Universidad Tecnológica Nacional
FACULTAD REGIONAL CORDOBA

PARADIGMAS DE PROGRAMACION

Unidad III

Paradigma de Programación con Orientación a Objetos

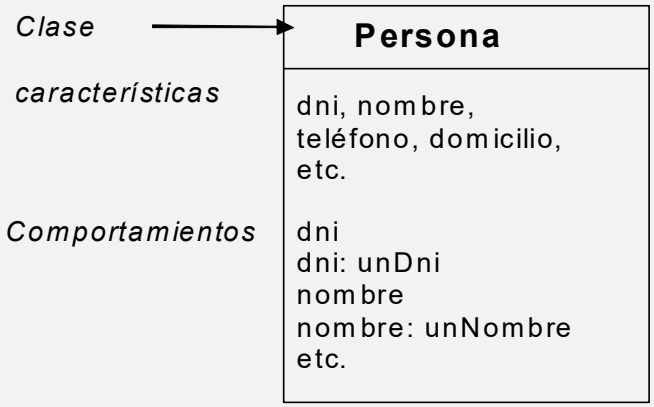
Clases

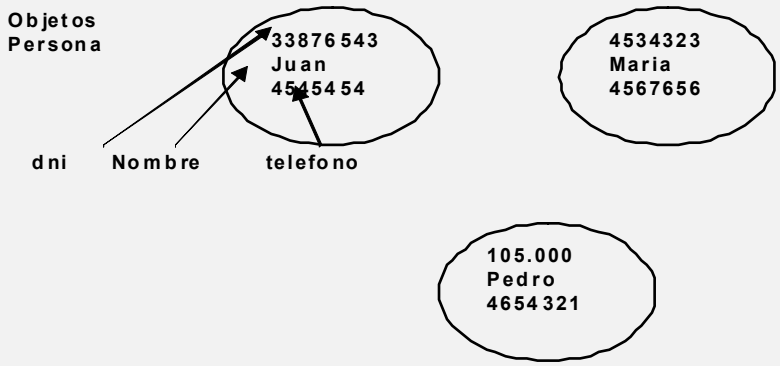
CONTENIDOS ABORDADOS

- Clases.
- Métodos.

Clases

- Es un molde o modelo para construir objetos.
- En lugar de definir cada objeto por separado, defino una clase con las características que serán comunes a los objetos, y luego voy a crear los objetos a partir de esta clase.





Clases en Smalltalk

- La definición de una clase en Smalltalk, es una instancia de una clase llamada **Metaclase**.
- Una clase es un objeto y en consecuencia también se comporta como tal.
- Una clase contiene elementos llamados miembros, que pueden ser variables de instancias (atributos) o variables de clase.
- La sintaxis para la definición de una clase en Smalltalk es:

```
Object subclass: #NombreClase
instanceVariableNames: ' '
classVariableNames: ' '
package: ' '
```

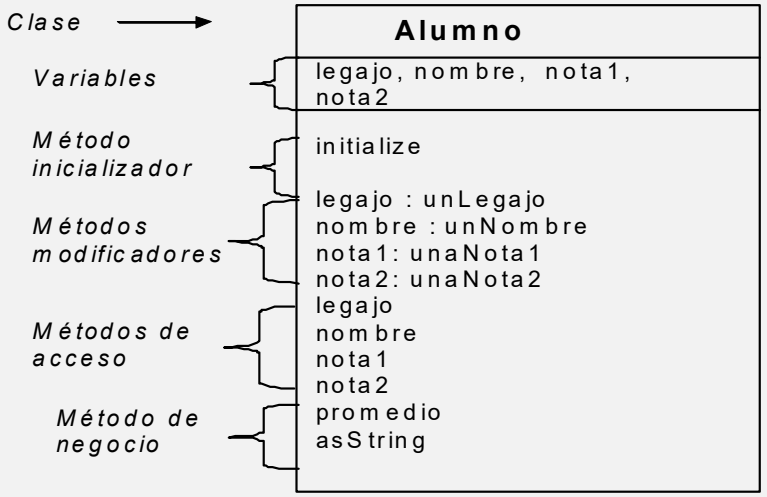
Clases en Smalltalk

- **Object subclass**: indica la clase base de la cual es clase derivada, la clase que se está creando.
- **#NombreClase**: identificador que se le asigna a la clase. Debe comenzar con mayúscula y si se encuentra constituido por más de una palabra, cada palabra debe comenzar con mayúsculas.
- **Variables de instancia**: son los atributos del objeto y existen durante el tiempo de vida del objeto instancia de la clase. Los nombres de atributos sólo pueden ser referidos desde el interior de los métodos de su clase.
- **Variables de clase**: son compartidas por todas las instancias de una clase. Hay una sola instancia de cada una de estas variables.
- **Package**: Paquetes de clases, cada uno tiene un nombre común que agrupa a varias clases.

