Universidad Tecnológica Nacional
FACULTAD REGIONAL CORDOBA

PARADIGMAS DE PROGRAMACION

Unidad III

Paradigma de Programación con Orientación a Objetos

Instancias y mensajes

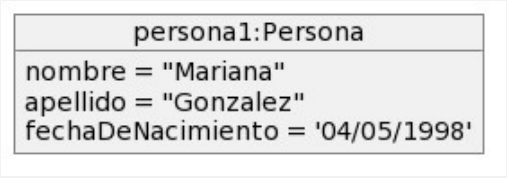
CONTENIDOS ABORDADOS

- Instancias.
- Mensajes.

Repaso: Clases e Instancias

- **Clase :** Molde a partir del cual se crean los objetos; y en el que se definen los métodos y el conjunto de variables que tendrán los objetos que se creen a partir del molde.

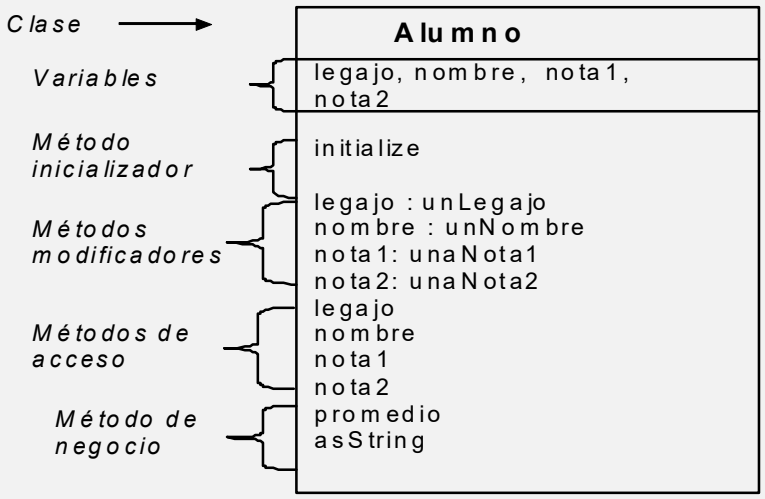
- **Instancia :** Cada objeto es instancia de la clase que se usó como molde para crearlo. Cada objeto es instancia de exactamente una clase.



Repaso: Clases en Smalltalk

Caso de estudio :

- Crear la clase Alumno que posea:
  - Atributos de instancia: legajo, nombre, nota1 y nota2.
  - Métodos: inicializador, modificadores, de acceso y de negocio.



Diseño de clase

Convención en Smalltalk:

- *Los selectores de acceso*: llevan el nombre de la variable, por ej: para dni es dni, no dameTuDni, getDni u otras variantes.
- *Los selectores modificadores*: reciben un objeto precedido del nombre de la variable y los (:), por ej: para dni es dni:unDni, no cambioTuDni, setDni u otras variantes.

Instancias en Smalltalk

Creación

- Para la creación de la instancia de una clase se debe enviar el mensaje new a la clase a instanciar. Por ejemplo:

```
unObjeto := Object new.
unAlumno := Alumno new.
otroAlumno := Alumno new.
```

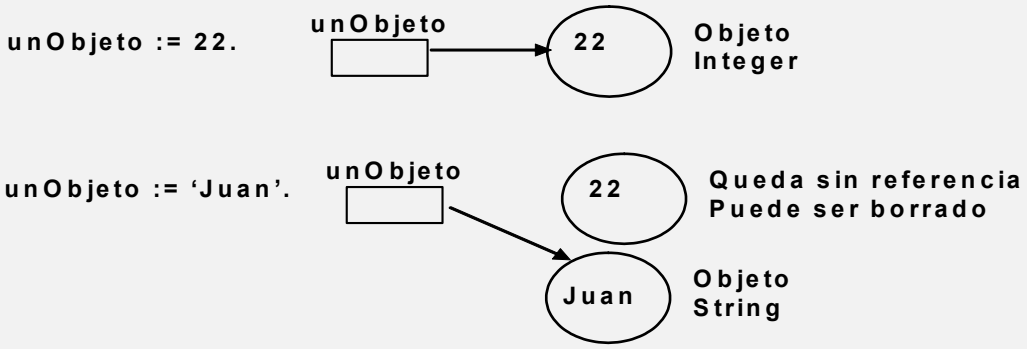
- Cuando se crea un objeto, todas sus variables de instancia (atributos) se inicializan en nil.
- Luego de instanciar un objeto, lo correcto sería invocar algún método de inicialización, para que sus variables de instancia asuman valores iniciales válidos. Por ejemplo usando el método initialize:

```
unAlumno := Alumno new initialize.
```

Instancias en Smalltalk

Destrucción

- Smalltalk posee un Recolector de basura automático (Garbage Collector) que automáticamente libera memoria.
- Cuando un Objeto deja de estar referenciado es desalojado y liberado de la memoria.



