Lehrstuhl für Angewandte Informatik I
Prof. Dr. Bernhard Westfechtel
Sandra Greiner

WS 2017/18
04. Januar 2018

Konzepte der Programmierung

Übung 10

Abgabe: 18.01.2018

Aufgabe 10.1 (Objektorientierte Programmierbausteine)
Geben Sie *handschriftlich* für die folgenden **OOP**-Konzepte jeweils eine kurze beispielhafte Implementierung an. Sie dürfen gerne mehrere Konzepte innerhalb einer Klasse demonstrieren und dort mit Kommentaren benennen.

- Expliziter Konstruktor
- Objektmethode
- Klassenfeld
- Objekterzeugung
- Klassenmethode
- Objektfeld
- Operationsklasse
- Konstante

(**4 Punkte**)

Aufgabe 10.2 (Plätzchenverwaltung: Doppelt verkettete Listen)
Wir setzen bzw. ändern den Zugriff auf Plätzchendosen:

Statt immer nur die oberste Plätzchendose vom Stapel aus Aufgabe 9.4 zu erhalten, wollen wir es nun ermöglichen, die Dosen auch an beliebigen Stellen in der Datenstruktur zu erhalten. So könnte man sich z.B. vorstellen, dass man vor einem Regal steht und die Dosen nebeneinander stehen und man eine beliebige auswählen kann.

Daher implementieren wir nun eine *verkettete Liste*, die beliebigen Zugriff auf Elemente innerhalb der Datenstruktur erlaubt, sodass Dosen an jeder beliebigen Stelle eingefügt und gelöscht werden können. Eine solche Datenstruktur könnte man auch sortieren, z.B. nach Plätzchensorten.

Das Grundgerüst für die Dosenliste sieht wie folgt aus:

```
1  public class DosenListe {
2      private Element erstes;
3      class Element {
4          private PlaetzchenDose dose;
5          private Element voriges;
6          private Element nachfolgendes;
7
8          public Element(PlaetzchenDose d) {
9              dose = d;
10         }
```

```
11      }
12
13      public DosenListe(PlaetzchenDose d) {
14          erstes = new Element(d);
15      }
16  }
```

Hinweis: Bei allen Methoden müssen Sie damit rechnen, dass die Objektfelder `voriges` und `nachfolgendes` mit `null` belegt sind. Berücksichtigen Sie dies mit Hilfe entsprechender Fallunterscheidungen. Tipp: *Für das Einfügen und Löschen von Listenelementen empfiehlt es sich, aufzumalen wie sich die Zeiger der einzelnen Elemente verändern. Achten Sie auch auf die Reihenfolge, in der Sie die Zeiger setzen.*

Die folgenden Methoden sollen nicht außerhalb der Klasse verwendbar sein!

(a) Implementieren Sie eine Methode `fuegeAlsNachfolgerEin(...)`. Sie bekommt zwei Elemente übergeben, das neue Element, das eingefügt werden soll, und dessen Vorgänger. Entsprechend soll das neue Element hinter dem Vorgänger eingefügt werden.

(b) Implementieren Sie eine Methode `loesche(...)`. Sie löscht das ihr übergebene Element, indem Sie das vorige Element auf das Nachfolgende zeigen lässt und umgekehrt.

(c) Implementieren Sie eine Methode `gibLaenge(...)`, die die Anzahl der Elemente in der Liste zurückgibt.

Mit diesen Vorarbeiten ist es möglich, nun die folgenden, von anderen Klassen benutzbaren Methoden zu implementieren. *Machen Sie so viel Gebrauch von den schon implementierten Methoden wie möglich.*

(d) Eine Methode `fuegeEin(...)`. Sie bekommt die Position übergeben, an der eine ebenfalls übergebene Plätzchendose eingefügt werden soll. Falls die Position größer ist als die aktuelle Anzahl an Listenelementen, wird die Dose einfach hinten angefügt. Wir nehmen an, dass die Position wie in der Informatik üblich bei 0 zu zählen beginnt.

(e) Eine Methode `loesche(...)`, die nur eine Position übergeben bekommt. Sie gibt eine Fehlermeldung auf der Konsole aus, wenn die Position ungültig ist. Ansonsten wird das Element an dieser Stelle gelöscht.

