Übung08: Dyn. Datenstrukturen

Abgabetermin: 17.05.19, 15:30 Uhr

Name: _____ Matrikelnummer: _____

Aufgabe	**schriftlich abzugeben**	**elektronisch abzugeben**	**gelöst**
Aufgabe21	Java Programm, Testausgaben	Java Programm	☐
Aufgabe22	Java Programm, Testausgaben	Java Programm	☐

Achtung!

Bitte auf diesem Deckblatt:

- **Name** und **Matrikelnummer** ausfüllen,
- gelöste Aufgaben **ankreuzen**

und dann der schriftlichen Abgabe als erste Seite anheften.

Aufgabe21: Wörterbuch als sortierte lineare Liste (12 Punkte)

Implementieren Sie ein Wörterbuch (Klasse `Dictionary`), das Paare von Wörtern in englischer Sprache und deren jeweilige deutsche Übersetzung speichert. Erstellen Sie dazu folgende Methoden:

- `public void insert(String word, String translation)` fügt ein englisches Wort mit der deutschen Übersetzung ein, wenn es noch kein solches Wort gibt.
- `public void delete(String word)` löscht einen vorhandenen Eintrag für das gegebene englische Wort.
- `public String lookup(String word)` sucht nach der deutschen Übersetzung für das gegebene Wort und liefert `null`, falls es keinen entsprechenden Eintrag gibt.
- `public String toString()` liefert eine `String`-Representation des Wörterbuchs.

Implementieren Sie das Wörterbuch als verkettete Liste. Erstellen Sie eine innere statische Klasse `Entry`, die intern zur Darstellung eines Eintrags und zum Aufbau der Listenstruktur verwendet wird. Dabei soll `Entry` das englische Wort und die Übersetzung, sowie eine Referenz auf das nächste `Entry` Objekt in der verketteten Liste speichern. `Dictionary` selbst besitzt dann eine Referenz auf das erste `Entry` Objekt in der Liste.

Hinweis:

Einträge sollen beim Einfügen **aufsteigend nach dem englischen Wort** sortiert werden. Verwenden Sie für den lexikographischen Vergleich zweier Zeichenketten die Methode `public int compareTo(String other)` der Klasse `String`. Diese liefert als Ergebnis einen Zahlenwert, der im jeweiligen Fall (s1 < s2, s1 == s2, s1 > s2) negativ, 0, oder positiv ist.

Testen Sie ihre Implementierung, indem Sie in einer `main`-Methode `Dictionary` Objekte anlegen, diese mit Übersetzungen füllen und Einträge wieder löschen. Überprüfen Sie die Sortierung, indem Sie nach jeder Operation die Wörterbücher ausgeben.

Sie können als einfache Anwendung auch z.B. einen Übersetzungsprozess realisieren. Lesen Sie Sätze in Englisch von der Konsole mittels `In.readWord()` ein bis ein Punkt eingelesen wird, und übersetzen Sie diese dann Wort für Wort. Befüllen Sie zuvor ein `Dictionary` Objekt mit entsprechenden Beispielen. Wörter, für die es keine Einträge gibt, können unverändert ausgegeben werden.

Beispiel:

```
car --> auto, i --> ich, love --> liebe, world --> welt

Geben Sie einen Satz zum Übersetzen ein (Ende mit .):
i love java .
ich liebe java .
Abbrechen? (y): y
```

Abzugeben sind:

- Java Programm
- Testausgaben

Aufgabe22: Bibliotheksbestand als binärer Suchbaum (12 Punkte)

Implementieren Sie für die Bestandsverwaltung einer Bibliothek einen binären Suchbaum, mit dem es möglich ist, Bücher und deren Vorräte einzufügen und den aktuellen Vorrat an Büchern für einen gegebenen Titel schnell abzufragen. Der Buchtitel ist der Schlüssel, nachdem entschieden wird, wo im Baum etwas eingefügt wird. Verwenden Sie für die Fallunterscheidung wieder die Methode `compareTo` der Klasse `String`. Gegeben ist die Klasse `Stock`, die den Vorrat eines bestimmten Buches abbildet:

```
public final class Stock {
  private final String title;
  private int amount;

  public Stock(String title, int amount) { … }

  public String getTitle()         { return title;    }
  public int getAmount()           { return amount;   }
  public void increment(int value) { amount += value; }
  public String toString()         { return String.format("[%s, %d]", title, amount); }
}
```

Erstellen Sie für den binären Suchbaum die Klasse `StockTree` mit der statischen inneren Klasse `Node`, um intern die Baumstruktur abzubilden. Jedes `Node` Objekt hat eine Referenz auf das linke und rechte Kind im Baum, sowie auf ein `Stock` Objekt. Implementieren Sie in der Klasse `StockTree` anschließend folgende Methoden:

- `public void insert(String title, int amount)` fügt das Buch mit dem gegebenen Titel und der Menge zum aktuellen Bestand und erzeugt dabei bei Bedarf einen neuen Knoten, falls das Buch noch nicht im Bestand vorhanden war.
- `public int amount(String title)` liefert die Menge im aktuellen Bestand für den gegebenen Buchtitel, indem im Binärbaum nach dem entsprechenden Knoten gesucht wird. Ist das Buch nicht vorhanden, soll 0 zurückgegeben werden.
- `public void printOrdered()` gibt den aktuellen Bestand mit allen Einträgen im Suchbaum aufsteigend sortiert nach den Buchtiteln auf der Konsole aus. Dies entspricht einer in-order Traversierung des Suchbaums.

Testen Sie Ihre Implementierung, indem Sie in einer `main`-Methode Suchbäume anlegen, diese mit Einträgen füllen, die Mengen im Bestand für gewisse Titel abrufen und die Einträge sortiert ausgeben. Sie können, um die Suchbäume aufzufüllen, folgende Auflistung als Bestand z.B. in eine Datei speichern und beim Programmstart einlesen.

Beispieldatei (stocks.txt):

```
1 Eternity's Sunrise
1 Paraguay (Bradt Travel Guide)
5 Treating Trauma and Traumatic Grief
3 A Terrible Revenge
2 Plastics
9 Microbiologically Safe Foods
1 Life Reimagined
2 It's St. Patrick's Day
6 Strands of Sorrow
7 Analytic Philosophy
9 The Modern Coral Reef Aquarium
10 Promises Kept
3 Vocabulary for the College Bound Student
```

```
5 Hunting the Rockies
5 The Slums of Aspen
3 Disaster Law and Policy
10 Mobile & Social Game Design
6 Ready, Freddy!
6 HSPT Flashcard Study System
10 On the Backroad to Heaven
```

Abzugeben sind:

- Java Programm
- Testausgaben