Programación Orientada a Objetos

Clase 2 - Smalltalk. Mensajes . Estructuras de Control. Jerarquía de clases.

PARADIGMAS DE PROGRAMACION *UTN - La Plata*

Smalltalk

1) Características de Smalltalk
2) Mensajes
    a) Tipos de Mensajes
    b) Orden de ejecución de mensajes
3) Concepto de Variable y asignación
4) Estructuras de control
    a) Selección condicional
    b) Repetición condicional
    c) Repetición de longitud fija
5) Actividad 2
6) Árbol jerárquico de clases. Class Hierarchy Browser.

Smalltalk. Características

1) Características de Smalltalk

- Es considerado el primero de los lenguajes orientados a objetos, aunque en realidad el primero en implementar la programación orientada a objetos fue Simula.

- Ha tenido gran influencia sobre otros lenguajes como **Java** o **Ruby**, y de su entorno han surgido muchas de las prácticas y herramientas de desarrollo promulgadas actualmente por las metodologías ágiles (refactorización, desarrollo incremental, desarrollo dirigido por tests, etc.).

- Es un lenguaje de programación reflexivo y con tipado dinámico.

Smalltalk. Características

Smalltalk es un lenguaje **orientado a objetos puro**, todas las entidades que maneja son objetos.

- El lenguaje se basa en conceptos tales como objetos y mensajes.
- Es mucho más que un lenguaje de programación, es un ambiente completo de desarrollo de programas.
- Algunos de los Entornos de trabajo son: Smalltalk Express, Pharo, Squeak, VisualWorks, Dolphin.

Smalltalk. Características

- Diseñar nuevas aplicaciones Smalltalk, requiere de conocimientos sobre las clases existentes en el sistema Smalltalk.

> Se dice que programar en Smalltalk es programar por extensión.

- Las nuevas aplicaciones son construidas por extensión de las librerías de clases de Smalltalk.
- El costo de aprendizaje se recupera por ser sus aplicaciones de alta productividad por el “reuso” y “extensión del código”.

Smalltalk. Mensajes

2) Mensajes

Componentes: receptor, selector, argumentos.

Tipos de mensaje: unarios, binarios y de palabra clave.

Ejemplos:

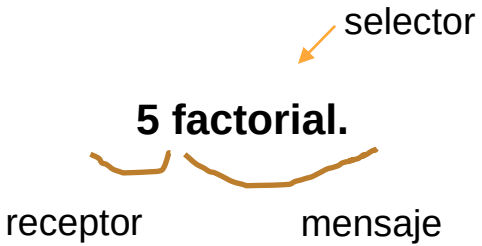
5 factorial.

3 < 5.

#(4 3 8 1) at: 4 put: 2.

Smalltalk. Mensajes

a) Mensajes unarios: no tienen argumentos



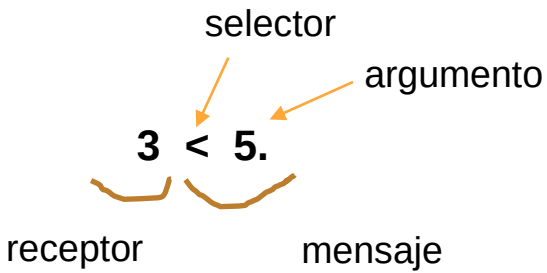
El selector es el nombre del método.

En este caso: factorial

Otros ejemplos:

```
19,76 rounded.
'abcd' size.
```

b) Mensajes binarios: tienen un solo argumento. Se utilizan para operaciones lógico, matemáticas.



El selector es el nombre del método.

En este caso: <

Otros ejemplos:

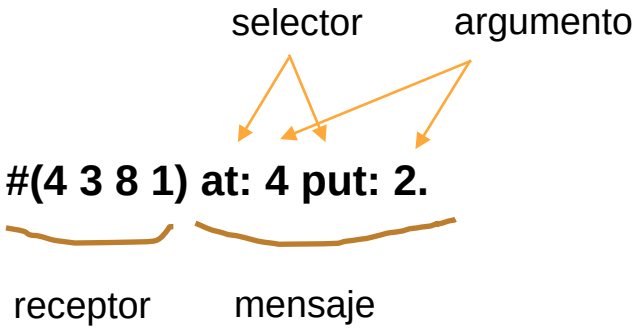
'abc' ~='def'.

true & false.

7 + 1.

Smalltalk. Mensajes

c) Mensajes de palabra clave: son mensajes con una o más palabras clave, cada palabra clave tiene un argumento asociado. Se reconoce por los dos puntos (:).



