

Paradigma Orientado a Objetos

Caso de estudio: Squeak Smalltalk

Lenguajes de Programación 2018

Paradigma Orientado a Objetos

Abstracción de datos

- Reusabilidad y mantenimiento
- Extensibilidad

Polimorfismo

- Flexibilidad

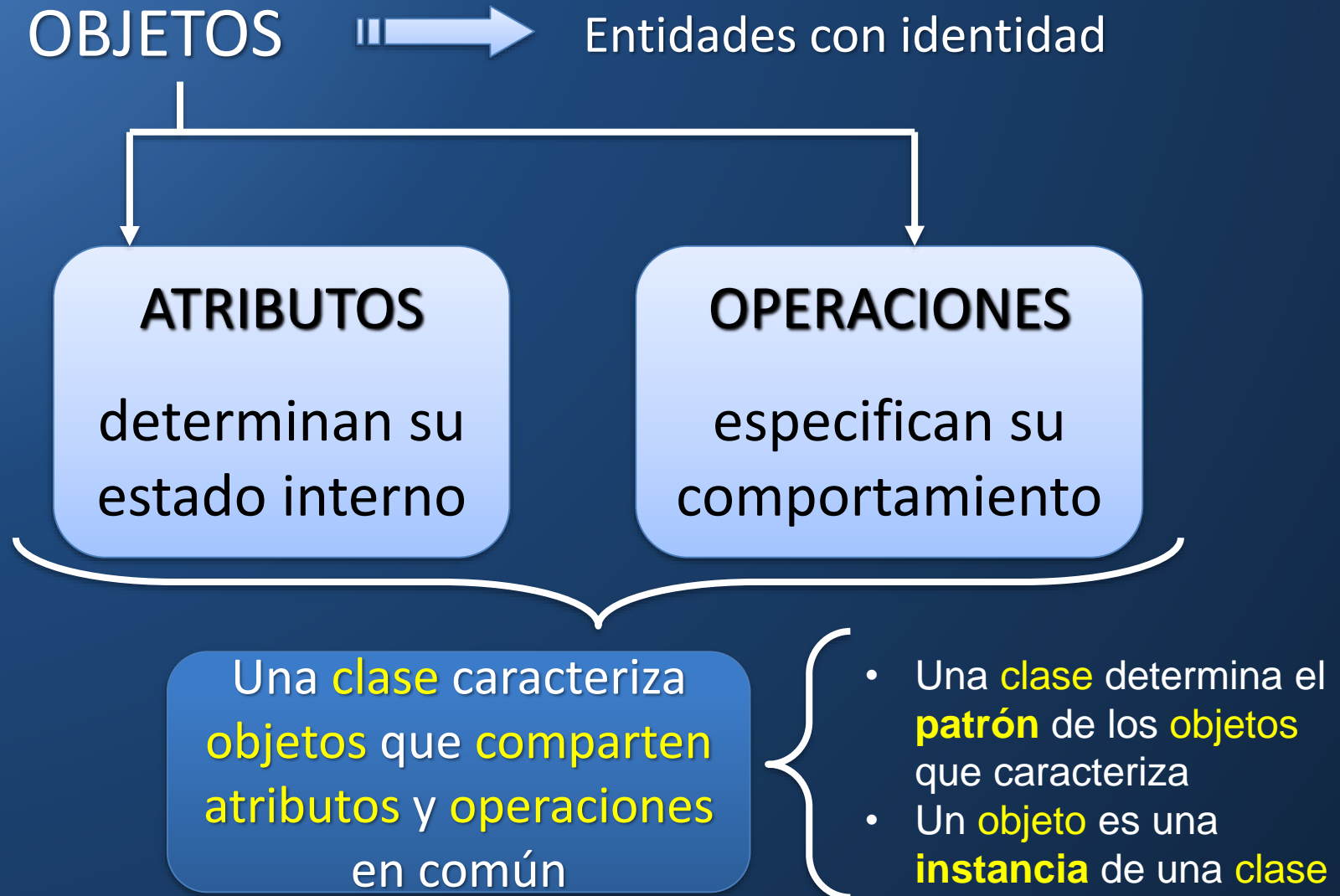
Nuevas
abstracciones como
extensiones o
refinamientos de
abstracciones
existentes

