

```
-----  
-- * Prog name ToursDeHanoi_obj_acc.adb  
-- *  
-- * Version 1.2  
-- * Last update 24/08/08  
-- *  
-- * Adapted by Pascal Pignard 2008 (http://blady.pagesperso-orange.fr),  
-- * based on "Les langages à objets",  
-- * Michel Beaudouin-Lafon, 1992 Armand Colin,  
-- * Licence CeCILL V2 (http://www.cecill.info).  
-- *  
-----  
  
with Ada.Text_IO;  
with Unchecked_Deallocation;  
procedure ToursDeHanoi_Obj_Acc is  
  
  subtype Entier is Integer;  
  
  package Fichier is  
    type Fichier is tagged null record;  
    procedure Ecrire (F : Fichier; Valeur : Entier);  
    procedure Ecrire (F : Fichier; Valeur : String := "");  
  end Fichier;  
  
  package body Fichier is  
    use Ada.Text_IO;  
    procedure Ecrire (F : Fichier; Valeur : Entier) is  
    begin  
      Put (Valeur'Img);  
    end Ecrire;  
    procedure Ecrire (F : Fichier; Valeur : String := "") is  
    begin  
      if Valeur = "" then  
        New_Line;  
      else  
        Put (Valeur);  
      end if;  
    end Ecrire;  
  end Fichier;  
  
  Erreur : Fichier.Fichier; -- objet global  
  
  package Pile is  
    type Tableau_Entier is array (1 .. 10) of Entier;  
  
    type Pile is tagged record  
      Pile : Tableau_Entier;  
      Sommet : Entier := 0;  
    end record;  
  
    procedure Empiler (P : in out Pile; Valeur : Entier);  
    procedure Depiler (P : in out Pile);  
    function ValSommet (P : Pile) return Entier;  
    -- nommée Sommet dans le texte original provoquant une confusion avec le composant  
    procedure Vider (P : in out Pile);  
    procedure Ecrire (P : Pile; Sortie : Fichier.Fichier);  
    -- les deux paramètres ont des types objets  
  end Pile;  
  
  package body Pile is  
    procedure Empiler (P : in out Pile; Valeur : Entier) is  
    begin  
      P.Sommet := P.Sommet + 1;  
      P.Pile (P.Sommet) := Valeur;  
    end Empiler;  
    procedure Depiler (P : in out Pile) is  
    begin  
      P.Sommet := P.Sommet - 1;  
    end Depiler;  
    function ValSommet (P : Pile) return Entier is  
    begin  
      return P.Pile (P.Sommet);  
    end ValSommet;  
  end Pile;  
end ToursDeHanoi_Obj_Acc;
```

```
end ValSommet;
procedure Vider (P : in out Pile) is
begin
  while P.Sommet > 0 loop
    P.Depiler;
  end loop;
end Vider;
procedure Ecrire (P : Pile; Sortie : Fichier.Fichier) is
begin
  for I in 1 .. P.Sommet loop
    Sortie.Ecrire (P.Pile (I));
  end loop;
end Ecrire;
end Pile;

package Tour is
type Tour is new Pile.Pile with null record;
-- héritage Ada 95 avec spécification d'extension des composants obligatoire même vide
-- -> spécialisation de l'objet : Tour est une Pile dont les valeurs sont décroissantes
type PTour is access all Tour;
procedure Initialiser (T : in out Tour; N : Entier);
function PeutEmpiler (T : Tour; Valeur : Entier) return Boolean;
overriding procedure Empiler (T : in out Tour; Valeur : Entier);
-- polymorphisme ad hoc (superposition)
function Constructor (N : Entier) return PTour;
-- devra être redéfinie par un descendant
procedure Placer (T : in out Tour; PosX, PosY : Entier) is null;
-- ne fait rien dans Tour mais prépare une dérivation vers TourG (call back de la
--procédure dérivée), également utile lors de l'appel (T'Class).Primitive
procedure Dessiner (T : Tour) is null;
-- attention après Free plus de primitive possible !
