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

WS 02/03

30. Oktober 2002

Practical 1 zur Vorlesung

Einführung in die Informatik: Programmierung und Software-Technik

Abgabe: spätestens 13.11.2002, 14:00 Uhr

Gegenstand dieses Practicals ist die einfache Grafikprogrammierung unter Java. Um am Ende der Vorlesung den Schein zu erhalten, müssen Sie dieses Practical sinnvoll bearbeitet haben. Dies ist allerdings keine hinreichende Bedingung für die Scheinvergabe.

Zur Bearbeitung brauchen Sie die `GraphicsWindow`–Klasse, die bereits in der Vorlesung vorgestellt wurde. Diese können Sie sich von der WWW-Seite der Vorlesung herunterladen. Beachten Sie, dass sich die Datei mit dieser Klasse im selben Verzeichnis befindet wie die, die Ihr ausführbares Programm enthält.

Damit das elektronische Abgabesystem automatisch überprüfen kann, ob Ihr Programm fehlerfrei kompiliert, muß Ihre Klasse einen bestimmten Namen haben. Für dieses Practical ist dies

```
Practical1
```

Wie beim ersten Practical müssen Sie Ihr Programm in einem `jar`-Archiv abgeben. Zur Wiederholung: Dies erzeugen Sie mit

`jar cvf` *ArchivName Datei1 Datei2* ...

Achten Sie darauf, dass *alle* Dateien außer den Standardbibliotheken, die von Ihrem Programm benutzt werden, in dem `jar`-Archiv vorhanden sind, also alle Klassen, die Sie für dieses Practical schreiben sollen sowie vorgegebene Klassen wie z.B. `GraphicsWindow.java`.

Bei der Bearbeitung bedenken Sie, dass die Lösung zu diesem Practical bereits in der folgenden Aufgabenstellung vorhanden ist. Alles, was Sie noch zu tun haben, ist die einzelnen Aufgabenschritte in Java-Code zu übersetzen. Dazu müssen Sie den Text aufmerksam lesen und herausfinden, was darin durch Variablen und was durch Methodenaufrufe zu modellieren ist. Achten Sie dabei darauf, dass Sie dieselben Variablen nicht für verschiedene Dinge benutzen und dass Sie aussagekräftige Bezeichner verwenden.

Zur sinnvollen Bearbeitung dieses wie auch der folgenden Practicals gehört auch die Beachtung einer wesentlichen Grundregel des Programmierens: Nur gut kommentierte Programme sind gute Programme. Bedenken Sie, dass die Korrektoren sich den Quell-Code Ihrer Programme ansehen und anhand dessen entscheiden werden, ob Sie die Problemstellung und Lösung verstanden haben.

Bei Problemen wenden Sie sich an die studentischen Hilfskräfte, die die Rechnerraumbetreuung durchführen. Diese sind an den folgende Terminen im CIP-Pool Sibirien bzw. Gobi zu finden. Montag 10:00 – 20:00, Dienstag 9:00 – 13:00, 14:30 – 18:30, Mittwoch 9:00 – 13:00, Donnerstag 12:00 – 14:00, Freitag 13:00 – 17:00.

1 Aufgabenstellung

1.1 Intuitive Aufgabenstellung

Klicken Sie auf der WWW-Seite der Vorlesung im Abschnitt `Practicals` auf den Link zum Demo–Programm, das dort bereitgestellt ist. Ziel dieses Practicals ist es, ein Programm zu erstellen, das sich genauso verhält.

1.2 Eine Klasse für achsenparallele Dreiecke

Zwei Punkte definieren ein achsenparalleles, rechtwinkeliges Dreieck. Vergleichen Sie dies mit den Punkten p_1 und p_2 aus Abbildung 1. Durch diese wird z.B. das Dreieck p_1, q_1, p_2 definiert.

Schreiben Sie eine Klasse für achsenparallele Dreiecke. Diese sollte folgendes enthalten.

- Variablen für zwei Punkte. Dazu müssen Sie die Klasse `Point` mittels

  ```
  import java.awt.Point;
  ```

  importieren.
- Einen Konstruktor, der zwei Punkte (Objekte vom Typ `Point`) erwartet, und die zwei Variablen mit diesen initialisiert.
- Eine Methode, die den Flächeninhalt eines dazugehörigen achsenparallelen Dreiecks berechnet. Dieser ergibt sich einfach als das Produkt der Absolutbeträge der Differenzen der jeweiligen x– und y–Koordinaten der beiden Punkte, geteilt durch 2.

