História da Programação Orientada a Objetos

Daniel Cordeiro

19 de fevereiro de 2016

Escola de Artes, Ciências e Humanidades | EACH | USP

“I invented the term Object-Oriented, and I can tell you I did not have C++ in mind.” – Alan Kay

Raízes do Conceito de Objetos

- Platão, 375 a.C., teoria das formas: uma forma ideal de uma cama é a base para todas as camas do mundo (A República).
- Aristóteles, 330 a.C., precursor da herança: “Se falamos dos animais, é claro que vamos falar as mesmas coisas sobre muitos deles”. Então, se duas classes tem características em comum, Aristóteles sugere que coloquemos estas características em uma entidade que hoje chamamos de superclasse.
- Carolus Linnaeus (ou Carl von Linné) em 1753 elaborou provavelmente a primeira grande hierarquia de classes (taxonomia de plantas) que se tornou um padrão internacional.

SIMULA

- Linguagem SIMULA criada na Noruega por Ole-Johan Dahl and Kristen Nygaard.
- Derivada do Algol.
- SIMULA I (1962-65) e Simula 67 (1967) foram as primeiras linguagens OO.
- Eram usadas para simulações do comportamento de partículas de gases.
- Os conceitos de objetos, classes e herança apareceram lá embora não necessariamente com estes nomes.
- Herança apareceu apenas em Simula 67 e se falava apenas em subclassing.
- Simula passou a ser usada com frequência da década de 70, sendo inclusive a linguagem utilizada para desenvolver as primeiras versões de Smalltalk.

SIMULA

```
Begin
   Class Glyph;
      Virtual: Procedure print Is Procedure print;
   Begin
   End;

   Glyph Class Char (c);
      Character c;
   Begin
      Procedure print;
        OutChar(c);
   End;

   Glyph Class Line (elements);
      Ref (Glyph) Array elements;
   Begin
      Procedure print;
      Begin
         Integer i;
         For i:= 1 Step 1 Until UpperBound (elements, 1) Do
            elements (i).print;
         OutImage;
      End;
   End;

   Ref (Glyph) rg;
   Ref (Glyph) Array rgs (1 : 4);

   ! Main program;
   rgs (1):- New Char ('A');   rgs (2):- New Char ('b');
   rgs (3):- New Char ('b');   rgs (4):- New Char ('a');
   rg:- New Line (rgs);        rg.print;
End;
```

Origens do Smalltalk

- XEROX PARC, 1970 a 1980: um lugar único na história da Ciência da Computação.
- Projeto Dynabook: um ambiente novo de computação para adultos e crianças com características altamente revolucionárias.
- Deste projeto saíram:
  - Interface Gráfica (com bitmaps)
  - Sistemas de Janelas
  - Mouse
  - Redes via Modem e via Ethernet
  - Placas de Som
  - etc., etc.

