33 - La notion de type abstrait

Programmation Concurrente - LI330
Université P. & M. Curie - année scolaire 2013/2014

PrC

Rappel: type = domaine + opérations

Type dont on ne connaît pas la structure qui ne se manipule que par l'intermédiaire d'opérations définies pour cet usage

- Objectifs
  - Protection des informations contenues dans le type
  - Définition «propre» des opérations réalisables
- Notion sous-jacente: l'encapsulation
  - Encapsuler = rendre aussi peu visible que possible tout ce qui a trait à l'implémentation

Première approche à travers un exemple: Le gestionnaire de fiches

- Contenu:
  - Nom + prénom
- Opérations:
  - Construction d'une fiche
  - Lecture des informations qu'elle contient

Spécification de gest_fich

```
-- Unité de gestion de fiches
package Gest_Fiches is

   -- Définition de la structure de données "fiche"
   type Une_Fiche is record
      Nom, Prenom : String (1..20);
      L_Nom, L_Prenom : Natural := 0;
   end record;

   -- Définition des primitives de manipulation d'une fiche
   procedure Nouvelle_Fiche (N, P : in String;
                             F : out Une_Fiche);
   function Lire_Nom (F : Une_Fiche) return String;
   function Lire_Prenom (F : Une_Fiche) return String;

end Gest_Fiches;
```

gest_fiches.ads

Spécification de gest_fich

```
-- Unité de gestion de fiches
package Gest_Fiches is

   -- Définition de la structure de données "fiche"
   type Une_Fiche is record
      Nom, Prenom : String (1..20);
      L_Nom, L_Prenom : Natural := 0;
   end record;

   -- Définition des primitives de manipulation d'une fiche
   procedure Nouvelle_Fiche (N, P : in String;
                             F : out Une_Fiche);
   function Lire_Nom (F : Une_Fiche) return String;
   function Lire_Prenom (F : Une_Fiche) return String;

end Gest_Fiches;
```

gest_fiches.ads

Corps de gest_fich (1)


```
-- Corps de l'unité de gestion de fiches
package body Gest_Fiches is

   procedure Nouvelle_Fiche (N, P : in String;
                             F : out Une_Fiche) is

   begin
      F.Nom (1.. N'Length)     := N;
      F.Prenom (1.. P'Length)  := P;
      F.L_Nom                  := N'Length;
      F.L_Prenom               := P'Length;
   end Nouvelle_Fiche;

   function Lire_Nom (F : in Une_Fiche) return String is

   begin
      return F.Nom (1 .. F.L_Nom);
   end;
```

gest_fiches.adb

Corps de gest_fich (2)


```
   function Lire_Prenom (F : in Une_Fiche) return String is

   begin
      return F.Prenom (1 .. F.L_Prenom);
   end;

end Gest_Fiches;
```

Le point de vue de l'utilisateur de gest_fich (un exemple)


```
with Ada.Text_Io, Ada.Integer_Text_Io, Gest_Fiches;
use  Ada.Text_Io, Ada.Integer_Text_Io, Gest_Fiches;

-- Corps de l'unité de gestion de fiches
procedure Test_Gest_Fiches is

S1, S2   : String (1..20);
L1, L2   : Natural;
La_Fiche : Une_Fiche;

begin
   -- Acquisition de la fiche
   Put_Line ("Votre nom");
   Get_Line (S1, L1);
   Put_Line ("Votre prenom");
   Get_Line (S2, L2);
   Nouvelle_Fiche (S1 (1..L1), S2 (1.. L2), La_Fiche);
   -- Affichage de la fiche
   Put_Line ("La fiche contient la personne suivante:");
   Put_Line (Lire_Prenom (La_Fiche) & " " & Lire_Nom (La_Fiche));
end Test_Gest_Fiches;
```

Le point de vue de l'utilisateur de gest_fich

- Cet utilisateur peut-il créer d'autres opérations sur le type Une_Fiche?

  Oui

  car il a accès aux composants du type

- Est-ce potentiellement dangereux

  Oui

  répercussions à prévoir si on change l'implémentation du type Une_Fiche

- Que dire...
  - Il n'y a pas encore de réelle encapsulation