



# Introducción a Smalltalk

Alicia Díaz

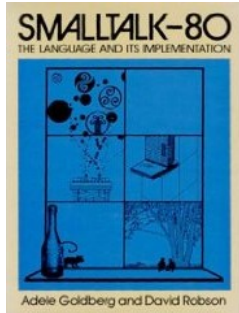
[alicia.diaz@lifa.info.unlp.edu.ar](mailto:alicia.diaz@lifa.info.unlp.edu.ar)

# ¿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





Smalltalks 2017 - November 8th, 9th & 10th, 2017

Facultad de Informática - Universidad Nacional de La Plata Ciudad de La Plata, Argentina.





# Sintaxis Básica de Smalltalk

Objetos, Mensajes, Variables

# El objeto "robotech" y sus mensajes

The screenshot displays the PharO software interface, which is used for creating and running simulations. The main window is titled "Pharo" and features a logo of a lighthouse. Below the logo, there are several panels and windows:

- Transcript:** A panel on the left side of the interface, currently empty.
- an OnTheFlyConfigurableSimulation - Stop condition reached:** A central window showing a grid-based simulation. The grid is green, and a black path is visible, representing the robot's movement. The path starts at the bottom left and moves upwards and then to the right. The window also displays "Steps: 23".
- Playground:** Two windows are open. The top one shows an arithmetic expression:  $(15 \times 19) + (37 \text{ squared})$ . The bottom one shows a series of messages sent to the "robotech" object, explaining how to move and orient the robot.

The messages in the bottom Playground window are:

- "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.`



# Otros objetos

- Literales

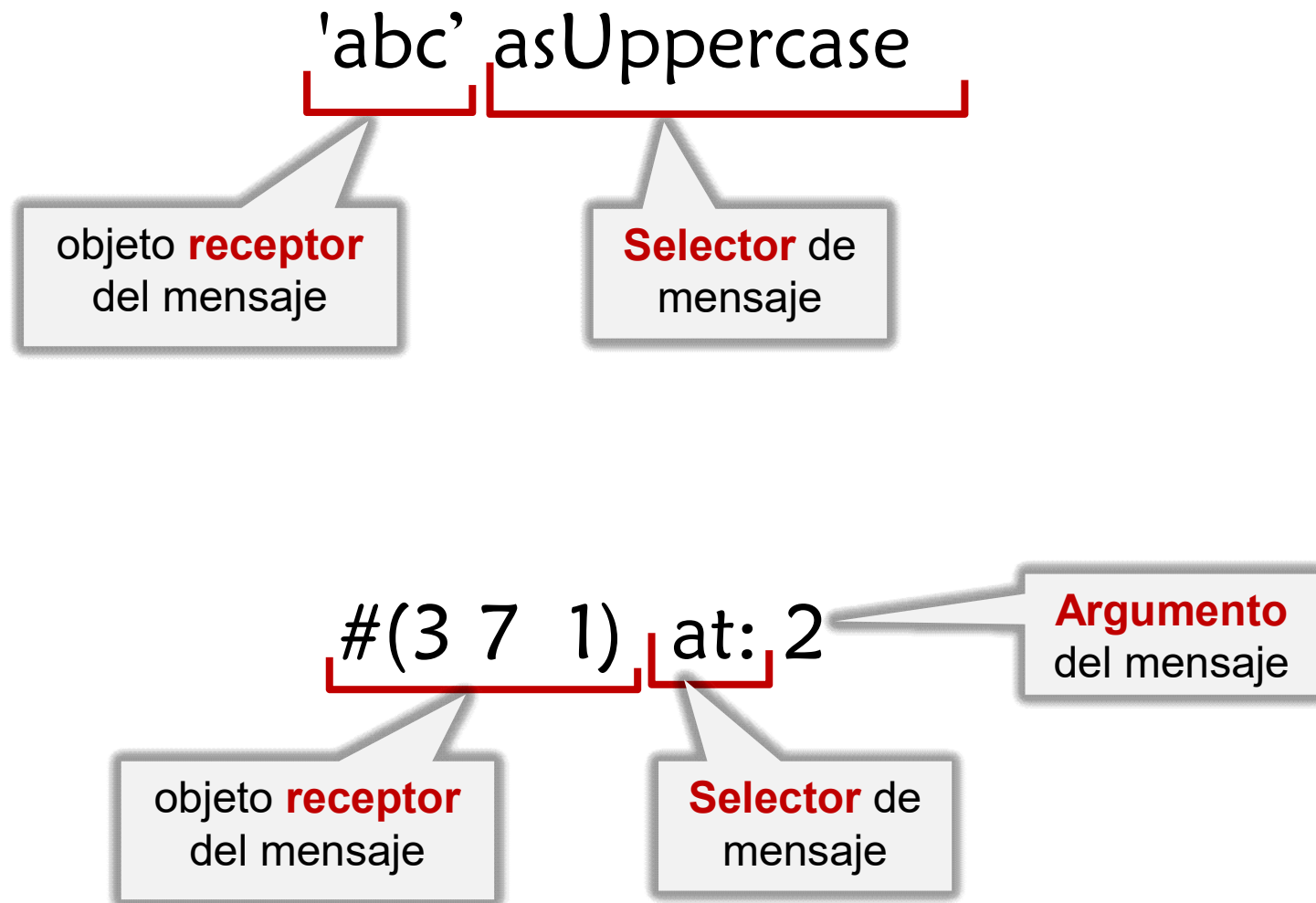
- numbers: 3 -5 0.56 1.3e5 16rA
- characters: \$A \$1 \$\$
- strings: 'hello' 'A' 'haven't'
- symbols: #Fred #dog
- arrays:   
#(3 7 1)  
#(3 \$A 'hello' #Fred #(4 'world' ) )
- punto 3@5
- bloques: lo veremos más tarde