procedure Free is new Unchecked_Deallocation (Tour, PTour);
end Tour;

package body Tour is
procedure Initialiser (T : in out Tour; N : Entier) is
begin
  T.Sommet := 0;
  for I in reverse 1 .. N loop
    -- empilement de valeurs décroissantes
    T.Empiler (I);
    -- invoquation de la méthode redéfinie dans Tour
  end loop;
end Initialiser;
function PeutEmpiler (T : Tour; Valeur : Entier) return Boolean is
begin
  if T.Sommet = 0 then
    return True;
  else
    return Valeur < T.ValSommet;
    -- n'a pas été définie mais hérite de Pile
  end if;
end PeutEmpiler;
procedure Empiler (T : in out Tour; Valeur : Entier) is
begin
  if T.PeutEmpiler (Valeur) then
    Pile.Pile (T).Empiler (Valeur);
    -- invoquation explicite de la méthode définie dans Pile avec conversion de l'objet hérité
  else
    Erreur.Ecrire ("Erreur : impossible d'empiler ");
    Erreur.Ecrire (Valeur);
    Erreur.Ecrire;
  end if;
end Empiler;
function Constructor (N : Entier) return PTour is
  T : constant PTour := new Tour;
begin
  T.Initialiser (N);
  return T;
end Constructor;
end Tour;

package Fenetre is
```

```
type Fenetre is tagged null record;
procedure Effacer (F : Fenetre);
procedure Ecrire (F : Fenetre; Sortie : Fichier.Fichier; Valeur : String := "");
end Fenetre;

package body Fenetre is
procedure Effacer (F : Fenetre) is
begin
  null;
end Effacer;
procedure Ecrire (F : Fenetre; Sortie : Fichier.Fichier; Valeur : String := "") is
begin
  Sortie.Ecrire (Valeur);
end Ecrire;
end Fenetre;

package Rectangle is
type Rectangle is tagged record
  X, Y : Entier;
  L, H : Entier;
end record;
procedure Centre (R : in out Rectangle; PosX, PosY : Entier);
procedure Taille (R : in out Rectangle; Largeur, Hauteur : Entier);
procedure Dessiner (R : Rectangle; F : Fenetre.Fenetre);
end Rectangle;

package body Rectangle is
procedure Centre (R : in out Rectangle; PosX, PosY : Entier) is
begin
  R.X := PosX;
  R.Y := PosY;
end Centre;
procedure Taille (R : in out Rectangle; Largeur, Hauteur : Entier) is
begin
  R.L := Largeur;
  R.H := Hauteur;
end Taille;
procedure Dessiner (R : Rectangle; F : Fenetre.Fenetre) is
begin
  F.Effacer;
  for I in 1 .. R.X loop
    F.Ecrire (Erreur, " ");
  end loop;
  for I in 1 .. R.L loop
    F.Ecrire (Erreur, "*");
  end loop;
  F.Ecrire (Erreur);
end Dessiner;
end Rectangle;

package TourG is
type TourG is new Tour.Tour with record
-- héritage Ada 95 avec spécification d'extension des composants
-- -> enrichissement de l'objet : TourG est une Tour qui a la possibilité de se dessiner
  F : Fenetre.Fenetre;
  X, Y : Entier;
end record;
type PTourG is access all TourG;
procedure Placer (T : in out TourG; PosX, PosY : Entier);
procedure Dessiner (T : TourG);
overriding procedure Empiler (T : in out TourG; Valeur : Entier);
-- redefinition pour ajouter l'affichage
overriding procedure Depiler (T : in out TourG);
-- redefinition pour ajouter l'affichage
not overriding function Constructor (N : Entier; X, Y : Entier) return PTourG;
-- attention après Free plus de primitive possible !
procedure Free is new Unchecked_Deallocation (TourG, PTourG);
end TourG;

package body TourG is
procedure Placer (T : in out TourG; PosX, PosY : Entier) is
begin
  T.X := PosX;
```

```
T.Y := PosY;
T.Dessiner;
end Placer;
procedure Dessiner (T : TourG) is
  R : Rectangle.Rectangle;
begin
  T.F.Effacer;
  for I in 1 .. T.Sommet loop
    R.Centre (T.X, T.Y - I);
    R.Taille (T.Pile (I), 1);
    R.Dessiner (T.F);
  end loop;
end Dessiner;
procedure Empiler (T : in out TourG; Valeur : Entier) is
begin
  Tour.Tour (T).Empiler (Valeur);
  -- n'a pas été définie mais hérite de Pile
  -- T.Dessiner;
end Empiler;
procedure Depiler (T : in out TourG) is
begin
  Tour.Tour (T).Depiler;
  -- n'a pas été définie mais hérite de Pile
  -- T.Dessiner;
end Depiler;
function Constructor (N : Entier; X, Y : Entier) return PTourG is
  -- invoqation explicite du constructeur ancêtre
  T : Tour.PTour := Tour.Constructor (N);
  TG : constant PTourG := new TourG'(T.all with F => <>, X => X, Y => Y);
begin
  Tour.Free (T);
  return TG;
end Constructor;
end TourG;

package Hanoi is
type TourPos is (Gauche, Centre, Droite);
