Karlsruher Institut für Technologie

Programmieren – Sommersemester 2018

Architecture-driven Requirements Engineering (ARE)
`https://sdqweb.ipd.kit.edu/wiki/Übung_Programmieren_SS18`
Jun.-Prof. Dr.-Ing. Anne Koziolek

Abschlussaufgabe 2

Ausgabe: 20.07.2018 – 13:00 Uhr
Abgabe: 18.08.2018 – 06:00 Uhr

Bearbeitungshinweise

- Achten Sie darauf, nicht zu lange Zeilen, Methoden und Dateien zu erstellen.[1]
- Programmcode muss in englischer Sprache verfasst sein.
- Kommentieren Sie Ihren Code angemessen: So viel wie nötig, so wenig wie möglich. Die Kommentare sollen einheitlich in englischer oder deutscher Sprache verfasst werden.
- Wählen Sie geeignete Sichtbarkeiten für Ihre Klassen, Methoden und Attribute.
- Verwenden Sie keine Klassen der Java-Bibliotheken ausgenommen Klassen der Pakete `java.lang`, `java.io` und `java.util`, es sei denn, die Aufgabenstellung erlaubt ausdrücklich weitere Pakete.[1]
- Achten Sie auf fehlerfrei kompilierenden Programmcode.[1]
- Halten Sie alle Whitespace-Regeln ein.[1]
- Halten Sie die Regeln zu Variablen-, Methoden- und Paketbenennung ein und wählen Sie aussagekräftige Namen.[1]
- Halten Sie die Regeln zur Javadoc-Dokumentation ein.[1]
- Nutzen Sie nicht das default-Package.[1]
- Halten Sie auch alle anderen Checkstyle-Regeln ein.[1]
- `System.exit` und `Runtime.exit` dürfen nicht verwendet werden.[1]
- **Wichtig**: Das Einreichen fremder Lösungen – seien es auch teilweise Lösungen – von Dritten, aus Büchern, dem Internet oder anderen Quellen – wie beispielsweise der Lehrveranstaltung oder dem Übungsbetrieb selbst – ist ein Täuschungsversuch und führt zur Bewertung **nicht bestanden**. Es werden nur rein selbstständig erarbeitete Lösungen akzeptiert. Für weitere Ausführungen sei auf die Einverständniserklärung (Disclaimer) verwiesen.

Abgabemodalitäten

Die Praktomat-Abgabe wird am **Freitag, den 03.08.2018 um 13:00 Uhr**, freigeschaltet.
Laden Sie die `Terminal`-Klasse nicht mit hoch.

- Geben Sie Ihre Antworten zu Aufgabe A als `*.java`-Dateien ab.

[1] Der Praktomat wird die Abgabe zurückweisen, falls diese Regel verletzt ist.

Terminal-Klasse
Laden Sie für diese Aufgabe die Terminal-Klasse herunter und platzieren Sie diese unbedingt im Paket `edu.kit.informatik`. Die Methode `Terminal.readLine()` liest eine Benutzereingabe von der Konsole und ersetzt `System.in`. Die Methode `Terminal.printLine()` schreibt eine Ausgabe auf die Konsole und ersetzt `System.out`. Verwenden Sie für jegliche Konsoleneingabe oder Konsolenausgabe die Terminal-Klasse. Verwenden Sie in keinem Fall `System.in` oder `System.out`. Fehlermeldungen werden ausschließlich über die Terminal-Klasse ausgegeben und müssen aus technischen Gründen unbedingt mit `Error` beginnen. Laden Sie die Terminal-Klasse **niemals** zusammen mit Ihrer Abgabe hoch.

Fehlerbehandlung
Ihre Programme sollen auf ungültige Benutzereingaben mit einer aussagekräftigen Fehlermeldung reagieren. Aus technischen Gründen muss eine Fehlermeldung unbedingt mit `Error,` beginnen. Eine Fehlermeldung führt nicht dazu, dass das Programm beendet wird; es sei denn, die nachfolgende Aufgabenstellung verlangt dies ausdrücklich. Achten Sie insbesondere auch darauf, dass unbehandelte `RuntimeExceptions`, bzw. Subklassen davon – sogenannte *Unchecked Exceptions* – nicht zum Abbruch Ihres Programms führen sollen.

