
Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing-Wiss. und Informatik
- Software-Entwicklung -

**Programmierung 1**

Wintersemester 2025/26
**Aufgabenblatt 11**

Die Online-Befragung zur genutzten alternativen Veranstaltungsform und zur Lehrevaluation ist online. Bitte ausfüllen: https://forms.gle/fZTBWPmUhK4oqLHw5. Sie werden eventuell aufgefordert sich bei Google anzumelden, das ist nur notwendig, wenn Sie in der Bearbeitung eine Pause machen und das Teilergebnis zwischenspeichern wollen. Die Befragung endet am 19.12., die Ergebnisse stehen in einem nachfolgenden Fragen&Antworten-Dokument auf der Webseite der Veranstaltung.

**0.11. Aufgabe (1 Punkt)**

Geben Sie die Lösungsworte der Quizze aus der Lernnotiz an.

**37. Aufgabe (11 Punkte, konsequente Anwendung dynamischer Polymorphie, VL 20)**

In dieser Aufgabe sollen gefüllte Figuren und Figuren, aus denen andere Figuren ausgeschnitten sind, gezeichnet werden. Zur Zeichnung gefüllter Figuren gibt es im Interaktionsbrett z. B. die Möglichkeit die einzelnen Punkte der jeweiligen Figur zu zeichnen.
Die Grundidee zum Darstellen komplexer, teilweise ausgefüllter Figuren wird in Abb. 1 skizziert. Gegeben ist eine Figur f, dann wird für jeden Punkt p, der zur Figur gehören könnte, mit f.innerhalb(p) geprüft, ob der Punkt in der Figur liegt (j) oder nicht (n). Ist der Punkt innerhalb, wird ein schwarzer Punkt an dieser Stelle gemalt. Sie können dabei Ihre Klassen Punkt, Kreis und Rechteck aus früheren Aufgaben in das auf der Veranstaltungsseite erhältliche Projekt AufgabeRahmen kopieren und modifizieren.
Abhängig vom Zoom-Faktor könnte die Darstellung im Interaktionsbrett ein Raster haben, was ok ist. Bei sehr großem Zoom (>=5) sollte eine durchgehende Fläche dargestellt sein. Grundlage des Programms ist die auch von der Veranstaltungsseite erhältliche vollständig abstrakte Klasse Geoobjekt mit folgenden Methoden.

| **Method Summary** | |
|---|---|
| void | **darstellen**(Interaktionsbrett ib)<br>Methode zum Zeichen des Objekts auf einem Interaktionsbrettobjekt, dabei wird das Objekt komplett schwarz dargestellt. |
| boolean | **innerhalb**(Punkt p)<br>Methode zur Berechnung, ob ein übergebener Punkt sich in diesem Objekt befindet, der Rand soll dabei zum Objekt gehören. |
| Punkt | **linksoben**()<br>Methode zur Berechnung der linken oberen Ecke des Objektes, dabei kann kein Punkt des Objektes sich weiter links oder oberhalb dieses Punktes befinden. |
| Punkt | **rechtsunten**()<br>Methode zur Berechnung der rechten unteren Ecke des Objektes, dabei kann kein Punkt des Objektes sich weiter rechts oder unterhalb dieses Punktes befinden. |
| void | **verschieben**(int dx, int dy)<br>Methode zum Verschieben des Objektes, dabei wird das gesamte Objekt auf der x-Achse um dx Punkte und auf der y-Achse um dy-Punkte verschoben. |

Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing-Wiss. und Informatik
- Software-Entwicklung -

**Programmierung 1**

Wintersemester 2025/26
**Aufgabenblatt 11**

a) Erweitern Sie Ihre Klasse Rechteck so, dass sie die abstrakte Klasse Geoobjekt realisiert, dabei soll ein Rechteck gefüllt dargestellt werden, wie es in Abb. 3 links-oben in der Ecke zu sehen ist.

b) Erweitern Sie Ihre Klasse Kreis so, dass sie die abstrakte Klasse Geoobjekt realisiert. Um den ausgefüllten Kreis zu zeichnen wird der Ansatz aus Abb. 1 genutzt: Es wird für jeden Punkt in dem von den Punkten linksoben() und rechtsunten() umrandeten Rechteck geprüft, ob dieser innerhalb() liegt; ist das der Fall, wird der Punkt gezeichnet (n=nein,nicht, j=ja).

