Java-Hamster

Der Java-Hamster ist ein Programmpaket, das auf spielerische Weise einen Einstieg in die Grundkonzepte der Programmierung vermittelt. Der Simulator lässt sich auf

http://www.java-hamster-modell.de

als zip-File herunterladen. Damit er funktioniert ist die Installation des Java-Development-Kits (JDK) erforderlich, das unter

http://www.oracle.com/technetwork/java/javase/downloads/index.html

heruntergeladen werden kann (bzw. suchen Sie im Internet nach „JDK download“).

Der Hamster verfügt „von Natur aus“ über einige wenige Grundbefehle:

Anweisungen:	
`vor();`	Ein Feld nach vorne
`linksUm();`	Drehen nach links
`nimm();`	Aufnehmen eines Korns
`gib();`	Hinlegen eines Korns.

Abfragen:	
`vornFrei()`	wenn Abfrage wahr ist, ist vor dem Hamster ist keine Mauer.
`kornDa()`	Abfrage, ob auf dem aktuellen Feld mindestens ein Korn liegt.
`maulLeer()`	Abfrage, ob keine Körner derzeit im Maul des Hamsters sind.

Der Hamster führt zunächst die Hauptroutine mit dem Namen `main()` aus, die z.B. so aussehen könnte:

```
void main() {
   vor();          // Hamster geht einen Schritt nach vorne
   linksUm();      // und dreht sich nach links
}
```

Kompliziertere Befehle können aus den Grundbefehlen Befehlen zusammengesetzt und mit einem neuen Namen versehen werden, z.B. ein neuer Befehl `drehUm()` wird folgendermaßen **definiert**:

```
void drehUm() {
   linksUm();      // 2x nach links drehen entspricht einem Umdrehen
   linksUm();
}
```

Der neue Befehl kann wie die Grundbefehle **ausgeführt**/aufgerufen werden:

```
void zurueck() {   // zurückgehen =
   drehUm();       //      umdrehen (eigener Hamster-Befehl),
   vor();          //      ein Feld „zurück“ gehen und
   drehUm();       //      wieder in die Anfangsrichtung drehen.
}
```

Der Hamster kann seine unmittelbare Umgebung wahrnehmen. Dies kann man in folgenden Konstrukten verwenden:

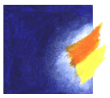
Abfrage: if = „wenn“

```
if ( kornDa() ) { // wenn ein Korn auf dem Feld
  nimm();  // liegt, wird dies ausgeführt,
} else {   // ansonsten (=es liegt kein Korn da)
  gib();   // wird dies ausgeführt.
}          // if-else: „entweder-oder“
```

Schleife: while = „solange“

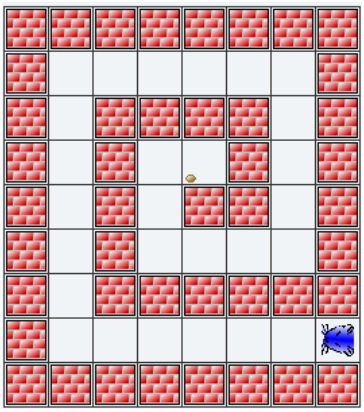
```
while ( !maulLeer() ) {  // '!' bedeutet NICHT
  gib(); // wird ausgeführt,
         // solange das Maul nicht(!) leer ist
         //         (d.h. Körner im Mund sind).
}        // => also mehrmals!
```

Hinweis: if-Abfragen und while-Schleifen können mehrere Befehle in den geschweiften Klammern haben und beliebig ineinander verschachtelt werden, d.h. eine Schleife kann in einer Schleife sein.



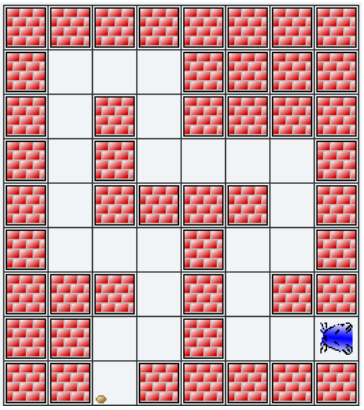
Vorbereitung: Erstellen Sie ein Verzeichnis `H:\AIT\01_Hamster` in das Sie alle Programme speichern.

Aufgabe 1: Schnecke / Labyrinth

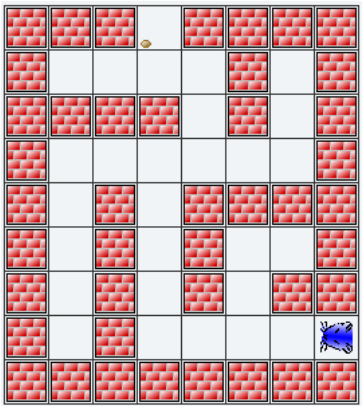


1.1. Erstellen Sie ein Programm (*Datei→Neu: imperatives Programm*) mit dem Namen *A11_Schnecke*.

1.2. Lassen Sie den Hamster im nebenstehenden Territorium das Korn fressen.

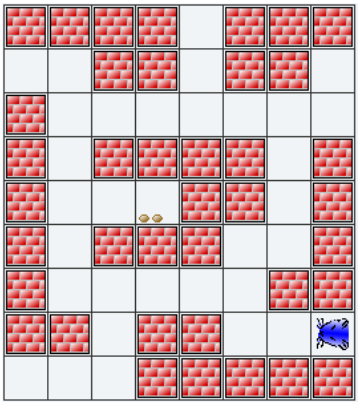


1.3. Erstellen Sie ihr eigenes Territorium, das nur einen Weg, aber Links- und Rechtskurven enthält. Kopieren Sie das Programm nach *A13_Schnecke* und ändern Sie es so, dass der Hamster auch hier zum Ziel findet.



1.4. Erstellen Sie ein Labyrinth, in dem es mehrere Abzweigungen als Sackgassen gibt. Der Ausgang soll mit einem Korn markiert sein.

Erstellen Sie ein Programm *A14_Schnecke*, so dass der Hamster in jedem denkbaren Labyrinth mit Sackgassen-Abzweigungen den Ausgang findet.



1.5. *Profi-Aufgabe*:
Erlauben Sie auch kreisförmige Gänge in Ihrem Labyrinth und Körner, die innerhalb dieser Kreise liegen.

Hinweis: mit dem Knopf  können Sie dem Hamster zu Startbeginn Körner ins Maul legen.