Universidad Tecnológica Nacional
FACULTAD REGIONAL CORDOBA

PARADIGMAS DE PROGRAMACION

Unidad III

Paradigma de Programación con Orientación a Objetos

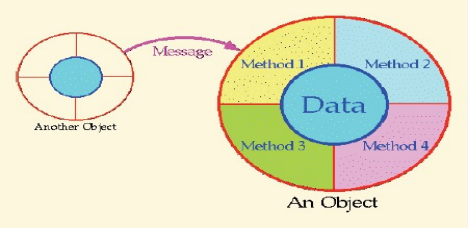
Lenguaje Smalltalk - Parte I

CONTENIDOS ABORDADOS

- Lenguaje Smalltalk:
  - Introducción.
  - Características.
  - Aspectos.
  - Entorno.
  - Sintaxis y semántica.

Introducción a Smalltalk

- Es un **lenguaje orientado a objetos puro**, todas las entidades que maneja son objetos.
- La programación consiste en:
  - **Crear clases.**
  - **Crear instancias.**
  - **Especificar la secuencia de mensajes entre objetos.**



Introducción a Smalltalk

Características

- Es un lenguaje altamente interactivo
- Se implementa como intérprete.
- Posee un ambiente completo de desarrollo de programas. Integra de una manera consistente:
  - Editor, compilador, debugger, utilitarios de impresión, sistema de ventanas, manejador de código fuente.

Aplicaciones

- Usado en las universidades, porque implementa el paradigma de objetos puro.
- Usado en aplicaciones comerciales. Por ejemplo:
  - En proceso de transacciones con tarjetas de crédito
  - En la supervisión de mercados internacionales de valores

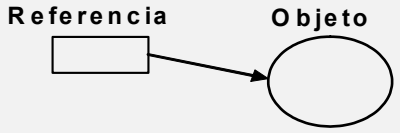
Aspectos importantes de Smalltalk

Asignación dinámica de memoria

- Solo se dispone de objetos, que son manipulados a través de apuntadores/referencias.
- Todas las referencias tienen un tamaño fijo constante y se le asigna espacio en la pila (stack).
- Los objetos referenciados se almacenan en otro sector de la memoria, llamado heap (montículo), que no está sujeto al protocolo de asignación de la pila.

Asignación de referencias

- Una asignación a una referencia, cambia el valor de la referencia por el valor contenido en otra referencia.
- si tenemos dos variables no solo tendrán el mismo valor sino que referencian al mismo objeto.
- Por ejemplo, x := y.



Aspectos importantes de Smalltalk

Asignación dinámica de tipos

- Las variables no se declaran con tipos específicos
- Posee asignación dinámica de tipos, donde cada valor debe llevar consigo una identificación que permite conocer la naturaleza exacta del valor, esta identificación queda ligada al valor a través de la asignación.
- A un identificador se le pueden pasar valores de tipos diferentes.

```
| identificadorVariable |
identificadorVariable := 10.
identificadorVariable := 'Hola'.
```

Objetos polimórficos

- Al ser un lenguaje con asignación dinámica todos los objetos son potencialmente polimórficos.
- Cualquier objeto puede tener valores de cualquier tipo.

Entorno de Smalltalk: Pharo

Las ventanas que utilizaremos del entorno Pharo son:

- **Browser**:
  - Permite inspeccionar y
  - Editar la biblioteca de clases del ambiente.
- **Playground**:
  - Contiene un editor de texto.
  - Se utiliza para evaluar expresiones.
- **Transcript**:
  - Es un editor de texto utilizado para evaluar expresiones.
  - El propio entorno también la utiliza para mostrar las salidas de los mensajes.
- **Inspect it**:
  - Permiten observar la estructura interna de un objeto y
  - Modificar el valor de las mismas.

Sintaxis y Semántica

- **Comentarios**
  - Cualquier texto que se encierra entre comillas dobles y que no provocará una ejecución del interprete.
  - Por ejemplo:
    “Esto es un comentario”
    “Esto es un comentario
    en varias líneas”

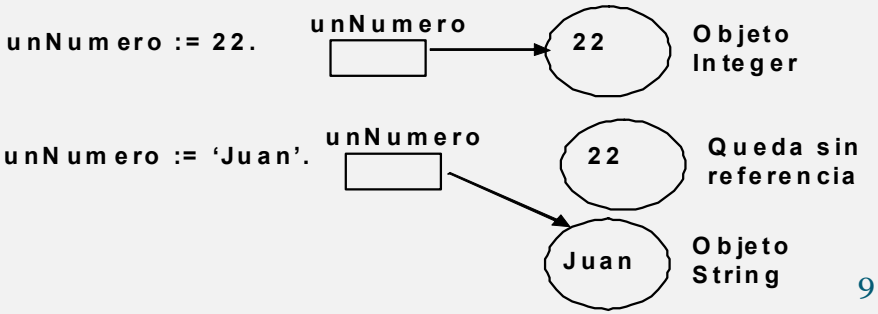
Sintaxis y Semántica

- **Operador de asignación (:=)**

  - El operador := se denomina operador de ligadura o asignación.
  - Asigna la variable de la izquierda al objeto calculado por la expresión de la derecha.
  - Es un operador especial, no es un mensaje.

