

DER HAMSTERSIMULATOR

Selbstlernübungen mit der Programmiersprache Java



Die nachfolgenden Übungen sollen es dir ermöglichen, selber die Syntax der Programmiersprache Java zu erlernen. Dabei gibt es eine Reihe von „Kopfnüssen“ zu knacken, die das Verständnis für Programmialgorithmen vertiefen.

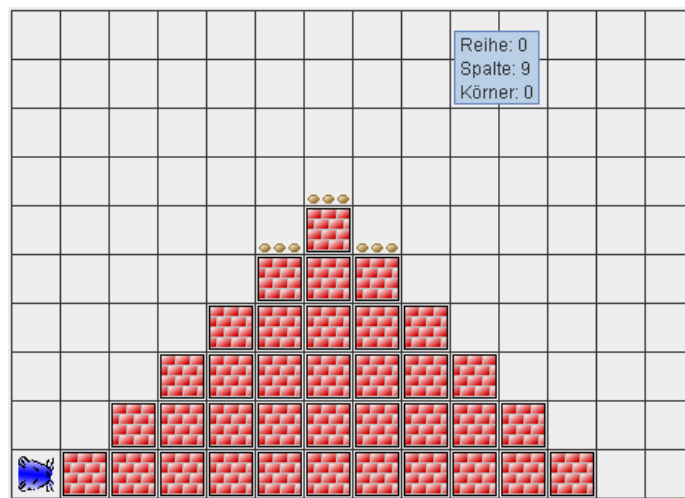
Grundlegende Informationen zu der Programmierumgebung findest du in der Datei „Hamster Handbuch.pdf“, die ebenfalls hier im Downloadbereich zur Verfügung steht.

01 Der Hamstersimulator

Prozeduren ersparen Tipparbeit

Aufgabe 1

- Für die nächste Aufgabe erstellt ihr zunächst ein Hamsterteritorium nach folgendem Muster.
- Erstellt ein neues Unterverzeichnis im Ordner **Hamster\Programme** mit Namen **\pyramide**
- Speichert das Territorium unter dem Namen **Tpyramide** in dem neuen Unterverzeichnis.
- Erzeugt ein neues Programm und speichert es im selben Verzeichnis unter dem Namen **prPyramide**.



Der Hamster soll bis zur Spitze der Pyramide laufen und dabei alle Körner, die er findet ins Maul nehmen. Danach läuft er auf der anderen Seite der Pyramide wieder herunter und legt die Körner am Boden ab.

Für das Programm sollt ihr nicht mehr als **50 Zeilen Code** schreiben. Dazu solltet ihr überlegen, welche Schritte ihr in Prozeduren zusammenfassen könntet.

Für die Aufgabe gelten folgende Spielregeln:

- In jeder Zeile steht nur ein Befehl, z.B. „vor()“;
- Prozeduren werden wie folgt auf Zeilen verteilt:

```
void machViel() {  
    machWas();  
    machWas();  
    machWas(); }  
}
```

- Hinter dem Hauptprogramm und den einzelnen Prozeduren folgt immer eine Leerzeile;

03 Der Hamstersimulator

Verzweigungen¹

Verzweigungen sind eine weitere mächtige Kontrollstruktur in Programmiersprachen. Mit ihnen kann eine Handlung abhängig von einer Bedingung gemacht werden.

Beispiel:

Mutter sagt zu Fritz: „Wenn du genug Geld hast, kaufe dir ein Mettbrötchen.“

In Java könnte diese Bedingung so aussehen

```
if2 (genugGeld()) kaufeBroetchen();
```

oder, falls mehrere Befehle ausgeführt werden sollen

```
if (genugGeld()) {  
  geheInDieMensa();  
  kaufeBroetchen();  
  verlasseDieMensa(); }  

```

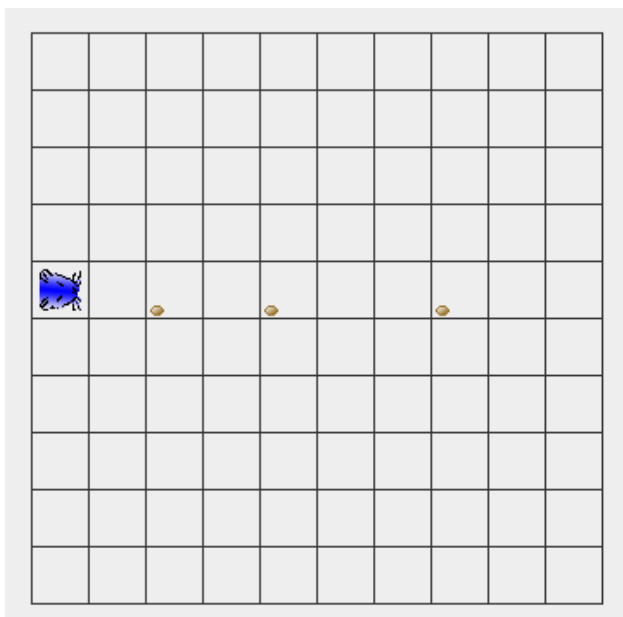
Aufgabe

Schreibe diesen Code mit Hilfe einer IF-Verzweigung und einer while-Schleife.