Dynabook

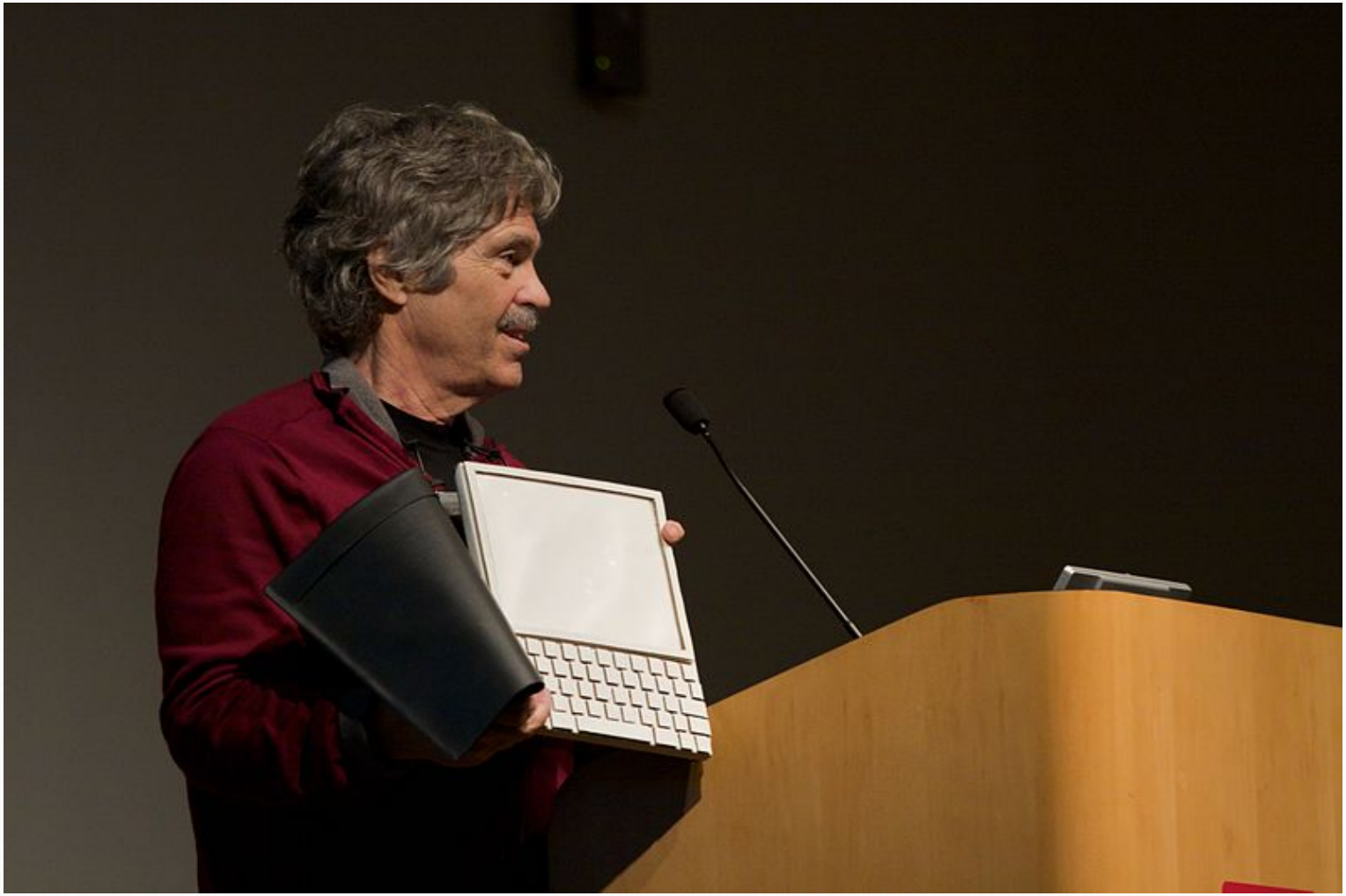


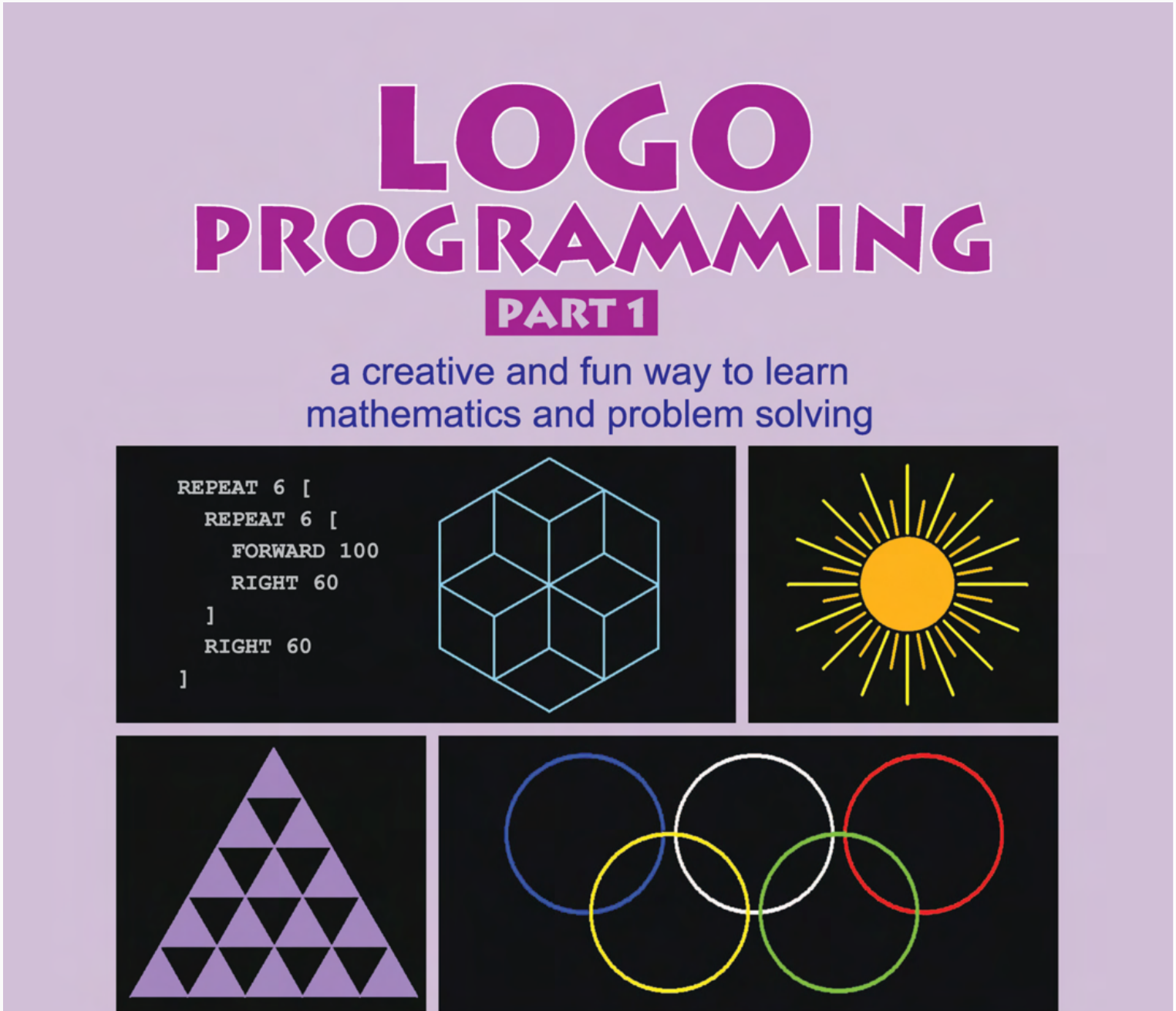
Figure 1: Alan Kay segurando uma maquete do Dynabook

Alan Kay

- Alan Kay, pesquisador de Utah, então propõe a criação de vários aplicativos novos para tal computador: editor de textos, programas para desenho, para modelagem de objetos tridimensionais, etc.
- Ele foi então convidado a entrar no projeto e criar uma nova linguagem de programação que fosse fácil de programar e que tivesse a sintaxe muito simples.
- Programar nesta linguagem deveria ser tão fácil quanto bater um papo furado (small talk) em inglês.

Alan Kay

- Kay se inspirou em:
  - LOGO: baseada em interface gráfica e destinada a crianças.
  - LISP: alto poder para manipulação de símbolos (as linguagens até então manipulavam apenas números), sintaxe super simples.
  - No final de 1972 a primeira versão de Smalltalk estava pronta. Ainda não possuía herança.
  - Em 1974, Smalltalk passou a incorporar interfaces gráficas e bitmaps. A partir daí foi criado o primeiro sistema para gerenciamento de janelas.
  - 1976: introduziu-se o conceito de herança, máquina virtual e bytecodes. A máquina virtual desvinculava o programa Smalltalk do hardware.
  - Paradoxo: O Dynabook nunca chegou a se tornar um produto comercial mas revolucionou o mundo da computação, ditando grande parte do que usamos hoje.
  - O projeto foi concluído mas a XEROX continuou investindo no Smalltalk, que foi amplamente divulgado, em fitas magnéticas, e era chamado de Smalltalk-80.


