Programmieren mit dem Java-Hamster-Modell

VP Orientierung Informatik
WS 2008/09
H.Hagenauer

Idee

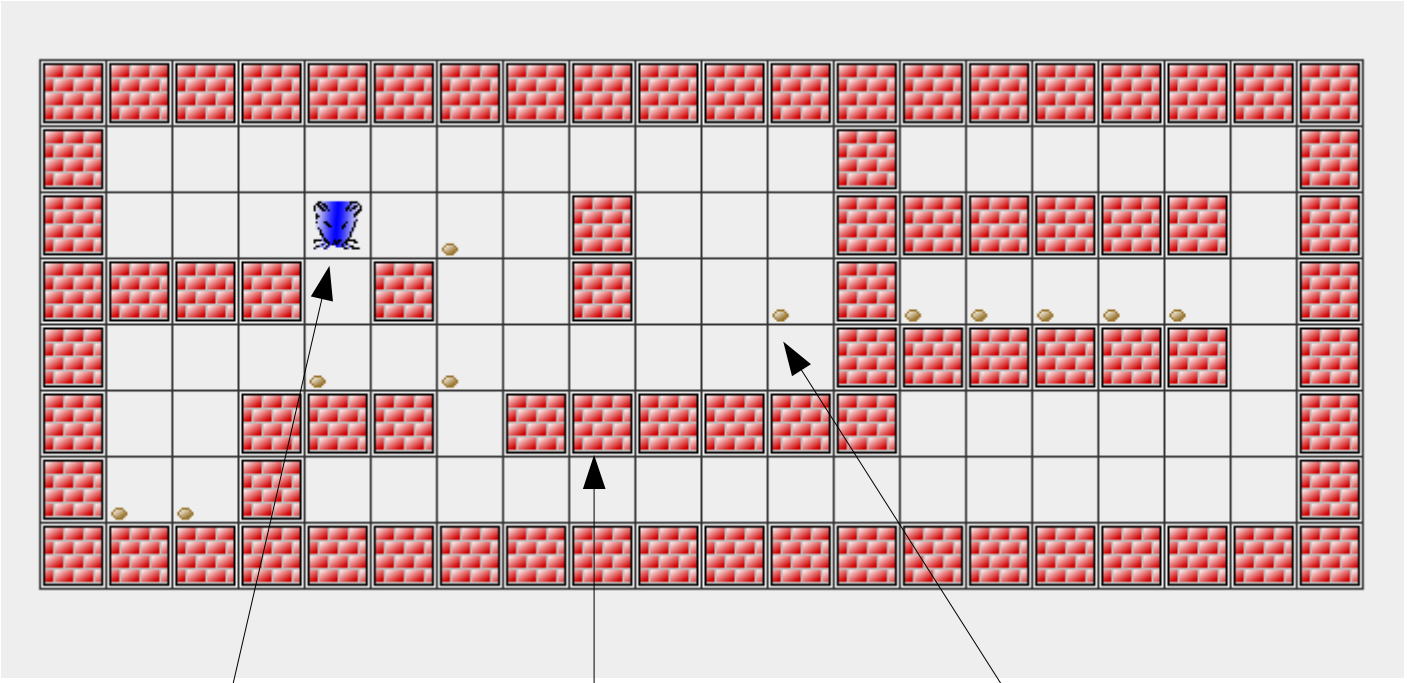
- didaktisches Modell zur Einführung in die Programmierung
- leichter und intuitiver Zugang („spielerisches“ lernen)
- mittels „Hamster-Programmen“ Grundlagen erlernen
- ... und schrittweise Steigerung der Komplexität

Grundlegendes Modell

- virtuelle Hamster in einer virtuellen Landschaft steuern
- Hamster müssen bestimmte Aufgaben lösen
- einfache „Hamster-Sprache“ mit wenig Grundbefehlen, orientiert sich an Java
- siehe auch
  http://www.java-hamster-modell.de/