  - Por ejemplo:

```
unNumero := 22.
unNumero := 'Juan'.
```



Sintaxis y Semántica

Expresión

- Smalltalk es un lenguaje basado en **expresiones**. Una expresión es una secuencia de caracteres que puede ser evaluada.
- Las expresiones terminan con un punto (.).
- Hay cuatro tipos de expresiones:
  - Literales
  - Nombres de Variables
  - Expresiones de mensajes
  - Expresiones de bloque

Sintaxis y Semántica

Literales (constantes literales u objetos constantes)

- El valor de una expresión literal es siempre el mismo objeto.
- Hay cinco tipos de constantes literales:

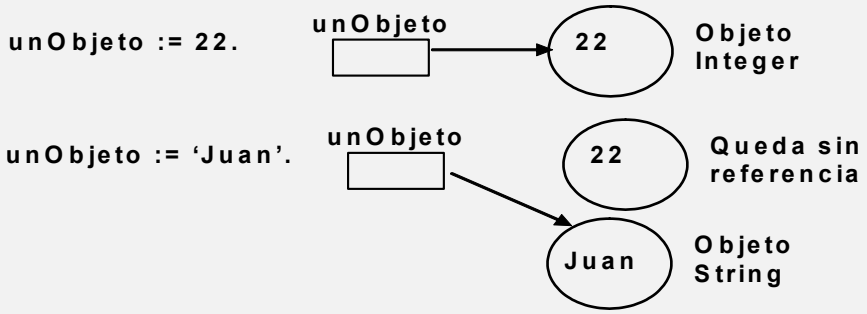
Números	Caracteres	Secuencia de caracteres
Símbolos	Arreglos	

- Ejemplos:
  - `10` número 10.
  - `$a` carácter a.
  - `'hola'` string con las letras h, o, l y a.
  - `#Alumno` símbolo (string constante)
  - `#(1 2 3)` arreglo de tres elementos enteros

Sintaxis y Semántica

Variables

- Las **variables** pueden hacer referencia a un objeto determinado en un instante de tiempo y, momentos después, a otro objeto completamente diferente.



- Los nombres de variables en SMALLTALK son identificadores que consisten en una secuencia de letras y dígitos que comienzan con una letra.

Sintaxis y Semántica

Tipos de variables

- Variables de instancia (Colaboradores internos, representan el estado del objeto).
- Variables locales o temporales (Se usan en los métodos, se declaran antes de usarse, entre pipes: `|var1 var2 var3|`).
- Colaboradores externos (Parámetros de los métodos).
- Variables globales (Son persistentes, pueden accederse desde cualquier parte del entorno: Integer, Number, Object, Transcript, etc.).
- Pseudo-Variables (son palabras reservadas y son globales y siempre referencian al mismo objeto: nil, true, false, self, super, thisContext).

Sintaxis y Semántica

Operaciones Aritméticas

- Una expresión que realiza una *operación aritmética* en Smalltalk tiene la siguiente forma:

  número **operación** número

- Una *operación* puede ser :

  + "suma“ - "resta"
  * "multiplicación" / "división"
  // "división entera (cociente)“ \\ "resto de una división"

- Ejemplos :

```
3 + 4.          "devuelve 7"
9 // 4.                "devuelve 2"
9 \\ 4.                "devuelve 1"
```

Sintaxis y Semántica

Comparaciones Lógicas

- Una *expresión de comparación* tiene el siguiente formato:

  valor **comparación** valor

- Las comparaciones lógicas devuelven un valor que puede ser *true (verdadero)* o *false (falso)*, que son instancias de las clases *True* y *False*, respectivamente.

>	"mayor que"	<	"menor que"
=	"igual en valor"	~=	"desigual en valor"
>=	"mayor o igual que"	<=	"menor o igual que"
==	"el mismo objeto que"		

- Ejemplos:

```
3 > 8  "devuelve false"
$e <= $f      "devuelve true"
```

Sintaxis y Semántica

Comparaciones Lógicas

"y" Lógico y "o" Lógico: El mensaje binario para el *y* lógico es &, y para el *o* lógico es |. Por ejemplo:

(a > 0) & (b < 0) "Devuelve true si *a* es positivo y *b* es negativo. Caso contrario devuelve false."

(a > 0) | (b < 0) "Devuelve true si *a* es positivo y/o *b* es negativo. Caso contrario devuelve false."

not : El mensaje unario *not* provee la función not. Este mensaje invierte un valor booleano (*true* se vuelve *false*, o *false* se vuelve *true*). El formato es: booleano not . Ejemplo:

```
(5 > 1) not                "El valor de retorno es false."
```

Sintaxis y Semántica

símbolo	Significado
`.`	Finalización de expresiones.
`:=`	Operador de ligadura o asignación.
`,`	Concatenación de String o cadenas.
`;`	Invocación de mensajes en cascada.
`“ ”`	Comentarios.
`' '`	Representación de String o cadenas.
`^`	Retorno de un método.
`|var1 var2 ...|`	Declaración de variables.
`nil`	Referencia vacía.
`self, super`	Referencia al objeto receptor de una clase actual y de una clase base respectivamente, se utilizan para invocar mensajes de la clase actual y/o clase base.

Sintaxis y Semántica

- Definición de una clase:

```
Object subclass: #NombreClase
instanceVariableNames: 'variable1 variable2 ... '
classVariableNames: ' '
package: ' '
```

- Definición de métodos:

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

Sintaxis y Semántica

- Para la creación de un objeto o instancia:

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

- Para la destrucción de objetos o instancias, se utiliza el recolector de basura automático (Garbage Collector), que libera la memoria ocupada por un objeto cuando deja de estar referenciado.

Sintaxis y Semántica

- La invocación de mensajes:

objeto mensaje

Ejemplo	Descripción
`3 factorial.`	El mensaje factorial es enviado al objeto 3 (instancia de la clase Integer) y se ejecutará el método de nombre factorial que se encuentra en la clase Integer o en sus superclases.
`unAlumno nota:10.`	Envía un objeto numero (10) al objeto alumno, a través del método nota:, el cual asigna el valor en el atributo nota.
`n := unAlumno nota.`	Retorna un objeto numérico que representa la nota del objeto alumno (atributo nota) a través del método nota y se asigna a un objeto numero llamado n.