LOGO
PROGRAMMING
PART 1
a creative and fun way to learn
mathematics and problem solving
REPEAT 6 [
REPEAT 6 [
FORWARD 100
RIGHT 60
]
RIGHT 60
]

LISP

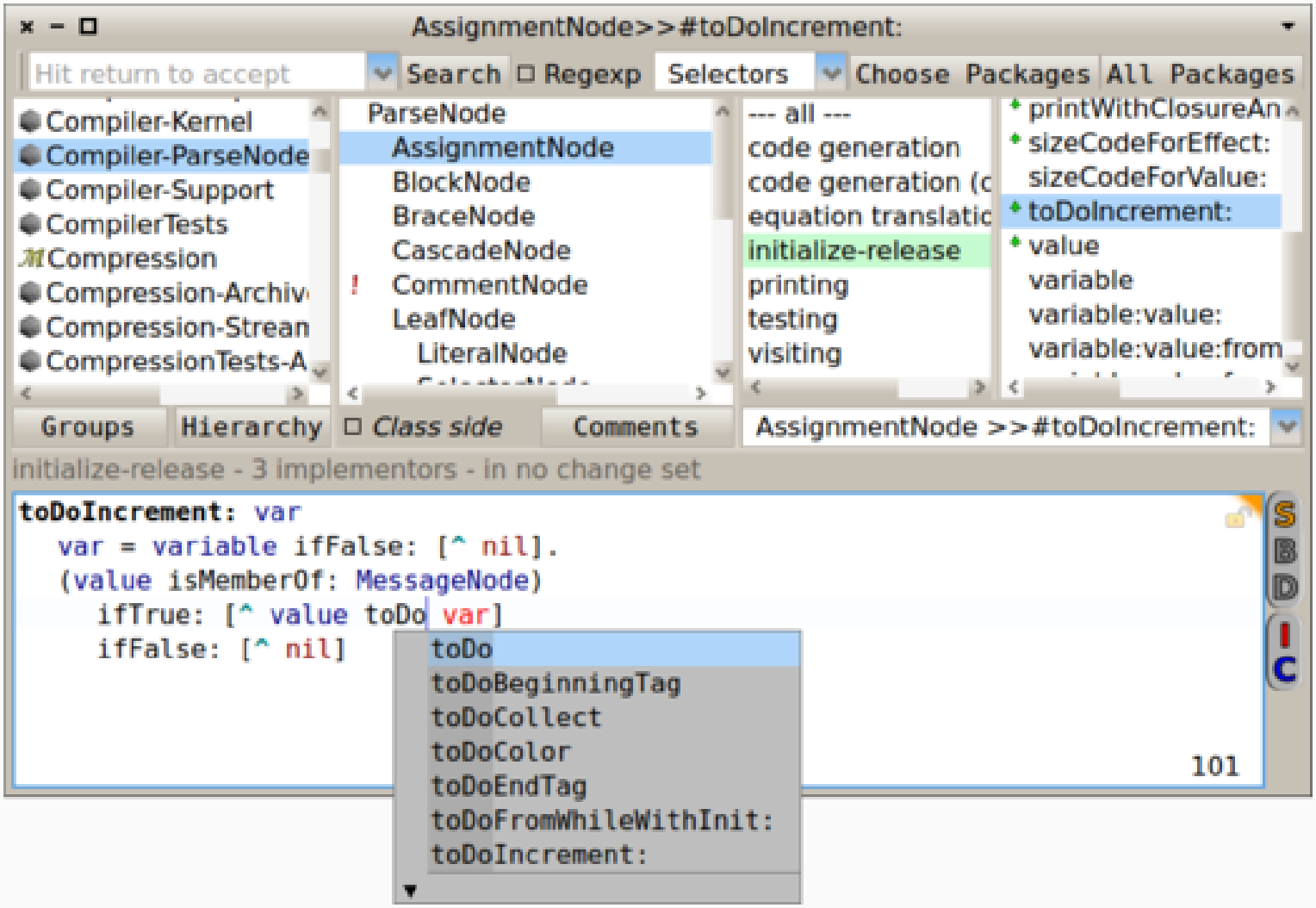
```
;-----------------------------------------------------------------------
; Author: Bob Dondero
; Insertion Sort in Common LISP
;-----------------------------------------------------------------------

; Return a sorted version of myList
(defun insertionSort (myList)
   (if (null mylist)
       '()
       (insertInPlace (car myList) (insertionSort (cdr myList)))))

;-----------------------------------------------------------------------
; Return the list that results from inserting e into sorted list
; myList at the correct place.
(defun insertInPlace (e myList)
   (if (null myList)
       (cons e '())
       (if (<= e (car myList))
   (cons e myList)
   (cons (car myList) (insertInPlace e (cdr myList))))))

;-----------------------------------------------------------------------
; Notes:
; (cons e myList)  evaluates to a list which is the same as myList
;                  but with e added to the front
; (car myList)     evaluates to the first element of myList
; (cdr myList)     evaluates to a list which is the same as myList
;                  but without the first element
```