type Tableau_Tour is array (TourPos) of access Tour.Tour'Class;
-- permet la référence dynamique de tous les descendants de Tour dont TourG
type Hanoi is tagged record
  Tours : Tableau_Tour;
end record;

procedure Construire (H : in out Hanoi; N : Entier);
procedure Initialiser (H : in out Hanoi; N : Entier);
procedure Deplacer (H : in out Hanoi'Class; De, Vers : TourPos);
-- permet l'invoqation dynamique pour la procédure Bouger
procedure Bouger (H : in out Hanoi; D : Entier; De, Vers : TourPos) is null;
-- ne fait rien dans Hanoi mais prépare une dérivation (call back de la procédure dérivée),
-- également utile lors de l'appel (T'Class).Primitive
procedure Jouer (H : in out Hanoi; De, Vers, Par : TourPos; N : Entier);
end Hanoi;

package body Hanoi is
procedure Construire (H : in out Hanoi; N : Entier) is
begin
  H.Tours (Gauche) := Tour.Constructor (N);
  H.Tours (Centre) := Tour.Constructor (0);
  H.Tours (Droite) := Tour.Constructor (0);
end Construire;
procedure Initialiser (H : in out Hanoi; N : Entier) is
begin
  H.Tours (Gauche).Initialiser (N);
  H.Tours (Centre).Vider;
  H.Tours (Droite).Vider;
end Initialiser;
procedure Deplacer (H : in out Hanoi'Class; De, Vers : TourPos) is
  Disque : constant Entier := H.Tours (De).ValSommet;
begin
  if H.Tours (Vers).PeutEmpiler (Disque) then
    H.Tours (De).Depiler;
    H.Tours (Vers).Empiler (Disque);
    H.Bouger (Disque, De, Vers); -- notifier le déplacement
```

```
        else
            Erreur.Ecrire ("Déplacer : coup impossible.");
            Erreur.Ecrire;
        end if;
    end Deplacer;
procedure Jouer (H : in out Hanoi; De, Vers, Par : TourPos; N : Entier) is
begin
    if N > 0 then
        Jouer (H, De, Par, Vers, N - 1);
        H.Deplacer (De, Vers);
        Erreur.Ecrire ("Déplacement de ");
        Erreur.Ecrire (TourPos'Pos (De));
        Erreur.Ecrire (" vers ");
        Erreur.Ecrire (TourPos'Pos (Vers));
        Erreur.Ecrire;
        Jouer (H, Par, Vers, De, N - 1);
    end if;
end Jouer;
end Hanoi;

package HanoiG is
    type HanoiG is new Hanoi.Hanoi with record
        F : Fenetre.Fenetre;
    end record;

    procedure Construire (H : in out HanoiG);

private
    procedure ConstruireTour (H : in out HanoiG; T : Hanoi.TourPos; X, Y : Entier);
    procedure Bouger (H : in out HanoiG; D : Entier; De, Vers : Hanoi.TourPos);
    -- animation du disque
end HanoiG;

package body HanoiG is
    procedure Construire (H : in out HanoiG) is
    begin
        H.ConstruireTour (Hanoi.Gauche, 10, 10);
        H.ConstruireTour (Hanoi.Centre, 20, 10);
        H.ConstruireTour (Hanoi.Droite, 30, 10);
    end Construire;
    procedure ConstruireTour (H : in out HanoiG; T : Hanoi.TourPos; X, Y : Entier) is
    begin
        H.Tours (T) := new TourG.TourG;
        H.Tours (T).Placer (X, Y);
        -- invocation dynamique de Placer de TourG
    end ConstruireTour;
    procedure Bouger (H : in out HanoiG; D : Entier; De, Vers : Hanoi.TourPos) is
    begin
        Erreur.Ecrire ("Animation du disque taille : ");
        Erreur.Ecrire (D);
        Erreur.Ecrire;
        for T in Hanoi.TourPos loop
            H.Tours (T).Dessiner;
        end loop;
    end Bouger;
end HanoiG;

P1 : Pile.Pile;
S : Entier;
T : Tour.PTour := Tour.Constructor (4);
TG : TourG.PTourG := TourG.Constructor (4, 1, 3);
H : Hanoi.Hanoi;
HG : HanoiG.HanoiG;

begin
    P1.Emplier (10);
    P1.Emplier (15);
    P1.Depiler;
    S := P1.Sommet; -- appelle le champ et non pas la fonction!
    S := Pile.ValSommet (P1); -- appelle la fonction
    S := P1.ValSommet; -- appelle la fonction
    S := P1.Pile (5);
```

```
T.Emplier (10);
T.Depiler;
T.Emplier (20);
T.Emplier (5);
T.Emplier (15);
Tour.Free (T);

-- TG.Initialiser (4);
-- TG.Placer (1, 3);
TG.Emplier (10);
TG.Depiler;
TG.Emplier (20);
TG.Emplier (5);
TG.Emplier (15);
TG.Dessiner;
TourG.Free (TG);

H.Construire (4);
-- H.Initialiser(4);
H.Deplacer (Hanoi.Gauche, Hanoi.Centre);
H.Deplacer (Hanoi.Gauche, Hanoi.Droite);
H.Deplacer (Hanoi.Droite, Hanoi.Centre);
H.Initialiser (3);
H.Jouer (Hanoi.Gauche, Hanoi.Droite, Hanoi.Centre, 3);

HG.Construire;
HG.Initialiser (3);
HG.Jouer (Hanoi.Gauche, Hanoi.Droite, Hanoi.Centre, 3);
end ToursDeHanoi_Obj_Acc;
```