(1+2+2+5+4 = 14 Punkte)

Aufgabe 10.3 (Binärbäume als rekursive Datenstrukturen)

In dieser Aufgabe entwickeln wir einen binären Suchbaum. Damit sollen Bücher einer Buchhandlung mittels ihrer ISBN *assoziativ* zugreifbar sein. Innerhalb des Suchbaums sind die Bücher gemäß ihrer ISBN sortiert. Die Sortierung wird beim Einfügen vorgenommen. Implementieren Sie dafür eine "rekursive" Klasse `ISBNBaum` mit folgenden Bestandteilen.

- Zwei Objektfeldern, `isbn`, die aus Buchstaben und Zahlen bestehen kann, und `titel` (des Buches), die beide im Konstruktor gesetzt werden und über geeignete Zugriffsmethoden lesend verfügbar sind.
  Achten Sie auf geeignete Sichtbarkeiten.
- Zwei Referenzen (Zeiger) auf einen linken bzw. rechten Teilbaum. Deklarieren Sie die entsprechenden Objektfelder öffentlich. Berücksichtigen Sie die Felder **nicht** im Konstruktor.

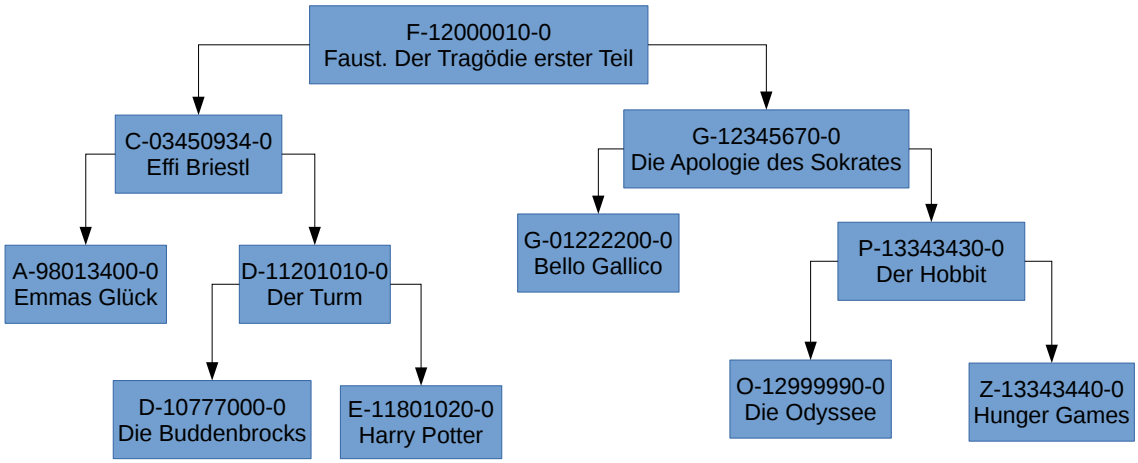
- Eine **rekursive** Objektmethode `findeBuch()`, die eine ISBN übergeben bekommt und den Titel des Buches zurückgibt.

  Unter Berücksichtigung der Suchbaumeigenschaft soll die Methode den Titel finden, der der übergebenen ISBN zugeordnet ist. Falls die ISBN nicht enthalten ist, soll `null` zurückgegeben werden.

- Eine rekursive boolesche Objektmethode `fuegeEin()`, der ein neuer Eintrag vom Typ `ISBNBaum` übergeben wird. Sie soll den gegebenen Eintrag an einer passenden Position als Seiteneffekt einfügen, wobei die Suchbaumeigenschaft bezüglich der ISBN erhalten bleiben soll. Falls es bereits einen Eintrag mit identischer ISBN gibt, soll nicht eingefügt werden und `false` zurückgegeben werden. Im Erfolgsfall wird `true` zurückgegeben.
  Hinweis: Der Baum muss nach dem Einfügen **nicht** balanciert sein.

- Eine rekursive Objektmethode `gibAus()`. Sie soll alle zu einem Eintrag (und dessen Teilbäumen) gehörenden Paare von ISBN und Titel zeilenweise ausgeben.

- Ein Hauptprogramm, in dem folgender Baum durch Objekterzeugung und Setzen von Objektfeldern erzeugt wird:



Danach sollen die beiden folgenden Einträge an der Wurzel eingefügt werden:
`D-12301230-0 -> Maria Stuart` und `1F-12121110-0 -> Der Zauberberg`.
Im Anschluss soll jeweils ein erfolgreicher und ein nicht erfolgreicher Aufruf von `findeName()` auf dem Wurzeleintrag erfolgen. Zum Schluss sollen alle Einträge von der Wurzel aus ausgegeben werden.

(**10 Punkte**)