Smalltalk

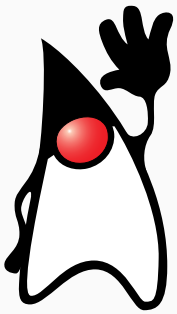

AssignmentNode>>#toDoIncrement:
Hit return to accept
Search
Regexp
Selectors
Choose Packages
All Packages
Compiler-Kernel
Compiler-ParseNode
Compiler-Support
CompilerTests
Compression
Compression-Archiv
Compression-Strean
CompressionTests-A
ParseNode
AssignmentNode
BlockNode
BraceNode
CascadeNode
CommentNode
LeafNode
LiteralNode
--- all ---
code generation
code generation (c
equation translatic
initialize-release
printing
testing
visiting
printWithClosureAn
sizeCodeForEffect:
sizeCodeForValue:
toDoIncrement:
value
variable
variable:value:
variable:value:from
Groups
Hierarchy
Class side
Comments
AssignmentNode >>#toDoIncrement:
initialize-release - 3 implementors - in no change set
toDoIncrement: var
var = variable ifFalse: [^ nil].
(value isMemberOf: MessageNode)
ifTrue: [^ value toDo var]
ifFalse: [^ nil]
toDo
toDoBeginningTag
toDoCollect
toDoColor
toDoEndTag
toDoFromWhileWithInit:
toDoIncrement:
101

Alan Kay

- Mais tarde a IBM criou a sua versão, Smalltalk V, que foi a primeira a rodar em DOS. Se tornou popular no final da década de 80. Hoje em dia se chama VisualAge for Smalltalk.
- Em 1987, a XEROX criou a empresa Parc Place para cuidar do Smalltalk. No final dos anos 90, a Parc Place foi comprada pela Cincom que hoje comercializa o Smalltalk que usamos nesta disciplina durante muitos anos.
- Em 1996, surgiu o projeto Squeak cujo objetivo é desenvolver uma versão de software livre para Smalltalk no ambiente Linux.
- O Squeak foi desenvolvido por um pequeno grupo de desenvolvedores, incluindo o Alan Kay, o pai da linguagem.
- Posteriormente o Squeak foi portado para outros SOs e o sistema que usamos agora nesta disciplina.

Java

- Dezembro de 1990: início do Green project por Patrick Naughton, James Gosling e Mike Sheridan da Sun Microsystems.
- Objetivo do projeto: substituir C++ por uma linguagem melhor, mais rápida e responsiva.
- Assim que o projeto ficou mais maduro, o objetivo mudou: criar uma linguagem para outros tipos de dispositivos além dos microcomputadores (no futuro ficaria mais claro que a linguagem seria desenvolvida pensando em sistemas embarcados)
- Gosling começou a melhorar C++, criando a linguagem `C++ ++ --` que depois passou a se chamar `Oak` e, finalmente, Java.



