Syntax & Semantik

Warum Eiffel in so guter Form ist

Programme (TRAFFIC oder FLAT_HUNT) sind in einer Sprache (Eiffel) geschrieben. Eiffel definiert einerseits die Grammatik und andererseits werden den einzelnen Schlüsselwörtern (Keywords) Bedeutungen zugewiesen.

Die Grammatik beschreibt zum Beispiel, dass in jedem Text einer Klasse am Anfang **class *KLASSEN_NAME*** haben muss. Und dass bei der Klasse auch noch ***feature*** definiert werden sollte in der Form:

```
feature -- Kommentar
        feature_name: KLASSEN_NAME
```

oder auch:

```
feature -- Kommentar
        feature_name is
                do
                        ...
                end
```

oder sogar:

```
feature
        feature_name( argument: ETWAS ) : ETWAS_ANDERES
```

Es gibt also verschiedene Grammatikregeln, welche eingehalten werden müssen. Diese Regeln definieren die Form bzw. die Syntax von Eiffel Programmen. Um die Syntax zu beschreiben, hatten Backus und Naur unabhängig von einander eine Sprache, die BNF, entwickelt. Selbst Eiffel ist in dieser Sprache definiert (siehe Referenzen).

Aber was hat das alles für einen Sinn?

Die Form allein genügt nicht. Ohne Sinn bzw. Semantik macht ein Programm mit korrekter Syntax noch keine Semantik bzw. keinen Sinn. Deshalb wird mit Eiffel auch noch eine ganze Reihe mit Beschreibungen und Definitionen von Konzepten mitgeliefert, die Antworten auf folgende Fragen geben: Was ist Objekt-Orientierung? Was sind Klassen? Was sind Features? Was sind Queries oder Commands? Diese Definitionen geben das Fundament für sinnvolle Programme.

Laurent Guenat an der Französischen Kunst Gallerie Berlin. "Der Leere Sinn geben, Sich selbst Sinn geben"

Die Syntax ist anhand der Grammatik weitgehendst definiert und unveränderbar (abgesehen von infix Operatoren beispielsweise). Die Semantik kann aber dank unserer Hilfe erweitert werden. Wir können sinnvollere Programme, schlauere Office-Assistenten-Büroklammern, oder intelligente Musik-Playlists entwickeln. Diese zusätzliche Semantik geben wir in Form von neuen Klassen mit sinnvollen Features.

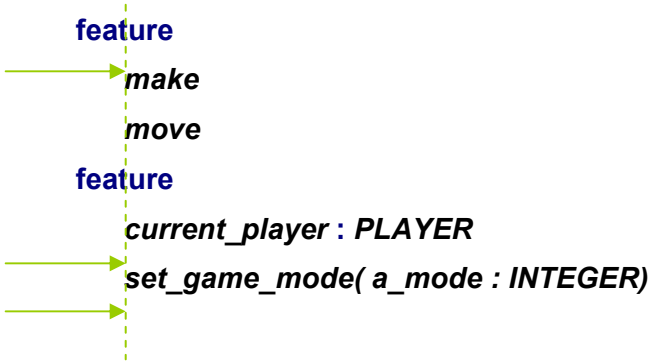
Was uns hilft, und wo wir auf uns selbst gestellt sind

Bei der Grammatik kann uns der Compiler helfen und uns auf Fehler hinweisen. Aber bei der Semantik ist der Informatiker weitgehendst auf sich selbst gestellt. Es gibt keine automatischen Programmierprogramm, welche automatisch neue, tolle Produkte entwickelt. Diese faszinierende Idee würde voraussetzen, dass die künstliche Intelligenz im Computer etwa gleich mächtig ist wie der Verstand eines Menschen. Weil es aussichtslos scheint, dies zu erreichen, begnügt man sich derzeit mit kopfloser Intelligenz. Ein weltführender Forscher doziert sogar an der Universität Zürich: Professor Pfeifer (siehe Referenzen).

Tipps zur Übungsserie 2

Features findet man immer an der gleichen Stelle in der Klasse. Darum ist es schlau, sich die BNF Syntax ein wenig anzusehen. Man wird dann merken:

- Kommentare können einfach imaginär gelöscht und ignoriert werden.
- Features stehen entweder direkt nach dem Keyword **feature**, oder dann nach einem anderem Feature:
  feature
  make
  move
- die Definition eines Features beginnt immer mit dessen Namen, gefolgt von
  - einem Doppelpunkt und einem Klassennamen, ***current_player : PLAYER***
  - dem Keyword **is**, ***make* is**, oder
  - einer Klammer mit Argumenten dazwischen und dann dem Doppelpunkt oder dem Keyword **is** ***set_game_mode (a_mode : INTEGER)***.
  - Manchmal steht auch noch ein **require** und **ensure** und **end** dahinter, oder auch gar nichts (aber nur in der "simplified and cropped version" in der Übung).
- Features sind immer gleich weit von links her eingerückt:



Befehlen und Nachfragen

Eine Query ist eine Nachfrage nach dem Zustand eines Objektes. Ein Feature ist eine Query, falls sie einen Wert zurückgibt. Das heisst, wenn es da einen Doppelpunkt und einen Klassennamen am Schluss der Zeile hat:

current_player : PLAYER

Ein Command ist ein Kommando zur Veränderung des Zustandes eines Objektes. Kommandos geben nichts zurück, verändern aber wahrscheinlich etwas am Zustand des Objektes, andernfalls würde es ja keinen Sinn machen, es aufzurufen:

make

Referenzen

Homepage von Professor Pfeifer --
http://www.ifi.unizh.ch/ailab/people/pfeifer

Artikel aus der Weltwoche zur kopflosen Intelligenz –
http://www.ifi.unizh.ch/ailab/people/pfeifer/ww1.html

Wikipedia Eintrag zu Semantik –
http://de.wikipedia.org/wiki/Semantik

Wikipedia Eintrag zu Syntax –
http://de.wikipedia.org/wiki/Syntax

Bild – http://dautrepart.free.fr/fkgb/pages/guenat.html

Eine Inoffizielle Eiffel Synatx in BNF –
http://www.gobosoft.com/eiffel/syntax/index.html