TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einführung in die Informatik 2 für Ingenieure (MSE)*

Alexander van Renen (renen@in.tum.de)
`http://db.in.tum.de/teaching/ss16/ei2/`

Blatt Nr. 5

Dieses Blatt wird am Montag, den 23. Mai 2016 besprochen.

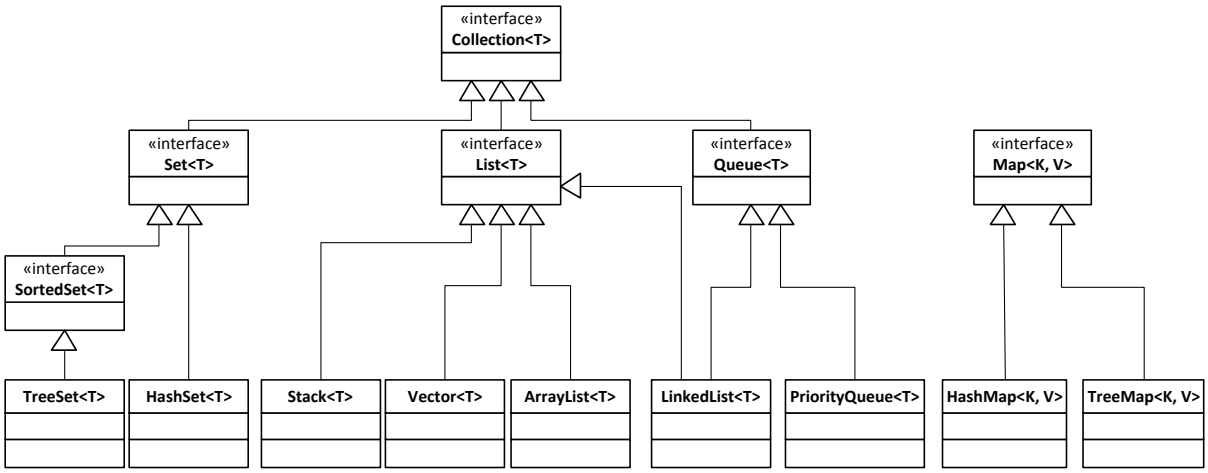

Abbildung 1: Ausschnitt aus dem Java Collections Framework (JCF)

Aufgabe 1: Java Collection Framework (JCF)

Das Java Collections Framework stellt viele häufig benötigte Datenstrukturen zur Verfügung, so dass Sie das Rad nicht neu erfinden müssen. Sie finden eine gute Einführung zu den Collections unter `http://docs.oracle.com/javase/tutorial/collections/`. In dieser Aufgabe geht es darum für verschiedene Anwendungsbeispiele jeweils die richtige Datenstruktur auszuwählen.

In dieser Aufgabe geht es um die Verwaltung von Filmen.[1] Auf der Webseite zur Vorlesung finden Sie vorbereitete Dateien. Sie sollen jeweils die effizienteste Datenstruktur wählen, wobei wir uns auf `PriorityQueue`, `HashMap`, `TreeMap` und `ArrayList` beschränken. Begründen Sie, welche Datenstruktur Sie gewählt haben und welche Nachteile die anderen gehabt hätten.

a) Die zu implementierende Klasse `MovieRangeSearch` soll Filme nach ihrem Erscheinungsjahr verwalten. Die Methode `getMoviesBetweenYears(int fromYear, int toYear)` soll alle Filme zurückgeben, die in einem bestimmten Zeitraum erschienen sind.

b) Die Klasse `MovieTitleSearch` verwaltet Filme nach ihrem Namen. Dabei gibt die Methode `getMovieWithTitle(String title)` den Film mit dem angegebenem Titel zurück.

[1] Basierend auf der Liste der Top 250 Filme von imdb.com: `http://www.imdb.com/chart/top`

c) Die Klasse `MovieVoteSearch` verwaltet Filme nach abgegebenen Bewertungen. Dabei soll die Methode `getMovieWithFewestVotes()` den Film mit den wenigsten Stimmen zurückgeben.

d) Die Klasse `MovieRankSearch` verwaltet Filme anhand ihrer Platzierung in der Top 250 Liste von IMDB. Die Methode `getMovieForRank(int rank)` gibt für eine Position den Film zurück.

Aufgabe 2: Generics

In dieser Aufgabe wollen wir einen generischen binären Suchbaum implementieren. Falls Sie sich noch unsicher mit Generics fühlen, können sie zunächst einen binären Suchbaum implementieren der lediglich Integer unterstützt und diesen dann in einem zweitern Schritt erweitern. Testen Sie Ihre Implmentierung in jedem Fall mit geeigeneten Beispielen.

1. Implementieren Sie einen binären Suchbaum,[2] der beliebige Elemente enthalten kann. Jeder Knoten sollte je einen Zeiger auf das linke und das rechte Kind haben. Der Baum sollte Methoden zum Einfügen und Suchen von Elementen anbieten. Wenn Sie eine Herausforderung suchen, können Sie auch noch Löschen implementieren.

2. Welches Problem gibt es, wenn man nacheinander die Zahlen $1, 2, 3, ..., 1.000.000$ in aufsteigender Reihenfolge einfügt?

Aufgabe 3: For Each Loops

Das folgende Programm soll alle Primzahlen im Intervall $[0, 10.000]$ bestimmen.[3] Leider wirft das Programm bei der Ausführung eine *Exception*. Wo und warum? Was ist eine mögliche Lösung?

```
1  import java.util.ArrayList;
2
3  public class Eratosthenes {
4
5     public static void main(String[] args) {
6        // Zahlen 2..10000 einfuegen
7        ArrayList<Integer> list = new ArrayList<Integer>();
8        for (int i = 2; i <= 10000; i++) {
9           list.add(i);
10       }
11       for (Integer zahl : list) {
12          // Probiere alle vorherigen Primzahlen als Teiler
13          for (Integer teiler : list) {
14             // Wenn der zu pruefende Teiler schon groesser als
15             // die Quadratwurzel ist , muss es eine Primzahl sein
16             if (teiler > Math.sqrt(zahl)) break;
17             // Zahlen mit Teiler sind keine Primzahlen und
18             // werden daher hier entfernt
19             if (zahl % teiler == 0) {
20                list.remove(zahl);
21                break;
```

[2] Falls Ihnen entfallen ist, was ein Binärbaum ist: `http://de.wikipedia.org/wiki/Binärer_Suchbaum`
[3] Basierend auf dem Sieb des Eratosthenes: `http://de.wikipedia.org/wiki/Sieb_des_Eratosthenes`

```
22                  }
23              }
24          }
25          // Alle verbliebenen Zahlen sind Primzahlen
26          for (Integer primzahl : list)
27              System.out.println(primzahl);
28      }
29  }
```