Objektorientierte Modellierung
Achten Sie darauf, dass Ihre Abgaben sowohl in Bezug auf objektorientierte Modellierung, als auch Funktionalität bewertet werden.

Öffentliche Tests
Bitte beachten Sie, dass das erfolgreiche Bestehen der öffentlichen Tests für eine erfolgreiche Abgabe nötig ist. Planen Sie entsprechend Zeit für Ihren ersten Abgabeversuch ein.

A Verwaltungssystem für Prüfungsergebnisse (20 Punkte)

In dieser Aufgabenstellung sollen die Prüfungsergebnisse in Form von einer Punktzahl für Lehrveranstaltungen mit Hilfe eines *Trie*, der ein spezieller Suchbaum darstellt, abgespeichert und verwaltet werden. Die Studenten werden hierbei über das fünfstellige Kürzel ihres SCC-Accounts in Form von *uxxxx* identifiziert, das im Folgenden als `Name` des Studenten bezeichnet wird. Für *uxxxx* gilt, dass das erste Zeichen u fest gewählt und x jeweils ein Element aus dem Alphabet [a-z] ist.

B Hinweise zur Datenstruktur

Ein sogenannter Trie (auch Präfixbaum) stellt eine spezielle Form eines Suchbaums dar, der zur gleichzeitigen Speicherung mehrerer Zeichenketten dient. Dieser besitzt in der Regel einen relativ hohen Verzweigungsgrad.

Im Allgemeinen ist ein Trie über eine Menge von beliebigen Zeichenketten aufgebaut. Beim Eintragen eines Wertes in den Baum wird die dazugehörige Zeichenkette auch zeichenweise betrachtet. Für jedes Zeichen der Zeichenkette erfolgt ein Abstieg im Baum über die dem Zeichen entsprechende Kante. Jeder innere Knoten der Ebenen 1 bis 4 kann einen maximalen Verzweigungsgrad von 26 haben.
Im Ausgangszustand besteht ein Trie nur aus einem Wurzelknoten. Sind beim Eintragen in den Trie die passenden Kanten und Knoten noch nicht vorhanden, so müssen diese erzeugt werden. Soll ein Wert aus dem Trie gelöscht werden, so wird zunächst zur entsprechenden Stelle navigiert und der Wert gelöscht. Anschließend muss überprüft werden, ob dadurch unnütze Knoten und Kanten entstanden sind, welche gelöscht werden müssen. Knoten sind immer dann unnütz, wenn weder sie selbst noch einer ihrer Kindknoten einen Wert beinhalten. Kanten sind unnütz, wenn sie auf einen unnützen Knoten verweisen. Ein Wurzelknoten ist niemals unnütz.

Alle Zeichenketten bestehen im Folgenden nur aus Kleinbuchstaben und Ziffern. Weiterhin sind Namensbezeichnungen immer eindeutig. Die Punktzahl ist eine natürliche Zahl $\mathbb{N} \cup \{0\}$, die den Wertebereich eines `Integer` nicht übersteigt.

