Name:

Matrikelnummer:

Bearbeitungszeit: 120 min.

Aufgabe 1. Gegeben ist das Interface

```
interface GeomFigure {
        double getFlaeche();
}
```

das eine 2-dimensionale geometrische Figur darstellt. Die Methode `getFlaeche()` liefert den Flächeninhalt der Figur zurück.

Schreiben Sie eine Klasse

```
class CompGeomFigure implements Comparator<GeomFigure>
```

die es mittels der Methode

```
public int compare(GeomFigure fig1, GeomFigure fig2)
```

erlaubt, zwei geometrischen Figuren bez. ihres Flächeninhalts zu vergleichen. Dabei liefert `compare(fig1,fig2)` den Wert -1 wenn `fig1` den kleineren Flächeninhalt hat, 0 wenn die Flächeninhalte gleich sind, und $+1$ wenn `fig1` den größeren Flächeninhalt hat.

Implementieren Sie auch die Klasse `Rechteck`, die das Interface `GeomFigure` implementiert. Die Klasse soll über den Konstruktor

```
Rechteck(double a, double b)
```

verfügen, der ein Rechteck mit den Seitenlängen `a` und `b` erzeugt. Weiters soll die Klasse `Rechteck` über die statische Methode

```
static void sortRechteckeAbsteigend(List<Rechteck> rechtecke)
```

verfügen, die die Rechtecke in der Liste so umsortiert, dass sie nach ihrem Flächeninhalt absteigend geordnet sind.

Hinweis: Zum Sortieren könnten Sie die statische Methode

```
static void sort(List<T> liste, Comparator<T> comp)
```

aus der Klasse `java.util.Collections` verwenden, die aufsteigend(!) sortiert.

Aufgabe 2. Schreiben Sie eine Klasse `Journaldienst`, die die Dienstzeiten von Mitarbeitern verwaltet. Der Konstruktor

```
Journaldienst(double[] von, double[] bis)
```

legt die Dienstzeiten von Mitarbeitern jeweils als `von[i]`–`bis[i]` fest, wobei `i=0,...,von.length-1` und `von.length==bis.length`. Sie können davon ausgehen, dass die Beginnzeiten der Dienst (im `von`-Array) bereits aufsteigend sortiert sind.

Die Methode

```
boolean dienstOK(double t)
```

soll überprüfen, ob zu jedem Zeitpunkt x, $0 \leq x \leq$ `t`, zumindest ein Mitarbeiter Dienst hat. In diesem Fall soll die Methode `true` zurück liefern, andernfalls `false`.

Beispiel: Mit den Dienstzeiten

`von`	0	2	3	7
`bis`	5	7	4	10

ist bis zur Zeit `t` $= 10$ zu jedem Zeitpunkt zumindest ein Mitarbeiter anwesend, während mit den Dienstzeiten

`von`	0	2	3	7
`bis`	5	6	4	10

zur Zeit $x = 6.5$ kein Mitarbeiter anwesend ist.

Aufgabe 3. Für diese Aufgabe werden Dominosteine mit ihren zwei Augenzahlen als `int`-Arrays der Länge 2 dargestellt. Schreiben Sie eine Klasse `Domino` mit der statischen Methode

```
static void vertausche(int[] stein),
```

die die Augenzahlen eines Dominosteins vertauscht, sodass aus `{a1,a2}` der Stein `{a2,a1}` wird.

Schreiben Sie weiters die Methode

```
List<int[]> getAnordnung(List<int[]> dominosteine, int n),
```

die als `ergebnis` eine Anordnung einer Auswahl der übergebenen Dominosteine zurückgibt, die folgende Bedingungen erfüllt:

1. für `i=1,...,ergebnis.size()-1` gilt, dass `ergebnis.get(i-1)[1]==ergebnis.get(i)[0]`,
2. jede Augenzahl $k = 0, \ldots,$ `n` $- 1$ kommt auf genau zwei der ausgewählten Dominosteine vor, und andere Augenzahlen kommen auf den ausgewählten Steinen nicht vor.

Es ist erlaubt, die Augenzahlen eines Dominosteins zu vertauschen. Das kann notwendig sein, um die erste Bedingung zu erfüllen.

Wenn es keine solche Anordnung gibt, soll die Methode `null` zurück liefern.

Beispiel: Für die Steine `{1,2},{2,3},{3,1},{3,4},{4,2},{1,0},{3,0},{4,0}` und `n` $= 5$ ist `{0,3},{3,4},{4,2},{2,1},{1,0}` ein korrektes Ergebnis, während es für die Steine `{1,2},{2,3},{3,1},{3,4},{3,0},{4,0}` keine korrekte Anordnung gibt.