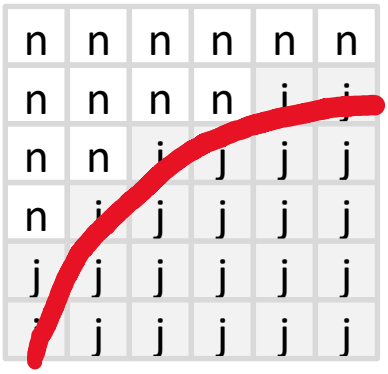



**Abbildung 1: punktweise Darstellung**

Um festzustellen, ob ein Punkt im Kreis liegt, kann der Satz des Pythagoras angewandt werden, wie es in Abb. 2 skizziert ist. Dazu wird der Abstand des zu untersuchenden Punktes auf der X-Achse (in der Abbildung a) und auf der Y-Achse (in der Abbildung b) vom Mittelpunkt des Kreises bestimmt. Gilt dann $a^2 + b^2 <= radius^2$, liegt der Punkt im Kreis (der Rand soll dazugehören).

Ein Beispiel für einen Kreis ist rechts neben dem Rechteck in Abb. 3 dargestellt.

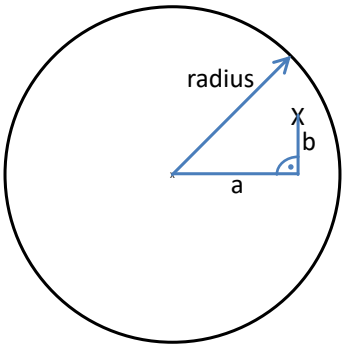



**Abbildung 2: liegt Punkt X im Kreis?**

c) Schreiben Sie eine Klasse Rahmen, die aus einem das Geoobjekt-Inferface realisierenden Objekt als Rahmen und mehreren daraus entfernten auch das Geoobjekt realisierenden Objekten besteht. Nutzen Sie dazu eine Methode `ausstanzen(Geoobjekt)` mit der aus dem Rahmen das übergebene Objekt ausgestanzt wird. Es soll z. B. folgender Code möglich sein.

```
Rahmen ra = new Rahmen(
   new Rechteck(
     new Punkt(10,80),120,80));
ra.ausstanzen(new Kreis(
        new Punkt(15,85),20));
ra.ausstanzen(new Kreis(
        new Punkt(85,115),20));
```

