



# Introducción a Smalltalk

Alicia Díaz

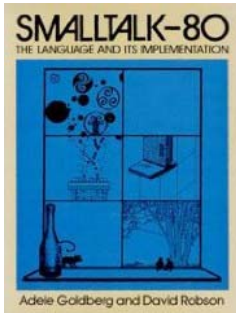
[alicia.diaz@lifa.info.unlp.edu.ar](mailto:alicia.diaz@lifa.info.unlp.edu.ar)

# ¿Qué es Smalltalk?

- Un lenguaje de programación OO puro
- Una librería de clases completa
- Un ambiente de programación interactivo (Pharo)
  - El ambiente en sí mismo está desarrollado en Smalltalk
  - Tiene su propio
    - Compilador
    - Debugger
    - Browser de la librería de clases
  - Es extensible



# Algo de historia



Smalltalk-80 fue desarrollado en *Xerox Palo Alto Research Center* entre fines de los 70s y principio de los 80s (Alan Kay, Dan Ingalls, Adele Goldberg)



En 1997 se presenta Squeak como un dialecto de Smalltalk derivado directamente de Smalltalk-80 (Dan Ingalls, Alan Kay).  
<http://www.squeak.org/>



En 2009 Pharo surge a partir de Squeak como una iniciativa open-source. Se centra en las técnicas de ingeniería de software y desarrollo modernos. (Pharo Board- Pharo Community)  
<http://pharo.org/>

# ¿porqué SmallTalk?

- Hay muchas razones que iremos aprendiendo durante el curso
- el lenguaje es **muy simples** y los conceptos subyacentes son simples y uniformes en todo el lenguaje
- No hay separación entre el lenguaje y el ambiente de programación.
  - El ambiente en sí es un universo viviente de objetos
- El código **fuentes es parte del ambiente**, está disponible para su estudio, extensión y modificación
- El ambiente de desarrollo es muy potente
- La **compilación es incremental**, favoreciendo el desarrollo y testing
- Es posible interrumpir la ejecución e **inspeccionar el estado** del programa e incluso modificar código y objetos
- Es **fuertemente tipado**, un objeto no puede responder a un mensaje que no se le programó
- Es **dinámicamente tipado**: cuando un programa tiene que evaluar su definición, este lo resuelve en run-time y no en la compilación
- es **portable**, permite que aplicaciones bien implementadas corran en plataformas distintas sin necesidad de hacer cambios



# Sintaxis Básica de Smalltalk

Objetos, Mensajes, Variables

# El objeto "robotech" y sus mensajes

The screenshot displays the Pharo IDE interface with the following components:

- Pharo Logo:** Located in the top left corner.
- Transcript:** A window on the left side, currently empty.
- an OnTheFlyConfigurableSimulation - Stop condition reached:** A central window showing a green grid with a black path. The path starts at the bottom left and moves right, then up, then right again. A small blue robot icon is at the end of the path. On the left side of this window, there are buttons: "Available Objects", "Browse Definition", "Open Workspace", and "Reset". Below these buttons, it says "Steps: 23".
- Playground (Top):** A window at the top right containing the following code:

```
"Expresión Aritmética"  
(15*19) + (37 squared)  
"Comparación 1"
```
- Playground (Bottom):** A window at the bottom right containing the following text and code:

```
"La forma de indicarle a un objeto que hacer, es mediante el envío de mensajes."  
  
" Para que el robot se mueva 10 celdas, le enviamos a robotech el mensaje #move: con  
parámetro 10"  
robotech brushDown.  
robotech move: 10.  
  
" Para que el robot se oriente hacia la derecha, le enviamos a robotech el mensaje  
#direction: con parámetro 90"  
robotech direction: 90.  
  
"Podemos enviarle a un objeto una serie de mensajes en cascada utilizando ;"  
robotech brushDown; direction: 0; move: 3; direction: 90; move: 3; direction: 180;  
move: 3; direction: 270; move: 3.
```

The bottom status bar shows the following tabs: "Playground", "an OnTheFlyConfigurableSimu...", "Playground", and "Transcript".

