Technische Universität Darmstadt
Fachbereich Informatik
Prof. Dr. Johannes Fürnkranz

Allgemeine Informatik 1 im WS 2006/07

Übungsblatt 5

Bearbeitungszeit: 27.11. bis 03.12.2006

Aufgabe 1: `while`-Schleife

In der Vorlesung haben Sie die **`while`**-Schleife kennen gelernt. Mit **`while`**-Schleifen kann man **`loop`**-Schleifen simulieren. Schreiben Sie hierzu die Klasse **`FarMoverA`** aus Übung 4 / Aufgabe 1 mit Hilfe einer **`while`**-Schleife.

*Hinweis: Um eine **`int`**-Variable **`x`** runterzuzählen, kann man die Anweisung **`x = x – 1;`** verwenden. Auf der rechten Seite der Zuweisung repräsentiert **`x`** immer den Wert vor der Zuweisung. Durch die Anweisung wird also der Variablen **`x`** der Wert von **`x`** vor der Zuweisung minus 1 zugewiesen, **`x`** wird also um 1 kleiner. Die Anweisung ist somit äquivalent zu:*

```
int temp = x;
x = temp – 1;
```

Eine praktische Kurzform für **`x = x - 1`** *ist* **`x--`** *(und für* **`x = x + 1`** *natürlich* **`x++`***).*

Aufgabe 2: Verbesserter Robot durch Verwendung von `while`

Schreiben Sie eine Roboterklasse **`EnhancedRobot`**, die die folgenden Methoden enthält:

```
void faceTo(direction dir) { ... }
void moveTo(int street, int avenue) { ... }
```

Nach einem Aufruf der Form **`karel.faceTo(dir)`** soll der **`EnhancedRobot karel`** in die übergebene Himmelsrichtung (**`dir`**) schauen, sich also so lange drehen, bis er in diese Richtung sieht.

Nach einem Aufruf der Form **`karel.moveTo(street, avenue)`** soll sich **`karel`** an der Kreuzung (**`street`**, **`avenue`**) befinden. Sie sollen dabei die Methode **`faceTo`** sinnvoll benutzen. Tipp: lassen Sie **`karel`** zuerst horizontal gehen, dann vertikal (oder umgekehrt).

Sie dürfen voraussetzen, dass die Welt, in der sich **`karel`** bewegt, keine zusätzlichen Wände enthält. Sie müssen aber davon ausgehen, dass die Anfangsposition und -richtung von **`karel`** unbekannt ist. Benutzen Sie für beide Methoden **`while`**-Schleifen!

Testen Sie Ihre Klasse, indem Sie **`karel`** nacheinander zu verschiedenen Punkten laufen lassen.

Aufgabe 3: Ameise

Diese Aufgabe soll zeigen, dass es mit sehr kleinen Programmen möglich ist, sehr komplexe Ergebnisse zu erzielen. Sie werden **`karel`** beibringen, sich wie eine „Langton-Ameise“ zu verhalten.

Schreiben Sie eine Roboterklasse **`Ameise`**, die eine Methode **`void step()`** hat. Diese Methode soll folgendes tun:

- wenn auf der Kreuzung ein Beeper liegt, nimmt die Roboter-Ameise diesen auf, dreht sich um 90 Grad nach rechts und geht einen Schritt nach vorne.
- wenn aber auf der Kreuzung kein Beeper liegt, legt der Ameisen-Roboter einen ab, dreht sich um 90 Grad nach links und geht einen Schritt nach vorne.

Überlegen Sie, wie Sie die Anzahl der Anweisungen minimieren können. Aus

```
if (!frontIsClear()) {
  turnLeft();
} else {
  turnLeft();
  turnLeft();
}
```

kann zum Beispiel, ohne jegliche Änderung des Sinns, folgendes werden:

```
turnLeft();
if (frontIsClear()) turnLeft();
```

Im **`task`** soll erstes mit Hilfe von **`World.setSize(streets, avenues)`** die Größe der Welt eingestellt werden. 40x40 oder größer ist vernünftig. Sie können mit **`World.setDelay(milliseconds)`** auch die Wartezeit zwischen den einzelnen Schritten verringern.

Erzeugen Sie in der Mitte der Welt die **`Ameise`** **`karel`** mit 999999 Beepern.

Dann soll **`karel`** in einer endlosen **`while`**-Schleife (die Bedingung muss also immer wahr sein!) die Methode **`step()`** aufrufen.

Führen Sie Ihr Programm (mit Delay 0!) eine Weile lang aus (mit dem Button **Stop** können Sie es dann anhalten). Erkennen Sie ein Muster in den Bewegungen der Ameise oder sind diese völlig chaotisch?

Sie können auch mehrere Ameisen erzeugen und sehen, wie sich diese gegenseitig beeinflussen.

Wenn Sie das Thema interessiert, ist hier ein guter Startpunkt im Internet:
http://www.mathematische-basteleien.de/ameise.htm

Viel Spaß!