Prof. Martin Hofmann, PhD
Dr. Steffen Jost

Ludwig-Maximilians-Universität München
Institut für Informatik
20. November 2017

5. Übung zur Vorlesung

Einführung in die Programmierung

Hinweise zu den Hausübungen Bitte achten Sie darauf, in sämtlichen abgegeben `.java`-Dateien jegliche Zeilen mit `package` zu entfernen (wird oft von IDEs am Anfang eingefügt). `package` Angaben in abgegebenen `.java`-Dateien können ab sofort zu Punktabzug führen, da dies den Zeitaufwand für die Korrektur unnötig erhöht! Das Thema „Packages“ wird in der Vorlesung später noch behandelt werden. Es werden nur `.java`-Dateien korrigiert, die kompilieren! Kommentieren Sie problematischen Code aus und schreiben Sie Fragen dazu.

Achten Sie ebenfalls darauf, dass die Dateinamen keine Umlaute enthalten, da UniWorX momentan leider Probleme mit Umlauten in Dateinamen hat.

A5-1 *BNF* Ein Text ist eine nicht-leere Sequenz von Sätzen. Ein Satz besteht aus einem Subjekt, gefolgt von einem Prädikat und einem Objekt. Jeder Satz wird durch einen Punkt abgeschlossen. Das Objekt kann ein Wort oder ein in (französischen) Anführungszeichen (≫ bzw. ≪) eingeschlossener Text sein.

Als Hilfe sind in dieser Aufgabe alle Terminalsymbole mit deutschen Anführungszeichen („ bzw. “) gekennzeichnet. Die deutschen Anführungszeichen sind kein Teil der Sprache.

a) Geben Sie eine BNF-Grammatik für die oben beschriebenen Texte an. Dabei können Sie folgende Regeln als bereits gegeben betrachten:

$$\begin{aligned}
\langle \textit{Subjekt} \rangle &:= \text{“}\texttt{Hund }\text{”} \mid \text{“}\texttt{Katze }\text{”} \\
\langle \textit{Prädikat} \rangle &:= \text{“}\texttt{sagt }\text{”} \mid \text{“}\texttt{brummt }\text{”} \\
\langle \textit{Wort} \rangle &:= \text{“}\texttt{wuff}\text{”} \mid \text{“}\texttt{miau}\text{”} \\
\langle \textit{Punkt} \rangle &:= \text{“}\texttt{.}\text{”} \\
\langle \textit{AnführungLinks} \rangle &:= \text{“}\gg\text{”} \\
\langle \textit{AnführungRechts} \rangle &:= \text{“}\ll\text{”}
\end{aligned}$$

Zur Erinnerung: Eine BNF-Grammatik ist ein Quadrupel (Σ, V, S, P), siehe Folie 5.11.

b) Leiten Sie aus Ihrer Grammatik den folgenden Text ab:

“`Hund`” “`brummt`” “≫” “`Katze`” “`sagt`” “`miau`” “`.`” “≪” “`.`”

A5-2 *Die Klasse GraphicsWindow* Prof. Hofmann hat Ihnen über die Vorlesungshomepage freundlicherweise sein Klasse `GraphicsWindow` zur Verfügung gestellt. In dieser Aufgabe wollen wir verstehen, wie wir die Klasse `GraphicsWindow` einsetzen können, auch wenn wir den Inhalt der Klassendefinition mit den bis jetzt in der Vorlesung behandelten Themen noch nicht verstehen können.

