E-Computer

Ergänzungsstunde

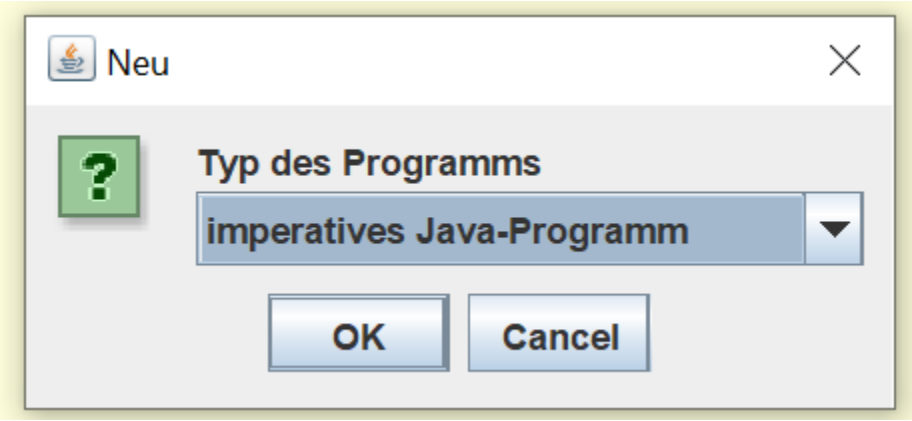
Herr Beinert

Wie lauten die vier Befehle, die wir dem Hamster geben können?

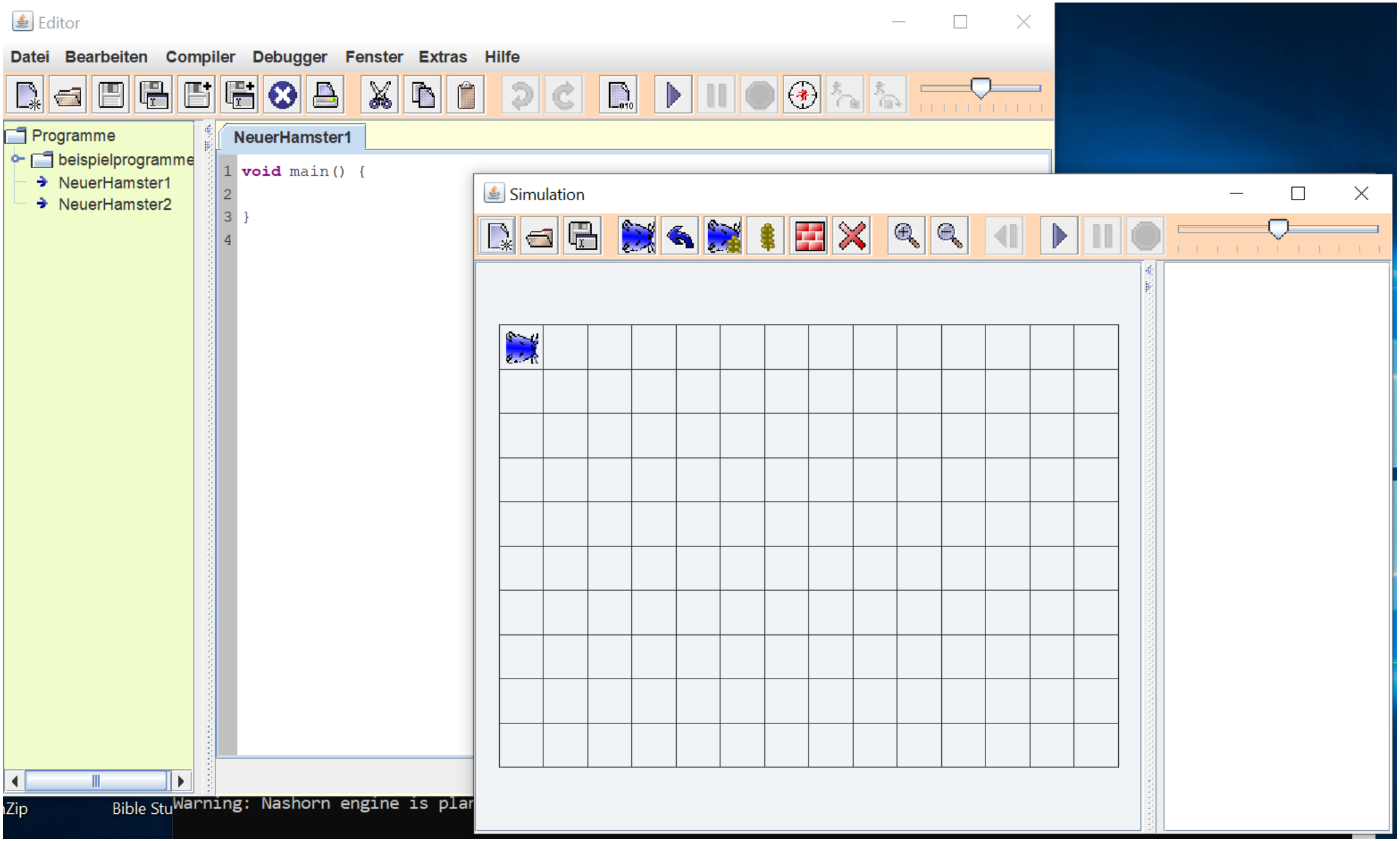
1. vor();
2. linksUm();
3. nimm();
4. gib();

