technische universität
dortmund

Fakultät für Informatik
Lehrstühle 11, 13, 14

Jun.-Prof. Dr. Maike Buchin
Łukasz Czajka, N. Kriege, D. Kurz, M. Ludewig

Sommersemester 2019
16. April

DAP2 Praktikum – Blatt 4

Abgabe: 29. April–3. Mai

Wichtig: Am Mittwoch, dem 1. Mai, fallen die Praktika aufgrund eines Feiertags aus. Besuchen Sie bitte in dieser Woche ein alternatives Praktikum an einem anderen Tag.

Langaufgabe 4.1: Euklidischer Abstand (4 Punkte)

Diese Aufgabe besteht aus mehreren Teilen. Bitte lesen Sie sich die Aufgabe vorher komplett durch und überprüfen Sie nach der Bearbeitung, ob Sie nichts vergessen haben.

- Legen Sie eine Klasse `Point` an, die einen Punkt im $\mathbb{R}^d$ beschreiben soll. Auch negative Werte und 0 sind für die jeweiligen Koordinaten zulässig.
- Die Klasse `Point` soll einen Konstruktor bereitstellen, der die Dimension d (`int`) und die d Koordinaten (`double... values`) entsprechend definiert. Dieser Konstruktor soll eine `IllegalArgumentException` werfen, wenn fehlerhafte Parameter übergeben werden.
  Hinweis: Suchen Sie nach „Java Varargs“ in der Suchmaschine Ihrer Wahl.
- Schreiben Sie eine Methode `public double get(int i)` und die Methode `public int dim()`, die die jeweilige Koordinate bzw. die Dimension des Punktes zurückgibt. Für eine ausgiebige Implementierung eines Punktes bzw. Vektors siehe:
  `https://commons.apache.org/proper/commons-math/javadocs/api-3.0/org/apache/commons/math3/linear/RealVector.html`
- Legen Sie außerdem eine abstrakte Klasse `Simplex` an, die $d+1$ dieser Punkte enthält. Die Klasse soll einen Konstruktor bereitstellen, der die Dimension d (`int`) und die $d+1$ Punkte (`Point... points`) entsprechend definiert. Diese geben ihre Eckpunkte an. Sie sollten auch eine passende Get-Methode schreiben, um die jeweiligen Punkte zurückzugeben. Legen Sie eine abstrakte Methode `public abstract boolean validate()` an, welche überprüfen soll, ob die jeweilige Instanz einen validen Simplex darstellt.
- Implementieren Sie eine Klasse `Triangle`, die von `Simplex` erbt. Die `validate`-Methode soll überprüfen, ob 3 Punkte vorhanden sind und diese jeweils genau zwei Koordinaten haben (d.h. $d = 2$).
- Schreiben Sie in die Klasse `Simplex` eine weitere Methode `public double perimeter()`, die die Summe der Seitenlängen des Simplex zurückgibt.
- Definieren Sie ein Interface `Distance` mit einer Methode `public double distance(Point p1, Point p2)`, welche den Abstand zweier Punkte berechnet. Implementieren Sie dieses Interface in der Klasse `EuclidDistance`, sodass die euklidische Distanz von der Methode `public double distance(Point p1, Point p2)` berechnet wird.

- Schreiben Sie eine Klasse `Application` und fügen Sie dieser eine `main`-Methode hinzu. Die `main` Methode soll entweder 6 Parameter annehmen, daraus ein Dreieck bauen und dessen Umfang dann ausgeben, oder ohne Parameter ein zufälliges Dreieck erstellen und dessen Umfang ausgeben. Die Koordinaten der Punkte dieser Dreiecke sollten sich aus Gründen der Nachvollziehbarkeit im Interval [-1000; 1000] befinden. Falls Sie unsicher sind, wie das funktioniert, lesen Sie noch einmal die Hinweise zu Zufallszahlen.
  Beispielaufruf: `java Application x1 y1 x2 y2 x3 y3`

Langaufgabe 4.2: Convex Hull (4 Punkte)

- Schreiben Sie eine Klasse `ConvexHull`, die für eine gegebene Menge an Punkten `P` im $\mathbb{R}^2$, die konvexe Hülle berechnet.
- Implementieren Sie die einfache iterative Methode, die in der Vorlesung als `SimpleConvexHull(P)` eingeführt wurde. Nennen Sie die Methode `public List<Point> simpleConvex(Point[] P)`.
- Es muss für je zwei Punkte aus `P` eine Gerade berechnet und entschieden werden, ob alle restlichen Punkte links oder rechts der Geraden liegen.
- Zum Schluss erzeugen Sie, wie in der Vorlesung beschreiben, eine verkettete Liste, die alle Punkte der konvexen Hülle enthält.
- Berechnen Sie die konvexe Hülle für 1000 zufällig erzeugte Punkte. Zum Testen dieser konvexen Hülle verwenden Sie ein Dreieck mit den Eckpunkten [(10,10), (10,100), (100,10)]. Die 1000 zufällig generierten Punkte sollen innerhalb dieses Dreiecks liegen.

Beachten Sie die Hinweise und Tipps auf der folgenden Seite.

Hinweise und Tipps

Zufallszahlen (noch einmal)

In Java steht ein Pseudozufallszahlengenerator zur Verfügung. Die Klasse `java.util.Random` stellt den Konstuktor für einen Pseudozufallszahlengenerator, sowie die Methode `nextDouble()` zur Verfügung.

Um Zufallszahlen zu erzeugen, kann wie folgt vorgegangen werden:

```
...
// Den Generator erzeugen (als Seed wird die Systemzeit verwendet)
java.util.Random generator = new java.util.Random();
...
// Wann immer man eine Zufallszahl zwischen 0 und 1 braucht:
double randomNumber = generator.nextDouble();
...
```

Um Zufallszahlen in bestimmte Bereiche einzugrenzen, geht man wie folgt vor:

```
double lowerBound = 3.14;
double upperBound = 9.65;
...
// Wann immer man eine Zufallszahl braucht:
double randomNumber = lowerBound
                    + (upperBound - lowerBound) * generator.nextDouble();
```

Damit werden Zufallszahlen zwischen `untereGrenze` (inklusive) und `obereGrenze` (exklusive) generiert. Für ganze Zahlen kann alternativ die Methode `nextInt(grenze)` verwendet werden, die eine ganze Zufallszahl zwischen 0 (inklusive) und `grenze` (exklusive) generiert.

Siehe auch: `http://docs.oracle.com/javase/8/docs/api/java/util/Random.html`