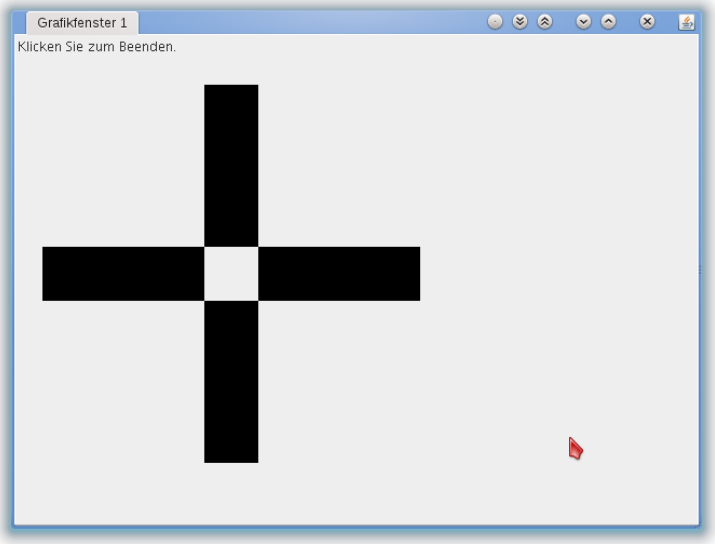
a) Speichern Sie die drei Dateien GraphicsWindow.java, Car.java und Test.java in ein frisches Verzeichnis. Kompilieren Sie die Klasse `Test` und starten Sie deren main-Methode anschließend. Was passiert?

Hinweis: Wer nicht weiß, was zu tun ist, sollte sich noch einmal Aufgabe A1-1 anschauen!

b) Welche Konstruktoren und Methoden bietet die Klasse `GraphicsWindow` an? Lösen Sie diese Aufgabe mit Hilfe von Javadoc!

Hinweis: Da die Dateivorlage in UTF-8 kodiert ist und deutsche Umlaute erhält, benötigt Javadoc ggf. die Parameter `-encoding UTF-8 -charset UTF-8 -docencoding UTF-8`.

c) Schreiben Sie eine eigene Klasse `Demo` zur Demonstration der Klasse `GraphicsWindow`. Orientieren Sie sich dabei an `Test`. Ihr Demo soll ungefähr das folgende Bild zeichnen:

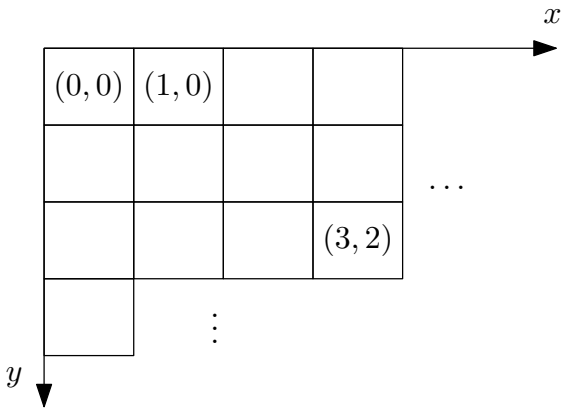


Hinweise:

- Die ihnen bereits bekannte Klasse `java.awt.Rectangle` könnte hierfür nützlich sein. Um ein Objekt der Klasse `Rectangle` zu zeichnen, übergeben Sie es als Parameter an die Methoden `draw` oder `fill` eines Objektes der Klasse `GraphicsWindow`.
- Sie können folgendes Programmgerüst verwenden:

