>`. Przetestuj wszystkie metody interfejsu `SimpleSequence<>` w kolekcjach list jednokierunkowych dla różnych typów parametrycznych: `SimpleList<Integer>`, `SimpleList<String>` i `SimpleList<Date>`. Sprawdź też możliwość przeglądania tych kolekcji za pomocą pętli *for-each*.

Uwaga.

Implementując klasę listy dwukierunkowej nie korzystaj z żadnej kolekcji standardowej!