Seminario de Programación en Ada

UNIVERSIDAD DE CANTABRIA

Anexo

- **Herencia y polimorfismo**

Lenguajes para OOP

UNIVERSIDAD DE CANTABRIA

Los lenguajes de programación orientada a objetos soportan:

a) ***encapsulamiento*** **de los objetos y sus operaciones**

b) extensión de objetos y ***herencia*** **de sus operaciones**

- **se crean objetos a partir de otros, y se programan sólo las diferencias**

c) ***polimorfismo*****, que significa invocar una operación de una familia de objetos parecidos; en tiempo de ejecución se elige la operación apropiada al objeto concreto utilizado**

- **a esto se llama** ***enlace tardío*** **o** ***enlace dinámico*** **(“late binding” o “dynamic binding”)**

Implementación de Objetos en Ada

UNIVERSIDAD DE CANTABRIA

***Encapsulado de objetos*: mediante paquetes**

***Clases de objetos*: tipos de datos abstractos y *etiquetados*, declarados en un paquete, o en un paquete genérico**

***Operaciones de un objeto*: subprogramas definidos en el paquete donde se declara el objeto o la clase de objetos**

Tipos etiquetados

UNIVERSIDAD DE CANTABRIA

Contienen los atributos de la clase. Declaración:

```
type Una_Clase is tagged record
   atributo1 : tipo1;
   atributo2 : tipo2;
end record;
```

Es habitual hacer los atributos privados

```
package Nombre_Paquete is
   type Una_Clase is tagged private;
   -- operaciones de la clase
private
   type Una_Clase is tagged record
      atributo1 : tipo1;
      atributo2 : tipo2;
   end record;
end Nombre_Paquete;
```

Operaciones primitivas

UNIVERSIDAD DE CANTABRIA

Los métodos de la clase se llaman *operaciones primitivas*

- **Tienen un parámetro del tipo etiquetado**
- **Están escritas *inmediatamente a continuación* del tipo etiquetado**

Extensión de tipos etiquetados

UNIVERSIDAD DE CANTABRIA

Declaración de un tipo extendido:

```
type Clase_Nueva is new Una_Clase with record
   atributo3 : tipo3;
   atributo4 : tipo4;
end record;
```

El nuevo tipo

- también es etiquetado
- hereda todos los atributos del viejo, y añade otros

Si no se desea añadir nuevos:

```
type Clase_Nueva is new Una_Clase with null record;
```

Extensión de tipos etiquetados (cont.)

UNIVERSIDAD DE CANTABRIA

La extensión puede ser con atributos privados:

```
with Nombre_Paquete; use Nombre_Paquete;
package Nuevo_Paquete is
   type Clase_Nueva is new Una_Clase with private;
   -- operaciones de la clase nueva
private
   type Clase_Nueva is new Una_Clase with record
      atributo3 : tipo3;
      atributo4 : tipo4;
   end record;
end Nuevo_Paquete;
```

Herencia de operaciones primitivas

UNIVERSIDAD DE CANTABRIA

Al extender una clase

- se heredan todas las operaciones primitivas del padre
- se puede añadir nuevas operaciones primitivas

La nueva clase puede elegir:

- *redefinir* la operación: se vuelve a escribir
  - la nueva operación puede usar la del padre y hacer más cosas: *programación incremental*
  - o puede ser totalmente diferente
- dejarla como está
  - en este caso, no escribir nada

Invocar las operaciones del padre

UNIVERSIDAD DE CANTABRIA

Para usar una operación del padre del objeto, se hace un cambio de punto de vista sobre ese objeto

- **Ejemplo: si definimos**

```
O : Clase_Nueva;
```

- **El cambio de punto de vista es como un cambio de tipo:**

```
Una_Clase(O)
```

Ejemplo de Herencia: Figuras

UNIVERSIDAD DE CANTABRIA

```
package Figuras is

   type Coordenadas is record
      X,Y : Float;
   end record;

   type Figura is tagged record
      Centro : Coordenadas;
   end record;

   procedure Dibuja (F : Figura);
   procedure Borra (F : Figura);
   function Esta_Centrada (F : Figura) return Boolean;

end Figuras;
```

Figuras (cont.)

```
with Figuras; use Figuras;
package Circulos is

   type Circulo is new Figura with record
      Radio : Float;
   end record;

   procedure Dibuja (C : Circulo); -- redefinida
   procedure Borra (C : Circulo);  -- redefinida
   -- hereda Esta_Centrada

end Circulos;
```

Figuras (cont.)

```
with Figuras; use Figuras;
package Rectangulos is

   type Rectangulo is new Figura with record
      Ancho, Alto : Float;
   end record;

   procedure Dibuja (R : Rectangulo); -- redefinida
   procedure Borra (R : Rectangulo);  -- redefinida
   -- hereda Esta_Centrada

end Rectangulos;
```

Notas:

UNIVERSIDAD DE CANTABRIA

En este ejemplo el tipo “Figura” es un registro etiquetado, y por tanto extensible. Tiene definidas tres operaciones primitivas (las operaciones primitivas son las que se definen inmediatamente a continuación del tipo y que tienen al menos un parámetro de ese tipo): Dibuja, Borra, y Esta_Centrada.