Starten des Hamster-Simulators

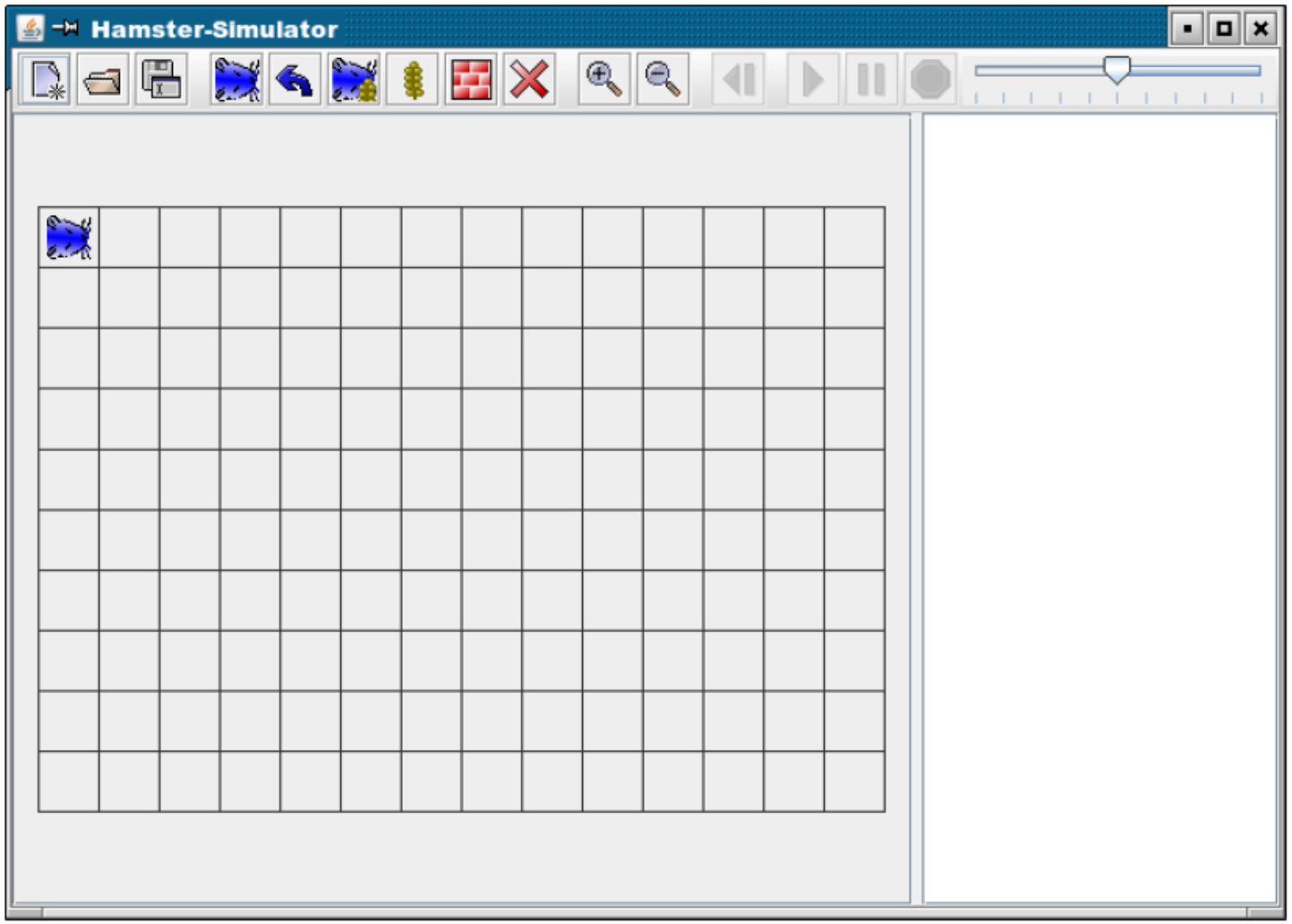
Im Editor-Fenster wählen wir:
Datei → Neu



Dann erhalten wir:


Editor
Datei Bearbeiten Compiler Debugger Fenster Extras Hilfe
Programme
beispielprogramme
NeuerHamster1
NeuerHamster2
NeuerHamster1
1 void main() {
2
3 }
4
Simulation
Zip
Bible Stu
Warning: Nashorn engine is plan

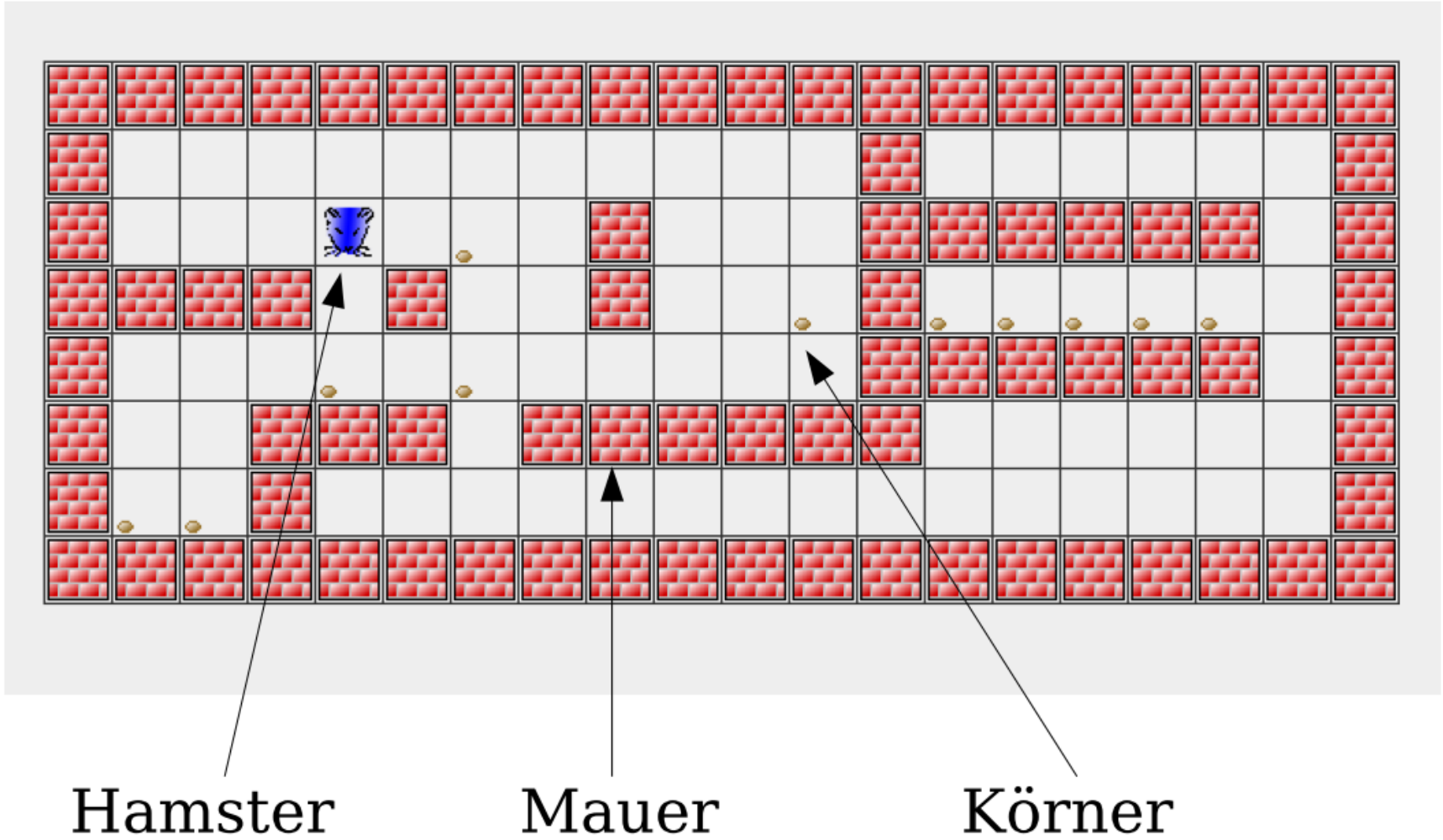
Simulator



- Erstellung und Bearbeitung virtueller Landschaften
- Hamster setzen
- Programmablauf verfolgen