- Técnicas de **clasificación** para capturar y factorizar **aspectos comunes** de las entidades

En este paradigma
promueve una disciplina
donde es **clave identificar**

OBJETOS

Paradigma Orientado a Objetos



Paradigma Orientado a Objetos

- Comunicación entre los OBJETOS

- envían y reciben **mensajes**

Los manejan de
acuerdo a las
habilidades nativas
de su clase

También hacen uso de las
habilidades de las clases
antecesoras a su clase

Paradigma Orientado a Objetos

En un sistema Orientado a Objetos

Los **objetos** procesan los mensajes escondiendo el conocimiento y la forma en que realizan ese procesamiento

Los **objetos** están clasificados de alguna manera.

Los **objetos** pueden enviar y recibir **mensajes** dinámicamente

Los **objetos** están caracterizados por atributos y tienen un comportamiento de acuerdo al cuál responden a los mensajes

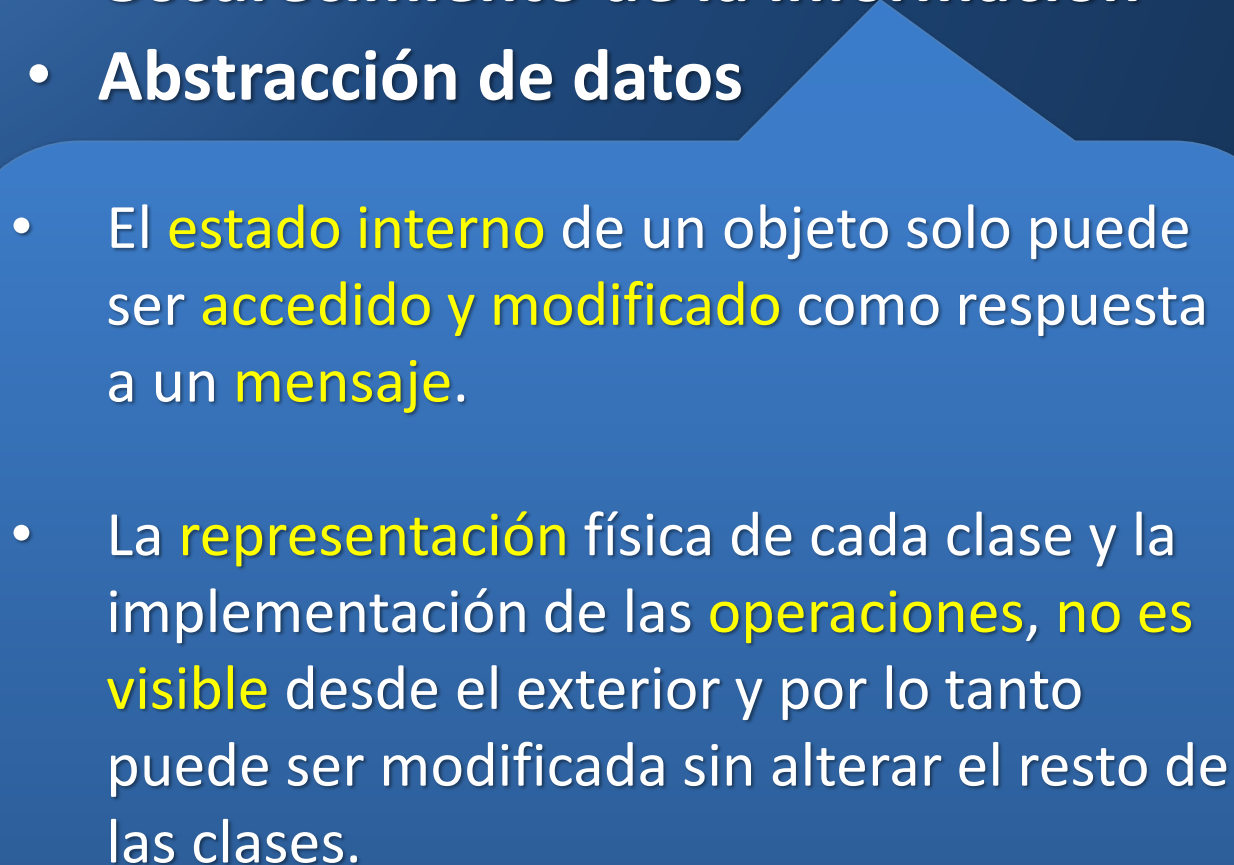
Lenguajes Orientado a Objetos

REQUERIMIENTOS MÍNIMOS

- Oscurecimiento de la información
- Abstracción de datos
- Clasificación (Herencia - Relación)
- Polimorfismo

Lenguajes Orientado a Objetos