El selector es el nombre del método.

En este caso: at: put:

Otros ejemplos:

5 between:8 and:10.

En este ejemplo: between: and:

Smalltalk. Mensajes

Orden de ejecución de los mensajes:

Los mensajes en Smalltalk se ejecutan en el siguiente orden:

1°) las expresiones que están entre paréntesis ().

2°) las expresiones unarias.

3°) las expresiones binarias.

4°) las expresiones de palabra clave.

Todas de izquierda a derecha.

Smalltalk. Mensajes

Ejemplos de mensajes anidados:

2 factorial negated.

3 + 4 * 6 + 3.

5 between:1 and:3 squared + 4.

Ejemplos de mensajes en cascada:

Cuando se envían mensajes sucesivos al mismo OR

'sol' size.

'sol' inspect. ←→ 'sol' size; inspect.

'sol' size inspect

Ejemplos de polimorfismo:

1) 5 + 100. valor de retorno= 105

2) (200 @ 200) + 100. valor de retorno= (300@300)

3) (1/5) + (5/3). valor de retorno= (28/15)

Las fracciones se escriben entre paréntesis en Dolphin.

Smalltalk. Variable y asignación

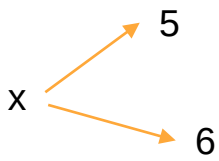
3) Variable y asignación

Una variable en Smalltalk representa un puntero a un objeto.

x :=5.

x ← x + 1

El mensaje +1 es enviado al objeto referido por x. La variable x luego apunta a la expresión resultado de evaluar x+1.



Una asignación en Smalltalk tiene el siguiente formato:

var := expresión.

Ejemplo: x := x + 1.

Todas las sentencias en Smalltalk terminan en **.** , salvo algunas excepciones.

Smalltalk. Variable y asignación

- Lenguaje no tipado, las variables NO se declaran, se asocian a un objeto y a una clase por BINDING DINÁMICO
  - z:= 3. z apunta al objeto 3 y por lo tanto es de clase Integer.
  - z:='hola'. z ahora apunta al objeto 'hola' y por lo tanto es de clase String.
  - z:= Array new:3. z ahora apunta a un objeto de la clase Array (vector).
- Hay alocación y eliminación dinámica de objetos (Garbage Collector)

Smalltalk. Tipos de variables.

- Variables de instancia: propias de cada objeto, representan el Estado Interno del mismo.
- Variables de clase: comunes a todos los objetos de la misma clase, tienen EL MISMO valor para todos los objetos de la clase.
- Variables temporales: locales a un método o aplicación.
- Argumentos: parámetros en los métodos.

4) Estructuras de control

En Smalltalk no existen las estructuras de control, se simulan, están implementadas en términos de objetos y mensajes.

a) Selección condicional:

- **(expresión booleana) ifTrue:[TrueBlock] ifFalse:[FalseBlock].**
- **(expresión booleana) ifTrue:[TrueBlock].**
- **(expresión booleana) ifFalse:[FalseBlock].**

Ejemplo:

```
|a b|
a:= 3.
b:= 8.
(a < b) ifTrue:[a:=a +1]
        ifFalse:[a:= b*2].
a inspect.
```

Interpretación del IfTrue: ifFalse:

Los objetos boolean true y false aceptan los mensajes de palabra clave ifTrue: ifFalse:

1) La expresión booleana es evaluada, dará un objeto true o false como resultado.
2) Si el resultado es true el mensaje ifTrue: ifFalse: con sus argumentos será enviado al objeto true, sino será enviado al objeto false.
3) El objeto true responde el mensaje evaluando la expresión TrueBlock.

Ejemplo:

```
|a b|
a:= 3.
b:= 8.
(a < b) ifTrue:[a:=a +1]
        ifFalse:[a:= b*2].
a inspect.
```

Smalltalk. Estructuras de control

b) Repetición condicional:

- **[expresión booleana] whileTrue:[cuerpo del loop].**
- **[expresión booleana] whileFalse:[cuerpo del loop].**

Ejemplo:

```
|suma i|
i:=1.
suma:= 0.
[i <=10] whileTrue:[suma:= suma + i.
                    i:=i+1].
Transcript show: suma displayString.
```

El mensaje show: lleva como argumento un objeto de clase String.

El mensaje displayString convierte al valor de suma en un string.

Interpretación del whileTrue:

1) El mensaje whileTrue [ ] es enviado al bloque [expresión booleana].

2) En respuesta el bloque se evalúa.