Princípios de Java

5 princípios nortearam a criação de Java:

1. ela deve ser “simples, orientada a objeto e familiar”
2. ela deve ser “robusta e segura”
3. ela deve ser “independente de arquitetura e portátil”
4. ela deve executar com alto desempenho
5. ela deve ser “interpretada, threaded e dinâmica”

Evolução das linguagens

C++

- No final dos anos 80, Bjarne Stroustrup estendeu a linguagem C com alguns conceitos de objetos e criou C++.
- Em termos de orientação a objetos, pode-se dizer que C++ está no extremo oposto de Smalltalk.

```
#include<iostream>
using namespace std;

class programming
{
private: int variable;
public:  void input_value() {
  cout << "In function input_value, Enter an integer\n";
  cin >> variable; }
void output_value() {
  cout << "Variable entered is ";
  cout << variable << "\n";
  }
};

main() {
   programming object;
   object.input_value();  object.output_value();
   return 0;
}
```

Objective C

Objective-C, por outro lado, é uma linguagem bem mais dinâmica realmente OO, influenciada por Smalltalk. Sua aceitação é pequena durante sua 1ª década de existência, mas no início da década de 2000, ela é adotada pela Apple para o seu ambiente gráfico Aqua e depois para o desenvolvimento de aplicativos para iPhone e passa a contar com grande popularidade.

```
// Programa simples de exemplo

#import <Foundation/Foundation.h>

int main (int argc, const char * argv[])
{
        NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
        NSLog (@"Hello, World!");
        [pool drain];
        return 0;
}
```

