

***Programación III***  
***I.T. Informática de Sistemas***  
***La estructura estática: Clases.***



Prof. Félix Prieto Arambillet  
Departamento de Informática  
Universidad de Valladolid  
Curso 2003/2004

# Primeras nociones

---

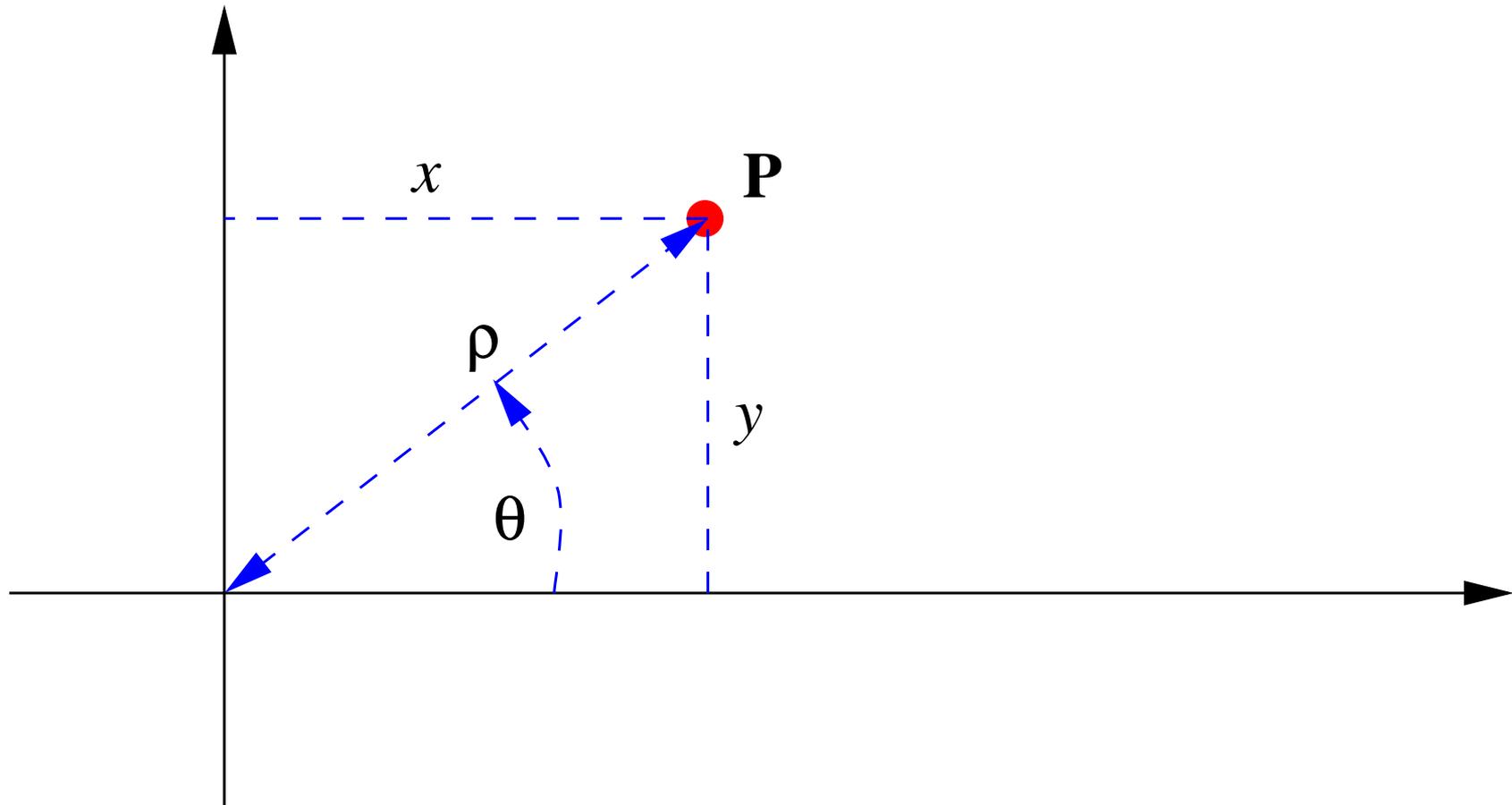
**Definición 1** *Una clase es un TAD dotado de una implementación (posiblemente parcial).*

- Las clases son **diferidas** si están parcialmente implementadas y **efectivas** en otro caso
- La instancia de una clase es un objeto del sistema