REQUERIMIENTOS MÍNIMOS

- Oscurecimiento de la información
 - Abstracción de datos
- 
- El **estado interno** de un objeto solo puede ser **accedido y modificado** como respuesta a un **mensaje**.
 - La **representación** física de cada clase y la implementación de las **operaciones**, **no es visible** desde el exterior y por lo tanto puede ser modificada sin alterar el resto de las clases.

Lenguajes Orientado a Objetos

REQUERIMIENTOS MÍNIMOS

- Oscurecimiento de la información
- Abstracción de datos
- Clasificación (Herencia y Relación)

Una clase define un **patrón de comportamiento** a partir del cual es posible crear varias **instancias**

Lenguajes Orientado a Objetos

REQUERIMIENTOS MÍNIMOS

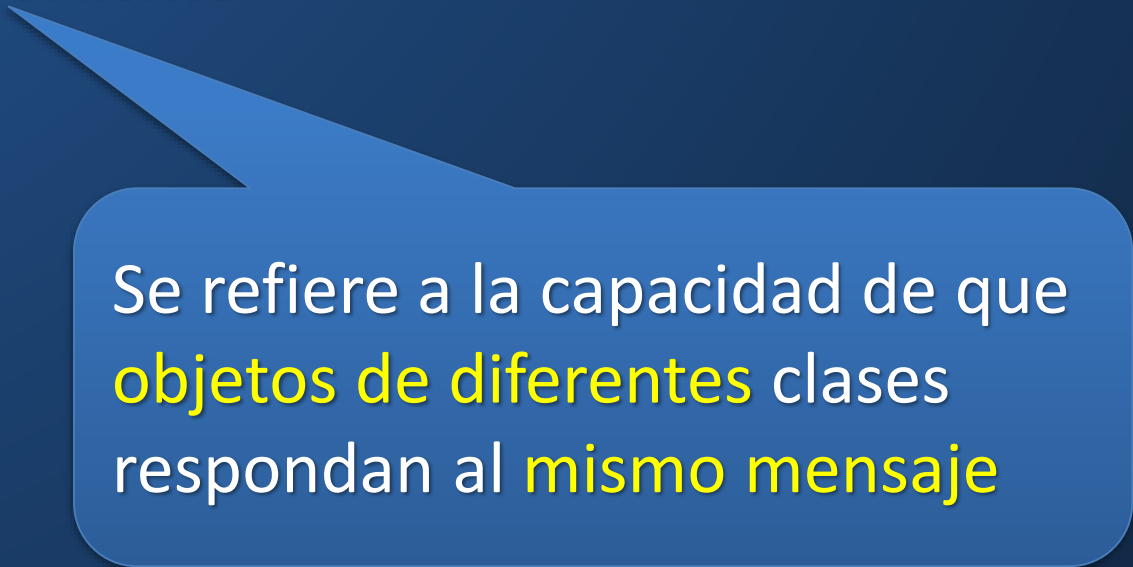
- Oscurecimiento de la información
- Abstracción de datos
- Clasificación (Herencia - Relación)
- Polimorfismo

Las propiedades de las clases pueden ser factorizadas para definir otras más **generales - generalización** - o a la inversa, que a partir de una clase es posible definir otras **más específicas - especialización**. La herencia reduce la redundancia en un sistema.

