Algorithmen und Datenstrukturen

Listen

Aufgabe 1 Erweiterung Klammertest – 2 Punkte

Programme werden oft erweitert. Beim Hinzufügen oder Anpassung der Funktionalität stellt sich die Frage, ob diese ohne Tests durchgeführt werden kann. Siehe auch die Folie bez. Vor-/Nachbedingung von Interfaces.

i) Führen Sie folgende Tests hinzu

```
test("(",false);
test(")",false);
```

Falls Ihr Program diese Tests auch besteht: Gratulation
Falls nicht: was haben Sie daraus bezüglich TDD gelernt?

ii) Es soll die weitere Klammerart "<" und ">" ebenfalls unterstützt werden. Passen Sie Ihr Programm entsprechend an.

```
test("<(<>)>",true);
test("<(<)>>",false);
```

Falls diese Anpassung in *einer Minute* gemacht werden kann: Gratulation

iii) Zusatzlich soll "<*" und "*>" als Klammerung genommen werden.

```
test("<*(<*<>*>)*>",true);
test("<(<**>)*>",false);
```

Falls diese Anpassung in *einer Minute* gemacht werden kann: Gratulation

iv) Falls Sie alle drei Aufgaben in 3 Minuten korrekt gelöst haben: Lösen Sie die Google Interview Frage.

v) Falls Sie diese ebenfalls korrekt lösen, dann lassen Sie sich von ADS dispensieren. Sie werden in dieser Vorlesung vermutlich nichts für Sie neues mehr lernen. Der Dozent leiht Ihnen gerne *The Art of Computer Programming* aus.

Aufgabe 2 Verkettete Liste – 4 Punkte

In dieser Aufgabe sollen Sie eine doppelt verkettete Liste programmieren. Entwickeln Sie eine Klasse MyList, die die folgenden Methoden des java.util.List Interfaces Implementiert.

- boolean add (Object o); // am Schluss der Liste anhängen
- boolean remove(Object obj); // Object mit dem gleichen Inhalt löschen (compareTo == 0)
- Object get(int pos); // beliebiges Objekt zurückliefern
- boolean isEmpty()
- int size();
- void clear();

Damit Sie nicht das vollständige Interface implementieren müssen, können Sie von AbstactList erben. Methoden die nicht implementiert sind, sollen die UnsupportedOperationException werfen.

Hinweise:

- Ihre IDE hat vermutlich eine Funktion, um ein Gerüst einer Klasse passend zu einem Interface zu generieren.
- Es hat sich gezeigt, dass eine zyklische Liste mit *einem* Header-Element zu der elegantesten (i.e. kürzesten) Implementation führt.

Aufgabe 3 Sortierte Liste – 4 Punkte

Implementieren Sie eine eigene SortedList Klasse, die wieder das java.util.List Interface implementiert. Ersetzen Sie Ihre Liste durch eine (eigene) SortedList und überprüfen Sie das korrekte Verhalten. Überlegen Sie sich, wie Sie die Reihenfolge bzw. die Ordnung der Objekte i.a. bestimmen können.

Hinweis:

- Die sortierte Liste soll von MyList erben