**Regla 1** *En Eiffel todo objeto del sistema es instancia de una clase, de modo que cada clase nos proporciona “un” tipo. Podemos utilizar “tipos” como **INTEGER**, **REAL**, **DOUBLE**, **BOOLEAN**, **CHARACTER**, pero **todos ellos son clases en Eiffel.***

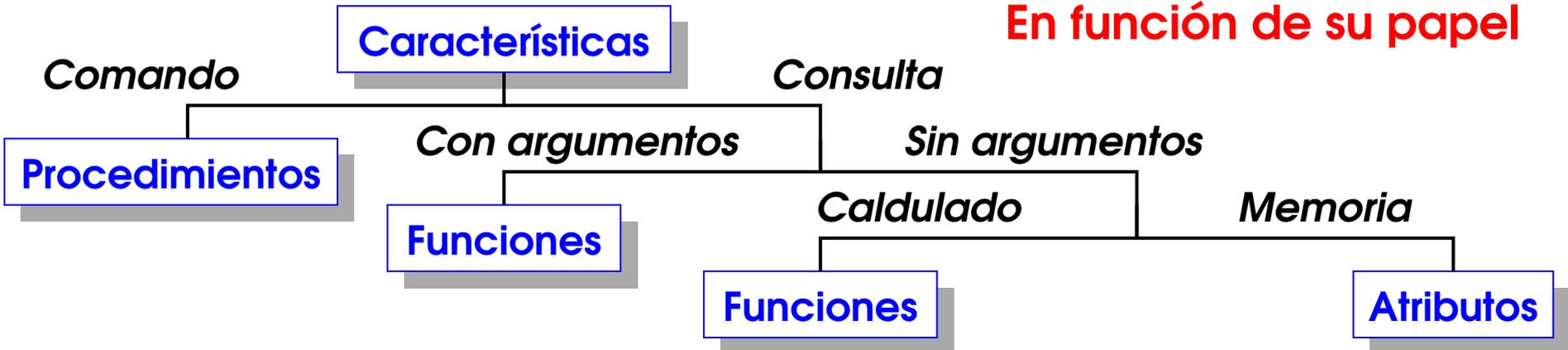
# Noción de punto en el plano

---

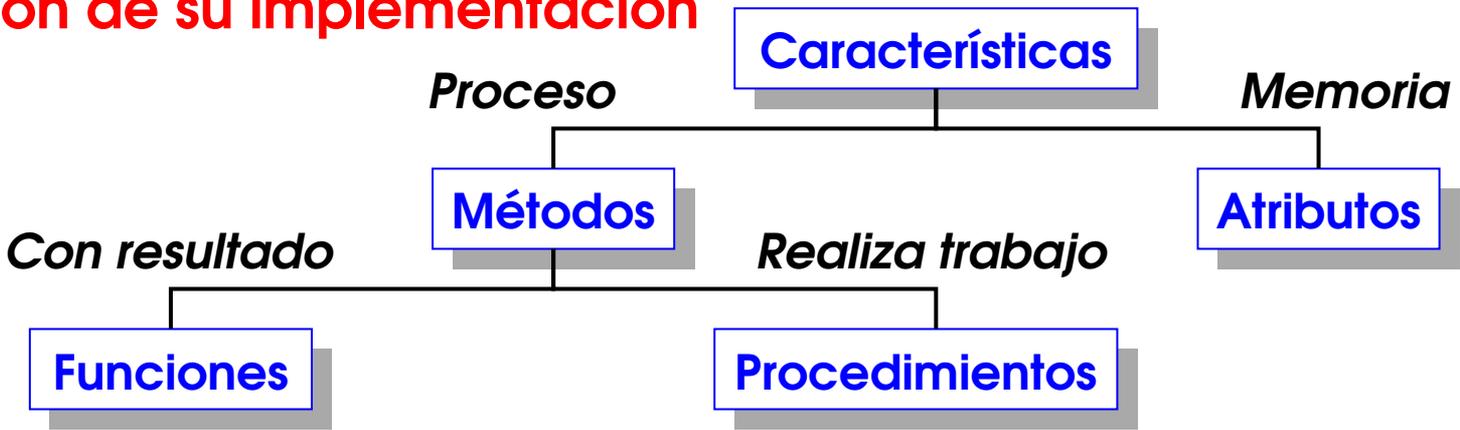


# Clasificaciones de las características

En función de su papel



En función de su implementación



# Una clase implementada en Eiffel

```

class PUNTO
  -- Representa un punto en el plano
feature
  x, y: DOUBLE;
    -- Coordenadas cartesianas.
  rho: DOUBLE is
    -- Distancia al origen.
  do
    Result := x ^ 2 + y ^ 2;
    Result := Result.sqrt;
  end;
  theta: DOUBLE is
    -- Angulo con el eje x
  do -- Sin resolver.
  end;
  escala(factor: DOUBLE) is
    -- Escala en factor.

  do
    x := factor * x;
    y := factor * y;
  end;
  distancia(p: PUNTO): DOUBLE is
    -- Distancia a un punto.
  do end; -- Sin resolver
  traslada(a, b: DOUBLE) is
    -- Traslación de coordenadas a,b
  do
    x := x + a; y := y + b;
  end;
  rota(p: PUNTO; angle: DOUBLE) is
    -- Rotación con centro y ángulo
  do -- Sin resolver
  end;
end -- class PUNTO

```

# Principio de acceso uniforme

---

**Regla 2** *En Eiffel la utilización de un atributo de una clase no se distingue de la llamada a un método sin parámetros.*

- Toda la documentación de la clase está en su “forma corta”, que se obtiene mediante el comando `short`
- El código de cada clase se guarda en un único fichero con el mismo nombre que esta, pero en minúsculas
