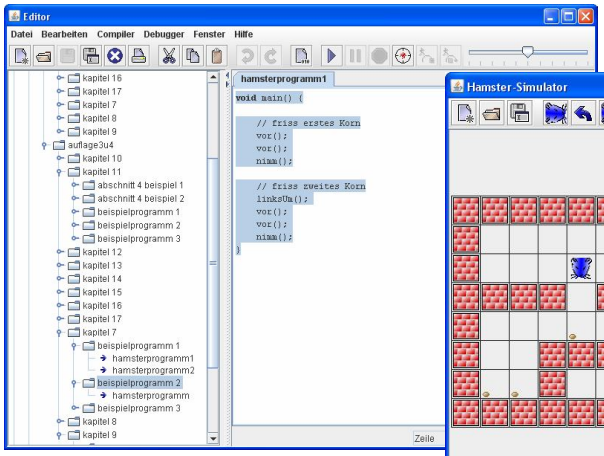
Der Hamster-Simulator und While-Schleifen

Allgemeine Informationen zu While-Schleifen gibt es auf dem Blatt „while-Schleifen mit C#“.

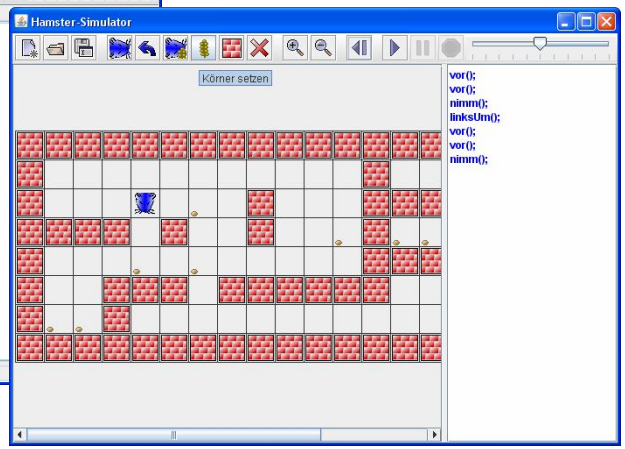
Der Hamster-Simulator

Der Hamster-Simulator ist ein Programm, mit dem Schüler und Studenten Grundlagen der Programmierung spielerisch lernen können. Der Simulator wurde an der Universität Oldenburg entwickelt. Man kann ihn auf der Seite http://www.java-hamster-modell.de herunterladen. Der Hamster ist in der Programmiersprache Java geschrieben, deshalb braucht man zum Betrieb des Hamster-Simulators eine Java-Laufzeitumgebung. Eine Downloadlink für Java gibt es auf der Hamster-Simulator-Homepage.
Java und damit auch der Hamster-Simulator haben die gleiche Syntax (gleiche Schreibweisen) wie C#, deshalb können wir den Hamster ohne große Einführung verwenden.

Der Hamster-Simulator besteht aus zwei Fenstern, dem Editorfenster und das eigentliche Simulatorfenster. Im Editorfenster können die Programme erstellt werden, die der Hamster auf dem Territorium im Simulatorfenster durchführt:



Editorfenster



Simulatorfenster

Die Vorgehensweise bei der Arbeit mit dem Hamster-Simulator ist immer gleich. Man erstellt ein Programm und ein Territorium, kompiliert das Programm und führt es aus. Das Programm wird dabei im Editor immer zwischen den geschweiften Klammern in der Main-Methode erzeugt:

```
void main(){
  // hier wird das Programm erstellt
}
```

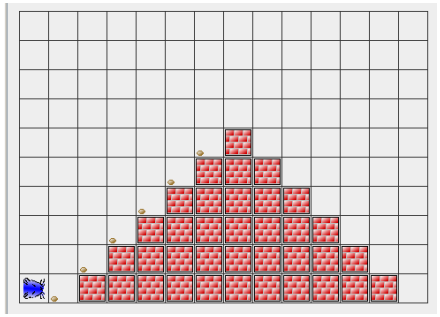

Aktionen des Hamsters:

Der Hamster kann die folgenden Aktionen ausführen:

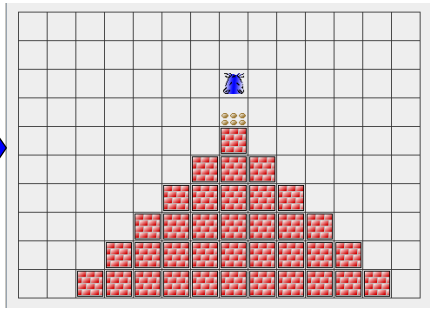
`vor();`	Der Hamster geht einen Schritt nach vorne.
`linksUm();`	Der Hamster dreht sich um 90° nach links.
`nimm();`	Der Hamster nimmt ein Korn auf.
`gib();`	Der Hamster legt ein Korn auf das Territorium.