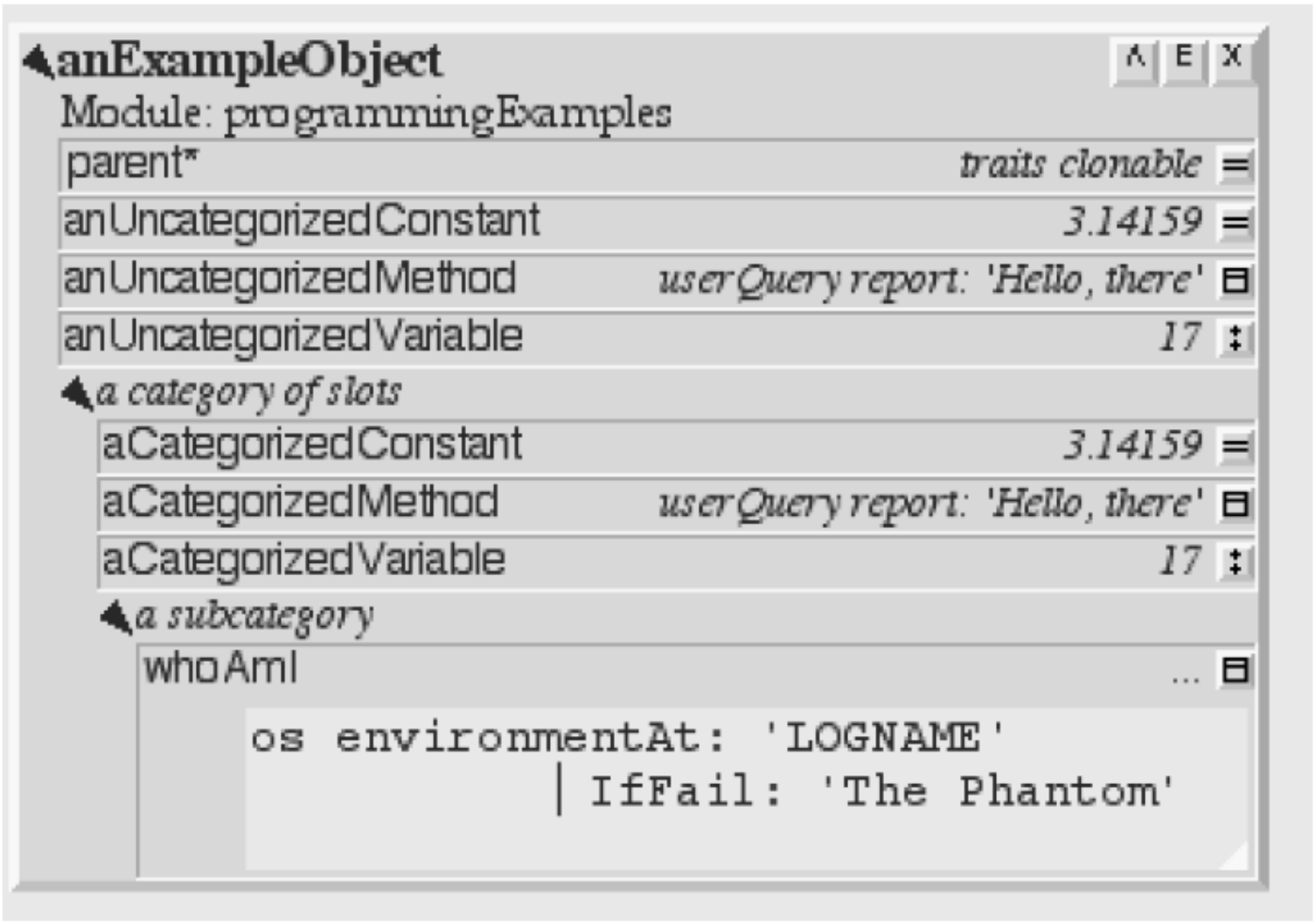
Evolução das linguagens

- Self é uma linguagem criada em meados dos anos 80 por David Ungar e Randall B. Smith na Sun. Sua maior virtude é a simplicidade (e tudo o que vem com ela: flexibilidade, agilidade, etc.).

```
assertPositive: x = (
                x > 0 ifTrue: [ ^ 'ok' ].
                error: 'non-positive x' )

( |     parent* = traits point.
        x <- 3 + 4.
        y <- 5.
| )
```