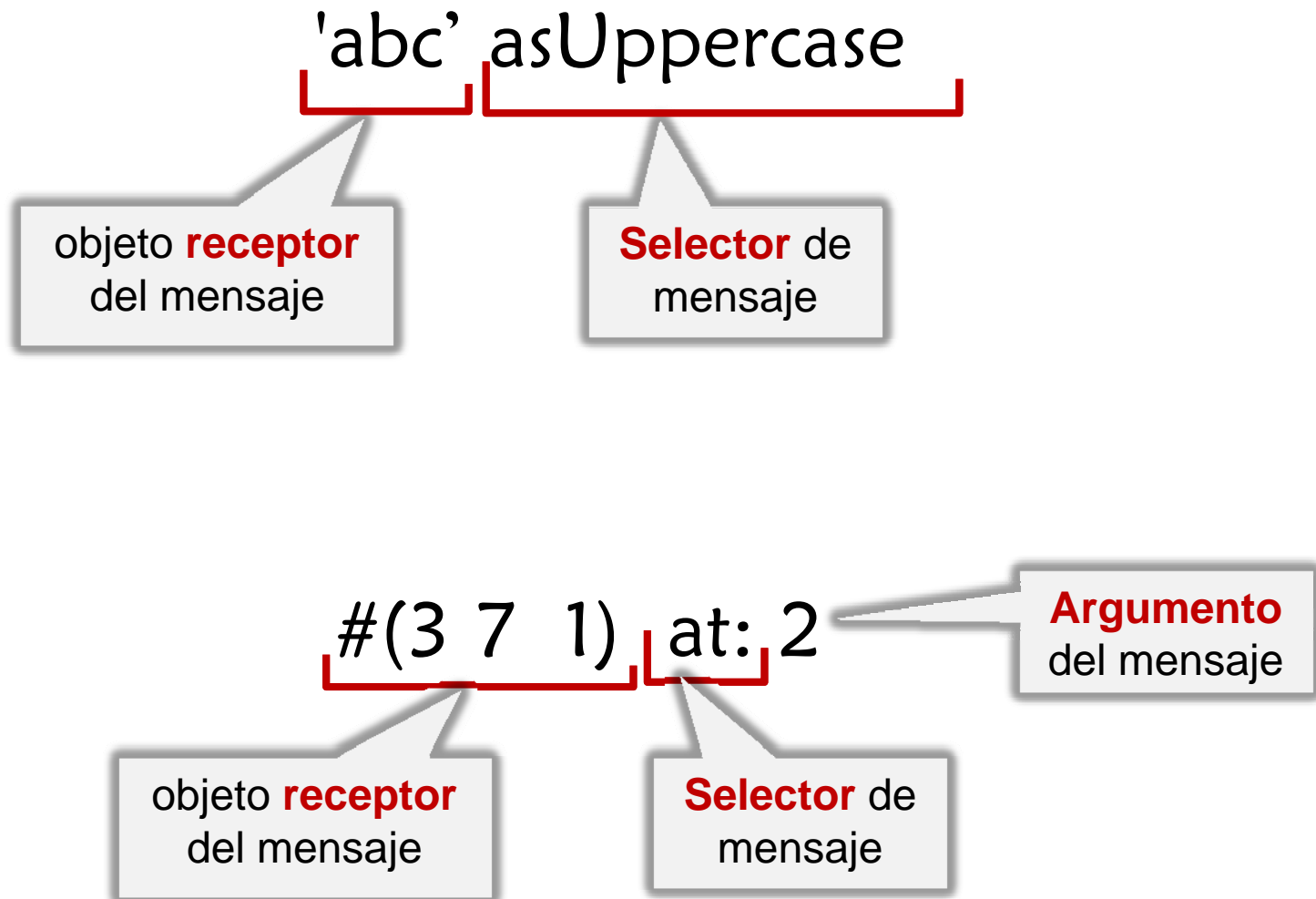
# Otros objetos

- Literales

- numbers: 3 -5 0.56 1.3e5 16rA
- characters: \$A \$1 \$\$
- strings: 'hello' 'A' 'haven't'
- symbols: #Fred #dog
- arrays:   
#(3 7 1)   
#(3 \$A 'hello' #Fred #(4 'world'))
- punto 3@5
- bloques: lo veremos más tarde

# Expresión Smalltalk

- Sintaxis básica





# Otros ejemplos de expresiones Smalltalk

"Expresión Aritmética"  
(15\*19) +(37 **squared**)

