SmallTalk

Victor Vallejo Rives

2017/18 Q2

Introducció

- Projecte del *Xeroc Parc* als anys 70.
  - Alan Kay
- Insipirat per *Simula* (Llenguatge de simulacions de programació).
- **SmallTalk** va ser guardonat amb la medalla "llenguatge de programació més estimat" en les enquestes de Stack Overflow al 2017.

Paradigma de programació

- **SmallTalk**: Orientat a Objectes pur, Reflexiu.
- **c++**
  - *typedef*
  - *struct*
  - *...*
- **java**
  - *int*
  - *boolean*

Altres Característiques

Compilat o Interpretat

Interpretat: es converteix a llenguatge màquina a mesura que s'executa el programa.
Normalment a *bytecode* i després interpretat per una màquina virtual o traduït dinàmicament en codi màquina natiu.

Sistema de tipus

Dinàmicament tipat: la comprovació de tipació es duu a terme en temps d'execució.

Fortament tipat: no es permeten violacions dels tipus de dades.

Disseny

- Dissenyat pensant en **OO**.
- Tots els programes es basen en l'enviament de ***missatges*** entre Objectes.

Orientació a Objectes a SmallTalk

- Un Objecte és una instància d'una classe (també és un objecte), que defineix l'estructura i comportament de les seves instàncies.

MetaClasse

Classe que crea classes com a instàncies, i també defineix l'estructura i comportament d'aquestes.(*Single inheritance*)

Orientació a Objectes a SmallTalk

- Un Objecte a SmallTalk pot fer exactament 3 coses:
  - Mantenir l'estat.
  - Rebre un missatge de si mateix o d'un altre Objecte.
  - En el transcurs del processament d'un missatge, enviar un missatge a si mateix o a un altre Objecte.

Els missatges entre Objectes a SmallTalk

Què és?

És una sol.licitud d'un objecte a un altre, per dur a terme una de les seves operacions. Única manera d'invocar l'operació d'un objecte.

Quina informació conté?

Quina operació es vol executar, però no cóm. Això ho decideix l'objecte receptor.
Pot portar un objecte com a paràmetre, per exemple l'operació d'afegir.

Interface

Set de missatges als quals pot respondre un Objecte.
Única manera d'interactuar amb ell.

Els missatges entre Objectes a SmallTalk

I si l'Objecte que rep el missatge No té un mètode per a ell?

Quan un Objecte rep un *missatge*, busca dins la seva *interface*, el mètode que el missatge demana. Si no el troba accedeix a la seva **Super Classe** i el busca allà. Va pujant dinàmicament per l'arbre mentre no el trobi, fins a arribar a l'arrel. Si aquí tampoc està el mètode desitjat, el programa donarà **ERROR**.

Sintaxis

La **Sintaxis** de SmallTalk és molt simple, només té 6 paraules clau reservades (pseudovariables):

- **true**
- **false**
- **nil** (*UndefinedObject*)
- **self** (*receptor actual del mètode*)
- **super** (*referència al pare de l'objecte al qual ens trobem*)
- **thisContext** (*referència a la pila en temps d'execució*)

(true, false i nil, són instàncies singleton)

Representacions a SmallTalk

Literals

```
10
-10
3.141592
2r0101
16r000A
```

Char

```
$C
```

String

```
'Hola, que tal?'
```

String dins un String

```
'Em va dir, ''Hola, que tal'' a mi.'
```

Representacions a SmallTalk

Symbol (a difèrencia dels Strings, no hi poden haver dos símbols iguals que siguin objectes diferents)

```
#'Soc un symbol'
#hola
```

Arrays

```
#(1 2 3 4)
#[1 2 3 4] (ByteArray)
```

Blocks (funció anònima)

[. . . Codi Smalltalk. . .]

Variables

Declaració

```
| laMevaVariable |

| var1 var2 |
```

Assignació

```
laMevaVariable := 'Hola, que tal?'
var1 := -1
var2 := #(1 2 3)
```

Composició de Missatges

Simples

```
5 factorial

10 + 5

x := 5 factorial

y := 'abababebeabab' quantsCops: $C

Taulell width: 50 height: 50 (Taulell és una classe,
 i es crea una nova instància amb aquestes mesures)
```

Composició de Missatges

Expressions

Sempre va d'esquerra a dreta, no té prioritats

```
4 + 2 * 3

6 * 3

18
```

Li hem d'especificar nosaltres mitjançant parèntesis

```
4 + (2 * 3)

4 + 6

10
```

Composició de Missatges

Expressions

```
3 factorial + 4 factorial between: 10 and: 100

6 + 4 factorial between:10 and: 100

6 + 24 betweem:10 and: 100

30 between:10 and: 100

True
```

Composició de Missatges

Expressions

```
(3 factorial + 4) factorial between: 10 and: 100

(6 + 4) factorial between:10 and: 100

10 factorial between:10 and: 100

3628800 between:10 and: 100

False
```

Blocks de codi

```
[:x | x + 5] -> f(x) = x + 5, (\x: x+5)
[:x | x + 5] param: 5
```

Composició de Missatges

Declaracio de if i else mitjançant blocks

```
expr ifTrue: [statements to evaluate if expr]
    ifFalse: [statements to evaluate if not expr]

True methods for evaluation
ifTrue: trueAlternativeBlock
        ifFalse: falseAlternativeBlock
     ^trueAlternativeBlock value

False methods for evaluation
ifTrue: trueAlternativeBlock
ifFalse: falseAlternativeBlock
    ^falseAlternativeBlock value
```

Classes

Definició

```
Object subclass: #MessagePublisher
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Smalltalk Examples'
```

Missatge enviat a la classe "*Object*" que acciona el seu mètode **subclass** i crea la subclasse "*MessagePublisher*". Per tant, es crea en temps d'execució.

Instanciar

```
jugador1 := Jugador new
```

Mètodes

metodeExemple

```
metodeExemple: x1 and: x2
    | resultat |
    resultat^ := x1 * x2
```

El *metodeExemple* seria un mètode que pertany a un Objecte. Al rebre un missatge referint-se a ell amb dos paràmetres numèrics, retornaria el seu producte.

Hello World!

```
publish
    Transcript show: 'Hello World!'
```

Per qui va ser Influit

- Flex (Alan Kay, 1969)
- Lisp (Interpreter, Blocks, Garbage Collection)
- Turtle graphics (The Logo Project, Programming for Children)
- Direct Manipulation Interfaces (Sketchpad, Alan Sutherland, 1960)
- NLS, (Doug Engelbart, 1968), “the augmentation of human intellect”
- Simula (Classes and Message Sending)
- Xerox PARC (Palo Alto Research Center)
- DynaBook: a Laptop Computer for Children

A qui ha Influit

- AppleScript
- Common Lisp Object System
- Dylan
- Falcon
- Java
- NewtonScript
- Object REXX
- Objective-C
- PHP 5
- Perl 6
- Python
- Ruby
- Scala
- Scratch
- Self
- etc.