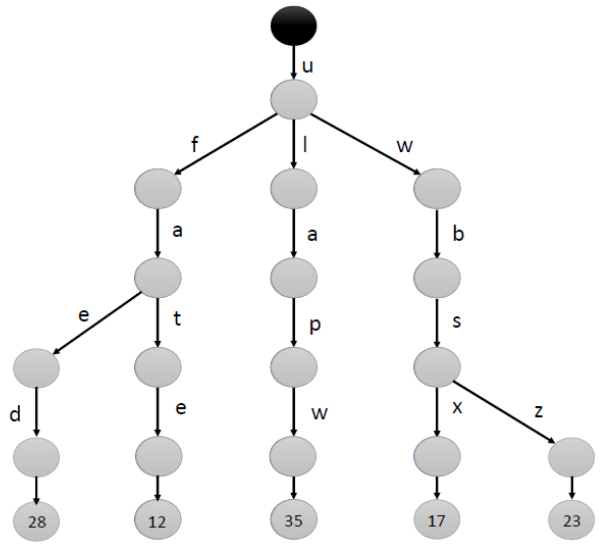

Abbildung 1: Beispiel eines Tries

Die Abbildung 1 repräsentiert ein graphisches Beispiel eines Tries, in welchem für die Zeichenketten „ufaed“, „ufate“, „ulapw“, „uwbsx“ und „uwbsz“ die jeweils erreichte Punktzahl 28, 12, 35, 17, 23 eingetragen wurden.

C Interaktive Benutzerschnittstelle

Ihr Programm arbeitet ausschließlich mit Befehlen, die nach dem Programmstart über die Konsole mittels `Terminal.readLine()` eingelesen werden. Nach Abarbeitung eines Befehls wartet das Programm auf weitere Befehle, bis das Programm durch die Eingabe des `quit`-Befehls beendet wird.

Beachten Sie, dass bei den folgenden Beispielen die Eingabezeilen mit dem `> `-Zeichen, gefolgt von einem Leerzeichen, eingeleitet werden. Diese beiden Zeichen sind ausdrücklich kein Bestandteil des eingegebenen Befehls, sondern dienen nur der Unterscheidung zwischen Ein- und Ausgabe.

Im Folgenden wird die Ausgabe für einen Erfolgsfall spezifiziert. Im Fehlerfall (z.B. bei falsch spezifizierten Eingaben) muss eine aussagekräftige Fehlermeldung beginnend mit `Error, ` ausgegeben werden.

C.1 Der `create`-Befehl

Mit dem `create`-Befehl legt man einen neuen Trie für eine Lehrveranstaltung an.
Eingabeformat:

```
create <Lehrveranstaltung>
```

`<Lehrveranstaltung>` ist ein beliebiger Kleinbuchstaben-`String` [a-z]+.
Ausgabeformat:

```
OK
```

`OK` lautet das Ausgabeformat dieses Befehls. Im Fehlerfall (z.B. wenn das hier spezifizierte Eingabeformat

nicht eingehalten wird oder der Trie für diese Lehrveranstaltung bereits existiert) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

C.2 Der `reset`-Befehl

Mit dem `reset`-Befehl wird der Trie für eine angelegte Lehrveranstaltung neu initialisiert.
Eingabeformat:

```
reset <Lehrveranstaltung>
```

`<Lehrveranstaltung>` ist ein Kleinbuchstaben-`String` [a-z]+.
Ausgabeformat:

```
OK
```

`OK` lautet das Ausgabeformat dieses Befehls. Im Fehlerfall (z.B. wenn das hier spezifizierte Eingabeformat nicht eingehalten wird oder der Trie für diese Lehrveranstaltung nicht existiert) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

C.3 Der `add`-Befehl

Der `add`-Befehl fügt für eine Lehrveranstaltung `<Lehrveranstaltung>` die ganzzahlige Punkte `<Punkte>` des Studenten `<Name>` in Form des SCC-Kürzels *uxxxx* in den Trie ein.
Eingabeformat:

```
add <Lehrveranstaltung>;<Name>;<Punkte>
```

`<Lehrveranstaltung>` und `<Name>` (in Form von *uxxxx*) sind jeweils ein Kleinbuchstaben-`String` [a-z]. `<Punkte>` ist eine positive `Integer`-Zahl inklusive 0. Alle Eingaben werden jeweils durch genau ein Semikolon separiert.
Ausgabeformat:

```
OK
```

`OK` lautet das Ausgabeformat dieses Befehls. Im Fehlerfall (z.B. wenn das hier spezifizierte Eingabeformat nicht eingehalten wird, die Lehrveranstaltung noch nicht angelegt wurde oder der Name nicht den hier angegebenen Konventionen entspricht) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

