```
with Text_IO; use Text_IO;
procedure Figures is

package Figure is
type Coordonnee is range -100 .. 100;
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
isible
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.Coordonnee);
with procedure Eteins (X, Y : Figure.Coordonnee);
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(RetourneX(Objet), RetourneY(Objet));
  else
    Eteins(RetourneX(Objet), RetourneY(Objet));
  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
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 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
  Positionne (Objet, 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 Class; Commentaire : String) is
  begin
  Put_Line(commentaire & Aire (Objet)'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
escendants
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(RetourneR(Objet)) * Figure.Surface(RetourneR(Objet));
  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
Figure.Positionne (MaFigure, 10, 20);
Put_Line("X : " & Figure.RetourneX (MaFigure)'Img);
Point.Positionne (MonPoint, 15, 25);
```

```
Put_Line("Y : " & Point.RetourneY (MonPoint)'Img);
Cercle.Positionne (MonCercle, 10, 10, 30);
Put_Line("R : " & Cercle.RetourneR (MonCercle)'Img);
Cercle.AfficheAire(MonCercle, "Aire cercle : ");
Carre.Positionne (MonCarre, 15, 20, 40);
Cercle.AfficheAire(MonCarre, "Aire carré : ");
MonAffiche (MonPoint, False);
end Figures;
```