

```
with Text_IO; use Text_IO;
procedure Figures2005 is

package Figure is
type Coordonnee is range -10000 .. 10000;
subtype Surface is Float;
type Instance is tagged private;
-- remarquez le mot clé "tagged" qui déclare l'objet et "private" qui rend sa structure interne inv
isable
subtype Class is Instance'Class;
-- noter l'attribut "'Class" pour permettre le polymorphisme, Class inclut l'instance et tous ces d
escendants
procedure Positionne (Objet : in out Instance; X, Y : Coordonnee);
function RetourneX (Objet : in Instance) return Coordonnee;
function RetourneY (Objet : in Instance) return Coordonnee;
private
type Instance is tagged record
-- remarquez le mot clé "tagged" qui déclare l'objet
    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;
function RetourneX (Objet : in Instance) return Coordonnee is
begin
    return Objet.X;
end;
function RetourneY (Objet : in Instance) return Coordonnee is
begin
    return Objet.Y;
end;
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 d
escendants
generic
with procedure Allume (X, Y : Figure.Cooronnee);
with procedure Eteins (X, Y : Figure.Cooronnee);
procedure Affiche (Objet : in Instance; EstVisible : Boolean);
private
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 Affiche (Objet : in Instance; EstVisible : Boolean) is
begin
    if EstVisible then
        Allume(Objet.RetourneX, Objet.RetourneY);
    else
        Eteins(Objet.RetourneX, Objet.RetourneY);
    end if;
end;
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 d
escendants
not overriding
```

```
procedure Positionne (Objet : in out Instance; X, Y, R : Figure.Coordonnee);
function RetourneR (Objet : in Instance) return Figure.Coordonnee;
function Aire (Objet : in Instance) return Figure.Surface;
procedure AfficheAire (Objet : in Instance'Class; Commentaire : String);
private
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;
function RetourneR (Objet : in Instance) return Figure.Coordonnee is
begin
    return Objet.R;
end;
function Aire (Objet : in Instance) return Figure.Surface is
begin
    return 3.14 * Figure.Surface(Objet.R) * Figure.Surface(Objet.R);
end;
procedure AfficheAire (Objet : in Instance'Class; Commentaire : String) is
begin
    Put_Line(commentaire & Objet.Aire'Img);
end;
end Cercle;

package Carre is
type Instance is new Cercle.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 d
descendants
overriding
function Aire (Objet : in Instance) return Figure.Surface;
private
type Instance is new Cercle.Instance with null record;
-- et le mot clé "with" pour ajouter des champs, ici on n'ajoute rien pour faire un carré à partir
du cercle
end Carre;

package body Carre is
function Aire (Objet : in Instance) return Figure.Surface is
begin
    return Figure.Surface(Objet.RetourneR) * Figure.Surface(Objet.RetourneR);
end;
end Carre;

MaFigure : Figure.Instance; -- instanciation de l'objet
MonPoint : Point.Instance; -- instanciation de l'objet
MonCercle : Cercle.Instance; -- instanciation de l'objet
MonCarre : Carre.Instance; -- instanciation de l'objet

procedure Allume (X, Y : Figure.Coordonnee) is
begin
    Put_Line("Allume");
end;
procedure Eteins (X, Y : Figure.Coordonnee) is
begin
    Put_Line("Eteins");
end;
procedure MonAffiche is new Point.Affiche(Allume, Eteins);

begin
-- Notation Ada 2005
```

```
MaFigure.Positionne (1005, 2005);
Put_Line("X : " & MaFigure.RetourneX'Img);
MonPoint.Positionne (1505, 2505);
Put_Line("Y : " & MonPoint.RetourneY'Img);
MonCercle.Positionne (1005, 1005, 3005);
Put_Line("R : " & MonCercle.RetourneR'Img);
MonCercle.Positionne (4005, 5005);
MonCercle.AfficheAire("Aire cercle : ");
MonCarre.Positionne (1505, 2005, 4005);
MonCarre.AfficheAire("Aire carré : ");
MonAffiche (MonPoint, False);
-- MonPoint.MonAffiche (False); -- illégal )-:
end Figures2005;
```