Unterverzeichnis \ifVerzweigung01

Programmname: prIfVerzweigung01

Territorium: tlfVerzweigung01



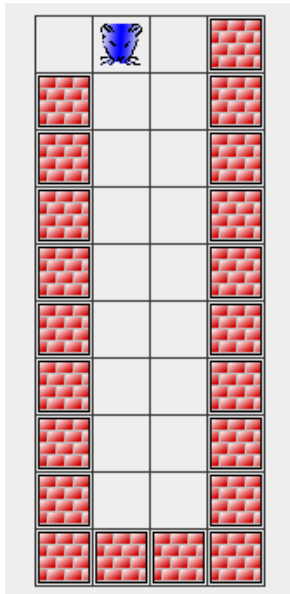
Der Hamster soll von links nach rechts bis zur Wand laufen.
Wenn er unterwegs ein Korn findet, soll er es fressen.

¹ Vokabel lernen!

² Wieder eine Vokabel

Für Profis

Gehe zurück zum Ordner: \SchleifeMitEingangProfi
und öffne das Territorium „t SchleifeMitEingangProfi“
und das Programm „prSchleifeMitEingangProfi“

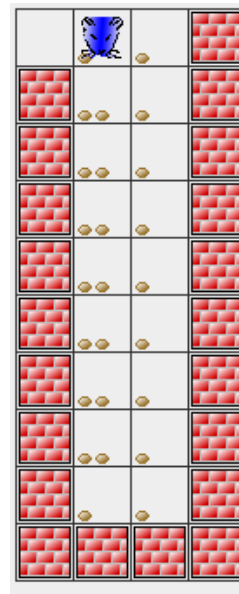


Aufgabe:

Nach dem erfolgreichen Ablauf
eures Programmes steht unser
Hamster nun ziemlich satt in dieser
Position (siehe links).

Er soll nun erneut solange das La-
byrinth durchlaufen und auf jedes
Feld ein Korn legen, bis sein Maul
leer ist.

Das könnte dann so aussehen, wie
auf dem Bild rechts.



03 Der Hamstersimulator

Verzweigungen¹

Verzweigungen sind eine weitere mächtige Kontrollstruktur in Programmiersprachen. Mit ihnen kann eine Handlung abhängig von einer Bedingung gemacht werden.

Beispiel:

Mutter sagt zu Fritz: „Wenn du genug Geld hast, kaufe dir ein Mettbrötchen.“

In Java könnte diese Bedingung so aussehen

```
if2 (genugGeld()) kaufeBroetchen();
```

oder, falls mehrere Befehle ausgeführt werden sollen

```
if (genugGeld()) {  
  geheInDieMensa();  
  kaufeBroetchen();  
  verlasseDieMensa(); }  

```

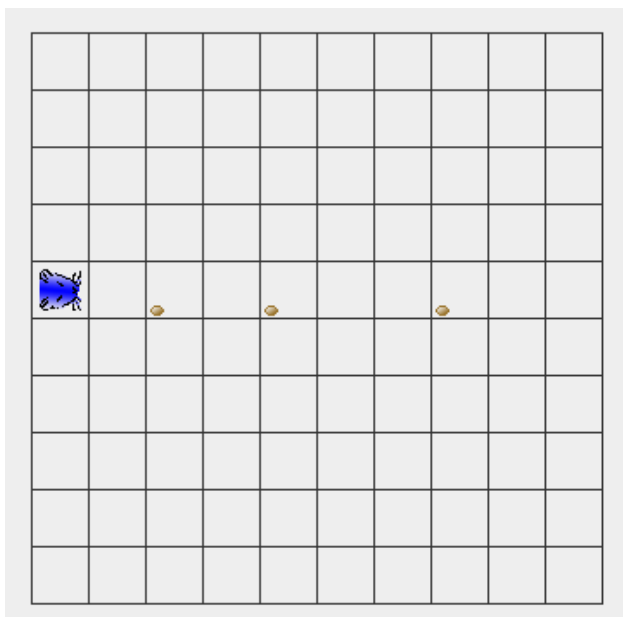
Aufgabe

Schreibe diesen Code mit Hilfe einer IF-Verzweigung und einer while-Schleife.