Lenguajes Orientado a Objetos

REQUERIMIENTOS MÍNIMOS

- Oscurecimiento de la información
- Abstracción de datos
- Clasificación (Herencia - Relación)
- Polimorfismo



Se refiere a la capacidad de que **objetos de diferentes** clases respondan al **mismo mensaje**

SQUEAK SMALLTALK



SQUEAK SMALLTALK

- **Paradigma puro:** Toda entidad es un objeto

Bloque son objetos, control se maneja mediante métodos (Iftrue, while, etc.)

- **No hay restricciones** para **ligar variables a objetos** ni para **redefinir métodos** (si se redefine con la misma signatura esconde al anterior) o definir nuevos.
- **Ligadura dinámica** del **mensaje** con el método.
- Las **variables** tienen **tipado dinámico**.
- **Herencia simple**

SQUEAK SMALLTALK

Identificadores

- Nombre de clase
- Variables de clase
- Variables globales
- variables de instancia
- variables temporales
- mensajes

Todos los identificadores de las **variables** cuyo alcance es **mayor al de un método u objeto**, empiezan con **mayúsculas**; y los de las variables cuyo **alcance es el método** (variable local o temporal) o el objeto (variable de instancia), comienzan con **minúsculas**

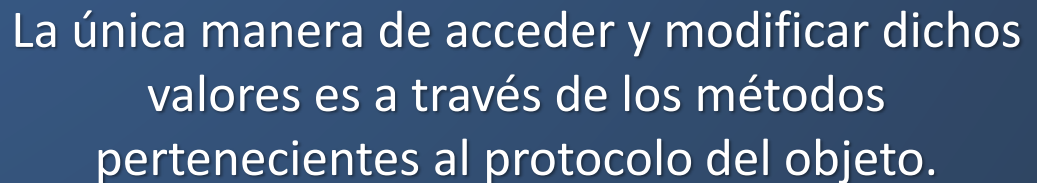
SQUEAK SMALLTALK

Variables de clase

Guardan atributos comunes a todos los objetos que son instancias de una misma clase

Variables de instancia

Tienen valores asociados únicamente con cada instancia u objeto creado a partir de una clase