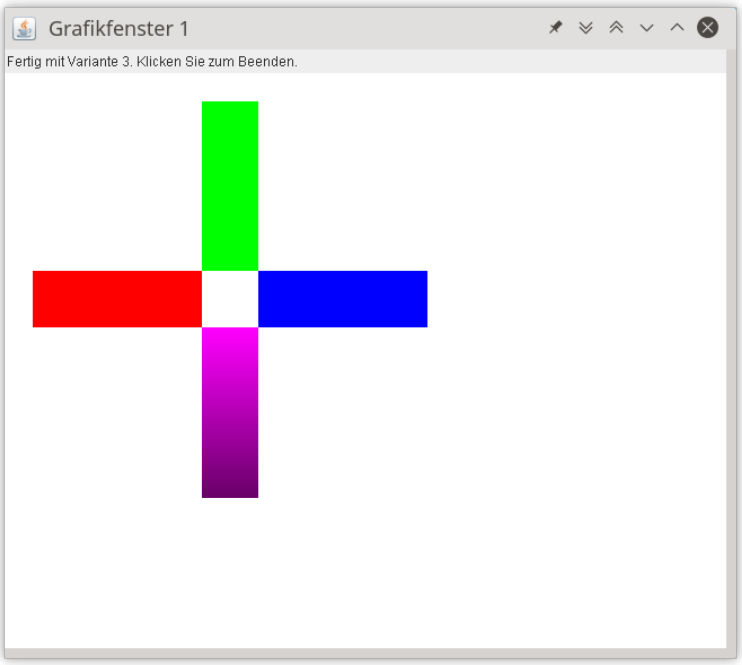
```
import java.awt.Rectangle;
public class Demo {
    public static void main(String[] args) {
        int margin     = 25;
        int box_height = 50;
        int box_width  = 150;
        Rectangle r1 = new Rectangle(margin, margin+box_width,
                                     box_width, box_height);
        GraphicsWindow gw = new GraphicsWindow();
        gw.fill(r1);
        // ...
        gw.setText("Klicken Sie zum Beenden.");
        gw.mouseClick();
        System.exit(0);
    }
}
```

- Das Koordinatensystem von `GraphicsWindow` ist folgendermaßen zu verstehen:



d) Ändern Sie Ihre Lösung zu Aufgabenteil c) so ab, dass Sie jede Zeichenoperation einzeln beobachten können. Hierzu könnten die Methoden `mouseClick` und/oder `sleep` der Klasse `GraphicsWindow` nützlich sein.

e) Ändern Sie Ihre Programm, so dass folgendes Bild entsteht:



Hinweise: Die Zeichenfarbe wird in `GraphicsWindow` mit der Methode `setColor` gesetzt. Als Argument wird ein Objekt der Klasse `java.awt.Color` erwartet. Diese bietet verschiedene Konstruktoren, z.B. `Color(int red, int green, int blue)` erwartet ganze Zahlen im Bereich 0–255 oder `Color(float r, float g, float b)` erwartet Zahlen im Bereich 0.0–1.0; den Datentyp `float` können Sie aus `double` konvertieren. In beiden Fällen gibt es eine Fehlermeldung, falls ein Argument nicht im erwarteten Zahlenbereich liegt!

H5-1 ***Backus-Naur Form*** (6 Punkte; Abgabe: `H5-1.txt` oder `H5-1.pdf`)
Gegeben sind folgende Produktionen einer BNF-Grammatik mit dem Startsymbol ⟨*Smiley*⟩:

$$
\begin{aligned}
\langle \mathit{Smiley} \rangle &::= \langle \mathit{FröhlicherSmiley} \rangle \mid \langle \mathit{TraurigerSmiley} \rangle \\
\langle \mathit{FröhlicherSmiley} \rangle &::= (\text{“:”} \mid \text{“;”} \mid \text{“8”})\ [\langle \mathit{Nase} \rangle]\ \langle \mathit{FröhlicherMund} \rangle \\
\langle \mathit{TraurigerSmiley} \rangle &::= (\text{“:”} \mid \text{“=”})\ [\langle \mathit{Nase} \rangle]\ \langle \mathit{TraurigerMund} \rangle \\
\langle \mathit{Nase} \rangle &::= \text{“o”} \mid \text{“-”} \\
\langle \mathit{FröhlicherMund} \rangle &::= \text{“)”} \mid \text{“\{”}\ (\text{“)”})^{+} \mid \text{“D”} \\
\langle \mathit{TraurigerMund} \rangle &::= (\text{“(”})^{+} \mid \text{“|”}
\end{aligned}
$$

a) Entscheiden Sie für die folgenden Wörter jeweils, ob sie in der Sprache dieser Grammatik sind. Begründen Sie Ihre Antwort in 1–3 Sätzen! Geben Sie weiterhin für Wörter, die in der Sprache sind, eine ausführliche Ableitung an! Führen Sie dabei jeden Schritt *einzeln* aus!