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.

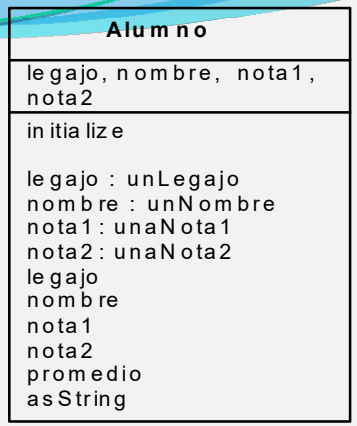


Diseño de clase

Convención en Smalltalk:

- *Los **selectores de acceso***: llevan el nombre de la variable,
  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 (:),
  Ej: para dni es dni:unDni, no cambioTuDni, setDni u otras variantes.

Clases en Smalltalk



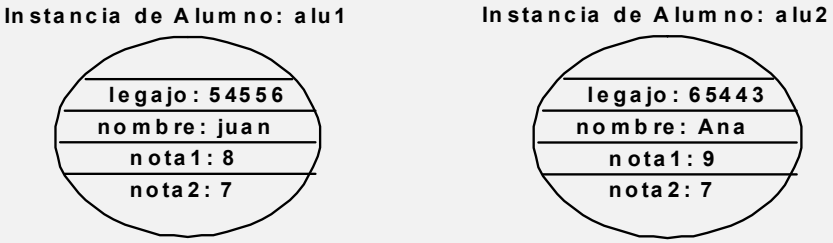
Creación de la clase

```
Object subclass: #Alumno
instanceVariableNames: 'legajo nombre nota1 nota2'
classVariableNames: ''
package: 'PPRClase1'
```

Clases en Smalltalk

Variables

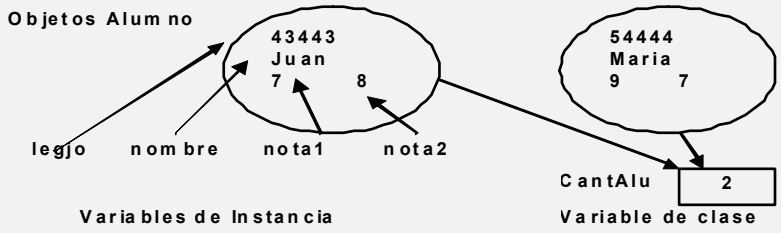
- **Variables de Instancia**: Existen durante todo el tiempo de vida de una instancia y representan el estado de la misma.
- Aunque todas las instancias de una clase tienen el mismo conjunto de variables de instancia, sus valores son únicos a cada una de las mismas.
- Por ejemplo: La clase Alumno tiene las **variables de instancia**: 'legajo nombre nota1 nota2'. Cada instancia de Alumno tendrá su propio juego de valores.



Clases en Smalltalk

Variables

- **Variables de Clase**: Es una variable que es compartida por todos los objetos de la clase.
- Todos los objetos de la clase referencian a esta única copia de la variable.
- Tienen las mismas convenciones de nombramiento que las variables de instancia, excepto que la primera letra de una variable de clase debe comenzar con mayúscula.



Clases en Smalltalk

Métodos

- Los métodos dan los detalles de implementación para los mensajes, y representan el comportamiento de la clase.
- En Smalltalk consta de dos partes:
  - **Patrón o Cabecera del método** que tiene el nombre del selector que identifica el método y los nombres de sus parámetros (colaboradores externos).
  - **Cuerpo del método**, que se encuentra constituido por tres componentes:
    - Un comentario para describir la actuación de ese método.
    - Una lista de variables temporales o locales que puedan ser utilizadas en las expresiones.
    - Las expresiones que responden al mensaje enviado.

Clases en Smalltalk

Métodos

Argumentos de métodos

- Permiten la participación de colaboradores externos en los métodos. Dichos parámetros/argumentos son necesarios para completar la funcionalidad del método.

Método sin argumentos

```
patrónDelMensaje

 "comentario"
 |variablesTemporales|
 expresiones.
```

Método con argumentos

```
patrónDelMensaje: colaboradorExterno1
 <nombreArgumento: colaboradorext2>
  "comentario"
|variablesTemporales|
  expresiones.
```

Clases en Smalltalk

Tipos de Métodos

Métodos de Clase

- Un método de clase es la implementación de un mensaje cuyo receptor es la clase misma. Los métodos de clase proveen un comportamiento de clase.
- Cada clase contiene el mensaje new, que retorna una nueva instancia.

```
miAlumno := Alumno new.
```

Clase Alumno

Métodos de Instancia

- Un método de instancia es la implementación de un mensaje cuyo receptor es un objeto.