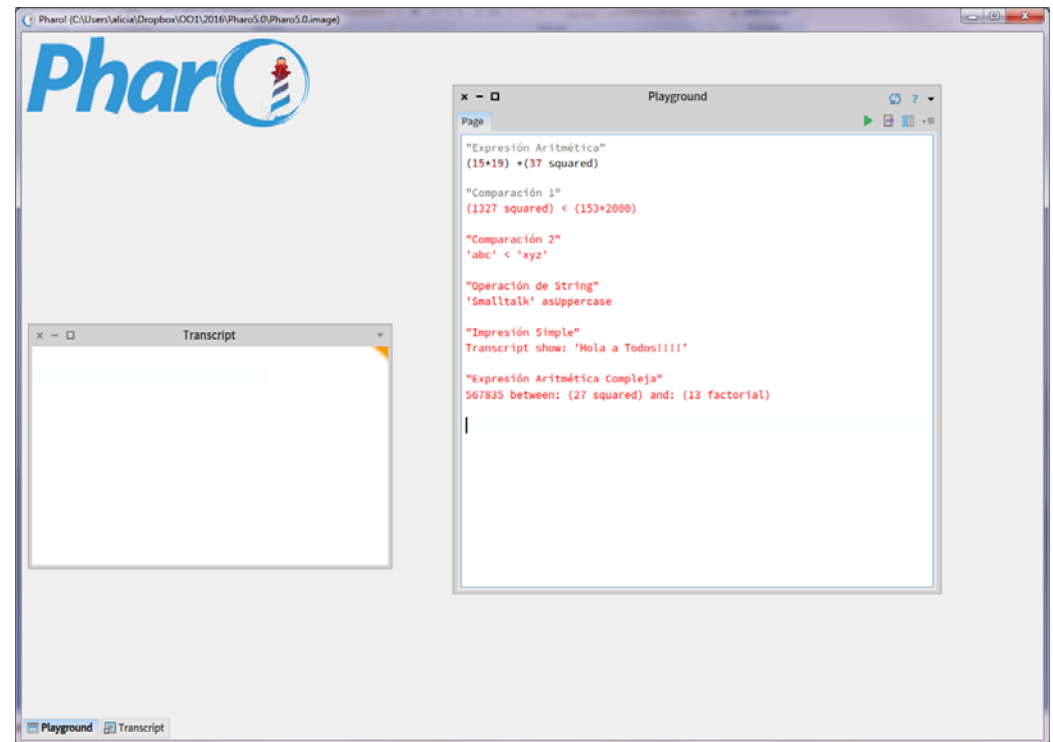
"Comparación 1"  
(1327 **squared**) < (153\*2000)

"Comparación 2"  
'abc' < 'xyz'

"Operación de String"  
'Smalltalk' **asUppercase**

"Impresión Simple"  
Transcript **show:** 'Hola a Todos!!!!'

"Expresión Aritmética Compleja"  
567835 **between:** (27 **squared**) **and:** (13 **factorial**)



## ST tiene 3 tipos de mensajes

- Unarios, Binarios y de Palabra Clave
- Difieren en:
  - La estructura del nombre del mensaje (su selector),  
y
  - La cantidad de argumentos que el mensaje espera

# Mensajes unarios

- No tienen argumentos

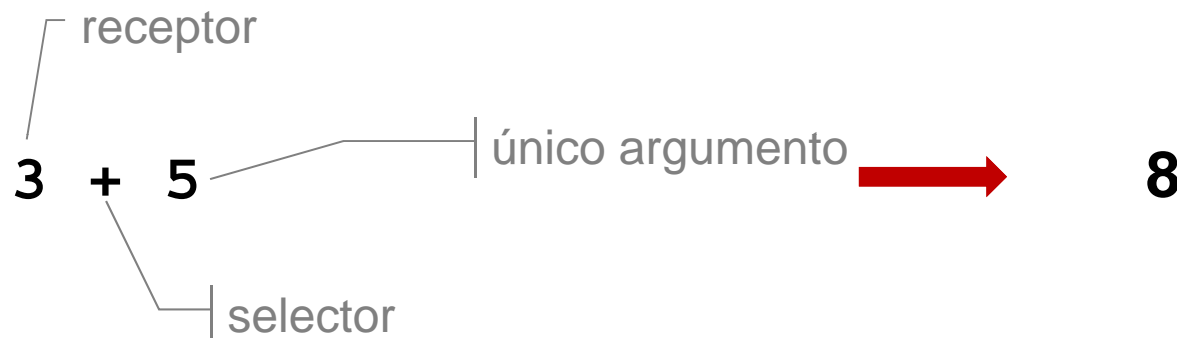
receptor  
3 negated  -3  
selector

'ABC' asLowercase  'abc'

#( 3 4 5 6 ) size  4

# Mensajes BÍNARIOS

- Usan uno o dos caracteres especiales como selector (+, \*, @, >=, =, “,” , ...) y tienen un solo argumento