Mensajes en Smalltalk

Paso de mensajes

- Cuando se envía un mensaje a un objeto, se busca en la clase receptora el método correspondiente. Si se encuentra, se hace una copia y se ejecuta la copia. Si no se encuentra, el proceso se repite en la superclase".
- En general, una expresión Smalltalk consiste en el nombre del objeto que recibe el mensaje, seguido por el nombre del mensaje. Por ejemplo:

```
unObjeto unMensaje.
unObjeto unNumero: 1.
```

- Los mensajes en Smalltalk se clasifican en:
  - Mensajes Unarios
  - Mensajes Binarios
  - Mensajes de palabra clave

Mensajes en Smalltalk

Paso de mensajes

Mensajes Unarios

- Es similar a la llamada de una función sin parámetros.
- Se compone de un nombre de mensaje y un operando.
- Ejemplos de mensajes unarios:
- "Los resultados de la invocación se asignan en variables que van a llevar la referencia del objeto retornado"

```
f := 5 factorial.          "5 recibe el mensaje factorial"
d := Date tomorrow.        "Date recibe el mensaje de clase
                             tomorrow"
leg := unAlumno legajo.    "unAlumno recibe el mensaje
                                  legajo"
```

Mensajes en Smalltalk

Paso de mensajes

Mensajes Binarios

- Son usados para especificar operaciones aritméticas, de comparación y lógicas.
- Un mensaje binario se puede encontrar constituido por uno o dos caracteres y pueden contener una combinación de los siguientes caracteres especiales:

  `+ / \ * - < > = @ % | & ? ! ,`

- Por ejemplo:

  `c := a + b.` devuelve la suma de a y b

  el mensaje **+** es enviado al objeto **a** con el parámetro **b**.

  Podría ser equivalente a la forma: `c := a sumar: b.`

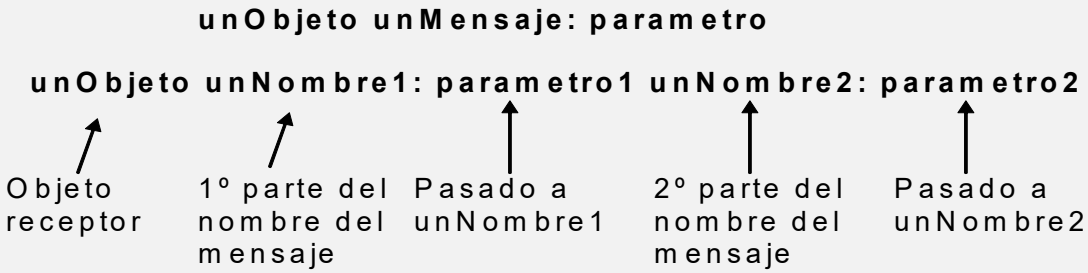
  pero el uso del operador resulta más intuitivo.

Mensajes en Smalltalk

Paso de mensajes

Mensajes De palabra clave

- Es equivalente a llamar a una función con uno o más parámetros con nombre.



- Ejemplo:

```
unAlumno nombre: 'Juan'.
 "Establece el valor de la variable de instancia del objeto Alumno de acuerdo a los parámetros"
```

Mensajes en Smalltalk

Mensajes en cascada

- Un punto y coma (;) entre dos mensajes indica que el mensaje que se encuentra inmediatamente a continuación de éste se va a enviar al mismo objeto receptor al que se ha enviado el anterior.
- Por ejemplo:

```
|unAlumno|
unAlumno := Alumno new.
unAlumno legajo: 123.
unAlumno nombre: 'Maria'.
```

ó

```
unAlumno legajo: 123; nombre: 'Maria'.
```

Mensajes en Smalltalk

Orden de evaluación de los Mensajes

1. Los mensajes se evalúan de izquierda a derecha.
2. Se ejecutan primero todas las expresiones que aparecen entre paréntesis, comenzando por la izquierda, y por aquella que está más anidada.
3. Dentro de una expresión, los mensajes unarios se ejecutan primero, luego los mensajes binarios, y finalmente los mensajes de palabra clave; siempre de izquierda a derecha.
4. Los mensajes binarios se ejecutan todos de izquierda a derecha, independientemente de las operaciones que realicen. Esto significa que **no hay un orden especial (precedencia de operadores)** para ejecutar operaciones aritméticas.

Aplicación del caso de estudio

- Usando la clase Alumno anterior, en Playground:
  - Crear objetos
  - Invocar a sus métodos

```
|unAlumno1 unAlumno2|

unAlumno1 := Alumno new initialize.

unAlumno1 legajo: 1; nombre: 'Mario'; nota1: 7; nota2: 10.

Transcript show: String cr, 'Los datos del alumno 1 son: ',
    unAlumno1 asString.
Transcript show: String cr, 'El promedio del alumno 1 es: ',
    unAlumno1 promedio asString.
```