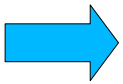
Wichtige Elemente

virtuelle Landschaft



Verwendung in dieser LV

- eigenhändige Installation des Hamster-Simulators
- Erstellung einer virtuellen Landschaft
- Lösung von einfachen Aufgaben mittels Hamster-Programmen



erste praktische Übungen an den Computern hier

dafür nötig: *Benutzerberechtigung*

(siehe VP Einführung UNIX)

Installation

- anmelden (einloggen) am Rechner mittels Benutzername und Passwort
- Terminal-Fenster starten
  $\rightarrow$ 2 Möglichkeiten
  - Symbol *Terminal-Applikation* schon vorhanden
  - unter *Applications* (oder *Anwendungen*) danach suchen

Installation Forts.

- kopieren der Datei
  `hamstersimulator-v25-02.zip`
  ins eigene Home-Verzeichnis mittels

```
cp /usr/local/sw/hamstersimulator/hamstersimulator-v24-01.zip .
```

- Kontrolle ob Datei vorhanden ist
  `ls -l`
- entpacken der Datei
  `unzip hamstersimulator-v25-02.zip`

Installation Forts.

- Kontrolle, ob Verzeichnis (Ordner, Directory) mit gleichem Namen eingerichtet wurde

```
ls -l
```

- wechseln ins neue Verzeichnis

```
cd hamstersimulator-v25-02
```

Starten des Simulators

- Voraussetzung: das aktuelle Verzeichnis lautet

  ```
  hamstersimulator-v25-02
  ```

  sonst „change directory“ Befehl anwenden:

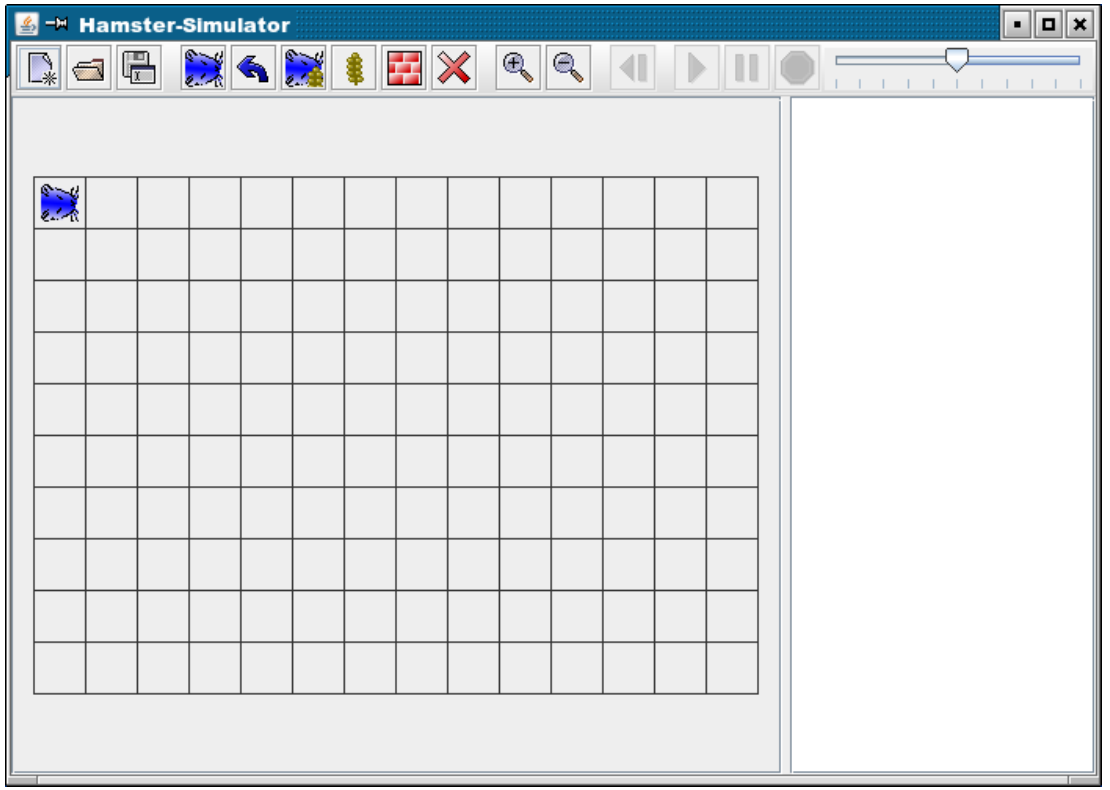
  ```
  cd hamstersimulator-v25-02
  ```

