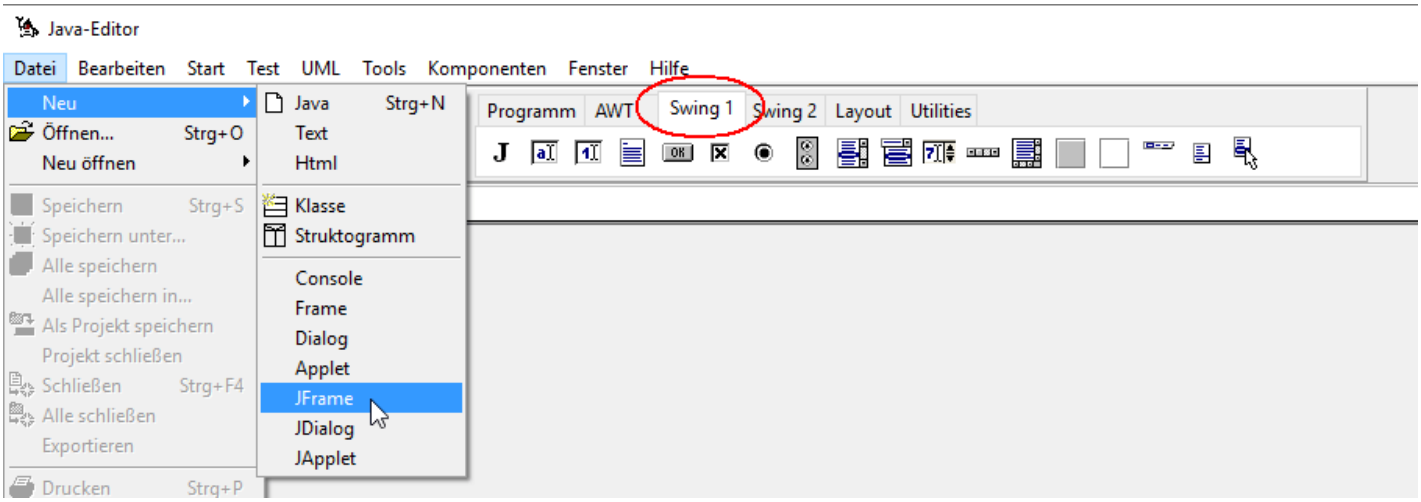
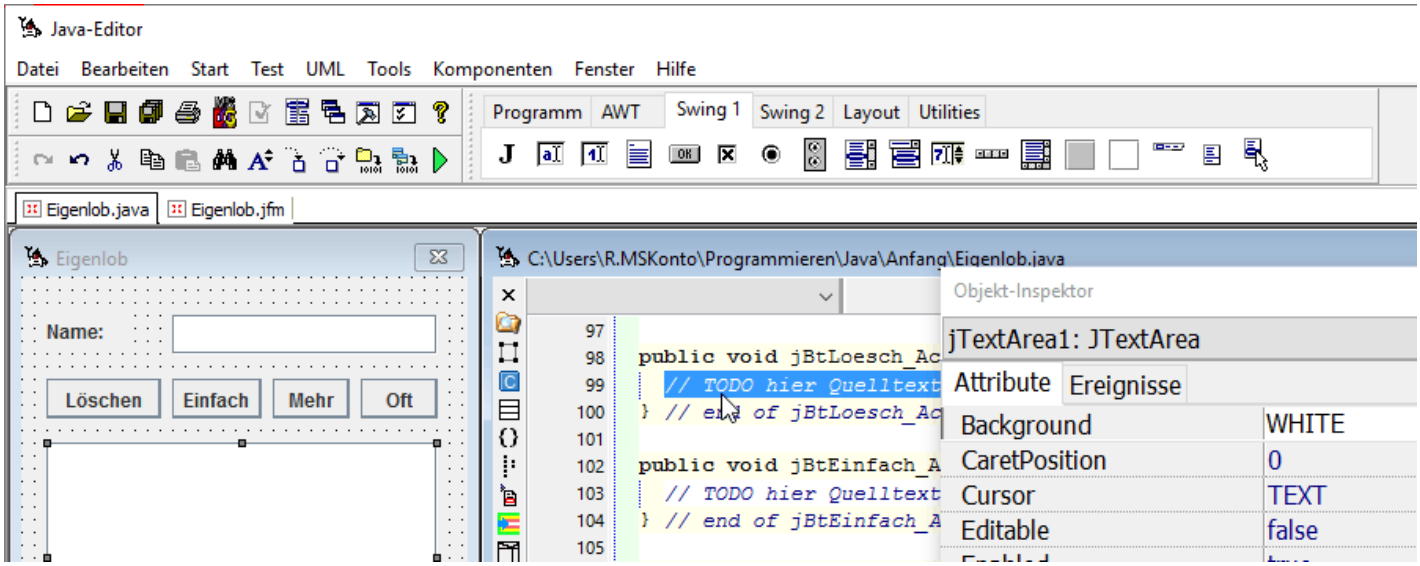
Programmieren in Java (mit dem Java-Editor [1])

(1) Wir entwickeln Anwendungen auf der Grundlage von *JFrames*, in die Objekte aus dem Reiter „Swing 1“ per Maus gezogen werden. Die Eigenschaften der Objekte werden nur mit dem Objekt-Inspektor verändert, der zugehörige Programmtext wird automatisch vom GUI-Builder erzeugt.