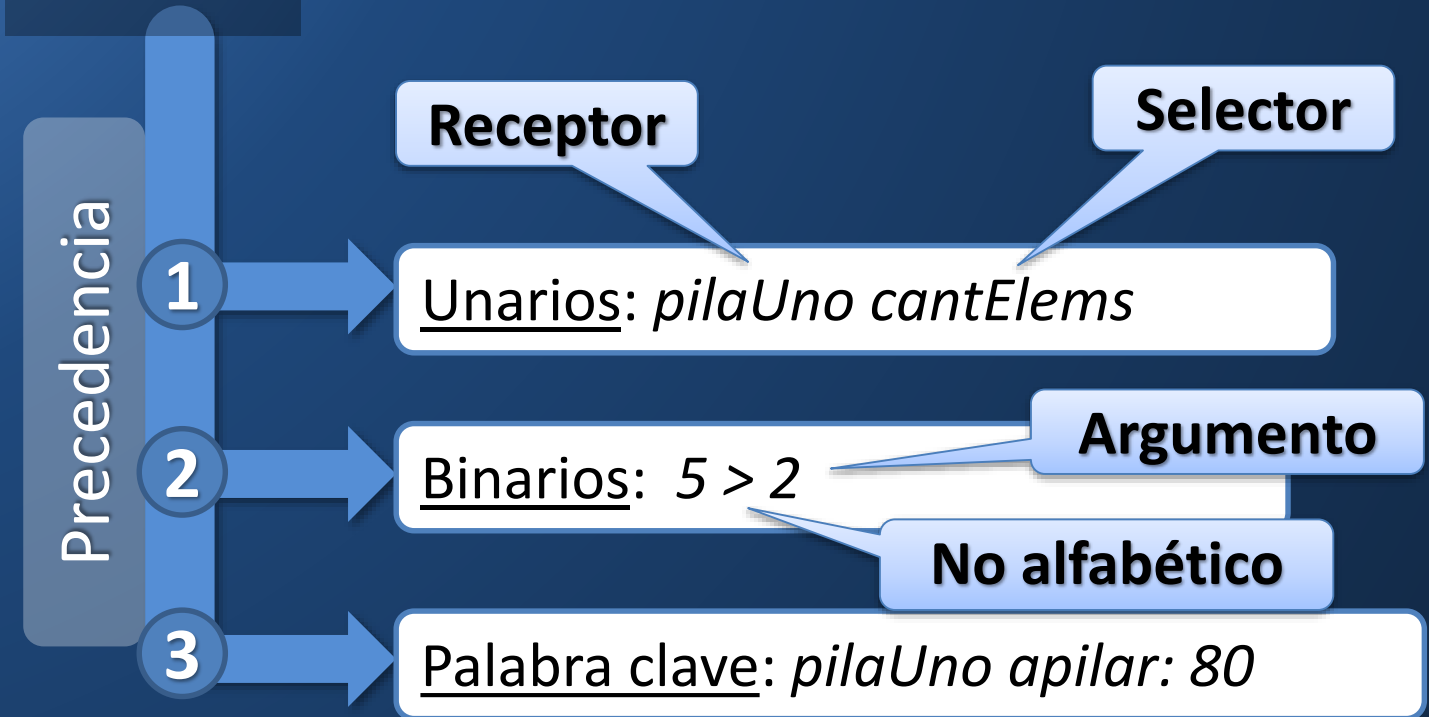


La única manera de acceder y modificar dichos valores es a través de los métodos pertenecientes al protocolo del objeto.

SQUEAK SMALLTALK

Computación

- Resultado de enviar mensajes a los objetos

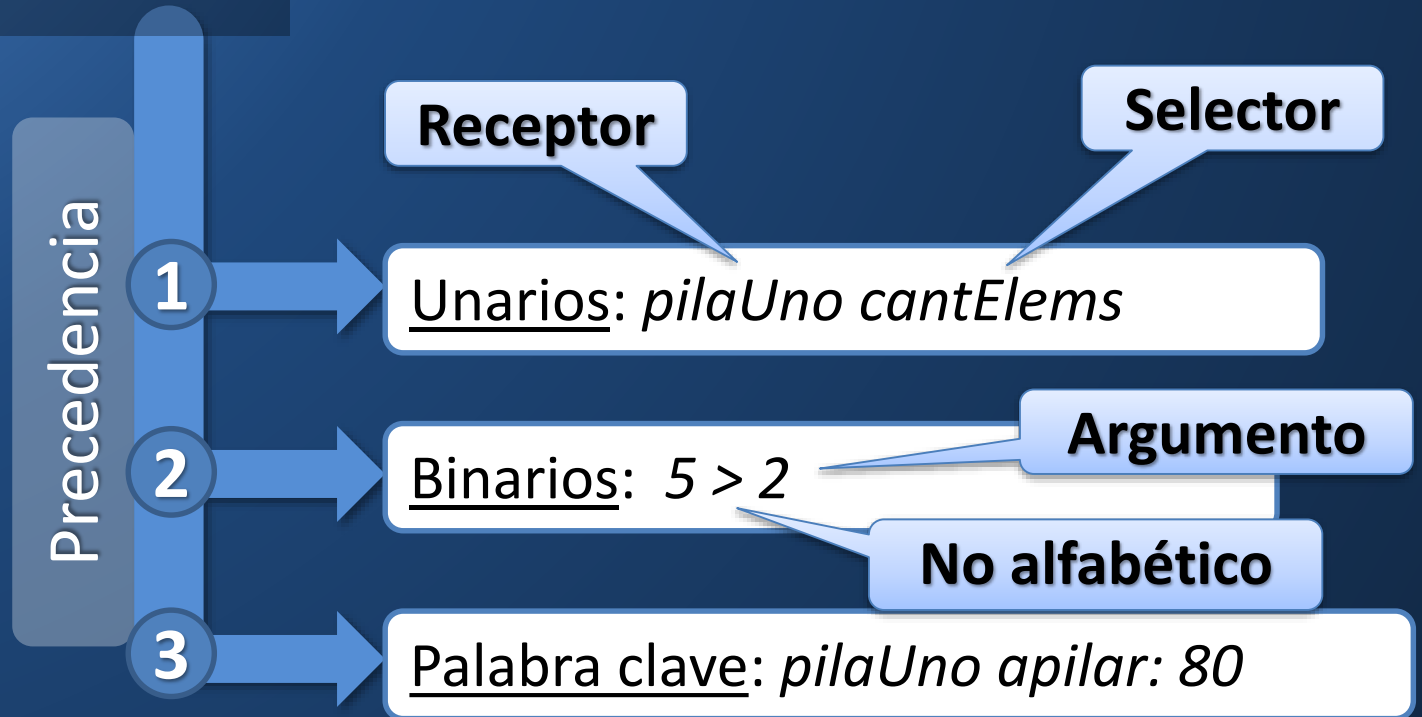
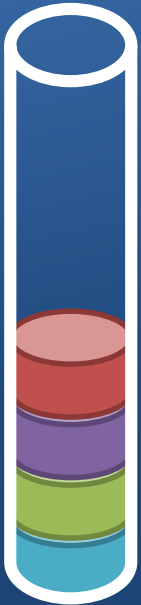