Wichtige Elemente

virtuelle Landschaft



Virtuelle Landschaft erstellen

Landschaft = Territorium

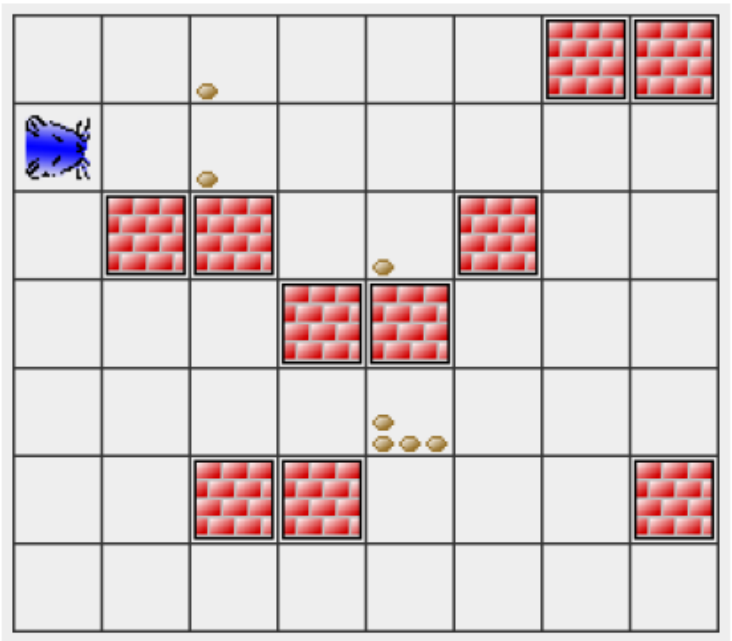

neues Territorium


Mauerkacheln setzen

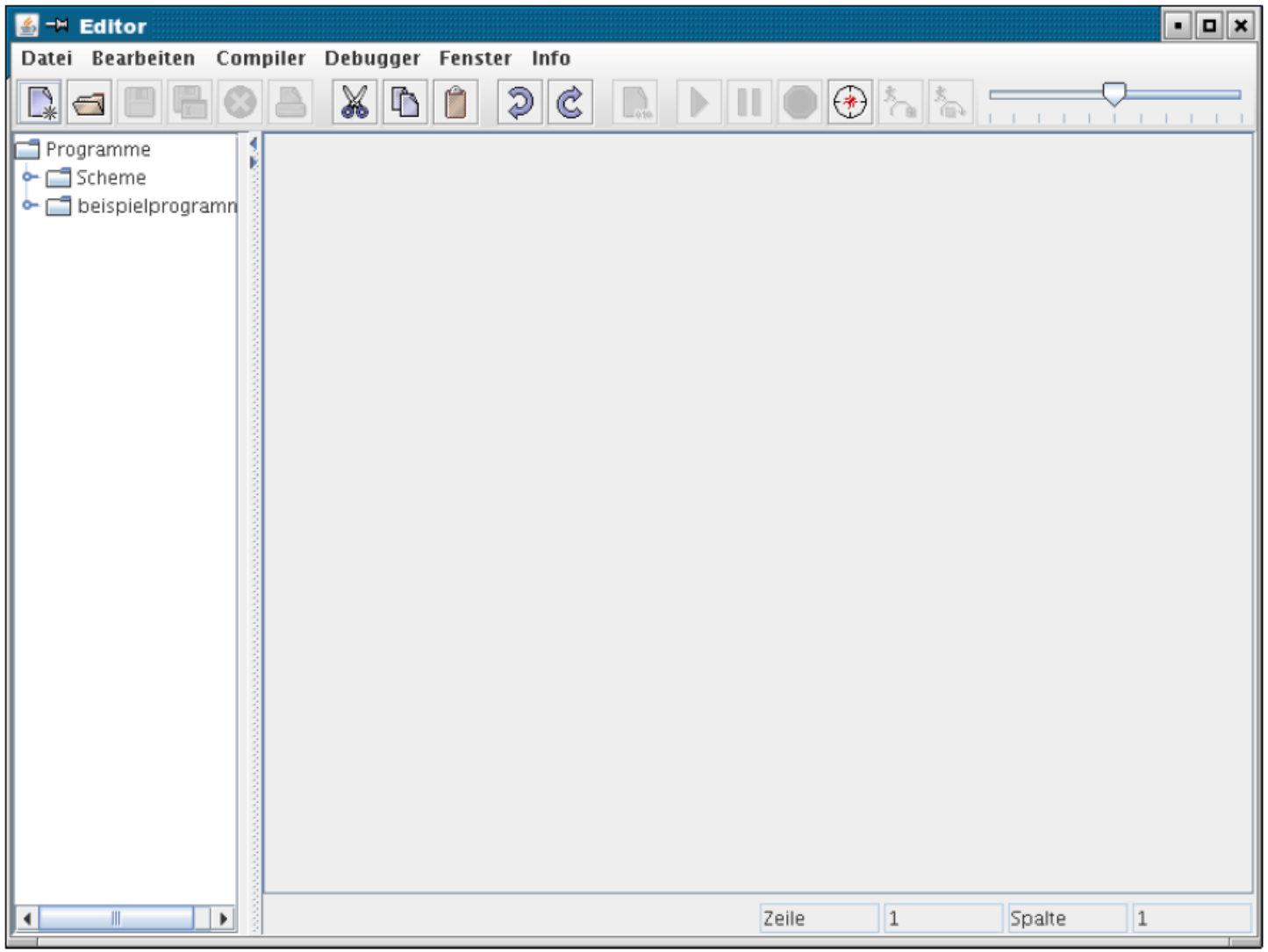

Körner verteilen


Hamster platzieren


Territorium speichern



Editor



Hamster-Programme

- erstellen
- ändern
- speichern
- laden

Programm erstellen