- Simulator und Editor starten

  ```
  java -jar hamstersimulator.jar
  ```

  Simulator- und Editorfenster werden geöffnet

Simulator



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

Virtuelle Landschaft erstellen

Landschaft = Territorium

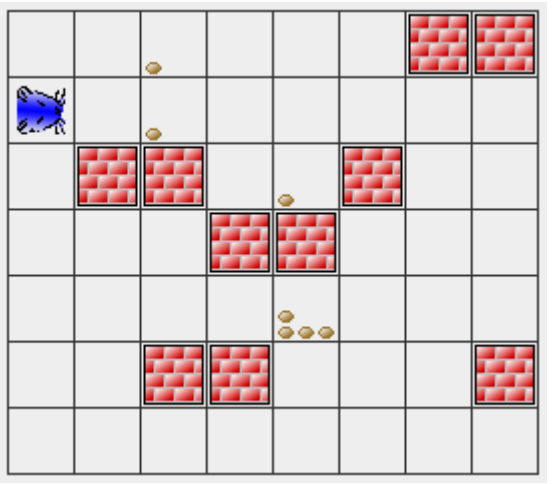

neues Territorium


Mauerkacheln setzen

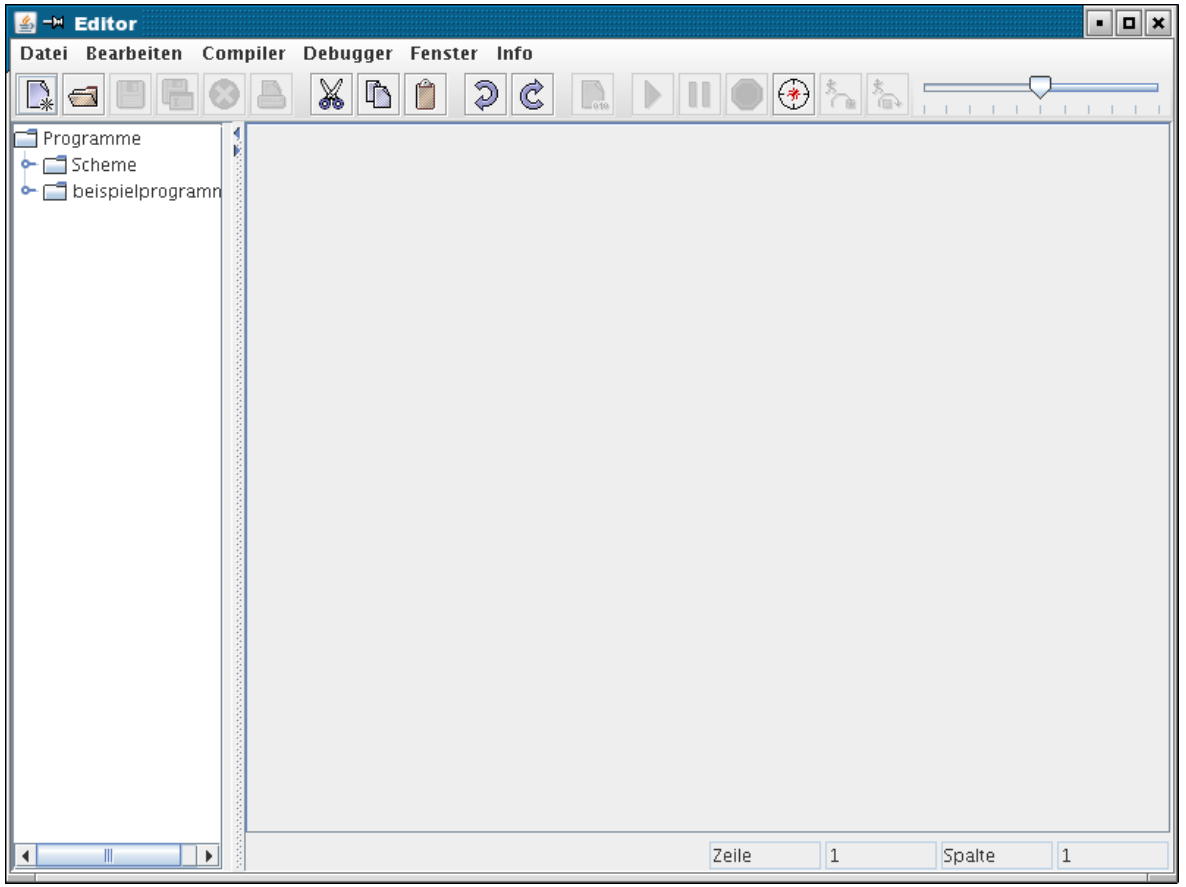

Körner verteilen


Hamster platzieren


Territorium speichern



Editor



Hamster-Programme

- erstellen
- ändern
- speichern
- laden

Grundbefehle für Hamster

`vor();`	springe 1 Feld nach vorne
`linksUm();`	Drehung um 90° nach links
`nimm();`	1 Korn vom aktuellen Feld aufnehmen
`gib();`	1 Korn auf aktuelles Feld ablegen