SQUEAK SMALLTALK

Computación

- Resultado de enviar mensajes a los objetos

pilaUno



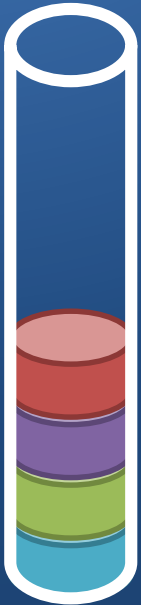
2 < pilaUno cantElems and: True

SQUEAK SMALLTALK

Computación

- Resultado de enviar mensajes a los objetos

pilaUno



Precedencia

1

Receptor

Unarios: *pilaUno cantElems*

Selector

2

Binarios: *5 > 2*

Argumento

No alfabético

3

Palabra clave: *pilaUno apilar: 80*

2 <

4

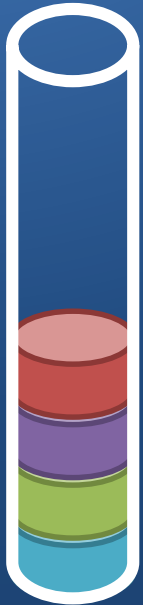
and: True

SQUEAK SMALLTALK

Computación

- Resultado de enviar mensajes a los objetos

pilaUno



Precedencia

1

Receptor

Selector

Unarios: *pilaUno cantElems*

2

Argumento

Binarios: *5 > 2*

No alfabético

3

Palabra clave: *pilaUno apilar: 80*

True

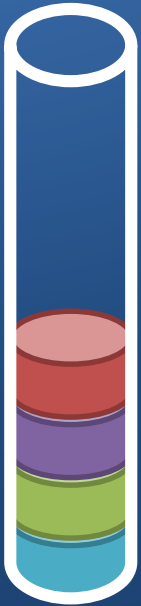
and: True

SQUEAK SMALLTALK

Computación

- Resultado de enviar mensajes a los objetos

pilaUno



Precedencia

1

Receptor

Selector

Unarios: *pilaUno cantElems*

2

Argumento

Binarios: $5 > 2$

No alfabético

3

Palabra clave: *pilaUno apilar: 80*

True

SQUEAK SMALLTALK

- Un objeto se puede enviar **un mensaje a sí mismo** mediante el uso de la pseudo variable **self**.

apilar: unElem

(self estaLlena)

ifTrue: [self error: 'pila llena']

ifFalse: [tope := tope + 1.

elementos at: tope put: unElem]



pilaUno *apilar: 80*

self hace referencia al objeto receptor del mensaje

SQUEAK SMALLTALK

• Métodos

Los **objetos responden** a los **mensajes** mediante la ejecución de métodos.

