Hamster-Programme

1 Hamster

a) Der Hamster kann auf zwei Arten erzeugt werden:

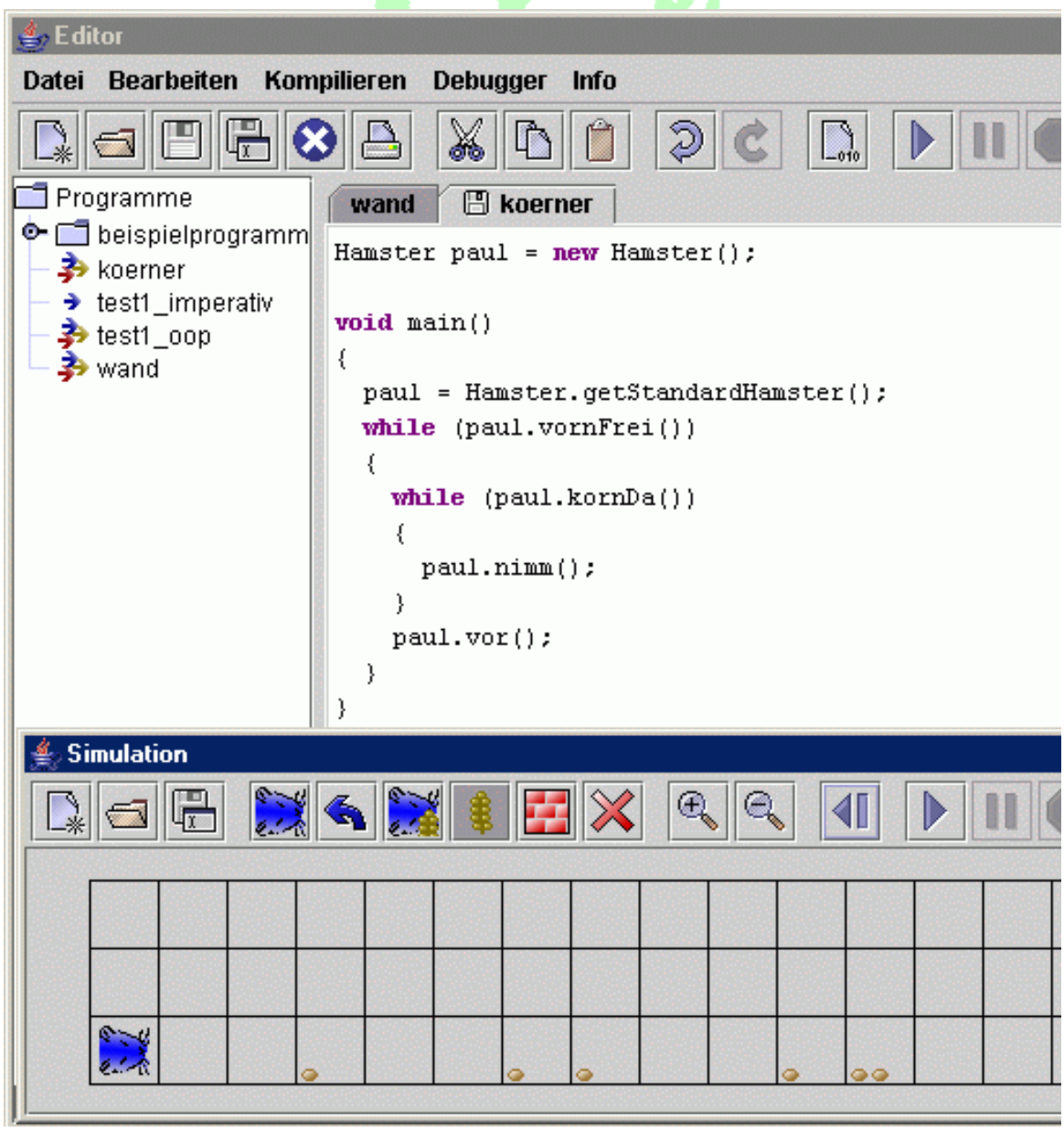
a1) Entweder wird einem bereits definierten Hamster (z.B. *Hamster egon;* oder auch *Hamster egon = new Hamster();*) der im Hamster-Territorium eingezeichnete Hamster mit *egon = Hamster.getStandardHamster();* zugewiesen

a2) oder ein vorher z.B. mit *Hamster egon = new Hamster();* definierter und erzeugter Hamster wird z.B. durch *egon.init(3,5, Hamster.SUED, 0);* auf das Feld mit den Koordinaten (3|5) gesetzt, blickt in südliche Richtung (nach unten) und hat 0 Körner im Maul.

b) Der Hamster kennt folgende Befehle: *vor(); linksUm(); nimm();* und *gib();*. Zum Aufruf muss der Hamstername mit Punkt vorangestellt werden, also z.B. *egon.nimm();* – worauf der Hamster namens *egon* 1 Korn von der Kachel aufnimmt, auf der er steht. Das Korn bleibt im Maul.

c) Weil der Hamster natürlich nur vor gehen kann, wenn vorne auch frei ist (er also nicht vor der Mauer oder dem Rand seines Territoriums steht) oder nur vorhandene Körner aufnehmen bzw. abgeben kann, verfügt er über drei Sensoren, die wahr (*true*) oder falsch (*false*) zurück melden: *vornFrei(); kornDa();* oder *maulLeer();*. Die Sensoren können in den bekannten Kontrollstrukturen wie Verzweigungen (*if*) oder Wiederholungen (*while*) verwendet werden – z.B. *if (egon.maulLeer()) {... }*. Die Verneinung erfolgt mit einem Ausrufezeichen *!*, Kombinationen mit und (&&) oder oder (| |) sind möglich: *while (! egon.vornFrei() && egon.kornDa()) ...*

2 Beispiel eines (objektorientierten!) Hamsterprogramms:



3 Beispiel für ein längeres Programm: Hier erleichtern eigene Methoden den Überblick und das Programmieren. Die Programme sollen auch für ähnliche Territorien gelten.

```
1  Hamster charlie = new Hamster();
2
3  void main()  /*mit dieser Methode geht's los!*/
4  {
5    charlie = Hamster.getStandardHamster();
6    charlieZurWand();
7    charlieAnWandHoch();
8    charlieÜberWand();
9    charlieAnWandRunter();
10   charlieBisZumKorn();
11 }
12
13 void charlieZurWand()
14 {
15   while (charlie.vornFrei())
16   {
17     charlie.vor();  /* Charlie geht immer wieder
18 einen Schritt, solange noch frei ist */
19   }
20 }
21
22 void charlieAnWandHoch()
23 {
24   while (!charlie.vornFrei())
25   {
26     charlie.linksUm();
27     charlie.vor();
28     charlieRechtsUm();
29   }
30 }
```

```
31 void charlieRechtsUm()
32 {
33   charlie.linksUm();
34   charlie.linksUm();
35   charlie.linksUm();
36 }
37
38 void charlieÜberWand()
39 {
40   charlie.vor();
41   charlie.vor();
42   charlieRechtsUm();
43 }
44
45 void charlieAnWandRunter()
46 {
47   charlieZurWand();
48   charlie.linksUm();
49 }
50
51 void charlieBisZumKorn()
52 {
53   while (! charlie.kornDa())
54   {
55     charlie.vor();  /* Charlie geht immer wieder
56 einen Schritt, solange kein Korn da ist */
57   }
58 }
```