3) Si el bloque retorna el objeto true se evalúa el cuerpo del loop y el mensaje whileTrue es nuevamente enviado al bloque [expresión booleana] y se repiten los pasos 1), 2) y 3).

Ejemplo:

```
|suma i|
i:=1.
suma:= 0.
[i <=10] whileTrue:[suma:= suma + i.
                    i:=i+1].
Transcript show: suma displayString.
```

Smalltalk. Estructuras de control

c) Repetición de longitud fija:

- valorInicial **to:** valorFinal **do: [**variable del loop **|** cuerpo del loop**].**

Interpretación del to:do:

El mensaje

valorInicial to: valorFinal do:unBloque

evalúa el argumento unBloque para cada entero que esté en el intervalo dado por el valor inicial hasta el valor final incluido.

Ejemplo:

```
|suma|
suma:= 0.
1 to: 10 do: [:i | suma:= suma + i ].
suma inspect.
```

5) Actividad 2

1) Resolver los siguientes mensajes anidados, indicando objeto receptor, mensaje, selector y argumentos, de cada uno de los mensajes. Indicar en cada caso también el valor de retorno. Resolver paso a paso indicando el procedimiento utilizado.

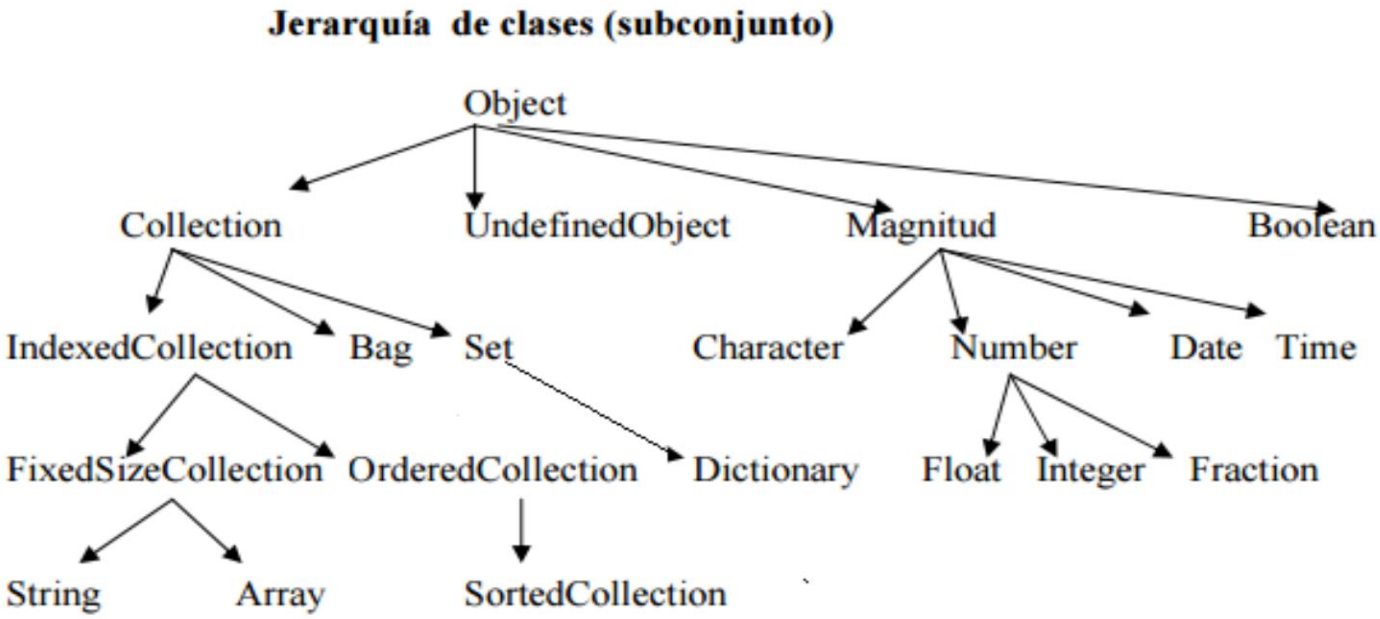
   a) 6 squared negated.

   b) 2 * 4 + 6 - 3.

   c) 5 between:1+2 and: 3 * 2 squared.

Jerarquía de clases (Class Hierarchy Browser)

Las clases están organizadas en forma jerárquica dando origen a un árbol cuya raíz es la clase Object.



Jerarquía de clases (Class Hierarchy Browser)

Clase Char.