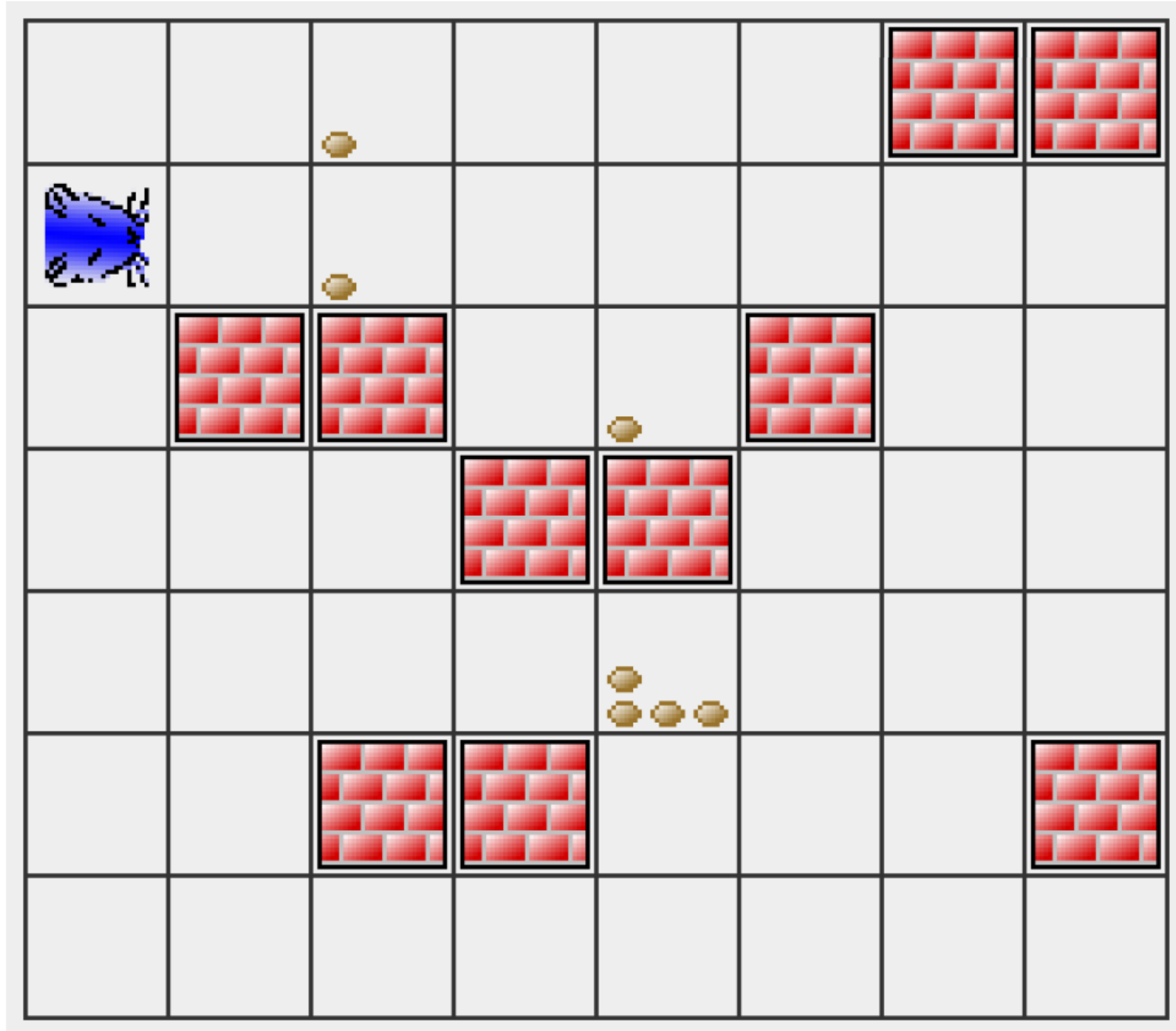
Aufgabe: der Hamster soll auf dem Bsp.-Territorium 2 Körner aufnehmen

neues Programm erstellen

imperatives Programm wählen

Programmname: ham1

Programm speichern



Erstes Programm

Programmname: `ham1`

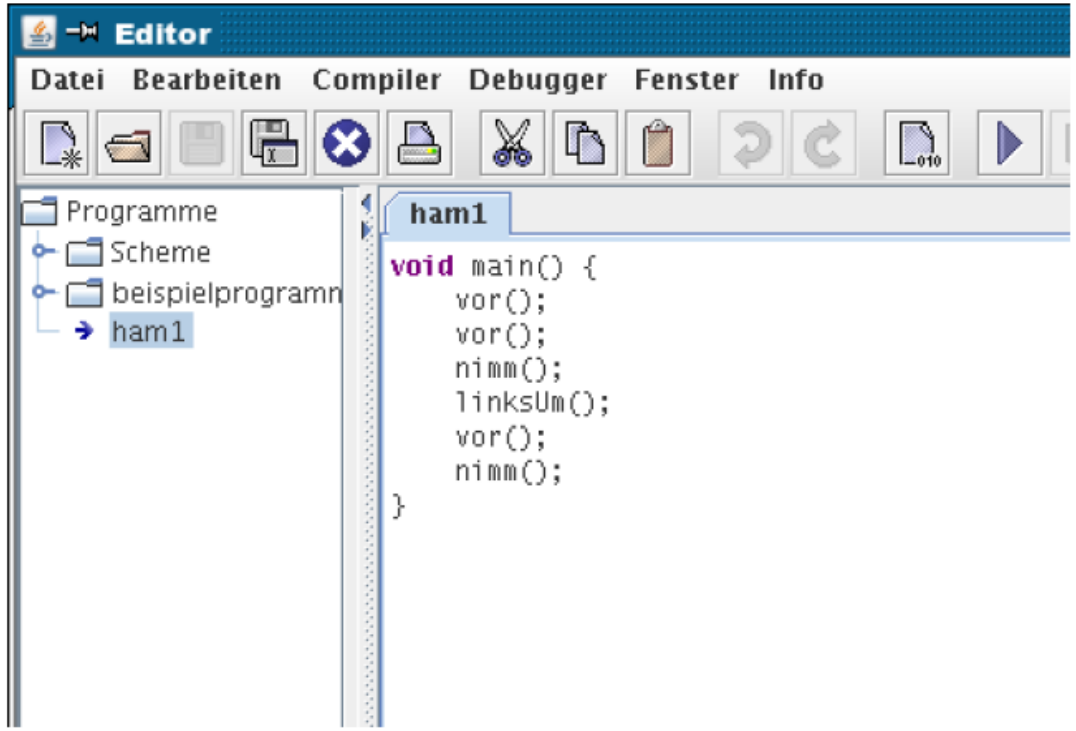
```
void main() {
    vor();
    vor();
    nimm();

    linksUm();
    vor();
    nimm();
}
```

Programm kompilieren (übersetzen)

