Einführung in die Programmierung

Präsenzpraktikum

Das Java Collections Framework

Aufgabe 1 (Algorithmen, Sortieren)

Schreiben Sie ein Java-Programm, das seine Argumente in lexikographischer (alphabetischer) Reihenfolge ausgibt. Sie können die folgende Datei verwenden

```
./PROG/collections/Sort.java
```

Aufgabe 2 (Algorithmen, zufällige Ausgabe)

Schreiben Sie ein Java-Programm, das seine Argumente in zufälliger Reihenfolge ausgibt. Sie können die folgende Datei verwenden

```
./PROG/collections/Shuffle.java
```

Aufgabe 3 (Die Schnittstelle `Set`)

Schreiben Sie ein Java-Programm, das seine Argumente durchsucht und alle mehrfach vorkommenden Argumente ausgibt. Außerdem soll die Zahl der unterschiedlichen Argumente sowie eine Liste der Argumente ausgegeben werden, die keine Duplikate mehr enthält. Verwenden Sie dazu ein Objekt der Klasse `HashSet`. Sie können die folgende Datei verwenden

```
./PROG/collections/FindDups1.java
```

Verwenden Sie nun statt eines `HashSet`-Objekts ein Objekt der Klasse `TreeSet`. Ändert sich die Ausgabe Ihres Programms?

Benutzen Sie nun die Schnittstelle `SortedSet` statt `Set` und die Klasse `TreeSet`. Definieren Sie einen `Comparator`, der beim Vergleichen Groß- und Kleinschreibung ignoriert.

Verändern Sie Ihr Programm, so dass es zwei Mengen verwaltet, eine, die alle unterschiedlichen Worte der Argumentliste enthält, und eine, die nur die Duplikate enthält. Die Differenz dieser beiden Mengen ergibt diejenigen Worte, die genau einmal in der Argumentliste enthalten sind.

Aufgabe 4 (Modifizieren einer Liste, `ListIterator`)

Schreiben Sie eine Methode, die als Argument ein Objekt vom Typ `List<String>` erhält und auf jedes Element der Liste die Methode `String.trim` anwendet. Die Listenelemente sollen also beim Durchlaufen der Liste verändert werden. Weder mit der for-each-Schleife noch mit einem `Iterator` können Sie das gewünschte Ergebnis erzielen. Verwenden Sie daher einen `ListIterator` zum Durchlaufen der Liste. Sie können die folgende Datei verwenden

```
./PROG/collections/Trim.java
```

Aufgabe 5 (Die Klasse `LinkedList`)

Die Dateien

```
./PROG/collections/ListeTest.java
./PROG/collections/ListeTestZwei.java
```

enthalten Testprogramme für die Implementierung der Klasse `Liste`. Verwenden Sie in diesen Programmen, statt der Objekte vom Typ `Liste`, Objekte vom Typ `LinkedList`. Versuchen Sie, statt der Methoden der Klasse `Liste` ausschließlich Methoden der Klasse `LinkedList` zu verwenden.

Aufgabe 6 (Die Schnittstelle `Queue`)

Die Datei

```
./PROG/collections/SchlangeTest.java
```

enthält ein Testprogramm für die Implementierung der Klasse `SchlangeMitListe`. Verwenden Sie in diesem Programm, statt des Objekts vom Typ `SchlangeMitListe`, ein Objekt vom Typ `LinkedList`. Benutzen Sie statt der Methoden der Klasse `SchlangeMitListe` ausschließlich Methoden der Schnittstelle `Queue`.

Aufgabe 7 (Vergleichen von Objekten)

Das folgende Java-Programm sollte eigentlich `"Blue"` ausgeben.

```
public class SortMe
{
    public static void main(String args[]) {
        SortedSet<StringBuffer> s = new TreeSet<StringBuffer>();
        s.add(new StringBuffer("Red"));
        s.add(new StringBuffer("White"));
        s.add(new StringBuffer("Blue"));
        System.out.println(s.first());
    }
}
```

Stattdessen wird eine Exception ausgeworfen. Woran liegt das? Wie können Sie das gewünschte Verhalten erreichen? Siehe die Datei

```
./PROG/collections/SortMe.java
```

Aufgabe 8 (Die Schnittstellen `Comparable` und `Comparator`)

Bitte lesen Sie die Datei

```
./PROG/collections/ObjectOrdering.html
```

und versuchen Sie, den darin enthaltenen Programmcode nachzuvollziehen.