Los caracteres son objetos que representan los símbolos que conforman un alfabeto.
Se los denota con un signo $: $a $c $D $+ $;

Entienden los siguientes mensajes:

unObjeto *isUpperCase*	(devuelve V o F si está en mayúscula o no)
unObjeto *isLowerCase*	(devuelve V o F si está en minúscula o no)
unObjeto *asUpperCase*	(lo pasa a mayúscula)
unObjeto *asLowerCase*	(lo pasa a minúscula)
unObjeto *isAlphabetic*	(devuelve V si es letra del alfabeto, F en caso contrario)
unObjeto *asciiValue*	(devuelve el ascii correspondiente)
unObjeto *isVowel*	(devuelve V o F si es o no una vocal minúscula o mayúscula)
unObjeto *isDigit*	(devuelve V o F si es o no $0 o $1 o ... $9)
unObjeto *isAlphanumeric*	(devuelve V o F si contiene o no letras y numeros)
unObjeto *isLetter*	(devuelve V o F si es $a, $b,...., $A, $B,.... o no lo es)
unObjeto *isSeparator*	(devuelve V o F si es o no: un espacio, tabulador, CR, salto)

Jerarquía de clases (Class Hierarchy Browser)

Un objeto de la clase **String** es una cadena de caracteres encerrados entre comillas simples. Es un objeto indexado y cada uno de sus componentes es un objeto de la clase Char.

Ej. ‘hola’ es igual a: $h,$o,$l,$a donde la coma concatena.
Además de las operaciones de comparación admite las siguientes operaciones:

unObjeto *isUpperCase*
unObjeto *isLowerCase*
unObjeto *asLowerCase*
unObjeto *asUpperCase*
unObjeto *size* (devuelve la cantidad de caracteres corrientes)
unObjeto, unObjeto (concatena)

unObjeto *at:* unaPosic (me devuelve el carácter de la posición unaPosic)
unObjeto *at:* unaP *put :* unCar (accede al elemento de la posición unaP y lo modifica con unCar)

unObjeto *copyFrom:* aPos1 *to :* aPos2 (retorna la subcadena comprendida entre los índices indicados)
unObjeto *asInteger* (si el contenido de unObjeto es numérico hace la conversión)
unObjeto *asFloat* (ideo si es real)
unObjeto *asDate* (lo convierte a fecha si el contenido aparece como mes, día y año en ese orden y separados por blancos)

Jerarquía de clases (Class Hierarchy Browser)

La clase Number encapsula protocolo común para sus subclases: Integer, Float y Fraction.

Algunos de los mensajes que entiende son:

Números

+, -, *, /	*operaciones típicas.*
unNumero // unNumero	*divide y trunca hacia -∞*
unNumero \\ unNumero	*resto tras truncar por //.*
unNumero **negated**	*invierte el signo del numero.*
unNumero **abs**	*valor absoluto.*
unNumero **sqrt**	*raiz cuadrada.*
unNumero **raisedTo:** unNumero	*eleva el primer número a la potencia indicada por el segundo.*
unNumero **squared**	*eleva un número al cuadrado*

Jerarquía de clases (Class Hierarchy Browser)

Mensajes que entienden todos los objetos

unaClase *new* (crea una instancia vacía)
unObjeto *class* (me devuelve la clase del objeto receptor)
unaClase *superClass* (me devuelve la superclase de una clase)
unObjeto *isKindOff*: aClass (me devuelve V o F si unObjeto pertenece a la clase aClass o no)
unObjeto *isNil* (devuelve true o false si es nil o no)
unObjeto *notNil*
unObjeto *yourself* (devuelve el objeto receptor del mensaje)
unObjeto *inspect* (abre una ventana especial que me permite inspeccionar al objeto receptor para ver su estado y/o modificarlo)

Jerarquía de clases (Class Hierarchy Browser)

Mensajes para ingresar datos por pantalla.

variable:= Prompter **prompt**: ‘ titulo del mensaje de ingreso de datos’. //ingresa un string siempre
resp:=MessageBox **confirm:** 'desea continuar?’. //ingresa um booleano
x:= (Prompter **prompt**: ‘ingrese valor') **asNumber**. //convierte a número entero o real

Mensajes para imprimir en pantalla.

MessageBox **notify**: ‘El nombre y apellido del alumno es:’**,** nomyape. //la coma concatena

MessageBox **notify**: 'la distancia entre los puntos es:'**,** (x **displayString**). //displayString convierte a string

suma **inspect.** //abre una ventana de inspección y muestra el contenido del OR