Aufgabe 1:

- Lege im Ordner „Eigene Dateien“ einen Unterordner „Hamsterprogramme“. Starte den Hamster-Simulator. Platziere das Editor- und Simulatorfenster so, dass sie sich nicht überlappen (u. U. Größen anpassen!).
- Erzeuge das unten bei der Anfangssituation abgebildete Territorium (14 x 10) und speichere es unter dem Namen „berg“.
- Schreibe ein Programm (Typ: „Imperatives Programm“), so dass die unten bei der Endsituation abgebildete Situation am Programmende entsteht. Speichere das Programm ebenfalls unter dem Namen „berg“. Die beiden Dateien müssen zwar nicht gleich benannt sein, aber es ist hilfreich.
- Kompiliere und teste das Programm.



Anfangssituation



Endsituation

Da Java und der Hamster-Simulator die gleiche Syntax wie C# besitzen, können wir - wie gewohnt – mit if-Anweisungen und for-Schleifen arbeiten.

Aufgabe 2:

Verändere das Programm von Aufgabe 1 so, dass die sich wiederholenden Anweisungen in einer for-Schleifen liegen

Sensoren des Hamsters

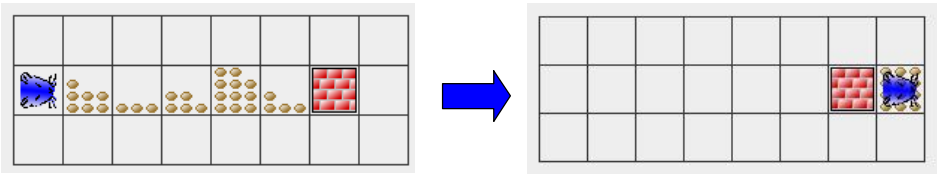
Wenn in Aufgabe 1 oder 2 nur ein Korn nicht da liegt, wo es sein sollte, bricht der Hamster-Simulator mit einer Fehlermeldung ab. Auch wenn man den Hamster auf ein Territorium loslässt, auf dem vorher nicht bekannt ist, wie viele Körner vorhanden sind, gibt es Probleme.
⇒ Der Hamster kann also bisher nicht auf seine Umgebung reagieren.
Im Hamster-Simulator wird dieses Problem durch die Sensoren des Hamsters gelöst.

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.

Aufgabe 3:

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!“

Aufgabe 4: Geschachtelte Schleifen

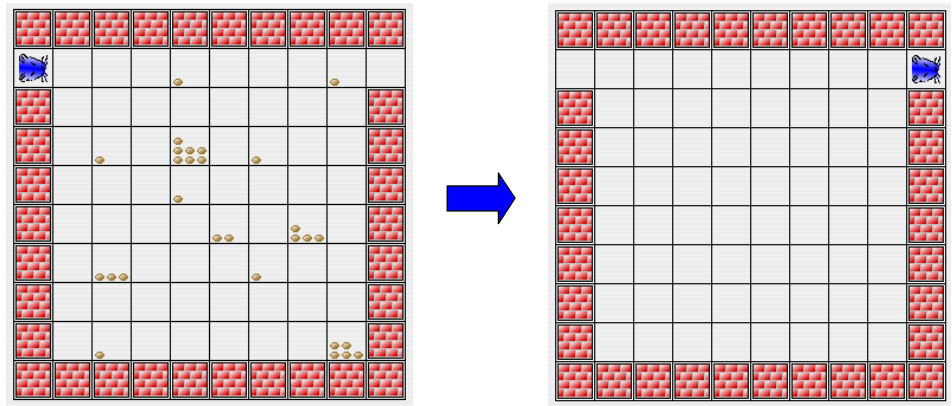
a) Was passiert, wenn das folgende Programm ausgeführt wird?

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

b) Baue ein Territorium, das vom folgenden Programm gelöst wird, und teste das Programm!

Aufgabe 5: Der Allesfresser

Erzeuge folgendes Territorium (10x10), auf dem Du wahllos Körner verteilen kannst. Der Hamster soll alle finden, fressen und am Ende im Ausgang rechts oben sitzen. Du kannst hierzu auch Teile des Programms aus Aufgabe 4 verwenden.



Aufgabe 6: Round Robin (Aufgabe für sehr Schnelle!)

Der Hamster soll zunächst auf einem Weg unbekannter Länge zu einer Mauer alle Körner fressen, die er findet.
Vor der Mauer beginnt nun die nächste Aufgabe: Der Hamster soll so oft um die Mauer tanzen, wie er Körner im Maul hat. Dabei soll er immer dann, wenn er gerade hinter der Mauer vorbeitanzt, ein Korn ablegen.
Teste Dein Programm mit unterschiedlichen Ausgangssituationen.

```
void main() {
    // sammle bis Mauer
    […]
    // in Ausgangsposition drehen
    […]
    // links herum tanzen
    […]
}
```