- El comando `finder` localiza el fichero
- El comando `pretty` adapta el formato del código al estándar de Eiffel

# El concepto de entidad

---

**Definición 2** *Una entidad es:*

- *Un atributo de una clase.*
- *Una entidad local de un método.*
- *Un argumento formal de un método.*

La noción de entidad corresponde a las variables utilizadas en programación procedimental, si bien con matices que analizaremos posteriormente.

# ¿Cómo usar las clases?

---

**Definición 3** Sea  $\mathbb{P}$  una clase. Diremos que la clase  $\mathbb{C}$  es cliente de  $\mathbb{P}$  si contiene alguna declaración de la forma  $a : \mathbb{P}$ . Diremos también que  $\mathbb{P}$  es proveedor de  $\mathbb{C}$

Luego la clase **PUNTO** es cliente de **DOUBLE**. Todas las operaciones aritméticas son en realidad mensajes enviados a objetos:

```
Result := x ^ 2 + y ^ 2 ;  
Result := Result.sqrt ;
```

## Relación de cliente entre clases

---

```
class GRAFICO feature
  — Realiza alguna tarea que requiere puntos
  p1: PUNTO;
  ...
  rutina is
    — Realiza algunas acciones sobre p1
  do
    — Crea una instancia de punto
    — referenciada por p1
    p1.traslada(4.1, -3.0)
  end
end
```

El paso de mensajes está sujeto a la existencia de la característica y, en su caso, a la concordancia de argumentos formales y actuales.

## El objeto Current(II)

---

```
distancia(p: PUNTO): DOUBLE is  
  -- Distancia a un punto.  
do  
  if (Current /= p)  
    then  
      Result := ((x-p.x)^2 + (y-p.y)^2).sqrt;  
    end;  
end;
```

En Eiffel todas las entidades son inicializadas con valores por defecto, cero en el caso de las entidades numéricas.