Unterverzeichnis \ifVerzweigung01

Programmname: prIfVerzweigung01

Territorium: tIfVerzweigung01



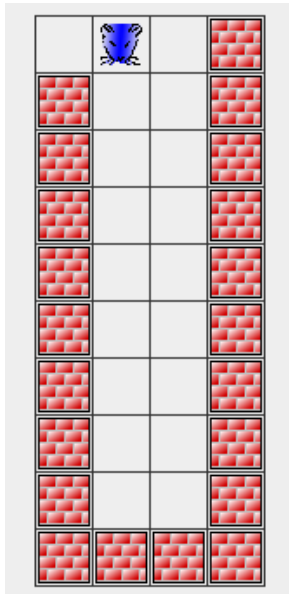
Der Hamster soll von links nach rechts bis zur Wand laufen.
Wenn er unterwegs ein Korn findet, soll er es fressen.

¹ Vokabel lernen!

² Wieder eine Vokabel

Für Profis

Gehe zurück zum Ordner: \SchleifeMitEingangProfi
und öffne das Territorium „t SchleifeMitEingangProfi“
und das Programm „prSchleifeMitEingangProfi“

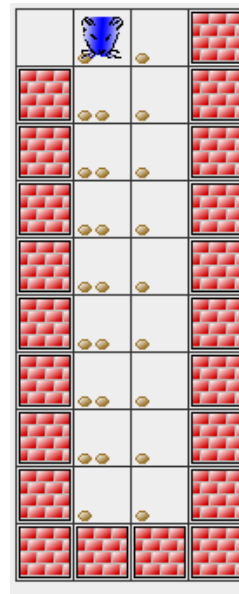


Aufgabe:

Nach dem erfolgreichen Ablauf
eures Programmes steht unser
Hamster nun ziemlich satt in dieser
Position (siehe links).

Er soll nun erneut solange das La-
byrinth durchlaufen und auf jedes
Feld ein Korn legen, bis sein Maul
leer ist.

Das könnte dann so aussehen, wie
auf dem Bild rechts.



02 Else hilft, wenn nichts mehr geht

Mutter sagt zu Fritz:

Wenn du deine Hausaufgaben gemacht hast, bekommst du ein Lob von Papa und darfst danach ins Kino, **andernfalls** bekommst du Ärger mit Papa und danach Stubenarrest.

Auch Programme können zwischen zwei Möglichkeiten unterscheiden. Zu diesem Zweck haben wir bereits die IF-Verzweigung kennengelernt. Aber was soll passieren, wenn die Bedingung, die If erwartet hat, nicht eingetreten ist?

Dann kann das Programm entweder einfach beim nächsten Schritt weitermachen, oder aber eine alternative Handlung ausführen. Dafür benötigen wir das Schlüsselwort „else“.

In Java sieht das so aus.

```
if (hausAufgabenGemacht()) {  
    papaLobtMich  
    darfInsKino(); }  
else  
    {aergerMitPapa();  
     kriegeStubenArrest();}
```

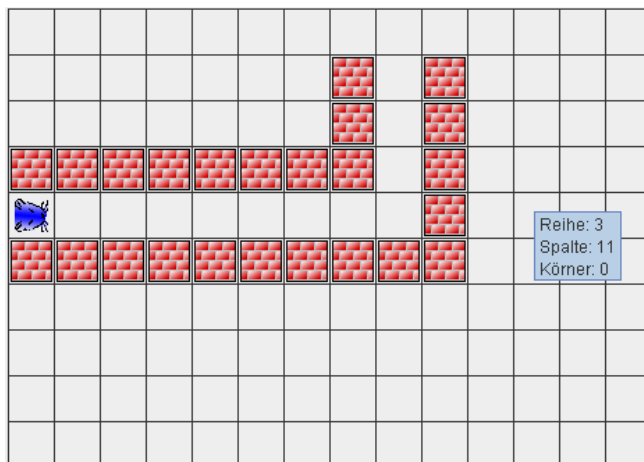
Aufgabe:

