Zürcher Hochschule für Angewandte Wissenschaften

Algorithmen und Datenstrukturen

Listen

Aufgabe 1: Überprüfung Klammertest

Aufgabe 1 und 2 basieren auf der Aufgabe 3 aus dem ersten Praktikum. Bei dieser Aufgabe wird einerseits der Unit-Test (ADS1_3_test.java) erweitert und andererseits die Funktionalität Ihrer Klassen. Guter (clean-code) sollte sich auch gut erweitern lassen.

Führen Sie folgende Tests zu ADS1_3_test.java hinzu:

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

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

Abgabe
Praktikum: ADS2.1
Filename: BracketServer.java

Aufgabe 2: Erweiterung Klammertest

Es soll die weitere Klammerart "<" und ">" ebenfalls unterstützt werden. Passen Sie Ihr Programm entsprechend an und fügen Sie folgende Tests hinzu:

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

Falls diese Anpassung in zwei Minuten gemacht werden kann: Gratulation!

Zusätzlich soll "/*" und "*/" Klammer genommen werden können:

```
test("/* hallo */",true);
test("/*/* */",false);
test("/*",false);
```

Falls diese Anpassung in fünf Minuten gemacht werden kann: Gratulation!

Abgabe
Praktikum: ADS2.2 Filename:
BracketServer.java

Aufgabe 3: Verkettete Liste

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 Dummy-Element zu der elegantesten (i.e. kürzesten) Implementation führt.

Gerüst:

```
public class MyList extends AbstractList {
```

Abgabe
Praktikum: ADS2.3
Filename: MyList.java

Aufgabe 4: Sortierte Liste

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.

Hinweise:

- Die sortierte Liste soll von MyList erben
- Collections.sort bei jedem Einfügen aufzurufen ist zu ineffizient.

Gerüst:

```
public class MySortedList extends MyList {

   @Override
   public boolean add(Object val){
   ...
```

Abgabe
Praktikum: ADS2.4
Filename: MySortedList.java