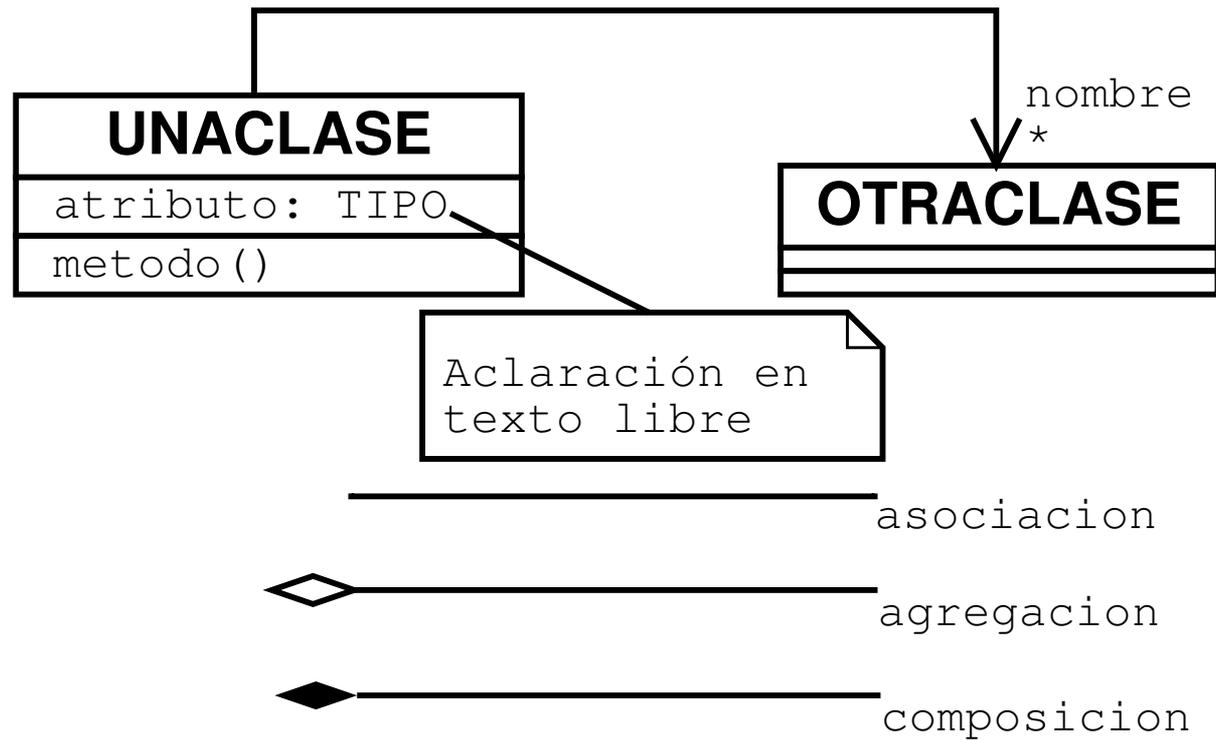
## El objeto **Current**(II)

---

Todo mensaje no cualificado se envía al objeto **Current**, de modo que en la definición de traslada  $x := \mathbf{Current}.x + a$  es equivalente a  $x := x + a$ . Sin embargo no podemos insertar **Current** por delante de  $a$  ni de la  $x$  del lado izquierdo de la expresión.

- Cualquier elemento ejecutado lo será como parte la ejecución de un método.
- La ejecución de un método siempre tiene como objetivo un objeto, al que se le pasó como mensaje el nombre del método.

# Notación UML: Diagrama de clases



Utilizaremos algunos diagramas de UML de modo poco formal.

# Operadores infijos

```
infix " | - | " (p: PUNTO): DOUBLE is  
  -- Distancia a un punto  
do  
  if (Current /= p)  
    then  
      Result := ((x-p.x)^ 2 + (y-p.y)^ 2).sqrt;  
    end;  
end;
```

Los operadores infijos deben comenzar por:

**+, -, \*, /, <, >, =, \, ^, @, #, |, &**

Además se pueden utilizar:

**not, and, or, xor, and then**(variante de **and**), **or else**(variante de **or**), **implies**

# Nivel de exportación de los atributos

---

```
class EJEMPLO
  -- Ejemplo de clase que oculta información
  feature
    -- Características públicas
  feature {A,B,C}
    -- Características utilizables por A, B, C
    -- y sus herederas
  feature {}
    -- Características privadas.
  feature {EJEMPLO}
    -- Características utilizables por EJEMPLO
    -- y sus herederas
end
```

# Niveles de protección de los atributos

---

1. Atributo privado
2. Atributo de lectura
3. Atributo de lectura escritura mediante un algoritmo específico.
4. Atributo de lectura escritura con restricciones en los valores, implementadas con precondiciones.
5. Atributo de lectura escritura sin restricciones.

# Elementos de un sistema Eiffel

---

- Un conjunto de clases
- Identificar a una de las clases del conjunto como **clase raíz** del sistema
- Identificar a un procedimiento sin argumentos de la clase raíz del sistema como **método de creación**

La ejecución del sistema se reduce a la instanciación del método de creación y la posterior ejecución de su método de creación.

# El “programa” Hola mundo

---

```
class SIMPLE
create
  make
feature {ANY}
  make is
    do
      -- ¿Quién recibe el mensaje print?
      print( "Hola_mundo%N" );
    end;
end -- class SIMPLE
```