Aufgabenblatt 2

Einführung in die Programmierung 2

LVA-Nr. 185.A92
2018 S
TU Wien

Thema:

lineare Liste, assoziative Datenstruktur

Ausgabe:

9. 4. 2018

Abgabe (Deadline):

16. 4. 2018, 6:00 Uhr

Lösungen hochladen und gelöste Aufgaben ankreuzen (TUWEL)

Skriptum:

Seiten 45 – 60
Aufgaben 2.12 – 2.26

Allgemeines zum Lösen von Aufgabenblättern

Verwenden Sie Ihr existierendes IntelliJ-IDEA-Projekt (Aufgabenblatt 1) und lösen Sie darin die Aufgaben. Eine gute Lösung erfüllt die Vorgaben der Aufgabenbeschreibung, ist kurz und einfach gehalten, ist mit informativen Kommentaren versehen und wurde gut getestet. Objektvariablen sind `private`.

Das Projekt mit den gelösten Aufgaben muss rechtzeitig vor der Deadline als ZIP-Datei in TUWEL hochgeladen werden. Programmtext für das Testen soll im Projekt enthalten sein. Bei mehrfachem Hochladen zählt die zuletzt hochgeladene ZIP-Datei. Es gibt keine andere Möglichkeit zur Abgabe.

Jede gelöste Aufgabe muss in TUWEL angekreuzt werden. Lösungen angekreuzter Aufgaben müssen in der Übungseinheit präsentiert werden können. Nach der Deadline ist das Ändern der Kreuzchen nicht möglich.

Aufgabe 1 (verpflichtend, 20%, 1 Punkt) **unbedingt lösen**

Diese Aufgabe wird in der Ad-hoc-Aufgabe erweitert. Wenn Sie sie nicht machen, werden Sie die Ad-hoc-Aufgabe vermutlich nicht schaffen.

Schreiben Sie eine Klasse `Playlist1` mit dem gleichen Konstruktor und den gleichen Methoden wie `Playlist` (Aufgabenblatt 1), wobei die Songs diesmal nicht in einem Array, sondern in einer einfach verketteten linearen Liste abgespeichert werden sollen.

Fragen

- Hat der Parameter des Konstruktors eine Funktion? Wenn doch, welche? Wenn nicht, ist es sinnvoll, den Parameter zu behalten? Warum, bzw. warum nicht?
- Welche Vor- und Nachteile hat die Implementierung von `Playlist1` im Vergleich zu `Playlist`? Wäre eine Array-Implementierung, die das Array gegen ein doppelt so grosses tauscht, wenn der Platz nicht mehr reicht, für diese Aufgabe weniger aufwändig? Ändert sich Ihre Einschätzung, wenn Sie auch die weiteren von Ihnen gelösten Aufgaben berücksichtigen?
- Muss Ihre Implementierung bei jedem `add`-Aufruf die ganze Liste durchgehen? Wenn ja, wie könnte man das vermeiden und wieviel Programmieraufwand wäre nötig? Wenn nein, wieviel Programmieraufwand hat das gekostet?

Aufgabe 2 (20%, 1 Punkt)

Fügen Sie zur Klasse `Playlist1` eine nicht-statische Methode

```
Song lookupTitle(String title)
```

hinzu, die den ersten Song in der Playlist zurückgibt, dessen Titel gleich `title` ist, bzw. `null`, wenn es keinen solchen Song gibt.

Fragen

Wie könnten Sie den letzten Song mit diesem Titel zurückgeben? Ist das aufwändiger?

Aufgabe 3 (verpflichtend, 20%, 1 Punkt)

unbedingt lösen

Auch diese Aufgabe sollten Sie machen um die Ad-Hoc-Aufgabe zu schaffen.

Fügen Sie einen weiteren Konstruktor zu `Playlist1` hinzu, wobei (mit `myplaylist` vom Typ `Playlist1`) der Aufruf

```
new PlayList1(myplaylist, l1, l2)
```

eine `Playlist1` mit einer Auswahl der Songs aus `myplaylist` (in der selben Reihenfolge) erstellen soll; und zwar soll die Auswahl alle Songs enthalten, deren Länge $\geq$ `l1` und $<$ `l2` ist.

`myplaylist` dabei nicht verändern

Fragen

Wäre die Aufgabe einfacher, wenn man die Reihenfolge der Songs nicht erhalten müsste? Warum bzw. warum nicht?

Aufgabe 4 (20%, 1 Punkt)

Fügen Sie eine nicht-statische Methode

```
void addAfter(String title, Song song)
```

zu `Playlist1` hinzu, die `song` in der `Playlist1` hinter dem ersten Song mit dem Titel `title` einfügt, bzw. am Ende, wenn es keinen Song mit dem Titel gibt.

Fragen

Wäre diese Aufgabe in `Playlist` (also mit Arrays) leichter zu lösen? Was hätte so eine Variante sonst für Vor- und Nachteile?

Aufgabe 5 (20%, 1 Punkt)

Fügen Sie eine nicht-statische Methode

```
void addBefore(String title, Song song)
```

zu `Playlist1` hinzu, die `song` in der `Playlist1` vor dem ersten Song mit dem Titel `title` einfügt, bzw. am Ende, wenn es keinen Song mit dem Titel gibt. Vertauschen Sie dabei keine Songs in Listenknoten.

zum Einfügen wird der Vorgänger des bei der Suche gefundenen Knotens benötigt

Fragen

Wäre diese Aufgabe in `Playlist` (also mit Arrays) leichter zu lösen? Was hätte so eine Variante sonst für Vor- und Nachteile?