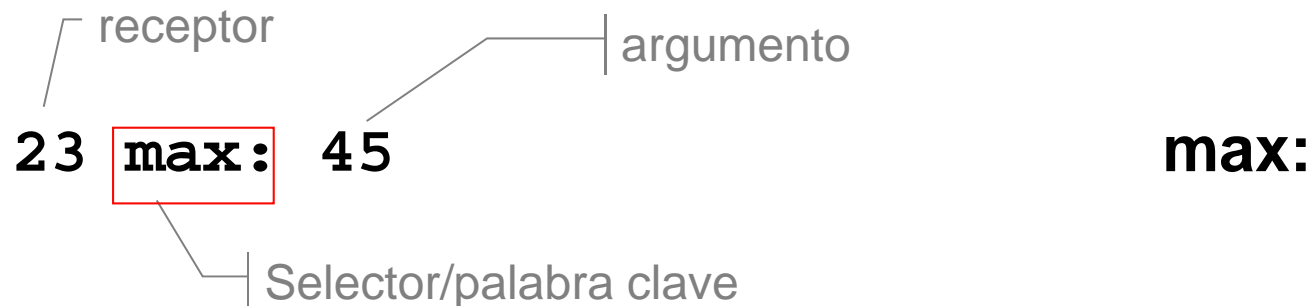


**'hi' , 'All'**  **'hiAll'**

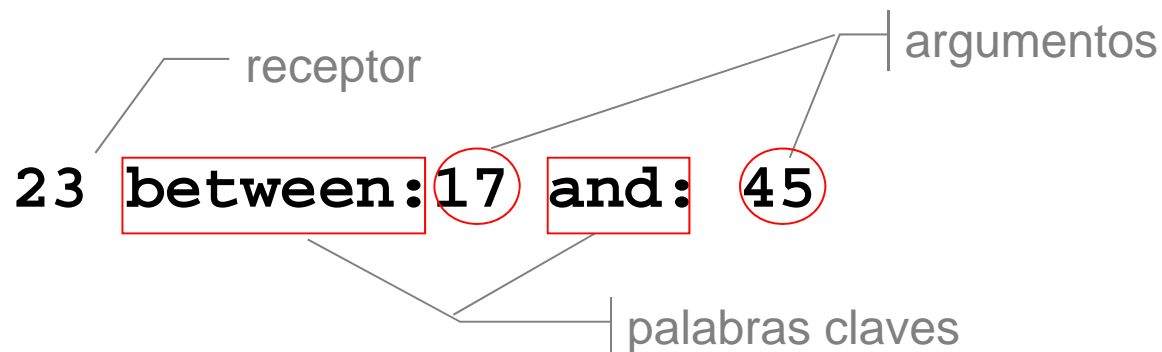
**23 <= 25**  **true**

# Mensajes de Palabra Clave

- Tienen uno o más argumentos
- Cada argumento es precedido por una palabra clave
- El selector tiene tantas palabras clave como argumentos
- Cada palabra es seguida por ‘ : ’




**max:**



**between: and:**

# Mensajes de Palabra Clave

- Otro ejemplo:

`#('blue' 'green' 'yellow') at: 2`  `'green'`


`#('blue' 'yellow' 'green') at: 3 put: 'red'`



`#('blue' 'yellow' 'red')`

# Cadenas Mensajes

- Cada mensaje devuelve un objeto  
.... entonces podemos escribir cadenas de mensajes

`'abc' asUppercase reverse`  
      
    `'ABC'`  
    `'CBA'`

*No siempre es tan simple.....*

`1 + 3 factorial`

*veamos que devuelve...*

`#(1 3 5) at:2 + 1`

*y acá que pasará? ..veamos...*

# Objetos y Mensajes

- Reglas de precedencia
  - los mensajes se ejecutan de izquierda a derecha
  - primero, las expresiones parentizadas
  - luego, los mensajes unarios,
  - luego, los binario
  - y por último los de palabra clave
- Única regla a tener en cuenta para poder leer y escribir programas Smalltalk

**5 factorial between: 3 squared and: 3 \* 5 + 9**

**(5 factorial) between: (3 squared) and: (3 \* 5 + 9)**

Los paréntesis permiten alterar la precedencia: **(1+3) factorial**



# Sentencias

- Cada expresión ST representa una sentencia
- Las sentencias se separan con ‘.’
- Una secuencia de expresiones representa un fragmento de código ST y sus componentes son sentencias

```
3 factorial.
```

```
2 squared
```

```
Transcript clear.
```

```
Transcript show: 'Hello!'
```

# Shortcut o mensajes en cascadas

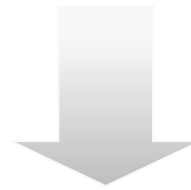
- Todos los mensajes van al mismo receptor y se separan con ' ; '

Transcript clear.

Transcript show: 'Hello!'.

Transcript cr.

Transcript show: 'How are you?'



Transcript clear;

show: 'Hello!';

cr;

show: 'How are you?'