Name:

Matrikelnummer:

Bearbeitungszeit: 120 min.

Aufgabe 1.

Eine 2-dimensionale geometrische Figur wird durch folgende abstrakte Klasse dargestellt:

```
public abstract class Figur2D
{
   public abstract void verschiebe(double x, double y);
}
```

Die Methode `verschiebe(x,y)` soll die Figur um die angegebenen Werte in x- bzw. y-Richtung verschieben.

Schreiben Sie eine Unterklasse `Komposition` von `Figur2D` mit dem Konstruktor

```
public Komposition(Figur2D[] bestandteile),
```

die eine Figur darstellt, die aus mehreren Teilen (die selbst Figuren sind) zusammengesetzt ist. Implementieren Sie auch die Methode

```
public void verschiebe(double x, double y)
```

entsprechend.

Aufgabe 2.

Gegeben ist die Klasse `Ort` mit der Methode

```
public double distZu(Ort andererOrt),
```

sodass `ort1.distZu(ort2)` den Abstand von `ort1` zu `ort2` zurück gibt. (ACHTUNG: Im allgemeinen ist `ort1.distZu(ort2) != ort2.distZu(ort1)`.)

Schreiben Sie eine Klasse `Auswahl` mit einem parameterlosen Konstruktor und der Methode

```
public List<Ort> getAuswahl(List<Ort> ortsliste,
                            double minDist, double maxDist),
```

die aus der Ortsliste Orte so auswählt, dass

1. es zu jedem Ort `einOrt` in der Ortsliste einen ausgewählten Ort `ort1` gibt mit `einOrt.distZu(ort1) <= maxDist`,
2. für jedes Paar ausgewählter Orte `ort1`, `ort2`, der Abstand `ort1.distZu(ort2) >= minDist` ist.

Der Abstand zwischen ausgewählten Orten soll also zumindest `minDist` sein, und zu jedem nicht ausgewählten Ort muss es im Abstand `maxDist` einen ausgewählten Ort geben.

Ist eine derartige Auswahl nicht möglich, dann soll die Methode `null` zurück geben.

Aufgabe 3.
Gegeben ist die Klasse `Auftrag` mit dem Konstruktor

```
public Auftrag(int ersterTag, int letzterTag, int anzahlTage),
```

der einen Auftrag erzeugt, der im Zeitraum von `ersterTag` bis `letzterTag` (jeweils inklusive) bearbeitet werden kann, und der insgesamt `anzahlTage` für seine Bearbeitung benötigt. Tage werden als ganzzahlige Werte durchnummeriert, und an einem Tag kann immer nur ein Auftrag bearbeitet werden. Ein Auftrag kann an mehreren Tagen bearbeitet werden, die nicht direkt aufeinander folgen müssen. Die Klasse `Auftrag` verfügt auch noch über die get-Methoden

```
public int ersterTag(),
public int letzterTag(),
public int anzTage(),
```

die die entsprechenden Werte zurück geben.

Weiters gegeben ist die Klasse `Scheduling` mit einem parameterlosen Konstruktor und der Methode

```
public boolean istMoeglich(List<Auftrag> auftragsliste),
```

die genau dann `true` zurück liefert, wenn die rechtzeitige Fertigstellung aller Aufträge in der Liste möglich ist.

Schreiben Sie eine Unterklasse `SchedulingTage` mit der Methode

```
public List<Auftrag> getMoegliche(List<Auftrag> auftragsliste,
                                  int tag),
```

die eine Liste jener Aufträge zurück gibt, die am angegebenen Tag bearbeitet werden können, sodass eine Fertigstellung aller Aufträge in der Auftragsliste gewährleistet ist. Wenn die Fertigstellung der Aufträge überhaupt nicht möglich ist, dann soll die Methode `null` zurück geben.

Beispiel: Für die Aufträge

```
Auftrag a1 = new Auftrag(11,20,3);
Auftrag a2 = new Auftrag(11,17,5);
Auftrag a3 = new Auftrag(11,15,2);
```

sind an den Tagen 11–15 jeweils die Aufträge `a2` und `a3` möglich, an den Tagen 16–17 nur der Auftrag `a2`, und an den Tagen 18–20 nur der Auftrag `a1`.

Beispiel: Für die Aufträge

```
Auftrag a1 = new Auftrag(11,20,3);
Auftrag a2 = new Auftrag(11,16,5);
Auftrag a3 = new Auftrag(11,15,2);
```

ist die Fertigstellung nicht möglich, und `getMoegliche()` soll `null` zurück geben.

Hinweis: Überlegen Sie, wie Sie durch Hinzufügen eines weiteren Auftrags einen Tag sperren können. Dann kann durch Modifikation der ursprünglichen Aufträge ausprobiert werden, welche Aufträge am gegebenen Tag bearbeitet werden können.