Universität Potsdam
Institut für Informatik

Praxis der Programmierung

11. Aufgabenblatt

1. Fügen Sie folgende Strings in dieser Reihenfolge nacheinander in ein `TreeSet` ein:
   Aal
   Tanne
   Birke
   Baum
2. Erzeugen Sie einen Iterator für Ihre Menge.
3. Durchlaufen Sie mit diesem Iterator die Menge vollständig und geben Sie die Elemente dabei auf die Standardausgabe aus.
   In welcher Reihenfolge werden die Elemente ausgegeben?

---

4. Ändern Sie nun den Typ Ihrer Menge in `HashSet`. Bauen Sie die Menge aber in der gleichen Weise auf, wie vorher mit dem `TreeSet`.
5. Erzeugen Sie wieder einen Iterator für Ihre Menge, durchlaufen Sie mit diesem Iterator die Menge vollständig und geben Sie die Elemente dabei auf die Standardausgabe aus.
   In welcher Reihenfolge werden die Elemente diesmal ausgegeben?

---

6. Schreiben Sie eine generische Klasse `Pair` mit zwei Typvariablen, die geordnete Paare von Zahlen (beliebigen Typs) repräsentiert.
   Hinweis: Nutzen Sie Typebounds.
   Die Klasse soll einen Initialisierungskonstruktor und Getter für beide Komponenten enthalten. Erzeugen Sie einen generischen Typ, wobei beide Komponenten ganze Zahlen speichern. Testen Sie mit einer kleinen Applikation.
7. Klassen, die das Interface `java.lang.Comparable<T>` implementieren, definieren eine Ordnungsrelation für ihre Exemplare. Beschäftigen Sie sich mit der Dokumentation dieses Interface und implementieren Sie es in `Pair`, so dass Paare nach der Größe des ersten Elements geordnet werden.
8. Erzeugen Sie in einer Applikation eine Liste von Paaren ganzer Zahlen und lassen Sie sie mit einer `foreach`-Schelife ausgeben.
9. Sortieren Sie jetzt die Liste mit der in Kollektionen dazu vorhandenen Methode. Lassen Sie die sortierte Liste zur Kontrolle ausgeben.
10. Ändern Sie jetzt die Definition der `compareTo`-Methode so ab, dass die Paare richtig sortiert sind, wenn sie als Brüche interpretiert werden, wobei die erste Komponente des Paars der Zähler und die zweite der Nenner eines Bruchs ist.