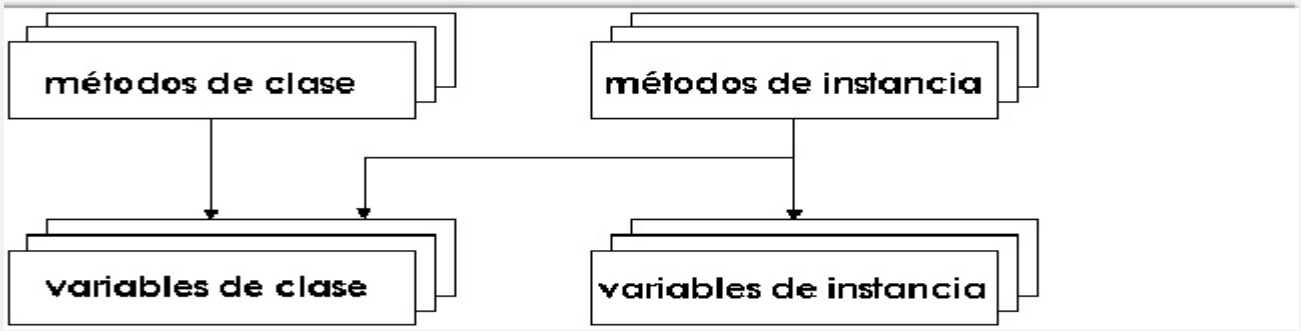
```
miAlumno nombre: 'Juan'.
```

Objeto miAlumno

Clases en Smalltalk

Relación entre métodos y variables de clase

- Los métodos de instancia y los métodos de clase pueden referirse a las variables de clase, pero a las variables de instancia sólo los métodos de instancia las pueden referir.



Clases en Smalltalk

Self

- Referencia al objeto receptor de un mensaje: objeto actual.
- Se utiliza cuando se necesita invocar a un método ya implementado en la misma clase.
- No se utiliza self para hacer referencia a un atributo de la clase.
- Invocación en Smalltalk:

Invocación fuera de la clase:

```
miAlumno legajo: 2. "modifico el legajo fuera de la clase"
```

Invocación dentro de la clase:

```
self legajo: 2. "modifico el legajo dentro de la clase"
```

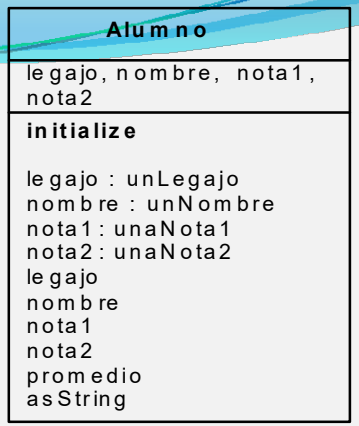
Clases en Smalltalk

Self

- Ejemplo:

```
legajo
    ^legajo.
nombre
    ^nombre.
muestraDatos
    "Retorna en forma de cadena el legajo y nombre"
    | nombre legajo|
    legajo := self legajo.      "invoca al mensaje legajo"
    nombre := self nombre.      "invoca al mensaje nombre"
    ^legajo asString, '-', nombre asString.
```

Clases en Smalltalk

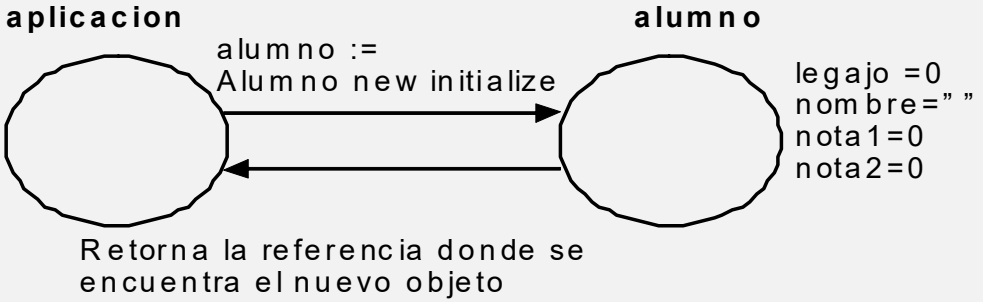


Métodos de inicialización

Variables de instancia

initialize

```
"inicializa la instancia"
legajo := 0.
nombre:= ' '.
nota1 := 0.
nota2 := 0.
```