1.3 Eine Klasse für Dreiecke

Ein allgemeines Dreieck ist im Gegensatz zum Achsenparallelen durch drei Punkte definiert. Schreiben Sie eine Klasse, die entsprechende Variablen und einen Konstruktor enthält, der diese initialisiert. Desweiteren sollte Ihre Klasse noch folgendes zur Verfügung stellen.

- Eine Methode, die das Dreieck verschiebt. Sie erwartet zwei Werte vom Typ `int`, die als Verschiebung in x– und y–Koordinatenrichtung zu interpretieren sind. Eine Verschiebung des Dreiecks ist dann einfach eine Verschiebung aller Punkte.
- Methoden, die die drei Punkte zurückliefern. Dies ist z.B. nach einer Verschiebung nötig.
- Eine Methode, die zu dem Dreieck das *umfassende Rechteck* (vom Typ `Rectangle`) berechnet. Dazu müssen Sie natürlich die Klasse durch

  ```
  import java.awt.Rectangle;
  ```

  importieren.

  Ein `Rectangle` läßt sich durch einen Punkt und zwei Werte vom Typ `int` erzeugen. Dabei ist der Punkt die linke, obere Ecke. Diese erhalten Sie durch die Minima aller x– und y–Koordinaten der Punkte des Dreiecks. Die beiden `int`–Werte sind als Breite und Höhe des Rechtecks zu interpretieren. Dazu müssen Sie noch die Maxima aller x– und y–Koordinaten berechnen, und von denen jeweils die Minima subtrahieren.
- Eine Methode, die den Flächeninhalt des Dreiecks berechnet. Dazu können Sie zuerst den Flächeninhalt des umfassenden Rechtecks berechnen, indem Sie einfach dessen Breite und Höhe multiplizieren. Dann bilden Sie für jede Seite des Dreiecks das achsenparallele Dreieck und ziehen dessen Fläche von der Rechtecksfläche ab. Betrachten Sie Abbildung 1 zur Veranschaulichung.

Diese Flächenberechnung liefert immer einen Wert. Allerdings funktioniert diese Konstruktion nur einwandfrei, wenn das Dreieck spitzwinkelig ist. Für stumpfwinkelige Dreiecke kann es den richtigen Wert liefern, muß es aber nicht. Dies hängt von der genauen Lage des Dreiecks ab. Schreiben Sie daher noch eine Methode, die überprüft, um welche Art von Dreieck es sich handelt.

Dabei wird der Winkel betrachtet, der der längsten Seite gegenüberliegt. Dazu müssen Sie natürlich herausfinden, welche Seite s_1, s_2 oder s_3 des Dreiecks die längste ist. Dazu können Sie Methodenaufrufe der Art

p_1`.distance(`p_2`)`

für p_1 und p_2 aus der Klasse `Point` benutzen.

Nehmen wir nun an, s_3 sei die längste Seite. Die folgende Vorschrift funktioniert auch dann, wenn s_3 nicht die einzige längste Seite ist.

Betrachten Sie die Summe der Quadrate von s_1 und s_2 minus dem Quadrat von s_3. Falls das Ergebnis 0 ist, dann ist das Dreieck rechtwinkelig (Pythagoras). Falls es echt größer als 0 ist, dann ist das Dreieck spitzwinkelig, ansonsten ist es stumpfwinkelig.

Sie können diesen Wert als Rückgabewert Ihrer Methode benutzen. D.h. Ihre Methode prüft das Dreieck nicht direkt dahingehend, welche Art von Winkeligkeit vorliegt. Ihr Hauptprogramm kann dies tun, indem es die Methode aufruft und testet, ob der Rückgabewert gleich 0 oder größer oder kleiner ist.

1.4 Das Hauptprogramm

Ihr Programm soll so arbeiten wie das Beispielprogramm, das auf der WWW-Seite zu finden ist.

