Name:

Matrikelnummer:

Bearbeitungszeit: 120 min.

Aufgabe 1

Gegeben ist die Klasse

```
public class Element implements Comparable<Element>.
```

Schreiben Sie eine Klasse `Sammlung`, die solche Elemente verwaltet. Die Klasse `Sammlung` soll über einen parameterlosen Konstruktor verfügen, der eine leere Sammlung anlegt. Weiters soll die Klasse `Sammlung` die folgenden Methoden besitzen:

```
boolean add(Element el);
Element getSortiert(int n);
Element getReihenfolge(int n).
```

Die Methode `add(el)` fügt das Element `el` der Sammlung hinzu und liefert `true` zurück, wenn das Element noch nicht enthalten ist. Ansonsten liefert die Methode `false` zurück.
Hinweis: Zwei Elemente `el1` und `el2` sind gleich, wenn `el1.compareTo(el2)==0`.

Die Methode `getSortiert(n)` liefert das `(n+1)`-kleinste Element aus der Sammlung zurück. Existiert ein solches Element nicht, dann liefert die Methode `null` zurück.
Beispiel: `getSortiert(2)` liefert das drittkleinste Element zurück.

Die Methode `getReihenfolge(n)` liefert das `(n+1)`-ste Element zurück, das zur Sammlung hinzugefügt wurde. Existiert ein solches Element nicht, dann liefert die Methode `null` zurück.
Beispiel: Nach

```
Sammlung s = new Sammlung();
s.add(el0); s.add(el1); s.add(el2); s.add(el3);
```

liefert `s.getReihenfolge(2)` das Element `el2` zurück.

Aufgabe 2

Schreiben Sie eine Klasse `PartialSum` mit der Methode

```
static int getBestSum(int[] a, int n),
```

die die maximale Summe von Zahlen im Array `a` zurück liefert, die in höchstens `n` nicht-überlappenden Teilfeldern von `a` liegen.

Beispiel: Für das Array `a` =

-15	20	-11	16	-13	17	-10	11	12	-12

soll `getBestSum(a,2)` den Wert 55 $(20 - 11 + 16 + 17 - 10 + 11 + 12)$ zurück liefern.

Hinweis: Diese Aufgabe kann relative einfach rekursiv oder auch iterativ gelöst werden.

Aufgabe 3

Ein quadratisches Gebäude mit den Eckpunkten `(0,0),(10,0),(0,10),(10,10)` soll auf einem flächenmäßig möglichst großen rechteckigen Grundstück ohne Baumbestand stehen. Eine Ecke des Grundstücks ist `(0,0)`, gesucht ist die gegenüber liegende Ecke `(gx,gy)`, sodass das Grundstück die Ecken `(0,0),(gx,0),(0,gy),(gx,gy)` besitzt. Der Baumbestand ist durch eine Liste von `(x,y)`-Koordinaten angegeben.

Schreiben Sie eine Klasse `Grundstueck` mit der Methode

```
static double[] getGxGy(ArrayList<double[]> baeume),
```

die ein Array der Länge 2 mit den Koordinaten `(gx,gy)` des optimalen Grundstücks zurück liefert. Die ArrayList `baeume` enthält die Koordinaten der Bäume ebenfalls jeweils als Array der Länge 2. Sie können annehmen, dass sich im Quadrat `(0,0),(10,0),(0,10),(10,10)` keine Bäume befinden.

Beispiel: In folgender Skizze ist das optimale Grundstück eingezeichnet.