   i) `8o|`

   ii) `:-(((`

b) Geben Sie einen Smiley in der Sprache dieser Grammatik an, der aus genau 5 Zeichen (Terminalsymbolen) besteht und das Terminalsymbol “`(`” nicht enthält!

c) Geben Sie eine Grammatik als 4-Tupel an, deren Sprache aus nicht-leeren Folgen von Smileys besteht. Dabei muss aber auf jeden traurigem Smiley mindestens zwei glückliche Smileys folgen. Hier ein paar Beispiele:

   Enthalten: “`;-)`”, “`;-) :o) :o)`”, “`:-| :) :) :)`” oder “`=((( :) :) :) =( :) ;)`”
   Nicht enthalten: “`:-|`”, “`;) :-| :)`”, oder “`:-| :( :) :) :) :) :)`”,

   Sie dürfen die Nichterminalsymbole der am Anfang gegebenen Grammatik verwenden, ohne die Regeln dafür zu wiederholen. Mögliche Leerzeichen zwischen den einzelnen Smileys müssen Sie zur Vereinfachung hier nicht beachten.

H5-2 ***Mit Schleifen malen*** (5 Punkte; Datei: Schleife.java)
Wir verwenden erneut die bereitgestellte Klasse `GraphicsWindow` aus Aufgabe A5-2.

Schreiben Sie ein Programm, das ein `GraphicsWindow` mit genau 640x480 Pixeln öffnet und dieses wie folgt einfärbt: Der Punkt mit den Koordinaten (x, y) soll mit dem RGB-Farbwert $\big(r(x,y), g(x,y), b(x,y)\big)$ eingefärbt werden, wobei die drei Funktionswerte in ganze Zahlen im Bereich 0–255 zu konvertieren sind. Die Funktionen sind dabei wie folgt spezifiziert:

$$f(x,y) = \left(\frac{300-y}{64} - \sqrt{\frac{|x-400|}{64}}\right)^2 + \left(\frac{x-400}{64}\right)^2$$

$$r(x,y) = \begin{cases} 255 - 42 \cdot f(x,y) & \text{falls } f(x,y) \leq 4.0 \\ 255 - \sqrt{22x} + \sqrt{32y} & \text{sonst} \end{cases}$$

$$g(x,y) = \begin{cases} \sqrt{100 \cdot b(x,y)} & \text{falls } f(x,y) > 4.0 \\ \frac{y}{6} & \text{sonst} \end{cases}$$

$$b(x,y) = \begin{cases} \sqrt{100 \cdot r(x,y)} & \text{falls } f(x,y) > 4.0 \\ \frac{y}{3} & \text{sonst} \end{cases}$$

Alle benötigten Java-Funktionen finden Sie in der Klasse `Math` aus der Standardbibliothek. Versuchen Sie Ihr Programm möglichst einfach zu gestalten.

Wenn Sie möchten, dürfen Sie dazu auch eigene statische Methoden deklarieren und verwenden; dies hat aber keinen Einfluss auf die Bewertung.

Abgabe: Lösungen zu den Hausaufgaben können bis Sonntag, den 26.11.17, mit UniWorX nur als `.zip` abgegeben werden. Aufgrund des Klausurbonus müssen die Hausaufgaben von Ihnen alleine gelöst werden. Abschreiben bei den Hausaufgaben gilt als Betrug und kann zum Ausschluss von der Klausur zur Vorlesung führen. Bitte beachten Sie auch die Hinweise zum Übungsbetrieb auf der Vorlesungshomepage (`www.tcs.ifi.lmu.de/lehre/ws-2017-18/eip/`).