Ada Magica

Christoph Grein

Einladung bei Lady Ada

Neulich saß ich wieder einmal vor den neuen Ada95-Manualen und kämpfte mich durch die Vergnügen der Objektorientiertheit: Polymorphie, Vererbung, Dispatching - alles was das Herz begehrt. Trotzdem ist das auf die Dauer eine recht trockene Angelegenheit. Meine Lider senkten sich, ich war kurz davor einzuschlafen, da plötzlich hörte ich eine leise Stimme, die mir zurief. Ich drehte mich um; eine junge Dame stand in der Tür meines Arbeitszimmer und winkte mir zu.
Ich blinzelte. Wer war sie? „Nun, was ist?“ Sie wurde ganz offensichtlich ungeduldig.
Ich schaute genauer hin – das war doch – Ja, tatsächlich, Lady Ada persönlich! Ich sprang auf und ging zu ihr. Freundlich wie einen alten Bekannten begrüßte sie mich und nahm mich beim Arm. „Gehen wir, ich finde, das war genügend von dem trockenen Zeug!“ sagte sie und zog mich aus dem Raum. Sie habe ein schönes Abendessen für zwei vorbereitet, ob ich nicht in einer Stunde bei ihr sein wolle, sie müsse sich vorher nur noch ein wenig zurecht machen? Damit entschwand sie.
Wie im Traum stand ich da …
Pünktlich zur verabredeten Zeit stand ich mit einem riesigen Blumenstrauß aufgeregt vor ihrer Tür. Auf mein Klingeln öffnete sich die Tür, ich trat ein, sanft fiel die Tür hinter mir ins Schloss. Leise orientalische Musik kam aus dem Wohnzimmer und ein betörender duft erfüllte die Wohnung. Zögernd schritt ich weiter, klopfte an.
Sie sah hinreißend aus in ihrem langen samtig grünen, tiefausgeschnittenen Kleid. Ihr Haar hatte sie mit einer weißen Nelke aufgesteckt. Der Speiseraum war intim von Kerzen erleuchtet. Sie führte mich an den festlich gedeckten Tisch:

```
procedure Tisch (Speisekarte: in String) is

  Gänge: constant := 10;
  subtype Menü is String (1 .. Gänge);

  -- *

  procedure Genieße (Wahl: in Menü) is separate;

begin

  Genieße
   (Menü (Speisekarte & (Speisekarte'Length + 1 .. Gänge => ' ')));

end Tisch;
```

Sie hieß mich Platz zu nehmen. „Halt, eine Kleinigkeit fehlt noch“, unterbrach sie mich, als ich zulangen wollte, und stellte auf den einzig noch freien Platz (*) eine Batterie Weinflaschen:

```
subtype Wein is Menü;
Getränke: array (1 .. Gänge) of Wein;
```

Der Erfolg war umwerfend. Als ich mit ihr anstieß und den ersten Schluck zu mir nahm, schwanden mir die Sinne. Ich sah gerade noch, wie Lady Ada plötzich eine Zauberstab schwang, ein spitzbübisches Grinsen, das so gar nicht zu ihr passen wollte, in ihrem hübschen Gesicht, dann wurde mir schwarz vor Augen.
Ich erwachte von einem schrillen Ton und einem unangenehmen Druck auf der Stirn. Als ich die Augen öffnete, saß ich in meinem Arbeitszimmer am Schreibtisch, mein Kopf war auf die Tastatur ge-

fallen. Der schrille Ton kam aus dem Computer, offensichtlich war der Eingabepuffer übergelaufen, der Bildschirm quoll über mit Fehlermeldungen des Compilers – und zu allem Überfluss kam auch noch mein Chef daher, um sich über den Fortschritt meiner Arbeit zu erkundigen. Ich wies nur hilflos und verwirrt auf den Monitor, wo der Compiler den Katenierungsoperator `"&"` als zweideutig zurückwies.
Wie konnte Lady Ada mit dem `Wein`, der noch dazu nicht einmal benutzt wird, das vorher so schmackhafte `Menü` ungenießbar machen?

Adresse des Autors: C. Grein, c/o ESG, Vogelweideplatz 9, 81677 München
eMail: Christ-Usch.Grein@T-Online.de

Nachtrag zum Problem der letzten Ausgabe:

Gefragt war, warum

```
declare
  Inhalt: Akten (Tresor);
begin
  Lies (Inhalt);
end;
```

`Constraint_Error` auslöst, wenn noch nie etwas im Tresor lag. Wir haben vergessen, unsere `Speicher`-Variable im Paketrumpf zu initialisieren. So enthält sie für jeden Eintrag die Variante für den Schreibtisch und beim Lesen schlägt der Diskriminantenvergleich zwischen `Inhalt` und `Speicher (Inhalt.Schlüssel)` fehl. Allerdings kann diese Initialisierung ziemlich mühevoll werden, wenn man sie nicht geschickt macht. Als Erstes fügen wir dem Verbund `Akten` noch eine Komponente hinzu, die angibt, ob überhaupt etwas vorhanden ist:

```
type Akten (Schlüssel: Schlüsselmuster := Schreibtisch) is
       record
         Leer: Boolean;
         ... -- Rest wie zuvor
       end record;
```

Der Paketrumpf erhält einen ausführbaren Teil (er wird während der Elaboration ausgeführt):

```
package body Archiv is

  ...  - Deklarationsteil wie bisher

begin

  for Schlüssel in Schlüsselmuster loop
    case Schlüssel is
      when Schreibtisch =>
        Speicher (Schlüssel) := (Schlüssel   => Schreibtisch,
                                 leer        => True,
                                 Vertraulich => ...);
      when Tresor =>
        Speicher (Schlüssel) := (Schlüssel   => Tresor,
                                 leer        => True,
                                 Geheim      => ...);
      when … => …
    end case;
  end loop;

end Archiv;
```

Zu unserem Leidwesen müssen wir bei dieser Methode die gesamte Verbunddeklaration in einem gewissen Sinne wiederholen. Wenn wir viele Schlüsselvarianten haben, kann das ziemlich mühevoll

werden. Glücklicherweise ist das auch völlig überflüssig, wenn wir bei der Deklaration von `Akten` gleich die richtigen Vorbelegungswerte mitgeben (soweit erforderlich):

```
type Akten (Schlüssel: Schlüsselmuster := Schreibtisch) is
       record
         Leer: Boolean := True;
         ... -- Rest ebenfalls mit geeigneten
             -- Vorbelegungswerten, wo nötig
       end record;
```

Die Initialisierung ist nun ganz einfach.

```
for Schlüssel in Schlüsselmuster loop
  declare
    Akt: Akten (Schlüssel);
    -- Akt: constant Akten (Schlüssel) := ???;
  begin
    Speicher (Schlüssel) := Akt;
  end;
end loop;
```

Man beachte die Variablendeklaration ohne Initialisierung – eine Konstante benötigte einen Initialwert und wir stünden vor dem Problem, wie wir ihn angeben könnten.
(Nebenbei: Deklarationen ohne Initialisierung sind durch manche Codingstandards streng verboten, vor allem, wenn der Variablen wie hier nie ein Wert zugewiesen wird! Wie wir gesehen haben, ist jedoch auch so scheinbar Unsinniges manchmal unumgänglich und – wenn man es richtig macht – auch völlig korrekt.)