

```
with Text_IO; use Text_IO;
procedure Figures_Abstraites2005 is

package Figure is
  type Coordonnee is range -100 .. 100;
  type Instance is abstract tagged private;
  -- remarquez le mot clé "abstract" qui déclare l'objet abstrait
  subtype Class is Instance'Class;
  -- noter l'attribut "'Class" pour permettre le polymorphisme, Class
  --inclut l'instance et tous ces descendants
  procedure Positionne (Objet : in out Instance; X, Y : Coordonnee);
  function RetourneX (Objet : in Instance) return Coordonnee;
  function RetourneY (Objet : in Instance) return Coordonnee;
  procedure Allume (Objet : in Instance) is abstract;
  procedure Eteins (Objet : in Instance) is abstract;
  procedure Affiche (Objet : in Instance'Class; EstVisible : Boolean);
  procedure Deplace (Objet : in out Instance'Class; DX, DY : Coordonnee);
private
  type Instance is abstract tagged record
    -- remarquez le mot clé "abstract" qui déclare l'objet abstrait
    X, Y : Coordonnee := 0;
  end record;
end Figure;

package body Figure is
  procedure Positionne (Objet : in out Instance; X, Y : Coordonnee) is
  begin
    Objet.X := X;
    Objet.Y := Y;
  end Positionne;
  function RetourneX (Objet : in Instance) return Coordonnee is
  begin
    return Objet.X;
  end RetourneX;
  function RetourneY (Objet : in Instance) return Coordonnee is
  begin
    return Objet.Y;
  end RetourneY;
  procedure Affiche (Objet : in Instance'Class; EstVisible : Boolean) is
  begin
    if EstVisible then
      Objet.Allume;
    else
      Objet.Eteins;
    end if;
  end Affiche;
  procedure Deplace (Objet : in out Instance'Class; DX, DY : Coordonnee) is
  begin
    Objet.Affiche (False);
    -- on cache la figure
    Objet.Positionne (Objet.X + DX, Objet.Y + DY);
    -- on déplace la figure
    Objet.Affiche (True);
    -- on affiche la figure
  end Deplace;
end Figure;

package Point is
  type Instance is new Figure.Instance with private;
  -- noter le mot clé "new" permettant l'héritage
  -- et le mot clé private pour l'encapsulation
  subtype Class is Instance'Class;
  -- noter l'attribut "'Class" pour permettre le polymorphisme, Class
  --inclut l'instance et tous ces descendants
private
  -- déclaration des méthodes réelles
  procedure Allume (Objet : in Instance);
  procedure Eteins (Objet : in Instance);
  type Instance is new Figure.Instance with null record;
  -- et le mot clé "with" pour ajouter des champs, ici on n'ajoute rien
  --pour faire un point
end Point;
```

```
package body Point is
  procedure Allume (Objet : in Instance) is
  begin
    Put_Line ("Allume un point");
  end Allume;
  procedure Eteins (Objet : in Instance) is
  begin
    Put_Line ("Eteins un point");
  end Eteins;
end Point;

package Cercle is
  type Instance is new Figure.Instance with private;
  -- noter le mot clé "new" permettant l'héritage
  -- et le mot clé private pour l'encapsulation
  subtype Class is Instance'Class;
  -- noter l'attribut "'Class" pour permettre le polymorphisme, Class
  -- inclut l'instance et tous ces descendants
  procedure Positionne
    (Objet : in out Instance;
     X, Y, R : Figure.Coordonnee);
  function Retourner (Objet : in Instance) return Figure.Coordonnee;
private
  -- déclaration des méthodes réelles
  procedure Allume (Objet : in Instance);
  procedure Eteins (Objet : in Instance);
  type Instance is new Figure.Instance with record
  -- noter le mot clé "new" permettant l'héritage
  -- et le mot clé "with" pour ajouter des champs
    R : Figure.Coordonnee := 0;
    -- l'initialisation par défaut à zéro ce qui revient à un Point ;- )
  end record;
end Cercle;

package body Cercle is
  procedure Positionne
    (Objet : in out Instance;
     X, Y, R : Figure.Coordonnee)
  is
  begin
    Objet.Positionne (X, Y);
    -- on utilise le constructeur hérité de Figure
    Objet.R := R;
  end Positionne;
  function Retourner (Objet : in Instance) return Figure.Coordonnee is
  begin
    return Objet.R;
  end Retourner;
  procedure Allume (Objet : in Instance) is
  begin
    Put_Line ("Allume un cercle");
  end Allume;
  procedure Eteins (Objet : in Instance) is
  begin
    Put_Line ("Eteins un cercle");
  end Eteins;
end Cercle;

MonPoint : Point.Instance; -- instanciation de l'objet
MonCercle : Cercle.Instance; -- instanciation de l'objet

begin
  MonPoint.Positionne (15, 25);
  Put_Line ("Y : " & MonPoint.RetourneY'Img);
  MonPoint.Affiche (True);

  MonCercle.Positionne (10, 10, 30);
  Put_Line ("R : " & MonCercle.Retourner'Img);
  MonCercle.Affiche (True);
  MonCercle.Deplace (10, 20);
end Figures_Abstraites2005;
```