Se han hecho dos extensiones del tipo Figura: Círculo y Rectángulo. Ambas extensiones heredan las tres operaciones primitivas de la Figura. Sin embargo las operaciones Dibuja y Borra no les sirven (porque son diferentes para las figuras, los círculos, y los rectángulos), y por ello las redefinen. La operación Esta_Centrada se hereda, lo que significa que, sin tener que declararlas, existen las dos nuevas funciones siguientes:

```
function Esta_Centrada (F : Círculo) return Boolean;
function Esta_Centrada (F : Rectángulo) return Boolean;
```

Estas funciones tienen el mismo cuerpo (las mismas declaraciones e instrucciones) que la original.

Las extensiones (Círculo y Rectángulo) podrían declarar nuevas operaciones primitivas, y podrían a su vez extenderse.

Tipos etiquetados abstractos y privados

UNIVERSIDAD DE CANTABRIA

El tipo `Figura` del ejemplo anterior no representa ninguna figura concreta

- existe sólo como padre de la jerarquía

Se podría definir como *abstracto*

- no se permite crear objetos
- pueden tener *operaciones primitivas abstractas* (sin cuerpo),
  - será obligatorio redefinirlas en los hijos no abstractos

El tipo etiquetado puede tener sus atributos privados

- en ese caso hacen falta operaciones para cambiarlos y usarlos

Figuras con atributos privados

UNIVERSIDAD DE CANTABRIA

```
package Figuras_Privadas is

   type Coordenadas is record
      X,Y : Float;
   end record;

   type Figura is abstract tagged private;

   procedure Cambia_Centro
     (F : in out Figura;
      Nuevo_Centro : Coordenadas);
   procedure Dibuja (F : Figura) is abstract;
   procedure Borra (F : Figura) is abstract;
   function Esta_Centrada (F : Figura) return Boolean;

private
   type Figura is abstract tagged record
      Centro : Coordenadas;
   end record;
end Figuras_Privadas;
```

Figuras con atributos privados (cont.)

UNIVERSIDAD DE CANTABRIA

```
package Figuras_Privadas.Circulos is

   type Circulo is new Figura with private;

   procedure Dibuja (C : Circulo); -- redefinida
   procedure Borra (C : Circulo);  -- redefinida
   procedure Cambia_Radio          -- nueva
     (C : in out Circulo;
      Nuevo_Radio : Float);
   -- hereda Esta_Centrada y Cambia_Centro

private
   type Circulo is new Figura with record
      Radio : Float;
   end record;
end Figuras_Privadas.Circulos;
```

Polimorfismo

UNIVERSIDAD DE CANTABRIA

Para cada tipo etiquetado existe un tipo de "ámbito de clase":

- **identifica los objetos de ese tipo etiquetado**
- **y todos sus descendientes**

Declaración:

```
Figura'class
-- Incluye a Figura, Circulo, Rectangulo,
-- y los que se definan después
```

Existen también punteros de ámbito de clase

```
type Figura_Ref is access Figura'class;
```

Las llamadas a operaciones primitivas en las que se usan tipos o punteros de ámbito de clase son *polimórficas*

Ejemplo de entrada/salida de texto

UNIVERSIDAD DE CANTABRIA

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure Test_Ficheros_Modo_Texto is
   File1     : File_Type;
   File_Name : String := "numeros.dat";
   Num_Total : Integer := 0;
   Dato      : Integer := 0;
begin
   --  Abrimos fichero nuevo
   Create (File1, Out_File, File_Name);
   Put_Line ("Numero de valores a introducir");
   Get (Num_Total);
   Skip_Line;
   for I in 1..Num_Total loop
      --  Salida estandar
      Put ("Valor numero " & Integer'Image (I) & ": ");
      Get (Dato);
      --  Salida especifica
      Put (File1, Dato);
   end loop;
   Close (File1);
end Test_Ficheros_Modo_Texto;
```

Ejemplo de polimorfismo: Figuras

UNIVERSIDAD DE CANTABRIA

```
with Figuras;
use Figuras;
procedure Mueve (F : in out Figura'Class; A : in Coordenadas) is
begin
     Borra(F);   -- no se sabe a priori a cuál de las
                 -- operaciones de borrar se invocará
     F.Centro:=A;
     Dibuja(F);  -- no se sabe a priori a cuál de las
                 -- operaciones de dibujar se invocará
end Mueve;
```