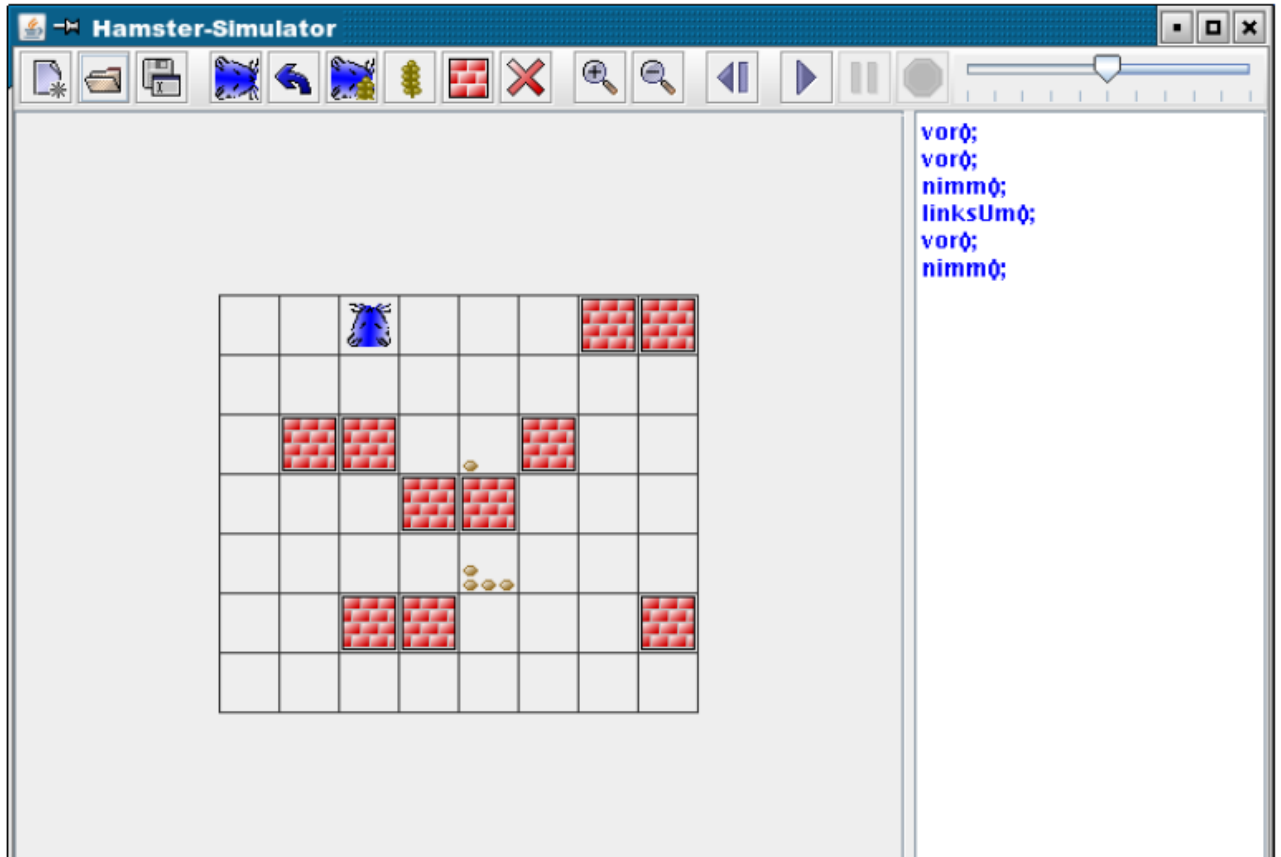
Programm starten

Ergebnis

Editor



Simulator



Eine neue Prozedur erstellen

- Beispiel:

```
void nimm2() {
        nimm();
        nimm();
}
```

- Der Prozedurkopf enthält das Schlüsselwort `void`
- Und den Prozedurnamen `nimm2`
- Der Prozedurrumpf enthält die Anweisungen

Aufgabe 2

Gegeben sei das Hamster-Territorium in Abbildung 1.1 (links).

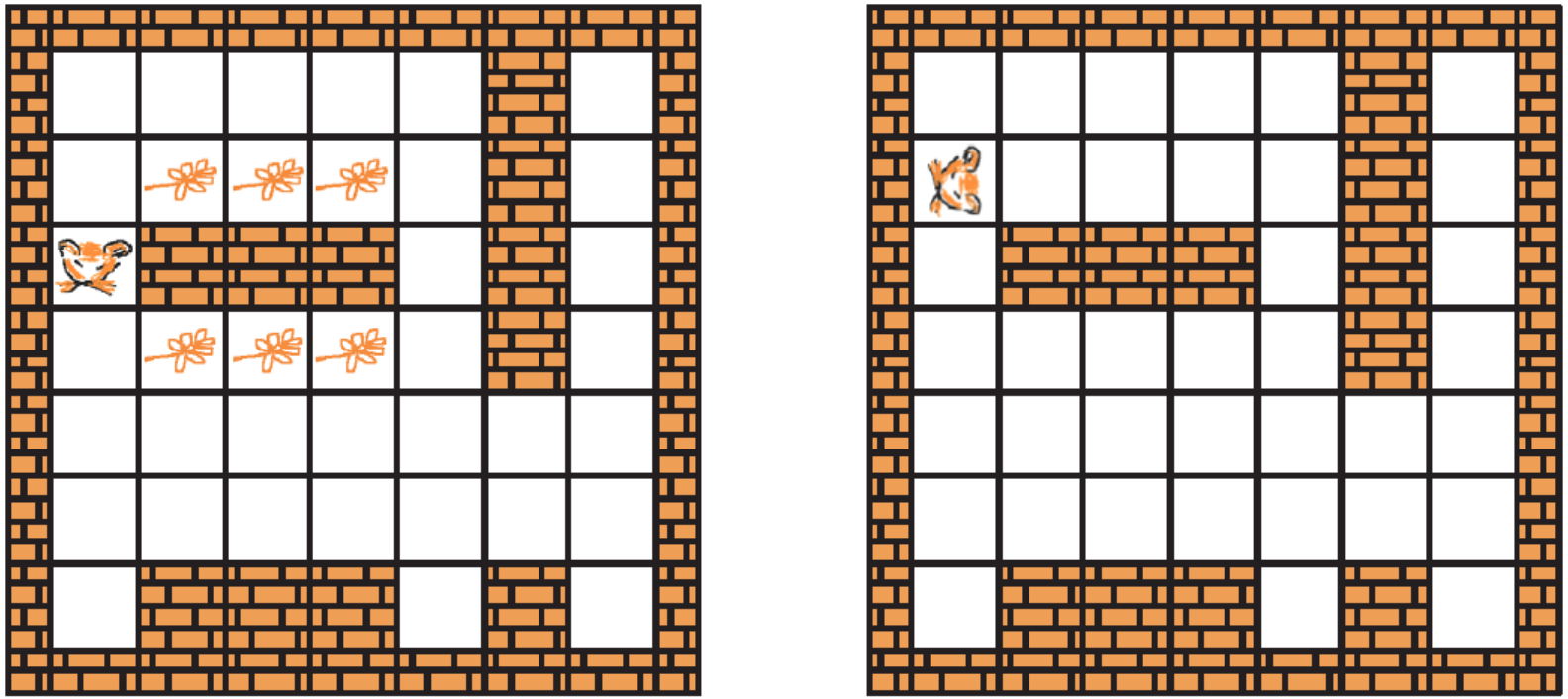


Abbildung 1.1

Dabei kann vorausgesetzt werden, dass auf allen Feldern, auf denen Körner eingezeichnet sind, jeweils genau zwei Körner liegen. Der Hamster soll alle Körner einsammeln. Nach Beendigung des Programms soll das Hamster-Territorium das in Abbildung 1.1 (rechts) skizzierte Erscheinungsbild besitzen.

Aufgabe 3

Gegeben sei das Hamster-Territorium in Abbildung 1.2 (links). Der Hamster habe mindestens sechs Körner im Maul. Er soll auf allen für ihn erreichbaren Feldern jeweils ein Korn ablegen und anschließend in seine Ausgangsposition zurückkehren, d.h. nach Beendigung des Programms soll das Hamster-Territorium das in Abbildung 1.2 (rechts) skizzierte Erscheinungsbild besitzen.