- El **método de instancia** en la clase de la cual un objeto es instancia, **sólo se interesa** por el **estado interno** de dicho **objeto**.
- Un método **no puede afectar** directamente las **variables** de instancia de **otro objeto**.

Ligadura entre la **invocación** (envío del mensaje) con la **implementación de la operación** (método) en tiempo de **ejecución**.

El retorno de un metodo se especifica mediante el simbolo: ^

SQUEAK SMALLTALK

Control

Clase
Boolean

```
(<expr. de prueba>) ifTrue: <Block> ifFalse: <Block>
```

apilar: unElem

```
(self estaLlena)
```

```
    ifTrue: [self error: 'pila llena']
```

```
    ifFalse: [tope := tope + 1.
```

```
                elementos at: tope put: unElem]
```

SQUEAK SMALLTALK

Control

Clase
Boolean

(<expr. de prueba>) ifTrue: <Block> ifFalse: <Block>

Clase
BlockContext

- [<sec. de expr.> whileTrue: <Block>
- [<sec. de expr.> whileFalse: <Block>

Clase Integer

- x timesRepeat: <Block>

do: select : collect : reject :

Clase
Collection

ITERADORES

SQUEAK SMALLTALK

Control

Pasaje de mensajes + clase *BlockContext*

El Objeto *Bloque*

- Permite encapsular una secuencia de expresiones
- Puede tener parámetros
- Puede ser pasado como parámetro o asignado
- Se evalúa (*value*) en el contexto en que fue definido

```
| I J |  
[ I := 3. J := 8. ] value
```

```
b:=[:char | char isVowel].  
b value: $a.
```

```
apilar: unElem  
(self estaLlena)  
  ifTrue: [self error: 'pila llena']  
  ifFalse: [tope := tope + 1.  
            elementos at: tope put: unElem]
```


SQUEAK SMALLTALK

Expresiones

Los NUMEROS son OBJETOS

La **evaluación** de las expresiones se realiza **de izquierda a derecha**

Todos los **operadores binarios** aritméticos y lógicos tienen la **misma precedencia** y su argumento se evalúa con “**evaluación ansiosa**”

Objeto:	Instancia de la clase:
3, 10, 12403	Integer
3.25, 500.5	Float
2/10, 58/2	Fraction

SQUEAK SMALLTALK

Expresiones lógicas

Clases True y False

&, |, and:, or:, not

and: y or: son mensajes de palabra clave:

evaluación no estricta

```
|a b|  
a:=true.  
b:=false.  
a | [b:=true].  
b
```

true

```
|a b|  
a:=true.  
b:=false.  
a or: [b:=true].  
b
```

false

Clase True:

and: bloque
 ^bloque value
& objeto
 ^objeto
not
 ^false

or: bloque
 ^self
| objeto
 ^self

SQUEAK SMALLTALK

Control, recursividad

factorial

self > 1

ifTrue: [^(self - 1) **factorial** * self].

self < 0

ifTrue: [^self error: 'negative factorial'].

^1

En que clase deberíamos
definir este método?

SQUEAK SMALLTALK

HERENCIA

Simple

Cada clase tiene solo una única **superclase** (excepto la clase Objeto, que no tiene ninguna) y una clase puede tener cualquier cantidad de **subclases**.

De
Implementación

Los métodos de una clase pueden acceder, además de a sus propias variables, a todas las variables de todas sus superclases; y pueden invocar a todos los métodos definidos en todas sus superclases.

Cuando se le envía un mensaje a un objeto, se ejecutará el método que este definido en la clase ubicada en el menor nivel en la jerarquía a partir de la clase de la cual el objeto receptor del mensaje es instancia.

Cuando se define un método que ya esta definido con igual signatura en alguna clase superior en la jerarquía de clases se dice que el método esta redefinido.