Der automatisch erzeugte Programmcode braucht und sollte nicht verändert werden; eigener Programmtext wird ausschließlich in die *Knopf_ActionPerformed*-Methoden an Stelle von „*// TODO hier Quelltext einfügen*“ geschrieben. Vorsicht: Nicht versehentlich Klammern { } löschen.



(2) Der eigene Programmtext enthält typischerweise Befehle wie
String eingabe = irgendeinTextfeld.getText(); oder *anderesTextfeld.setText("haha");* oder auch *jTextAreaAusgabe.append("zusätzliche Zeile\n“);*.
Benötigte Variablen (wie etwa *eingabe*) werden einmal mit Angabe des Typs (z.B. *String*, *int* oder *double*) deklariert – z.B. *int zähler = 0*;. Bei weiterer Verwendung der Variable wird der Typ nicht wiederholt. Bei *zähler = zähler + 1;* wird zunächst der Term rechts vom Zuweisungszeichen = berechnet (hier wird zum bisherigen Inhalt der Variablen *zähler* eine *1* addiert) und dann wird das Ergebnis in der Variablen links vom = gespeichert, in diesem Fall wieder in der Variablen *zähler*.
Alle hintereinander stehenden Befehle, Aufrufe und Anweisungen werden vom Computer in der Reihenfolge nacheinander abgearbeitet; jeder Befehl wird genau einmal ausgeführt („Sequenz“).

---

[1]) kostenlos von http://www.javaeditor.org/ ; vgl. auch www.r-krell.de/if-java-a.htm

3 Kontrollstrukturen

Durch Kontrollstrukturen wird erreicht, dass – abweichend von der normalen Sequenz – Anweisungen vielleicht gar nicht bzw. nur unter bestimmten Bedingungen (nämlich in einer Verzweigung) oder auch mehrfach ausgeführt werden können (Wiederholung):

a) Verzweigungen

a1) einseitige Verzweigung (Wenn-Dann)

```
if (Bedingung) {
  Befehl(e) nur, wenn Bedingung zutrifft
}
```

a2) zweiseitige Verzweigung (Wenn-Sonst)

```
if (Bedingung) {
  Befehl(e) nur, wenn Bedingung zutrifft
} else {
  Befehl(e), falls Bedingung falsch ist
}
```

a3) mehrseitige Verzweigung

```
switch (Selektorvariable) {
  case Wert1 : Befehle für Wert 1; break;
  case Wert2 : Befehle für Wert 2; break;
  case Wert3 : Befehle für Wert 3; break;
}

evtl. als Letztes noch
  default : Befehle, falls kein Wert gewählt;
}
```

b) Wiederholungen

b1) mit vorgeschalteter Kontrolle

```
while (Bedingung) {
  Anweisungen werden wiederholt,
  solange die Bedingung zutrifft
}
```

b2) mit nachgeschalteter Kontrolle

```
do {
  Befehle werden mindestens 1x augeführt
  und dann solange wiederholt, wie die
  Bedingung zutrifft
} while (Bedingung);
```

Ist die Wiederholungs-Bedingung von Anfang an falsch, werden die Befehle im Schleifenkörper bei b1) gar nicht ausgeführt, während die Befehle in b2) mindestens einmal ausgeführt werden, weil die Bedingung erst nach dem Durchlauf überprüft wird. In beiden Fällen muss darauf geachtet werden, dass die Bedingung irgendwann falsch wird, damit der Computer nicht in einer Endlos-Schleife gefangen wird.

Weiß man schon vor Eintritt in den Schleifenkörper, wie oft die Befehle wiederholt werden sollen, bietet sich als Abkürzung von b1) (links) eine Zählschleife an (rechts):

```
int i = 0;
while (i < zahl) {
  Befehle;
  i = i+1;
}
```

Kurzschreibweise:

```
for (int i=0; i < zahl; i = i+1) {
  Befehle;
}
```

4 Konventionen: Klassen- bzw. Programmname groß, alle anderen Namen klein beginnen, nur Buchstaben und Ziffern oder _, keine Leerstellen. Programm-, Klassen-, Datei- sowie GUI-Builder-Objektnamen ohne Umlaute und Sonderzeichen!