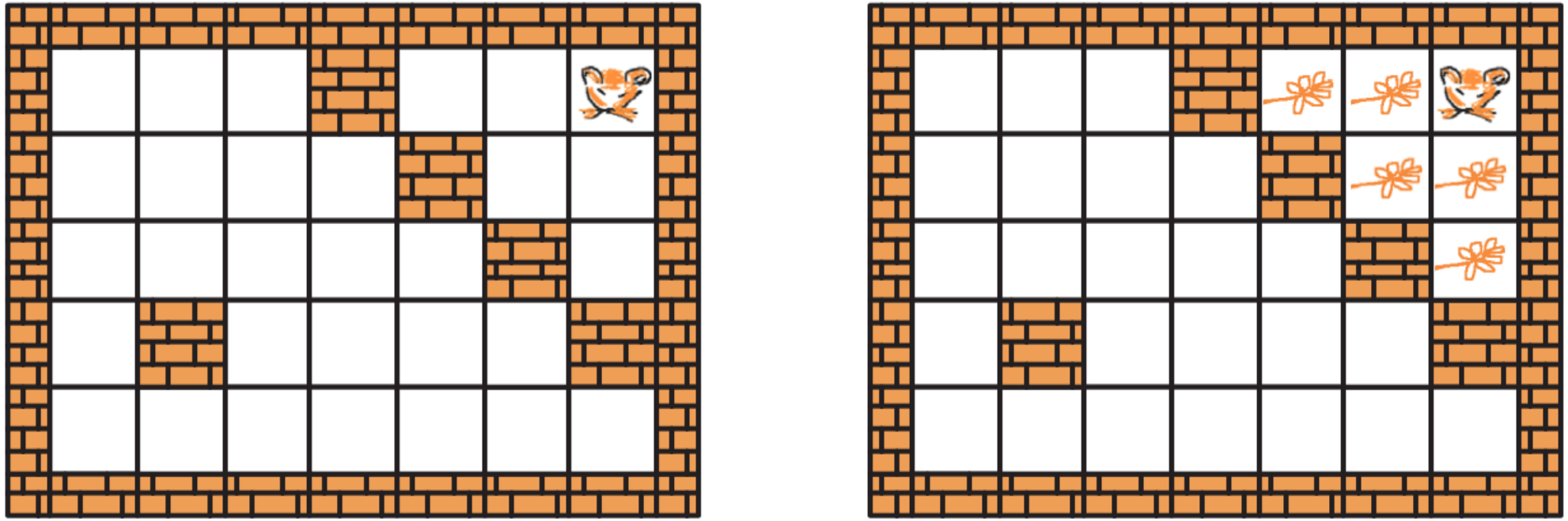


Abbildung 1.2

Aufgabe 4

Gegeben sei das Hamster-Territorium in Abbildung 1.3. Der Hamster soll das Korn am Ende d
Ganges fressen.

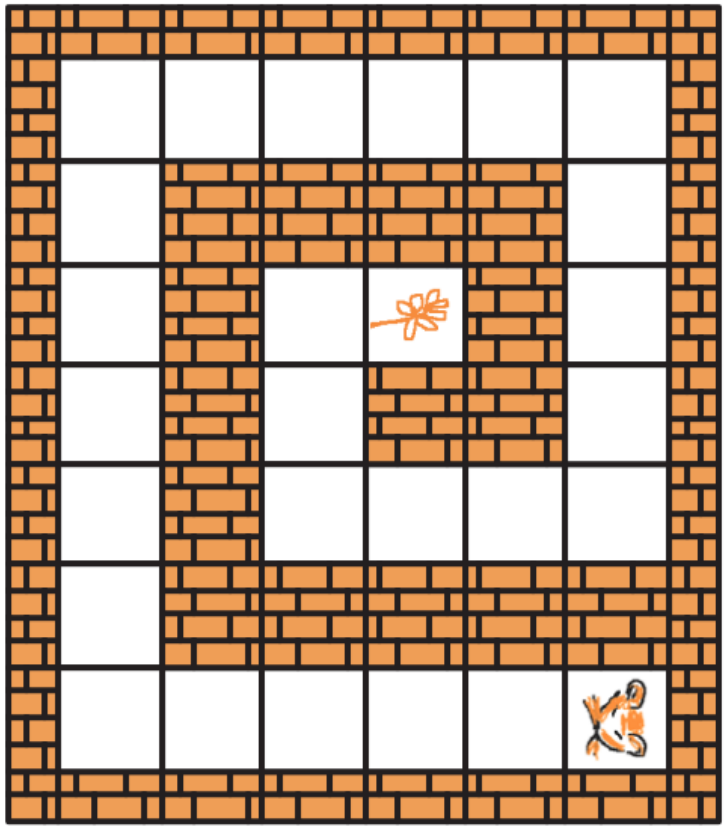


Abbildung 1.3

Neue Prozedur mit Übergabeparameter

- Beispiel:

```
void nimm(int i)
{
  while (i > 0)
  {
    nimm();
    i--;
  }
}
```

- Überlagerung der Funktion `nimm()`
- Wir übergeben der Funktion `nimm()` den Parameter, oder die Variable `i`
- `i` ist eine ganze Zahl, das heißt `int`
- `while` ist eine Schleife, in der Klammer steht die Bedingung, d.h. solange `i` größer als `0` ist
- `i--` bedeutet, dass ich von `i` 1 abziehe.

Aufgabe 5

Es ist Herbst. Der Mais ist über 2 Meter hoch. Da hat sich der Bauer überlegt, für Kinder ein Labyrinth ins Kornfeld zu mähen (siehe Abbildung 1.4). Dieses Vergnügen will sich der Hamster natürlich auch nicht entgehen lassen. Er steht am Anfang des Labyrinths. Finden Sie einen Weg (die freien Kacheln), über den Sie ihn zum Ausgang des Labyrinths steuern.