Programm erstellen

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

neues Programm erstellen

imperatives Programm wählen

Programm eingeben

Programm speichern

Erstes Programm

Programmname: ham1

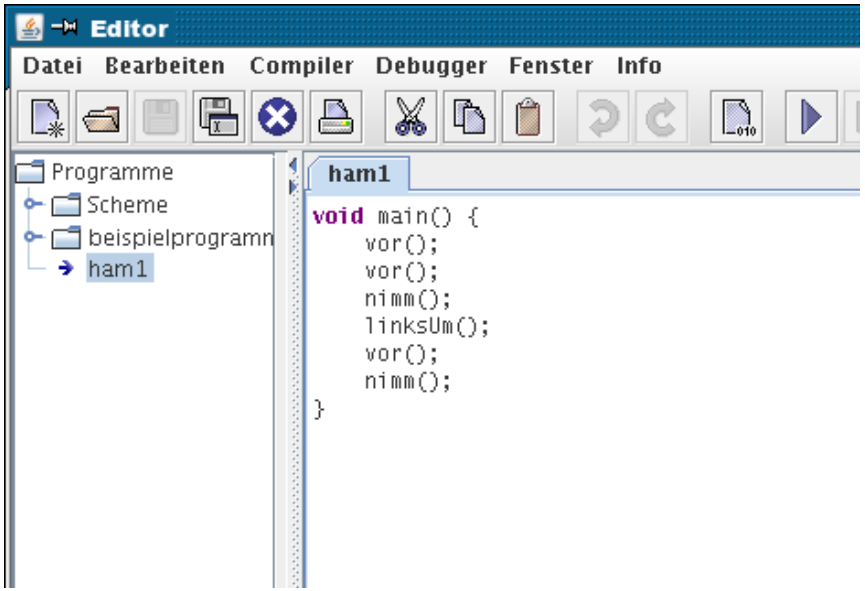
```
void main() {
    vor();
    vor();
    nimm();

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

Programm kompilieren (übersetzen)

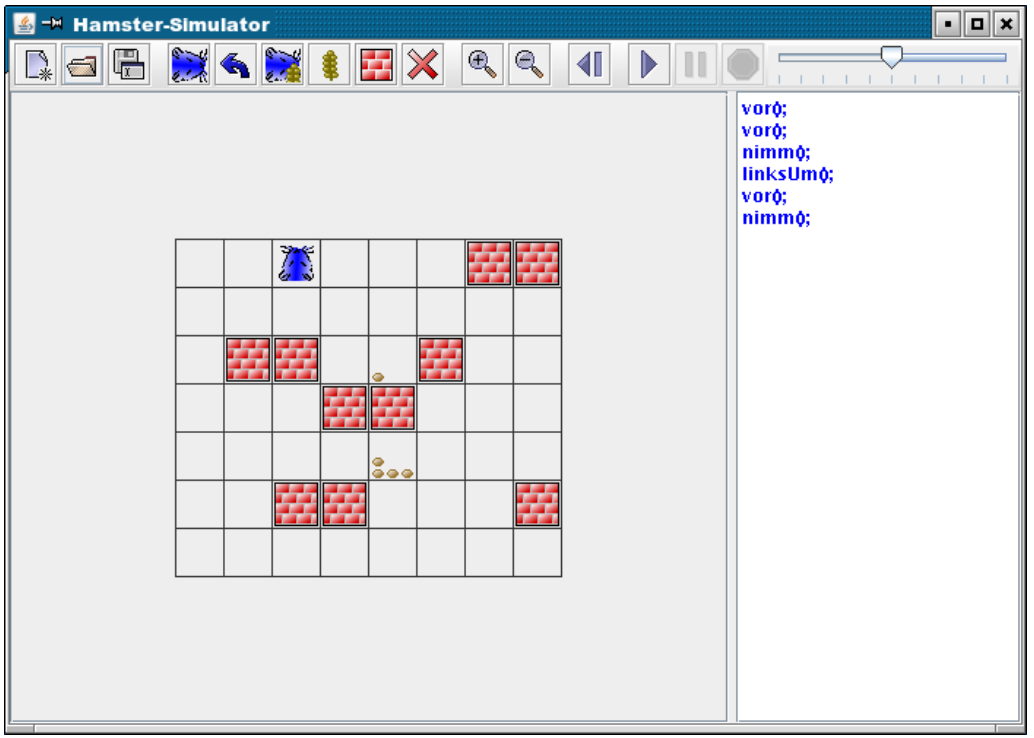
Programm starten

Ergebnis

Editor



Simulator



Erweiterte Aufgabe

Aufgabe: der Hamster soll auf dem Bsp.-Territorium 3 Körner aufnehmen und dann 1 Korn am Feld gerade voraus ablegen (wenn möglich)

Benötigen:

neue Befehle (als Prozeduren/Methoden)

Bedingungen und Abfragen

Neue Befehle

Neues Programm anlegen: ham2

```
void rechtsUm(){
    linksUm(); linksUm(); linksUm();
}
```

```
void main() {
    rechtsUm();

    vor(); vor(); vor();
    linksUm();
    vor(); vor(); vor(); vor();

    nimm(); nimm(); nimm();
}
```

neu definierter Befehl für Drehung um 90° nach rechts

Bedingungen

Bedingungen ergeben immer
wahr (true) oder *falsch (false)*

`vornFrei()`	ist das Feld vor dem Hamster frei?
`maulLeer()`	ist das Maul des Hamsters leer?
`kornDa()`	ist ein Korn auf dem aktuellen Feld vorhanden?

Bedingungen & Abfragen

```
void rechtsUm(){.....}

void main() {
    rechtsUm();

    vor(); vor(); vor();
    linksUm();
    vor(); vor(); vor(); vor();

    nimm(); nimm(); nimm();

    if (vornFrei()){
        vor();
        gib();
    }
}
```

Ergänzung von ham2 nötig, um 1 Korn abzulegen!

Abfrage: wenn Feld vor dem Hamster frei ist, dann gehe vor und lege 1 Korn ab

Wiederholungen (Schleifen)

Aufgabe: der Hamster soll sich so lange wie möglich nach vorne bewegen

Neues Programm anlegen: `ham3`

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

Schleife: solange das Feld vor dem Hamster frei ist, bewegt er sich um 1 Feld weiter