C.4 Der `modify`-Befehl

Der `modify`-Befehl ändert die bereits eingepflegten Punkte eines Studenten mit seinem SCC-Kürzel `<Name>` für eine bestimmte `<Lehrveranstaltung>`.
Eingabeformat:

```
modify <Lehrveranstaltung>;<Name>;<Punkte>
```

`<Lehrveranstaltung>` und `<Name>` sind ein Kleinbuchstaben-`String` [a-z]. `<Punkte>` ist die neue Punktzahl des Studenten. Sie repräsentiert eine positive `Integer`-Zahl inklusive 0. Alle Eingaben werden jeweils durch genau ein Semikolon separiert.

Ausgabeformat:

```
OK
```

`OK` lautet das Ausgabeformat dieses Befehls. Im Fehlerfall (z.B. wenn das hier spezifizierte Eingabeformat nicht eingehalten wird, die Lehrveranstaltung nicht existiert oder Student in dem Trie noch nicht gespeichert ist) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

C.5 Der `delete`-Befehl

Mit dem `delete`-Befehl löscht man einen Studenten mit `<Name>` aus dem Trie der angegebenen Lehrveranstaltung.

Eingabeformat:

```
delete <Lehrveranstaltung>;<Name>
```

`<Lehrveranstaltung>` ist ein beliebiger Kleinbuchstaben-`String` mit [a-z]+. Alle Eingaben werden jeweils durch genau ein Semikolon separiert.

Ausgabeformat:

```
OK
```

`OK` lautet das Ausgabeformat dieses Befehls. Im Fehlerfall (z.B. wenn das hier spezifizierte Eingabeformat nicht eingehalten wird, die Lehrveranstaltung nicht existiert oder der Student nicht im Trie gespeichert ist) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

C.6 Der `credits`-Befehl

Mit dem `credits`-Befehl kann die Punktzahl eines Studenten mit seinem SCC-Kürzel `<Name>` für eine Lehrveranstaltung `<Lehrveranstaltung>` in Erfahrung gebracht werden.

Eingabeformat:

```
credits <Lehrveranstaltung>;<Name>
```

`<Lehrveranstaltung>` und `<Name>` sind jeweils ein Kleinbuchstaben-`String` [a-z]. Alle Eingaben werden jeweils durch genau ein Semikolon separiert.

Ausgabeformat:

```
<Punkte>
```

Die Ausgabe `<Punkte>` ist in dem Fall die gespeicherten Punkte `<Punkte>` eines Studenten für eine bestimmte Lehrveranstaltung. Im Fehlerfall (z.B. wenn das hier spezifizierte Eingabeformat nicht eingehalten wird, die Lehrveranstaltung nicht existiert, die Punkte des Studenten im Trie nicht eingetragen sind) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

C.7 Der `print`-Befehl

Mit dem `print`-Befehl gibt man den Inhalt des Tries für eine bestimmte Lehrveranstaltung aus.
`<Lehrveranstaltung>` ist ein beliebiger Kleinbuchstaben-`String` [a-z]+.

Eingabeformat:

```
print <Lehrveranstaltung>
```

Die Ausgabe erfolgt nach einem hier geforderten Format:

- Die textuelle Ausgabe des Tries erfolgt in einer einzigen Zeile.
- Abgesehen vom Wurzelknoten hat jeder Knoten im Trie einen Vaterknoten. Die Kante über die der Knoten von seinem Vaterknoten aus erreichbar ist, entspricht einem Zeichen im Alphabet. Dieses Zeichen wird als das dem Knoten zugeordnete Zeichen betrachtet. Das zugeordnete Zeichen des Wurzelknotens ist `#`.
- Bei der Ausgabe eines Knotens wird zuerst das ihm zugeordnete Zeichen ausgegeben.
- Enthält ein Knoten (Blattknoten) einen Punktewert, so wird dieser ohne Leerzeichen in runden Klammern bei der Ausgabe angefügt. Ansonsten wird nichts Weiteres angefügt.
- Falls ein Knoten Kindknoten hat, so werden diese der alphabetischen Reihenfolge ihrer zugeordneten Zeichen entsprechend ('a' bis 'z', dann '0' bis '9') in eckigen Klammern der Ausgabe angefügt.