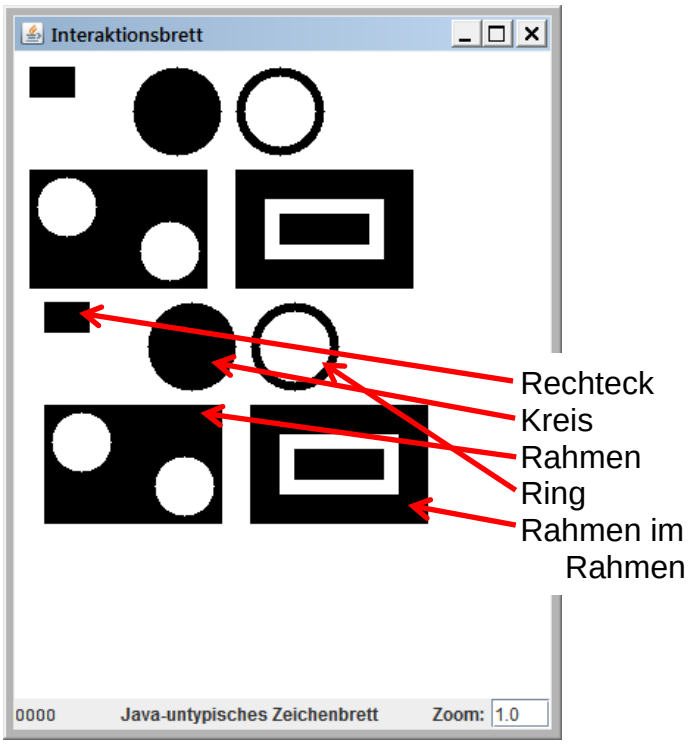



**Abbildung 3: Interaktionsbrett mit Beispielfiguren**

Abb. 3 zeigt unter dem Rechteck z. B. einen Rahmen, der aus einem Rechteck-Objekt besteht, aus dem zwei Kreise entfernt wurden. Realisieren Sie die Klasse Rahmen so, dass sie selbst die abstrakte Klasse Geoobjekt realisiert. Sichtbar sind Punkte des Rahmens, die nicht auch in einem (oder mehreren) entfernten Objekten liegen. Der Ansatz mit innerhalb() aus Abb. 1 kann hier auch hilfreich sein.

Insgesamt kann damit ein Rahmen selbst wieder ein Rahmen in einem anderen Objekt sein. Die ineinander geschachtelten Rechtecke rechts neben den entfernten Kreisen in Abb. 3 sind so entstanden, indem aus einem äußeren Rahmen ein innerer Rahmen entfernt wurde.

d) Lassen Sie eine Klasse Ring von der Klasse Rahmen erben, wobei bei einem Ring aus einem Kreis ein kleinerer Kreis mit gleichem Mittelpunkt entfernt wird.
Ein Konstruktor eines Ringes sollte wie folgt aussehen:
`public Ring(Kreis aussen, int ringdicke)`
Ein Ring-Beispiel ist in Abb. 3 neben den gefüllten Kreisen dargestellt.

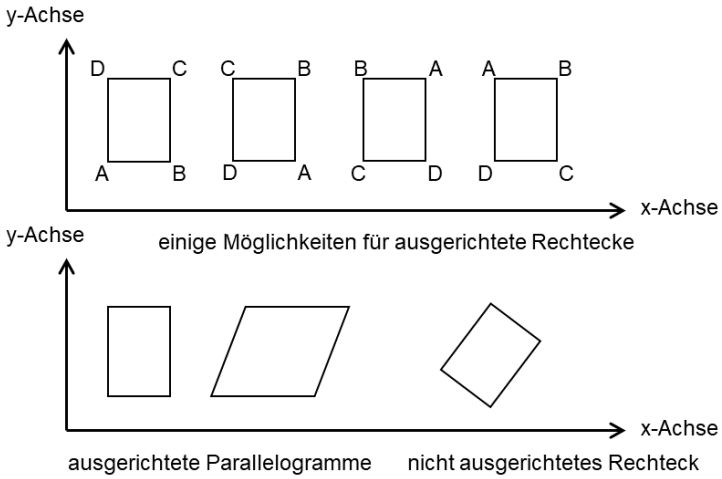
**Abbildung 4: robbi()**

Zur Überprüfung Ihrer Ergebnisse ist die Klasse GeoSpielerei beigelegt, die das in Abb. 3 gezeigte Bild mit malMal() berechnet, bei dem mehrere Objekte auch einmal verschoben dargestellt werden. Die zweite Methode robbi() liefert das Bild aus Abb. 4.
Freiwillig: Da Code-Wiederholungen vermieden werden sollen, sind hier auch andere Ansätze sinnvoll. Überlegen Sie sich eine Lösungsvariante, in der es nicht zur Code-Wiederholung des Darstellungsverfahrens kommt.