Programmiere dafür eine überlagerte Funktion `vor(int i)` und benutze sie

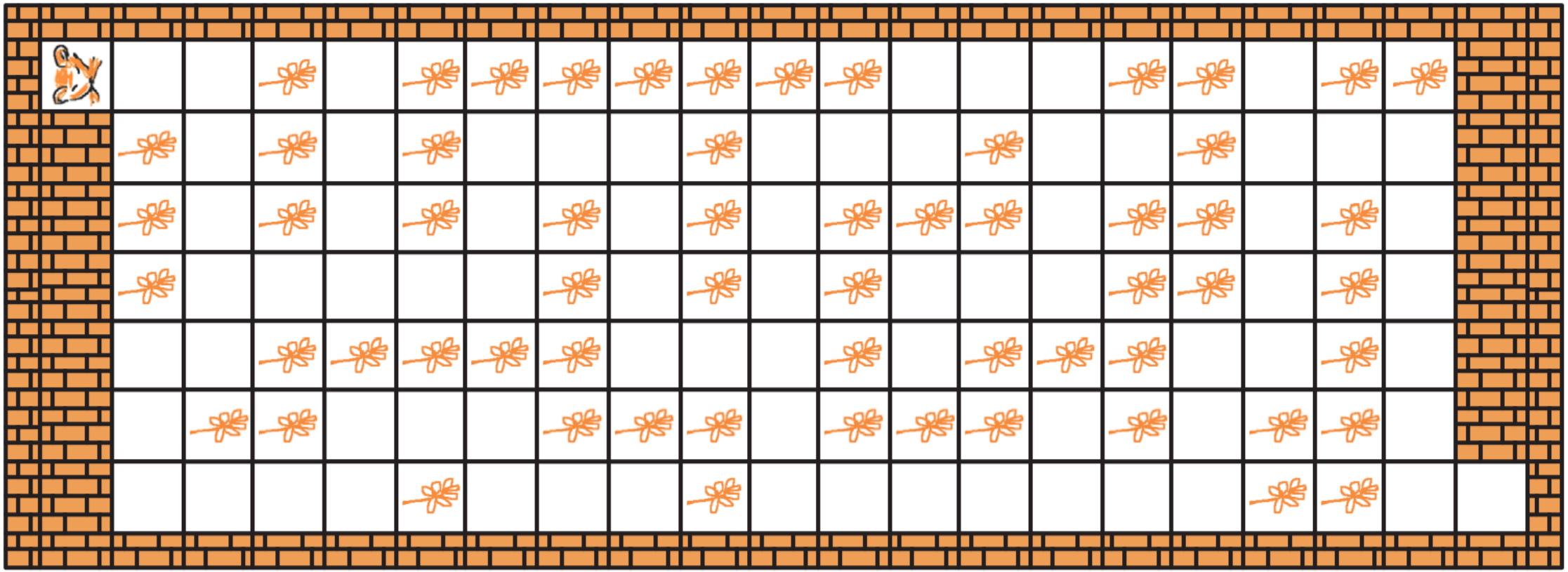


Abbildung 1.4

Der Hamster besitzt folgende Sensoren, die alle true (wahr) oder false (falsch) zurückliefern:

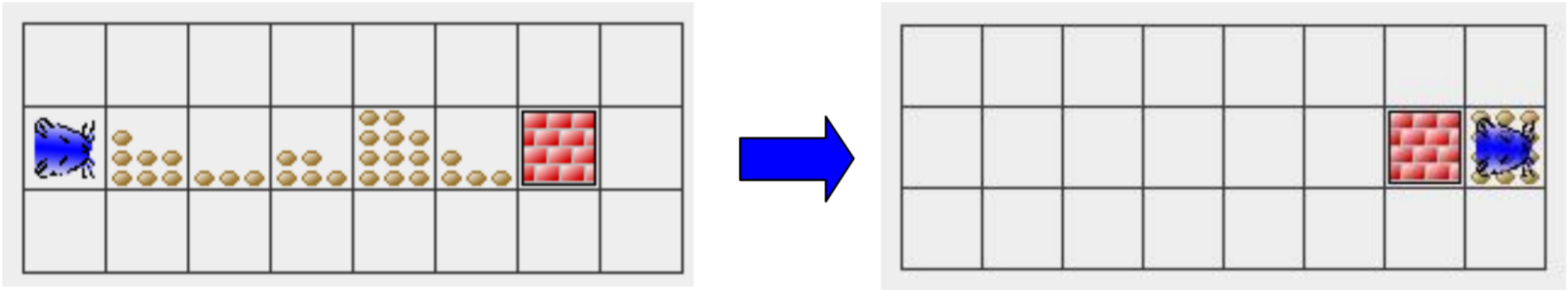
`vornFrei()`	Prüft, ob der Hamster nicht vor einer Wand steht.
`kornDa()`	Prüft, ob auf dem Feld, auf dem der Hamster gerade steht, mindestens ein Korn enthalten ist.
`maulLeer()`	Prüft, ob der Hamster ein Korn im Maul hat.

Beispielprogramm: Solange vorne frei ist, gehe nach vorne.

```
void main()
{
   while ( vornFrei() )
     {
       vor();
     }
}
```

Aufgabe 5

Schreibe ein Programm, das alle Körner hinter eine Mauer bringt, egal wie viele im Weg liegen. Verwende dazu eine `while`-Schleife.



Hinweis: Für das Ablegen der Körner hinter der Mauer benötigst Du die Information, ob der Hamster noch Körner im Mund hat. Da es keinen Sensor gibt, der `maulVoll` lautet, muss man den Sensor `maulLeer()` negieren. Dies geschieht mit einem Ausrufezeichen „!“, das für unser Wort „nicht“ steht:

```
while(!maulLeer()) gib();
```

⇒ Dies bedeutet direkt übersetzt: „So lange das Maul nicht leer ist, lege ein Korn ab!“ oder kurz: „Lege alle Körner ab!“

Variablen

Der Hamster lernt zählen. Was braucht er dazu?

Ein Speicherplatz, wo er sich merken welche Zahl man gerade erreicht hat.

Diesen Speicherplatz nennen wir eine Variable und geben ihm einen Namen z.B. `zaehler` oder einfach `i`.

Wir müssen noch festlegen (definieren), was wir speichern wollen: Zahlen (genauer welche Art von Zahlen) oder Sätze

Der Speicherpatz `byte` für eine Zahl sind 8 Bit (kleinste Einheit).

Wie viele positive und negative Zahlen kann er sich darin merken?

Datentyp	**Bits**	**Wertebereich**
`byte`	8	-128 bis 127
`integer`	32	-2.147.483.648 bis +2.147.483.647

`for`-Schleifen

Manchmal kann es aber auch sinnvoll sein, eine Schleife eine bestimmte Anzahl von Malen zu wiederholen. (100 mal abschreiben)

In Java sieht das so aus

```
for (int zaehler=1; zaehler <=100; zaehler++) schreibAb();
```

Hier wird ein Zaehler angemeldet. Das Schlüsselwort „int“ sagt Java, dass es sich um eine ganze Zahl handelt. „int“ ist die Abkürzung für „integer“ (ganze Zahl).
Der Anfangswert des Zählers ist 1.

Hier wird festgelegt, wie oft die Schleife wiederholt werden soll: 100 mal

Nach jeder Wiederholung der Schleife, wird der Zähler um einen Schritt erhöht. Das erreicht man in Java durch die Ergänzung „++“.