# Expresión Smalltalk

- Sintaxis básica



# Otros ejemplos de expresiones Smalltalk

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

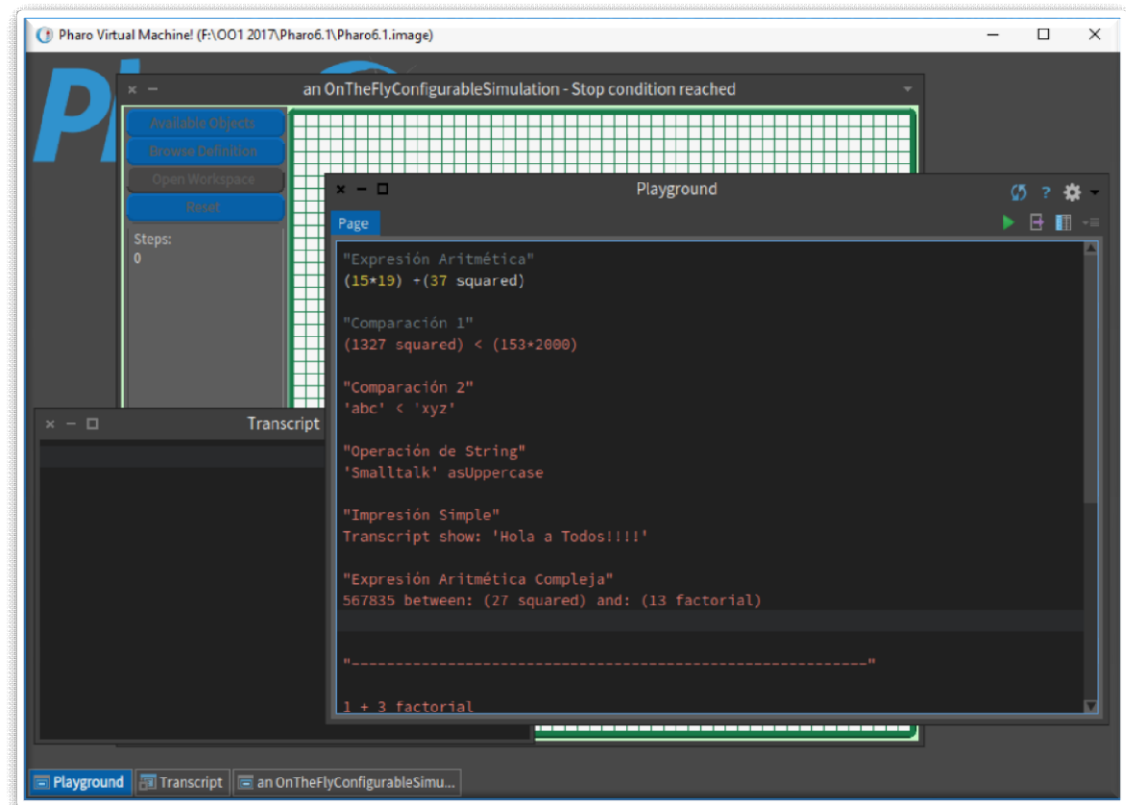
"Comparación 1"  
(1327 **squared**) **<** (153\*2000)

