Institut für Informatik
der Universität München
Prof. Dr. M. Hofmann
Dr. J. Johannsen, Dr. R. Matthes

SS 2004

28.5.04

Übungen zur Vorlesung
Informatik II

Blatt 6

Abgabe der Hausaufgaben bis spätestens am 14.6.04, 14:00 Uhr über
`http://lehre.tcs.ifi.lmu.de/info2/Abgabe/abgabe.php`,
Bearbeitung in Gruppen zu max. 3 Personen ist zulässig.

Bitte halten Sie sich an folgende Grundsätze zur Bearbeitung der Aufgaben. Die Bewertung Ihrer Abgaben kann andernfalls deutlich herabgesetzt werden.

Verwenden Sie genau die Signatur der Aufgabenstellung, insbesondere die Namen der Klassen und Methoden in der vorgegebenen Klein- und Großschreibung.

Verwenden Sie keine „magischen" Größen in Ihren Programmen, d. h., weisen Sie Konstanten (außer den Zahlen 0 und 1 und dem leeren String) immer einer mit `final` deklarierten und in Großbuchstaben geschriebenen „Variablen" zu und verwenden Sie anschließend nur diese. Wer gute Gründe hat für eine Abweichung von dieser Regel, der kommentiere seinen Code entsprechend.

Jede Klasse und `public`-Methode muß angemessen mit `javadoc`-Kommentaren dokumentiert werden.

Programmieraufgabe P-15 (`geometrie.jar`): **20 Punkte**

Im *homogenen Koordinatensystem* wird ein Punkt in der Ebene durch einen Vektor $(x, y, z) \in \mathbb{R}^3$ repräsentiert, wobei Vektoren, die sich nur um einen skalaren Faktor unterscheiden, wie z.B. $(1, 0, 1)$ und $(2, 0, 2)$, den gleichen Punkt repräsentieren. Die geometrische Vorstellung ist, dass ein Vektor v für die Gerade $\{\lambda v \mid \lambda \in \mathbb{R}\}$ steht, und der dargestellte Punkt der Schnittpunkt mit der Ebene $\{(x_{\mathsf{cart}}, y_{\mathsf{cart}}, 1) \mid (x_{\mathsf{cart}}, y_{\mathsf{cart}}) \in \mathbb{R}^2\}$ ist. Also hat der durch den Vektor (x, y, z) repräsentierte Punkt die kartesischen Koordinaten $x_{\mathsf{cart}} = x/z$ und $y_{\mathsf{cart}} = y/z$ in dieser Ebene. (Ignorieren Sie in dieser Aufgabe generell, daß $z = 0$ hier bei der Division zu einem Fehler während der Ausführung des Programms führen würde. Dieser Fall kommt in unseren Testbeispielen auch nicht vor.) Offenbar wird dann durch den Vektor $(x_{\mathsf{cart}}, y_{\mathsf{cart}}, 1)$ derselbe Punkt repräsentiert wie durch (x, y, z).

Vorteil dieser Darstellung ist, dass sich so mehr Operationen als Multiplikation mit einer Transformationsmatrix darstellen lassen als in kartesischen Koordinaten. So ist z.B. das Verschieben um einen Vektor $(\Delta x, \Delta y, 1)$ durch die Matrix

$$\begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix}$$

gegeben. Die Matrizen für die Rotation um den Vektor $(x, y, 1)$ als Mittelpunkt und Winkel α und die zentrische Streckung mit dem Vektor $(x, y, 1)$ als Zentrum und Streckfaktor λ sind

$$\begin{pmatrix} \cos\alpha & -\sin\alpha & x(1-\cos\alpha) + y\sin\alpha \\ \sin\alpha & \cos\alpha & y(1-\cos\alpha) - x\sin\alpha \\ 0 & 0 & 1 \end{pmatrix} \quad \text{und} \quad \begin{pmatrix} \lambda & 0 & (1-\lambda)x \\ 0 & \lambda & (1-\lambda)y \\ 0 & 0 & 1 \end{pmatrix}$$

– bitte wenden –

Bemerkungen:

- Die Verkettung von Operationen entspricht dann der Matrizenmultiplikation.
- Die Drehung erfolgt um den Winkel α im mathematisch positiven Sinn, d. h., gegen den Uhrzeigersinn. Dies gilt aber nur für die in der Mathematik übliche Ausrichtung des Koordinatensystems mit der x-Achse nach rechts und der y-Achse nach oben. In den Java-Bibliotheken geht die y-Achse nach unten, weshalb sich der Drehsinn in der grafischen Ausgabe umdreht.

Schreiben Sie ein package `geometrie`, das geometrische Objekte im homogenen Koordinatensystem verwaltet. Das package muß eine abstrakte Klasse `Figur` enthalten, sowie deren Unterklassen `Punkt`, `Dreieck`, `Viereck`, `Rechteck`, `Quadrat` und `Kreis`, die die entsprechenden geometrischen Figuren implementieren. Ein Punkt wird – wie oben beschrieben – durch homogene Koordinaten dargestellt. Ein Dreieck ist durch drei (Konstruktor `Dreieck(Punkt p , Punkt q , Punkt r)`) und ein Viereck durch vier beliebige Punkte gegeben. Rechtecke und Quadrate sind spezielle Vierecke, und ein Kreis ist durch Mittelpunkt und Radius gegeben mit Konstruktor `Kreis(Punkt p, double r)`.

Alle diese Klassen sollen die Methoden

- `void verschieben(double dx, double dy)`
- `void rotieren(Punkt zentrum, double winkel)`
- `void strecken(Punkt zentrum, double faktor)`

anbieten, die die oben genannten Operationen durch Anwendung der entsprechenden Transformationsmatrizen implementieren. Weiterhin soll es in jeder Klasse eine Methode

- `void zeichnen(Graphics2D g2)`

geben, die das Objekt in dem übergebenen Grafikkontext zeichnet.

Überlegen Sie sich, welche Methoden Sie in welcher Klasse implementieren, um eine unnötige Duplizierung von Code zu vermeiden.

Konstruktormethoden mit folgendem Verhalten müssen implementiert werden:

- `Punkt(double x, double y)` erzeugt einen Punkt mit den beiden kartesischen Koordinaten `x` und `y`.
- `Rechteck(Punkt p, Punkt q, double breite)` erzeugt ein Rechteck, dessen erste Seite durch `p` und `q` gegeben ist. Die nächste Seite ab `q` erhält man gedanklich wie folgt: Man dreht die Seite von `p` nach `q` um den Punkt `q` um 90 Grad im Uhrzeigersinn und streckt/dehnt die Seite genau auf die Länge `breite`. Der vierte Punkt ist dann ebenfalls festgelegt. Beachten Sie, daß das Rechteck nicht achsenparallel sein muß wie in `java.awt.Rectangle`. Stattdessen steht hier die andere Seite der Länge `breite` senkrecht auf der Verbindung zwischen `p` und `q`.
- `Quadrat(Punkt p, Punkt q)` erzeuge das Quadrat, das durch Aufruf des eben beschriebenen Rechtecks-Konstruktors mit passender `breite` erzeugt würde.

Als Test finden Sie auf der Vorlesungshomepage zwei Applets, die diese Klassen verwenden. Wenn Sie sie ausführen, sollten die ebenfalls dort als Screenshot zu findenden Grafiken erzeugt werden. Das erste Applet `SimpTest` soll Sie auf einen häufig gemachten Programmierfehler hinweisen. Das zweite testet die gesamte Funktionalität der Klassen.