Self

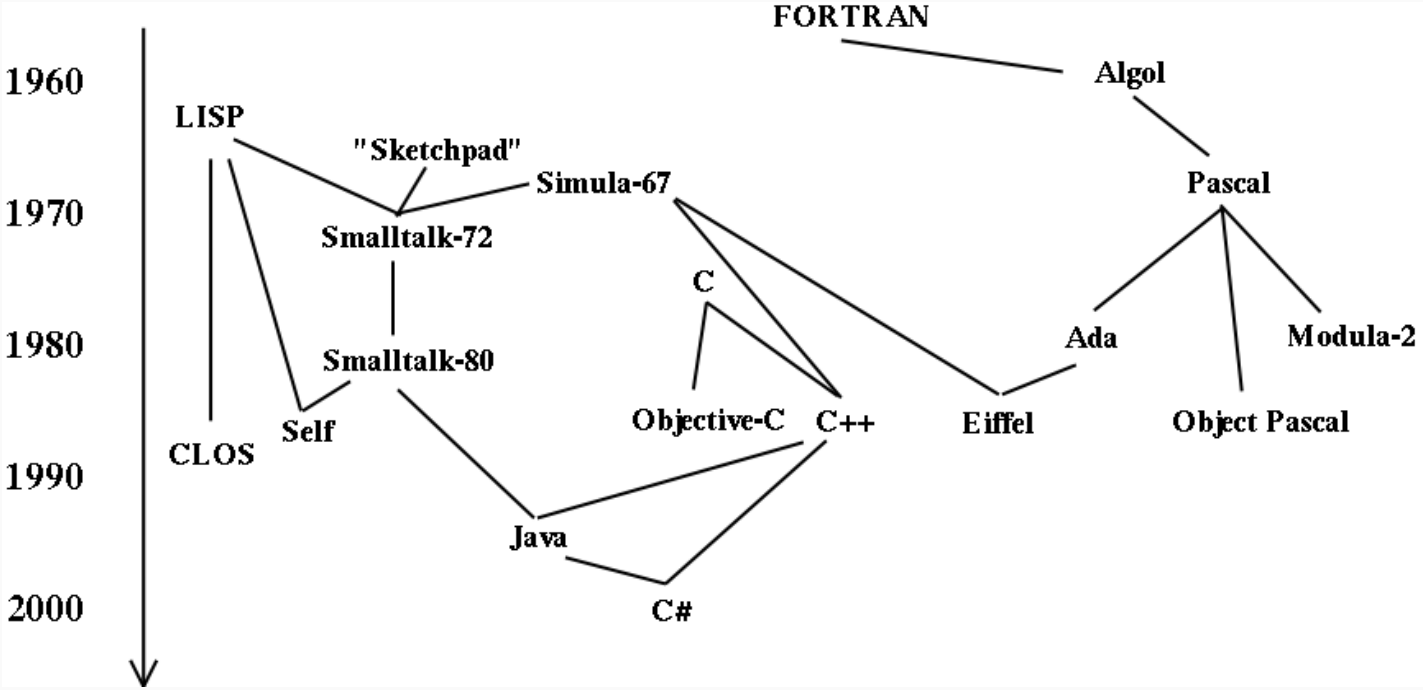

anExampleObject
Module: programmingExamples
parent*
traits clonable
anUncategorizedConstant
3.14159
anUncategorizedMethod
userQuery report: 'Hello, there'
anUncategorizedVariable
17
a category of slots
aCategorizedConstant
3.14159
aCategorizedMethod
userQuery report: 'Hello, there'
aCategorizedVariable
17
a subcategory
whoAmI
...
os environmentAt: 'LOGNAME'
| IfFail: 'The Phantom'

Evolução das linguagens

- Em 1995, a Sun lança o ambiente Java, que inclui a linguagem Java, altamente influenciada por Smalltalk mas com sintaxe parecida com C++.
- Java apresenta um crescimento enorme tanto no meio acadêmico quanto industrial.
- A Microsoft se sente ameaçada e tenta criar a sua própria versão de Java com algumas incompatibilidades em relação à versão padrão controlada pela Sun. A Sun reclama na justiça e Bill Gates declara Java linguagem non grata na Microsoft.
- Microsoft lança sua versão de Java: C#
- C# é mais complexa do que Java, voltando a incorporar alguns aspectos de C++ que Java tinha optado por não incluir. Com estes aspectos adicionais, perde em elegância e clareza mas ganha em desempenho.

Evolução das linguagens

- Ruby foi lancada em 12/2005 e foi influenciada por Perl, Smalltalk, Eiffel e Lisp.
- A motivação de seu criador, Yukihiro Matsumoto, era produzir uma linguagem que fosse mais poderosa que Perl e mais orientada a objetos do que Python.
- Ruby passou a se tornar popular a partir de 2005, com o Ruby on Rails.



O que faz de uma linguagem Orientada a Objetos?

- Em 87, Peter Wegner, publicou um artigo na OOPSLA que resolveu a questão (pelo menos entre aqueles que concordam com o Wegner :-)
- Para uma linguagem ser Orientada a Objetos ela precisa:
  1. ser baseada em objetos, ou seja, deve ser fácil programar objetos que encapsulam dados e operações;
  2. ser baseadas em classes, ou seja, cada objeto pertence a (ou é fabricado a partir de) uma classe; e
  3. permitir herança, ou seja, deve ser fácil agrupar classes em hierarquias de subclasses e superclasses.
- Outros estudiosos (chatos?) acrescentam outros itens à lista:
  1. enlace dinâmico, tardio (late binding)
  2. coleta automática de lixo
  3. herança múltipla
  4. agregação
- A linguagem Self é uma grande exceção entre as linguagens claramente orientada a objetos: ela não possui classes, ao invés usa delegação.

Bibliografia

- Notas de aula do prof. Fabio Kon (IME/USP)
  `http://www.ime.usp.br/~kon/`