Hier wird festgelegt, was innerhalb der Schleife ausgeführt werden soll: SchreibAb();

Beispiele: `for`-Schleifen

Der Hamster geht 10 Schritte vor und legt ein Korn auf jedes Feld:

```
main()
{
  for (int i = 1; i <= 10; i++)
  {
    gib();
    vor();
  }
}
```

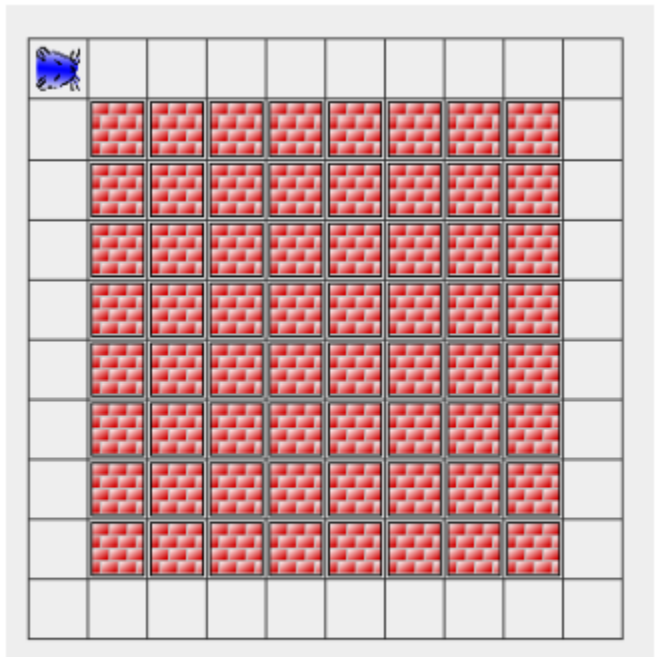
Der Hamster geht 10 Schritte vor und legt ein Korn auf jedes Feld:

```
main()
{
  int zaehler = 0;
  while ( kornDa())
  {
    nimm();
    zaehler++;
  }
```

Aufgabe 6

Verwende eine For-Schleife um den Hamster 19 Schritte vorwärts laufen zu lassen.

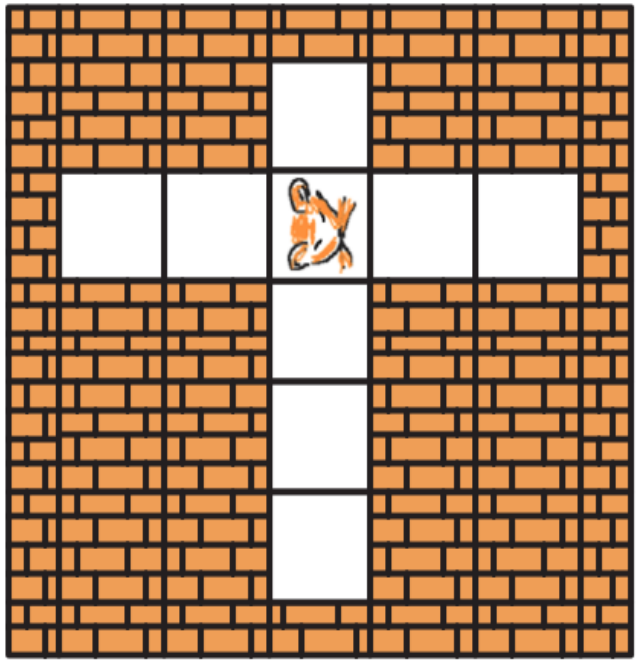
Aufgabe 7

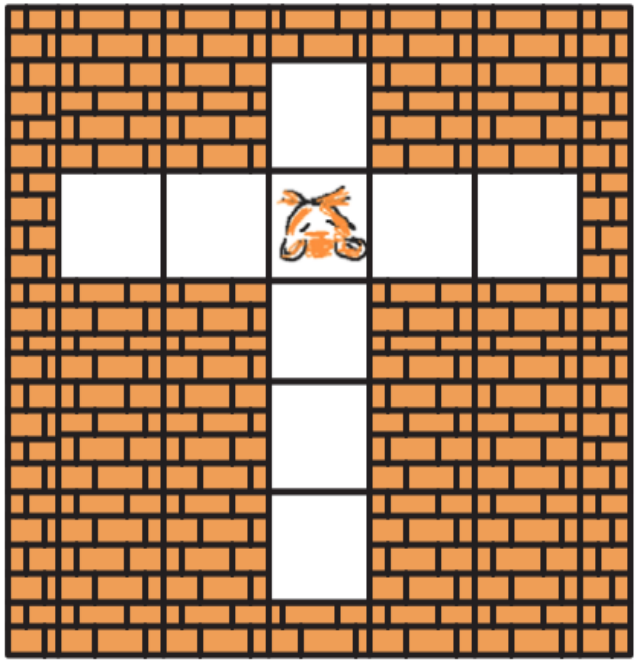


Unser Hamster soll 10 Mal um die Mauer herumlaufen. Wenn er vor eine Wand läuft, ändert er selbständig die Richtung.

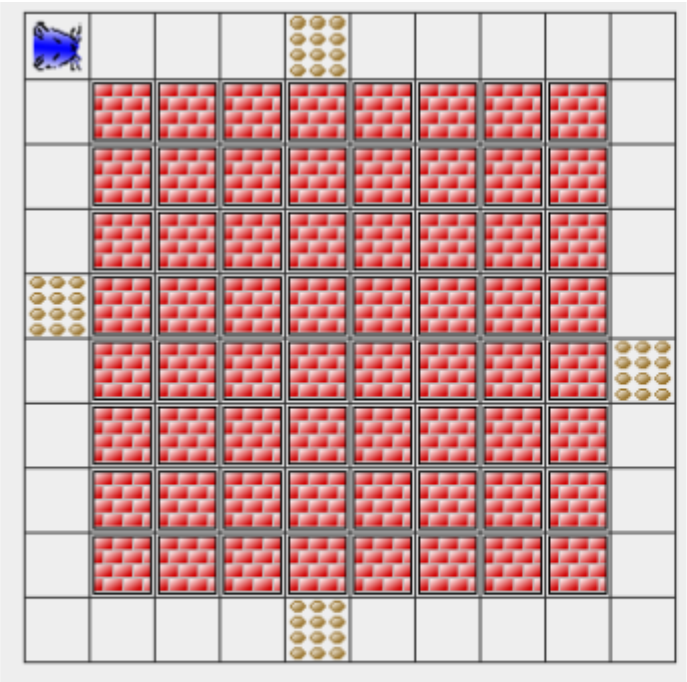
Aufgabe 8

Der Hamster steht in einem Kreuz, wie in Abbildung s.u. skizziert. Er hat neun Körner im Maul. Er soll auf allen neun freien Kacheln genau ein Korn ablegen und dann auf seine Ausgangskachel zurückkehren. Es gibt jedoch ein Problem: Es ist nicht festgelegt, in welche Richtung der Hamster anfangs schaut. Schreiben Sie ein Hamster-Programm, dass die gegebene Aufgabe unabhängig von der anfänglichen Blickrichtung des Hamsters löst.





Aufgabe 9



Der Hamster umkreist die Mauer und sammelt dabei alle Körner ein. Anschließend geht er schrittweise 8 Felder vor und legt dabei auf jedes Feld 5 Körner.