Verzeichnis = hamsterSchlau01

Programm = prHamsterSchlau01

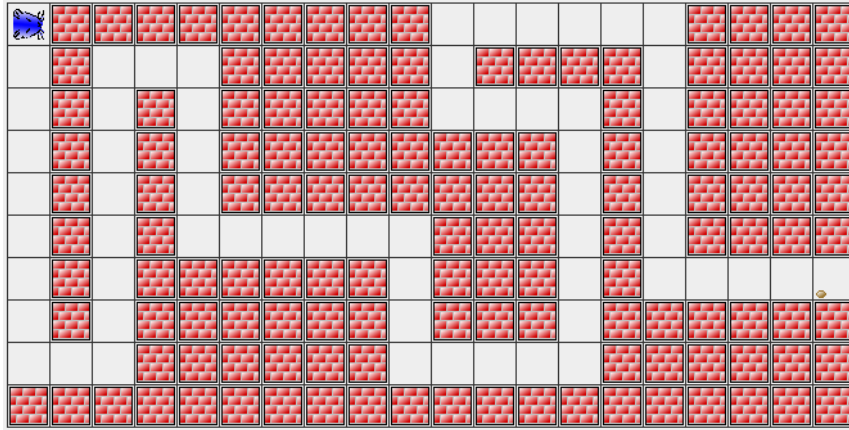
Territorium = tHamsterSchlau01

Unser Hamster soll bis an das Ende des Labyrinths finden. Dabei weiß er vorher nicht, in welche Richtung er an der Abzweigung abbiegen muss. Er soll also eine selbständige Entscheidung treffen. Das gleiche Programm muss also für Rechts- und Linkskurven gleichermaßen funktionieren. (> 15 Zeilen)



Für Profis

Verzeichnis = hamsterSchlau02
Programm = prHamsterSchlau02
Territorium = tHamsterSchlau02



Unser Hamster hat Hunger. Es soll nun völlig selbständig den Weg durch dieses Labyrinth finden. Er weiß weder, wieviele Kurven er laufen muss, noch ob es an den Verzweigungen nach rechts oder links geht. Er weiß nur, dass am Ende ein Korn auf ihn wartet, welches er gierig fressen wird. Danach geht der Hamster schlafen, d. h. das Programm endet. (<23 Zeilen = Meisterklasse)

Der Hamstersimulator 06

For-Schleifen mit festgelegter Wiederholungszahl und Variablen

Fritzchen hat im Informatikraum ein tiefendes Softeis gegessen und damit drei Tastaturen vollständig zugeschmiert. Natürlich gibt es dafür eine Sonderaufgabe. Der Lehrer schreibt an die Tafel „Ich darf im Unterricht kein Softeis essen.“

Dann sagt er

„Das schreibst du jetzt ab bis zum Klingeln.“

Fritzchen grinst. Es schellt in drei Minuten. Doch auch der Lehrer hat eine Uhr und er verbessert sich schnell:

„Du wiederholst das Abschreiben dieses Satzes hundert Mal!“. Mist!

Bisher haben wir eine Schleife mit einer Abbruchbedingung kennengelernt. Das heißt ein Vorgang wird so lange wiederholt, bis eine bestimmte Bedingung erfüllt ist. (Abschreiben bis zum Klingeln).

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: 50 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();

Aufgaben:

1. Der Neunzehner-Lauf*

Verzeichnis = neunzehnerlauf

Programm = prNeunzehnerlauf

Territorium = tNeunzehnerlauf



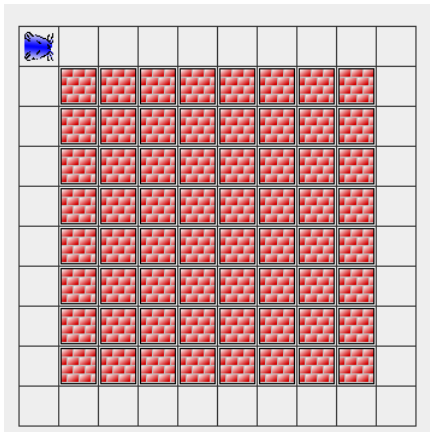
Verwende eine For-Schleife um den Hamster 19 Schritte vorwärts laufen zu lassen.

2. Kreisläufer **

Verzeichnis = kreislaeuer

Programm = prKreislaeuer

Territorium = tKreislaeuer



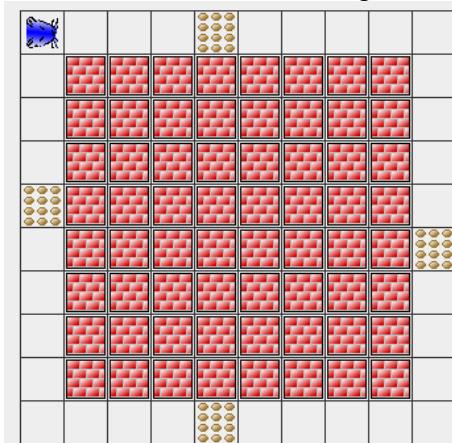
Unser Hamster soll 10 Mal um die Mauer herumlaufen. Wenn er vor eine Wand läuft, ändert er selbständig die Richtung.

3. Häufchenleger***

Verzeichnis = haefchenleger

Programm = prHaeufchenleger

Territorium = tHaeufchenleger



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.