Ada Magica

Christoph Grein
c/o ESG, 81677 München;
eMail: Christ-Usch.Grein@T-Online.de
http://home.T-Online.de/home/Christ-Usch.Grein

Geheimniskrämerei

Ein hehres Prinzip guter Software-Entwicklung ist das Geheimnisprinzip (*information hiding*). Es besagt, dass man alles, was der Anwender nicht zu wissen braucht, verstecken soll. Auch Lady Ada liebt es, Verstecken zu spielen; sie ist ein rechter Geheimniskrämer.

```
with Ada.Finalization;
use  Ada.Finalization;

package Geheimniskrämer is

   type Wurzel is tagged private;

private

   type Wurzel is new Controlled with
     null record;

   procedure Initialize
                (Objekt: in out Wurzel);
   procedure Adjust
                (Objekt: in out Wurzel);
   procedure Finalize
                (Objekt: in out Wurzel);

end Geheimniskrämer;
```

```
package Geheimniskrämer.Abgeleitet is

   type Etwas is tagged private;

   Nichts: constant Etwas;

private

   type Etwas is new Wurzel with
     null record;

   procedure Initialize
                (Objekt: in out Etwas);
   procedure Adjust
                (Objekt: in out Etwas);
   procedure Finalize
                (Objekt: in out Etwas);

   Nichts: constant Etwas :=
     (Wurzel with null record);

end Geheimniskrämer.Abgeleitet;
```

Hier wird durch Ableitung vom vordefinierten Typ `Ada.Finalization.Controlled` eine Klasse kontrollierter Typen definiert mit dem Ursprungstyp `Wurzel`. Kontrolliertheit ist eine neue Ada95-Errungenschaft, die uns zusätzliche Kontrolle über die drei fundamentalen Operationen auf Objekten ermöglicht: Erzeugung, Zuweisung, Vernichtung – ihnen sind die drei Prozeduren `Initialize`, `Adjust` und `Finalize` zugeordnet. Die exakte Definition ist im Ada RM Kapitel 7.6 *User-Defined Assignment and Finalization* zu finden. Wir wollen uns das Verhalten kontrollierter Typen an Hand des folgenden einfachen Hauptprogramms einmal anschauen:

```
with Geheimniskrämer.Abgeleitet;
use  Geheimniskrämer.Abgeleitet;

with Ada.Text_IO;
use  Ada.Text_IO;

procedure Ausprobieren is
  Ding: Etwas;
begin
  Put_Line ("Rumpf von Ausprobieren" &
            " beginnt jetzt.");
  Ding := Nichts;
  Put_Line ("Rumpf von Ausprobieren" &
            " endet jetzt.");
end Ausprobieren;
```

Hier geschieht im wesentlichen Folgendes:

- Als erstes werden das Paket `Geheimniskrämer` und sein Nachkomme `Abgeleitet` elaboriert. Die Konstante `Nichts` wird erzeugt. Da sie einen expliziten Initialwert zugewiesen bekommt, wird `Initialize` nicht aufgerufen. `Adjust` wird ebenfalls nicht aufgerufen, da ein Aggregat vermöge des bindenden Ada Issue AI95_00083 an Ort und Stelle erzeugt wird. (Aus gutem Grund werden diese Operationen hier nicht aufgerufen, denn sie sind zur Zeit der Erzeugung von `Nichts` noch nicht elaboriert. Ohne diese Regelung könnte kein Objekt eines kontrollierten Typs zusammen mit seinem Typ in derselben Paketspezifikation deklariert werden.)
- `Etwas` ist von `Wurzel` abgeleitet, also muss `Initialize` bei der Erstellung des Aggregats für `Nichts` für ein (anonymes) Objekt diesen Typs gerufen werden, nämlich für die Ansichtsumwandlung `Wurzel (Nichts)`.
- Hiermit ist die Elaborierung der Kontextklausel beendet.

- Nun wird `Ding` elaboriert. Dies ist eine uninitialisierte Variable, somit wird `Initialize` für sie gerufen.
- Damit ist die gesamte Elaboration beendet.
- Es beginnt die Ausführung des Programms mit dem Aufruf von `Put_Line`.
- Die Zuweisung zu `Ding` erfolgt in mehreren Schritten. Zuerst wird `Finalize (Ding)` gerufen (sein augenblicklicher Wert wird ja vernichtet). Dann wird ihm der neue Wert zugewiesen. Als Abschluss dieser Zuweisungsoperation wird `Adjust (Ding)` aufgerufen.
- Die Ausführung des Programms endet mit einem weiteren Aufruf von `Put_Line`.
- Nun werden alle Objekte in der umgekehrten Reihenfolge ihrer Erzeugung wieder vernichtet, zuerst `Ding` und dann `Nichts`.

Dieser gesamte Ablauf sei hier noch einmal zusammengefasst:

`Initialize` gerufen für Objekt des Typs `Wurzel`	→	Initialisierung des `Wurzel`-Teils von `Nichts`
`Initialize` gerufen für Objekt des Typs `Etwas`	→	Initialisierung der lokalen Variablen `Ding`
`Rumpf von Ausprobieren beginnt jetzt.`		
`Finalize` gerufen für Objekt des Typs `Etwas`	→	Vor-Zuweisungsfinalisierung von `Ding`
`Adjust` gerufen für Objekt des Typs `Etwas`	→	Nach-Zuweisungsausrichtung von `Ding`
`Rumpf von Ausprobieren endet jetzt.`		
`Finalize` gerufen für Objekt des Typs `Etwas`	→	Ende des Gültigkeitsbereichs von `Ding`
`Finalize` gerufen für Objekt des Typs `Etwas`	→	Ende des Gültigkeitsbereichs von `Nichts`

Nun bekam Lady Ada neulich einen verzweifelten Brief von einem ihrer neuen Verehrer. Er sandte ihr ein kleines Programm, das er verfasst hatte, und schrieb dazu:

```
with Geheimniskrämer;
use  Geheimniskrämer;

package Neuling is

  type Etwas is tagged private;

  Nichts: constant Etwas;

private

  type Etwas is new Wurzel with
    null record;

  procedure Initialize
              (Objekt: in out Etwas);
  procedure Adjust
              (Objekt: in out Etwas);
  procedure Finalize
              (Objekt: in out Etwas);

  Nichts: constant Etwas :=
    (Wurzel with null record);

end Neuling;
```

```
with Neuling;
use  Neuling;

with Ada.Text_IO;
use  Ada.Text_IO;

procedure Überraschung is
  Ding: Etwas;
begin
  Put_Line ("Rumpf von Überraschung" &
            " beginnt jetzt.");
  Ding := Nichts;
  Put_Line ("Rumpf von Überraschung" &
            " endet jetzt.");
end Überraschung;
```

„Verehrte Lady Ada,

wie Sie in der Anlage sehen können, habe ich genau dasselbe wie Sie gemacht, allerdings mit für mich unerfahrenen Adepten gänzlich unerwartetem Ergebnis – zwar ähnlich wie oben, doch wird keine der drei Operationen für `Etwas` aufgerufen, sondern stets die für `Wurzel`. Wieso?

Um Aufklärung heischend verbleibe ich Ihr allerergebenster Diener, ...“

Eine übersetzbare und ausführbare Version des Programms ist unter der Internet-Adresse
http://home.T-Online.de/home/Christ-Usch.Grein/AdaMagica/Geheim.zip
ladbar.