"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 Binaríos

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



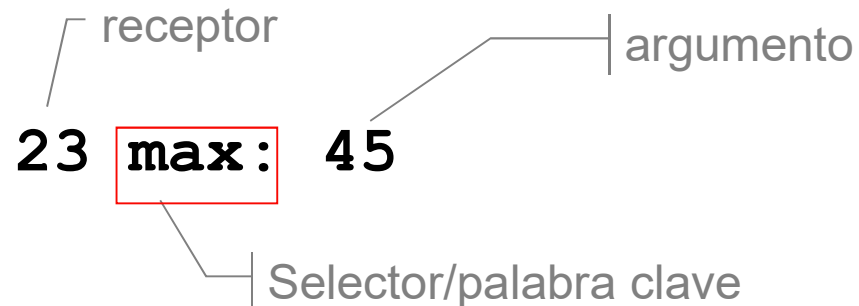
**'hi' , 'All'**  **'hiAll'**

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

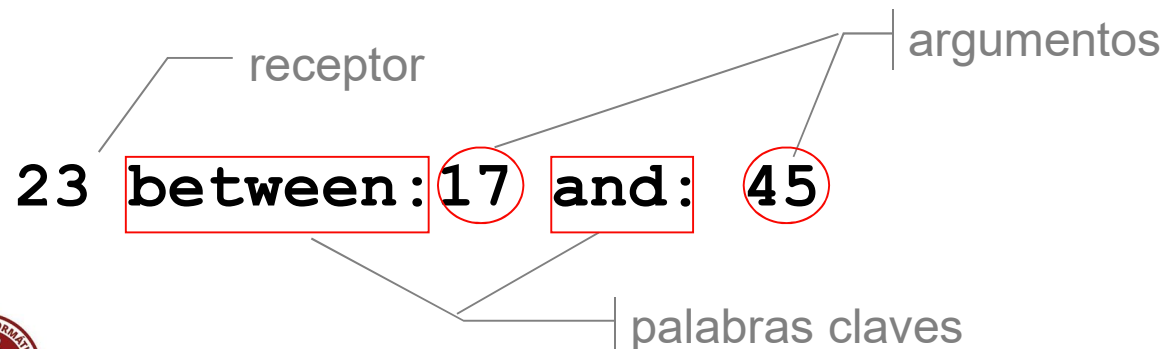


# Mensajes de Palabra Clave

- 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

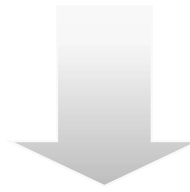
- Todos los mensajes van al mismo receptor y se separan con ' ; '

Transcript clear.

Transcript show: 'Hello!'.

Transcript cr.

Transcript show: 'How are you?'



Transcript clear;

show: 'Hello!';

cr;

show: 'How are you?'