Zu Beginn öffnet es ein Grafikfenster. Dies können Sie über

```
GraphicsWindow fenster = new GraphicsWindow()
```

erreichen. Mit dem Methodenaufruf

`fenster.setText(`*Zeichenkette*`)`

können Sie Text am oberen Rand des Grafikfensters ausgeben.

Das Programm erwartet drei aufeinanderfolgende Mausklicks. Ein jeder definiert einen Punkt in dem Grafikfenster. Die Anweisung

```
fenster.mouseClick()
```

wartet auf einen Mausklick im Fenster `fenster` und liefert als Rückgabe ein Punktobjekt mit den Koordinaten des Mausklicks.

Diese drei Punkte definieren dann ein Dreieck. Als nächstes soll dieses Dreieck gezeichnet werden. Dazu können Sie einen Aufruf der Form

`fenster.drawLine(`p_1`,`p_2`)`

benutzen.

Im nächsten Schritt soll das umfassende Rechteck gezeichnet werden. Bedenken Sie, dass die Methode in Ihrer Dreiecks-Klasse ein Objekt vom Typ `Rectangle` zurückliefert. Dessen Koordinaten sowie Höhe und Breite können aus folgenden Größen auslesen.

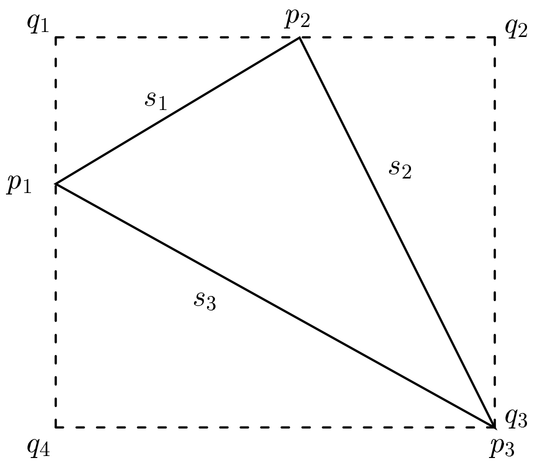

Abbildung 1: Ein Beispiel.

Rechteckname`.x` *Rechteckname*`.y` *Rechteckname*`.width` *Rechteckname*`.height`

Daraus müssen Sie dann vier Punkte machen, welche durch q_1 bis q_4 in Abbildung 1 beispielhaft dargestellt sind.

Löschen Sie nach dem nächsten Mausklick das umfassende Rechteck wieder. Dazu können Sie einfach mit dem Methodenaufruf

```
fenster.switchToBackgroundColour()
```

die Zeichenfarbe für das Fenster `fenster` auf die Hintergrundfarbe umstellen und das Rechteck nochmal zeichnen. Mit

```
fenster.switchToForegroundColour()
```

schalten Sie wieder auf schwarze Zeichenfarbe um.

Testen Sie nun das Dreieck auf seine Winkeligkeit. Falls es spitzwinkelig oder rechtwinkelig ist, dann geben Sie eine entsprechende Meldung am oberen Fensterrand aus und auch den Flächeninhalt des Dreiecks. Falls es stumpfwinkelig ist, dann geben Sie eine Meldung aus, dass aufgrund dessen die Fläche nicht berechnet werden kann.

Jetzt soll das Programm wieder auf einen Mausklick, dessen Koordinaten uninteressant sind, warten. Danach gilt es, das Dreieck im Grafikfenster zu löschen.

Zum Schluss soll dasselbe Dreick zusammen mit seinem umfassenden Rechteck noch einmal an einer anderen Stelle gezeichnet werden, allerdings jetzt in der Vordergrundfarbe. Dazu wird wieder auf einen Mausklick gewartet. Dieser wird als der neue erste Punkt des Dreiecks interpretiert.

Aus den Differenzen der x– und y–Koordinaten des ersten Punktes des Dreiecks und des neuen Punktes ergeben sich die Parameter für die Verschiebemethode Ihrer Dreiecksklasse. Danach müssen Sie sich die neuen Eckpunkte des Dreiecks holen und es noch einmal zeichen. Achten Sie darauf, dass die Eckpunkte des umfassenden Rechtecks ebenfalls verschoben sind.

Das Programm können Sie mit dem Aufruf

```
System.exit(0);
```

beenden.