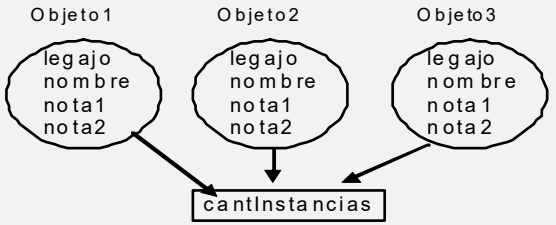
Clases en Smalltalk

Métodos de inicialización

Variables de clase

- Para la inicialización de variables de clase se debe disponer de un método de clase que tenga **como** objetivo inicializar las variables de clase.
- Este mensaje debe ser enviado a la clase antes de la creación de sus instancias, siempre que sus instancias así lo requieran.

```
inicializarVariablesDeClase
   "Inicializa una variable de clase"
   (cantidadInstancias  isNil)
      ifTrue: [ cantidadInstancias := 0].
```



Clases en Smalltalk

Métodos Modificadores

legajo : unLegajo

```
"asigna unLegajo al legajo del alumno"
    legajo := unLegajo.
```

nombre : unNombre

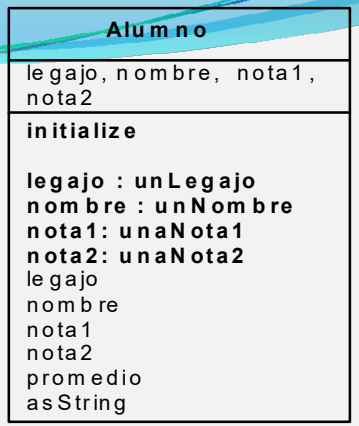
```
" asigna unNombre al nombre del alumno"
    nombre := unNombre.
```

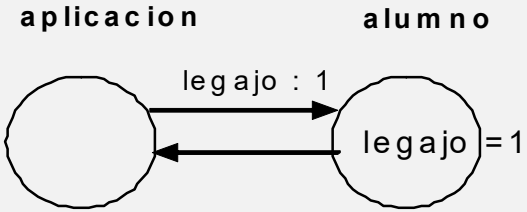
nota1 : unaNota1

```
" asigna unaNota1 a la nota1
    del alumno"
    nota1 := unaNota1.
```

nota2 : unaNota2

```
" asigna unaNota2 a la nota2 del alumno"
    nota2 := unaNota2.
```





Retorna sin valor porque su proposito es cambiar el legajo con el valor 1

Clases en Smalltalk

Métodos de Acceso

legajo

```
"retorna el legajo del alumno"
    ^ legajo.
```

nombre

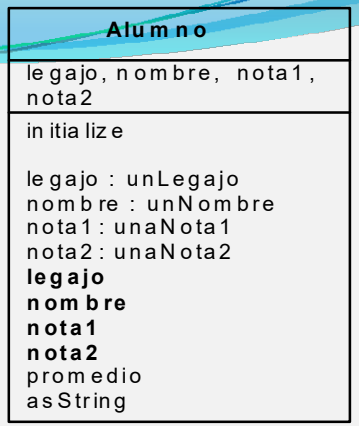
```
"retorna el nombre del alumno"
    ^ nombre.
```

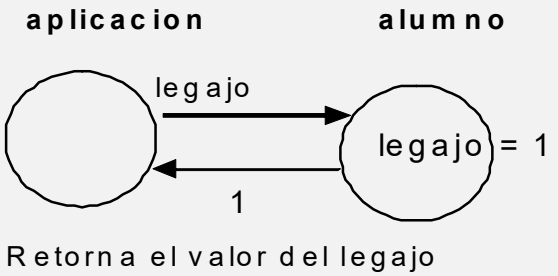
nota1

```
"retorna la nota1 del alumno"
    ^ nota1.
```

nota2

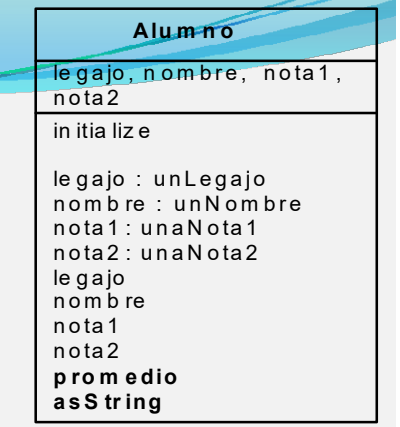
```
"retorna la nota2 del alumno"
    ^ nota2.
```





Retorna el valor del legajo

Clases en Smalltalk



Métodos de Negocio

asString

```
"retorna una cadena con los datos del alumno"
|res|
res := String cr, 'Legajo: ', self legajo asString,
       ' Nombre:', self nombre asString,
       ' Nota1: ', self nota1 asString,
       ' Nota2: ', self nota2 asString ,
       ' Promedio: ', self promedio asString.
^res.
```

Hay que realizar conversión explícita

promedio

```
"retorna el promedio de notas del alumno"
|promedio|
promedio := ((self nota1 + self nota2) asFloat) /2.0.
^promedio.
```

Aplicación del caso de estudio

- En Pharo
  - Crear la clase Alumno
  - Implementar los métodos