Problemorientiertes Programmieren: RoboCode
Sommersemester 2011
Dipl.-Inf. Christian Rieß, Dipl.-Inf. Eva Eibenberger

Aufgabenblatt 4 – Termin 4

Nehmen wir an, wir wollen Roboter verwalten (beispielsweise Gegner in der RoboCode-Arena). Ein Roboter hat einen Namen und seine Batterie hat Energie (eine Kommazahl zwischen 0 und 100). Aušerdem interessieren uns die x-y-Koordinaten, an denen er die letzten Male auf dem (2D-)Spielfeld gesichtet wurde.

Aufgabe 4-1: Buchhaltung

Angenommen, wir haben drei Robotergegner. Welche Arrays (und von welchem Typ?) sollten wir anlegen, um von den drei Gegnern Name, Energie, und die letzten fünf Positionen zu speichern?

Aufgabe 4-2: Ein eigener Typ

Arrays vordefinierter Typen sind nützlich, aber nicht alles. Wenn wir abwechselnd auf verschiedene Eigenschaften der drei Roboter zugreifen wollen, haben wir mit den Arrays der letzten Aufgabe schon eine Menge Verwaltungsaufwand.

Wie lässt sich der Überblick behalten? Durch die Einführung eines eigenen Typs, z.B. `MeinGegner`. Damit lässt sich eine Liste der Gegner anlegen, und jeder Gegner (für sich) enthält seinen Namen, Energie und die letzten 5 Positionen – alles ist aufgeräumt!

Es gibt unterschiedliche Wege, eigene Datentypen zu realisieren. Ein für RoboCode nützlicher Mechanismus sind *inneren Klassen*. Ist die Hauptklasse unseres Programms beispielsweise `RoboterVerwaltung`, kann `MeinGegner` als eine innere Klasse angelegt werden, d.h. eine Klasse in der Klasse. Innerhalb von `RoboterVerwaltung` kann dann `MeinGegner` genauso benutzt werden wie andere Klassen.

- Schreibe ein Programm `RoboterVerwaltung`, das drei Instanzen von `MeinGegner` anlegt und für jeden Gegner sinnvolle Werte einfügt.
- Gib die Werte von jedem Gegner auf der Konsole aus. Etwas genauer: Implementiere in der Klasse `MeinGegner` eine Methode `printStatus()`, das die Roboterdaten auf die Konsole schreibt. Laufe in der Klasse `RoboterVerwaltung` mit einer Schleife über ein Feld von `MeinGegner`-Instanzen.
- (Optional): Suche einen Gegner nach Namen (mit der `String`-Methode `equals`, suche in der Java-API), und gib seine Daten aus.

Achtung! Bei dem Arbeiten mit Listen muss man einen Unterschied zwischen primitiven und komplexen Typen beachten: Listen von primitiven Typen konnten direkt genutzt werden:

```
int[] zahlen = new int[3];
zahlen[0] = 42;
```

Bei dem Anlegen einer Liste von `MeinGegner`-Instanzen muss

1. erst die Liste mit `new` angelegt werden, und anschließend dann
2. jede `MeinGegner`-Instanz der Liste mit `new` instanziiert werden.

Wir brauchen also etwas in der Art von

```
MeinGegner[] gegnerListe = new MeinGegner[3];
gegnerListe[0] = new MeinGegner();
```

um das Objekt an Position 0 der Liste benutzen zu können.

Aufgabe 4-3: Zu schnell gewesen?

Jetzt geht's los. Suche im Netz nach RoboCode, lade Dir die RoboCode-Umgebung herunter und installiere sie. Wenn Du Lust hast, lies Dich auf der Homepage ein.