**38. Aufgabe (7 Punkte, Nutzung Arrays, VL 21)**

Gegeben sei das Projekt AufgabeViereck von der Webseite der Veranstaltung. Ergänzen Sie die Klasse Viereck, das aus vier beliebigen Punkten besteht, die *in einer Objektvariablen, in einem Array,* verwaltet werden.

```
private Punkt[] ecken;
```




Viereck soll das gegebene Interface ViereckInterface realisieren. Gehen Sie davon aus, dass die Punkte in der Reihenfolge A, B, C, D, wie in der Abbildung gezeigt, eingegeben werden und die Kanten von A nach B, B nach C, C nach D und D nach A verlaufen, dabei muss A aber nicht die linke untere Ecke sein. Es gibt damit vier verschiedene Vierecke, die das gleiche Viereck beschreiben.
Sollte bei den Aufgaben c) bis g) einer der Punkte null oder das Viereck nicht OK sein, ist das Ergebnis false. Prüfen Sie Ihre Implementierungen mit den vorher zu lesenden JUnit-Tests.

a) Ergänzen Sie in der gegebenen Klasse Punkt eine Methode abstand(), die einen Punkt übergeben bekommt und den Abstand der Punkte als double-Wert zurückgibt. Sollte der übergebene Punkt null sein, ist das Ergebnis 0.0. Sie können Klassenmethoden wie Math.sqrt(.) nutzen und sich an Pythagoras erinnern.
b) Schreiben Sie einen Konstruktor für Viereck, der vier Punkte übergeben bekommt und in den Array der einen Objektvariable setzt.
c) Schreiben Sie eine Methode istOK(), die genau dann true als Ergebnis liefert, wenn es sich um ein echtes Viereck handelt, also alle vier Punkte existieren und unterschiedlich sind und maximal zwei Punkte den gleichen x-Wert sowie maximal zwei Punkte den gleichen y-Wert haben. (Man kann sich überlegen, dass es trotzdem noch „hässliche“ Vierecke geben kann (z. B. (1,1), (2,2), (3,3), (4,4) ), das ist hier zu ignorieren. Dies ist aber eine spannende, nicht triviale Zusatzaufgabe.)
d) Schreiben Sie eine Methode istAusgerichtet(), die genau dann true als Ergebnis liefert, wenn es sich um ein echtes „ausgerichtetes“ Viereck handelt. Dabei heißt ein Viereck hier ausgerichtet, wenn zwei nicht verbundene Kanten parallel zur x-Achse verlaufen. Oben sehen Sie 6 ausgerichtete Vierecke (rechts unten nicht).
e) Schreiben Sie eine Methode istAusgerichtetesParallelogramm(), die genau dann true als Ergebnis liefert, wenn es sich bei dem Viereck um ein echtes „ausgerichtetes“

(siehe d)) Parallelogramm handelt, also die parallelen Seiten gleich lang sind. Oben sehen Sie 6 ausgerichtete Parallelogramme (rechts unten nicht).

f) Schreiben Sie eine Methode istAusgerichtetesRechteck(), die genau dann true als Ergebnis liefert, wenn es sich bei dem Viereck um ein echtes „ausgerichtetes“ (siehe d)) Rechteck handelt. Also ein Rechteck parallel zur x-Achse und y-Achse. Oben sehen Sie 5 ausgerichtete Rechtecke.

g) Schreiben Sie eine Methode istRaute(), die genau dann true als Ergebnis liefert, wenn alle vier Kanten gleich lang sind. Erlauben Sie bei der Prüfung der Werte eine minimale Ungenauigkeit, um Probleme mit der Rechengenauigkeit zu vermeiden.

h) Schreiben Sie eine equals() Methode für Vierecke, dabei reicht es aus, wenn die vier Kanten übereinstimmen (auch null-Werte als Punkte erlaubt) und es ist unerheblich ob ein Viereck OK ist. Es müssen also nur die vier Punkte in der gleichen Weise verbunden sein. Also ein Viereck mit punkten {p1,p2,null,p3} ist u. a. equals mit {null,p3,p1,p2}.
(freiwillig: die Vierecke sind gegen den Uhrzeigersinn definiert; bei equals könnten auch Vierecke gleich sein, die im Uhrzeigersinn orientiert sind und zur gleichen Darstellung führen (in obigen Abbildungen z. B. A und C vertauschen).