Beispielablauf

```
> print programmieren
#[u[e[w[x[v(32)]]]h[f[z[i(20)]]]w[y[o[p(37)]]]]]
```

Im Fehlerfall (z.B. wenn das hier spezifizierte Eingabeformat nicht eingehalten wird oder die Lehrveranstaltung nicht existiert) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

C.8 Der `average`-Befehl

Mit dem `average`-Befehl wird der Durchschnittspunktewert für eine Lehrveranstaltung ermittelt.

Eingabeformat:

```
average <Lehrveranstaltung>
```

`<Lehrveranstaltung>` ist ein beliebiger Kleinbuchstaben-`String` [a-z]+.

Ausgabeformat:

```
<Durchschnittspunkte>
```

Die Ausgabe besteht aus dem Punktedurchschnitt `<Durchschnittspunkte>`, der sich aus einer Division der Gesamtpunktzahl und der Anzahl der Studierenden der Lehrveranstaltung ergibt.

Im Fehlerfall (z.B. wenn das hier spezifizierte Eingabeformat nicht eingehalten wird, die Lehrveranstaltung nicht existiert oder noch keine Punktzahl eingetragen ist) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

C.9 Der `median`-Befehl

Mit dem `median`-Befehl wird der Zentralwert hinsichtlich der Punktzahl für eine Lehrveranstaltung ermittelt.
Eingabeformat:

```
median <Lehrveranstaltung>
```

`<Lehrveranstaltung>` ist ein beliebiger Kleinbuchstaben-`String` [a-z]+.
Ausgabeformat:

```
<Zentralwert>
```

Der Median `<Zentralwert>` ist in dem Fall eine positive Zahl inklusive 0. Werden die Werte für die Punkte in aufsteigender Reihenfolge sortiert, so kann der Median $\tilde{x}$ folgendermaßen berechnet werden:

$$\tilde{x} = \begin{cases} x_{\frac{n+1}{2}} & \text{für n ungerade} \\ \frac{1}{2} \cdot \left(x_{\frac{n}{2}} + x_{\frac{n}{2}+1}\right) & \text{für n gerade} \end{cases}$$

Beispiel: Die Punkteverteilung für fünf Studenten einer Lehrveranstaltung verhält sich wie folgt: 5, 11, 17, 3, 6. Die mittlere Zahl in 3, 5, 6, 11, 17 ist der `<Zentralwert>`. Daher ist in diesem Fall der Wert 6 der Median.

Im Fehlerfall (z.B. wenn das hier spezifizierte Eingabeformat nicht eingehalten wird, die Lehrveranstaltung nicht existiert oder noch keine Punktzahl eingetragen ist) wird eine aussagekräftige Fehlermeldung beginnend mit `Error,` ausgegeben.

C.10 Der `quit`-Befehl

Dieser Befehl beendet das Programm. Dabei findet keine Konsolenausgabe statt.
Hinweis: Denken Sie daran, dass hierfür keine Methoden wie `System.exit` oder `Runtime.exit` verwendet werden dürfen.

D Beispielinteraktion

Beachten Sie im Folgenden, dass Eingabezeilen mit dem `>`-Zeichen, gefolgt von einem Leerzeichen, eingeleitet werden. Diese beiden Zeichen sind ausdrücklich kein Bestandteil des eingegebenen Befehls, sondern dienen nur der Unterscheidung zwischen Ein- und Ausgabe.

Einfacher Beispielablauf

```
> create programmieren
OK
> print programmieren
#
> add programmieren;ufate;12
OK
> add programmieren;ulapw;35
OK
> credits programmieren;ulapw
35
> print programmieren
#[u[f[a[t[e(12)]]]]l[a[p[w(35)]]]]]
> modify programmieren;ulapw;0
OK
> average programmieren
6
> reset programmieren
OK
> print programmieren
#
> quit
```