Aufgabe 8.1

a) Definieren Sie eine Klasse Rechteck, die Rechtecke über ihre Länge und Breite definiert. Die Klasse soll über zwei Methoden umfang() und flaeche () verfügen, die Umfang bzw Fläche des Rechtecks liefern.

(Für alle, die das in der Schule verschlafen haben: der Umfang u eines Rechtecks berechnet sich nach $u = 2 \cdot (laenge + breite)$, die Fläche f ist $f = laenge \cdot breite$.)

Definieren Sie zwei verschiedene Komparatoren für Rechtecke: einen, der die Rechtecke nach ihrem Umfang vergleicht und einen, der sie nach Fläche vergleicht.

b) Definieren Sie in einer Util-Klasse eine statische Methode min( ), die eine Folge von Rechtecken *und* einen Komparator für Rechtecke als Eingabe erhält und das (im Sinne des Komparators) kleinste Rechteck der Folge liefert.

Nutzen Sie zum Testen eine der Implementierungen von Folge<T> aus der Vorlesung.

Aufgabe 8.2

a) Modifizieren Sie die unten angegebene (abgespeckte Version der) Klasse Stud so, dass die Klasse das Interface Comparable<Stud> implementiert.
Stud-Objekte sollen anhand ihrer Matrikelnummer verglichen werden.

b) Schreiben Sie JUnit-Tests um Vergleiche von Stud-Objekten zu testen.

```
public class Stud {

        private static int nextNr = 100;

        private String name;
        private int matNr;

        public Stud(String name) {
                this.name = name;
                this.matNr = nextNr++;
        }

        public String name() {
                return name;
        }

        public int matNr() {
                return matNr;
        }

        public String toString() {
                return name + "(" + matNr + ")";
        }
}
```

Aufgabe 8.3
Implementieren Sie in der Klasse StudUtil eine statische Methode

public static Folge<Stud> sort(Menge<Stud> m)

die eine Menge von Stud-Objekten als Eingabe erhält und eine Folge von Stud-Objekten zurückliefert. In der Folge sollen die Stud-Objekte der Menge m nach Matrikel-Nummern sortiert sein.

Als Sortier-Algorithmus bietet sich „Sortieren durch Einfügen“ (insertion sort) an:

- füge das erste Element der Menge m in die Folge an Position 0 ein
- wiederhole, solange m noch weitere Elemente enthält:
  - nimm das nächste Element aus m
  - füge es in der Folge an der richtigen Position ein, sodass die entstehende „Teilfolge“ nach jedem Einfügen sortiert ist

Eine Test-main finden Sie in der Datei StudUtil.java.

Aufgabe 8.4
Definieren Sie einen weiteren Komparator für Personen, der Personen zuerst nach Namen vergleicht, und falls der Name zweier Objekte gleich ist, als zweites Kriterium das Geburtsjahr heranzieht.
Testen Sie auch diesen Komparator.

Aufgabe 8.5
Aus der Vorlesung zu „Einführung in die Programmierung“ kennen Sie vielleicht schon das Bubble-Sort-Verfahren, mit dem man ein Array von int-Werten (aufsteigend) sortieren kann:

```
public static void bubbleSort(int[] a) {
  for (int n = a.length; n > 1; n--)
     for (int i = 0; i < n-1; i++)
        if (a[i] > a[i+1])
        // tausche die Eintraege an den Positionen i und i+1
           swap(a, i, i+1);
}
```

a) Implementieren Sie das Sortierverfahren BubbleSort nun als generische Methode, die nicht ein Array, sondern eine (beliebig implementierte Folge) von Objekten eines beliebigen Typs erhält. Die Methode soll als Sortier-Kriterium einen externen Comparator verwenden.

b) Testen Sie die Methode mit einer Folge von Personen und den verschiedenen Comparatoren für Personen.

   Wie verhält sich die Methode bubbleSort(Folge<T> a, Comparator<T> c), wenn in der Personen-Folge mehrere Personen gleichen Namens (aber unterschiedlichem Geburtsjahr), bzw mehrere Personen mit gleichem Geburtsjahr (aber mit unterschiedlichen Namen) enthalten sind?
   Wie verhält sich die Methode, wenn Personen mit gleichem Alter **und** gleichem Namen enthalten sind?

c) Implementieren Sie eine weitere generische Variante von BubbleSort, die die *innere Ordnung* des Datentyps verwendet und dabei garantiert, dass die Methode nur für solche Typen aufgerufen wird, die über eine innere Ordnung verfügen!

   Testen Sie diese Methode mit einer Folge von Stud-Objekten.