Aufgabenblatt 1

Einführung in die Programmierung 2
LVA-Nr. 185.A92
2018 S
TU Wien

Thema:
Abstrakte Datentypen implementieren

Ausgabe:
12. 3. 2018

Abgabe (Deadline):
19. 3. 2018, 6:00 Uhr
Lösungen hochladen und gelöste Aufgaben ankreuzen (TUWEL)

Skriptum:
Seiten 31 – 45
Aufgaben 2.1 – 2.11

Allgemeines zum Lösen von Aufgabenblättern

Holen Sie sich das bereitgestellte IntelliJ-IDEA-Projekt aus TUWEL 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 (19. März, 6:00 Uhr für Aufgabenblatt 1) 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 und in künftigen Aufgabenblättern verwendet. Wenn Sie sie nicht machen, werden Sie die Ad-hoc-Aufgabe vermutlich nicht schaffen, und dann müssen Sie diese Aufgabe für die späteren Aufgabenblätter nachholen.

Schreiben Sie eine Klasse `Song` mit den Objektvariablen `titel`, `band` und `laenge`. Schreiben Sie einen Konstruktor, sodass man einen Eintrag für einen Song z.B. mit

```
new Song("When I'm Sixty-Four", "Beatles", 157)
```

erstellen kann. Schreiben Sie folgende nicht-statische Methoden in der Klasse `Song`:

Sichtbarkeit beachten: Objektvariablen private, verlangte Methoden und Konstruktoren public, Hilfsmethoden private

```
String getTitel()
String getBand()
long getLaenge()
void print()
```

Dabei soll `print()` den obigen Eintrag wie folgt ohne Zeilenumbruch am Ende anzeigen:

auf die Bedeutung von static bzw. nicht-static achten

```
Beatles: When I'm Sixty-Four (157s)
```

Fragen

Fragen sollen in der Übung beantwortet werden können

Sollte es auch Setter-Methoden für die Felder geben? Warum, warum nicht?

Aufgabe 2 (30%, 1.5 Punkte)

Schreiben Sie eine Klasse `Playlist` mit den Objektvariablen `songs` (ein Array von Songs aus Aufgabe 1) und `n` (die Anzahl der benutzten Array-Einträge). Schreiben Sie einen Konstruktor, sodass man mit dem Aufruf

```
new Playlist(100)
```

eine Playlist mit Platz für 100 Songs (oder einem anderen übergebenen Parameterwert) erstellen kann. Schreiben sie nicht-statische Methoden

public Methoden und Konstruktoren durch Kommentare beschreiben (Außensicht eines abstrakten Datentyps)

```
void add(Song song)
void print()
long getLaenge()
```

wobei `print()` die Songs in der Reihenfolge des Hinzufügens durch Verwendung der `print()`-Methode von `Song` ausgibt und nach jedem Song die Zeile umbricht (eine neue Zeile beginnt). `getLaenge()` gibt die Summe der Längen der Songs in der Playlist zurück.

Fragen

- Es könnte versucht werden, mehr Songs hinzuzufügen als Platz vorgesehen ist. Auf welche sinnvollen Arten kann man damit umgehen? Was macht Ihr Programm?
- Wie implementieren Sie `getLaenge()`? Gibt es noch andere Möglichkeiten? Welche Vor- und/oder Nachteile haben Ihre Implementierung und die Alternativen?

Aufgabe 3 (50%, 2.5 Punkte)

Fügen Sie zur Klasse `Playlist` einen weiteren Konstruktor hinzu, sodass man z.B. mit

```
new Playlist(myplaylist, 3600)
```

eine Auswahl der Songs aus der Playlist `myplaylist` in der selben Reihenfolge wie in `myplaylist` erhält, deren Gesamtlänge 3600s nicht überschreitet, die aber möglichst nahe an 3600s liegt. Entsprechendes muss natürlich auch für andere übergebene Parameterwerte gelten.

Ein rekursiver Algorithmus für dieses Auswahlproblem ist, dass man sich zunächst dafür entscheidet, den ersten Song zu nehmen und berechnet, welche von den restlichen Songs die restliche Zeit möglichst gut füllen; und dann probiert man aus, wie es ist, wenn man den ersten Song nicht nimmt und die Gesamtzeit mit den restlichen Songs möglichst gut füllt. Am Ende entscheidet man sich für die bessere der beiden Lösungen.

richtig eingesetzt kann Rekursion die Lösung wesentlich vereinfachen

Implementieren Sie diesen Algorithmus. Dafür ist es sinnvoll, mindestens eine (möglicherweise statische) Hilfsmethode zu erstellen.

Fragen

- Wieviele Aufrufe Ihrer rekursiven Methode erfolgen beim Übergeben einer Playlist mit n Songs maximal?
- Fällt Ihnen ein iterativer Algorithmus für dieses Problem ein?
  Wenn ja, wie funktiert er? (Schwer; ein „Nein“ auf die erste Teilfrage führt nicht zu einem Abzug.)