Programowanie obiektowe

Wykład 15

Marcin Młotkowski

23 czerwca 2022

Plan wykładu

Plan wykładu

C. A. R. Hoare, 1966

Koncepcja obiektu

Wprowadzenie koncepcji obiektu (rzeczywistego) modelowanego w systemach komputerowych jako rekord.

C. A. R. Hoare, 1966

Koncepcja obiektu

Wprowadzenie koncepcji obiektu (rzeczywistego) modelowanego w systemach komputerowych jako rekord.

Klasy

Klasy odzwierciedlają klasyfikację obiektów.

Inne koncepcje

Referencja

"Adres" rekordu w pamięci. Referencje mają określone typy, które są sprawdzane podczas kompilacji.

Inne koncepcje

Referencja

"Adres" rekordu w pamięci. Referencje mają określone typy, które są sprawdzane podczas kompilacji.

Tworzenie rekordów

Rekrody są tworzone dynamicznie i usuwane za pomocą odśmiecania.

Inne koncepcje

Referencja

"Adres" rekordu w pamięci. Referencje mają określone typy, które są sprawdzane podczas kompilacji.

Tworzenie rekordów

Rekrody są tworzone dynamicznie i usuwane za pomocą odśmiecania.

Podklasy

Deklarownie klas rekordów poprzez rozszerzanie zadeklarowanych wcześniej klas.

Inne pomysły Hoare

null

Wprowadzenie pojęcia pustej referencji.

Inne pomysły Hoare

null

Wprowadzenie pojęcia pustej referencji.

QCon, London, 2009

I call it my billion-dollar mistake. It was the invention of the null reference in 1965.

Powstanie języka SIMULA

Zespół

Kristen Nygaard i Ole-Johan Dahl (Univac, Norwegian Computing Center)

Główne cele projektu

Stworzenie języka symulującego dyskretne zjawiska.

Przykład deklaracji klasy

```
Class Figura(X, Y); Real X, Y;
    Begin
    Real Pole;
    Procedure Drukuj
    Begin
        OutText ("figura");
        OutFix (X, 3, 10); OutFix (Y, 3, 10);
        OutImage;
    End;
    OutText ("Konstrukcja obiektu");
    Figura;
End;
```

Przykład deklaracji klasy

```
Class Figura(X, Y); Real X, Y;
    Begin
    Real Pole;
    Procedure Drukuj
    Begin
        OutText ("figura");
        OutFix (X, 3, 10); OutFix (Y, 3, 10);
        OutImage;
    End;
    OutText ("Konstrukcja obiektu");
    Figura;
End;

Ref (Figura) fig;
fig :- New Figura(1.0, 1.0);
```

Podklasy i metody wirtualne

```
Class Figura;
virtual: Real Procedure Pole;
Begin
    ...
End;

Figura Class Prostokąt(W, S); Real W, S;
Begin
    Real Procedure Pole;
        Pole := W * S;
    End
End
```

Ukrywanie implementacji

Pola i metody mogą być oznaczone jako **Hidden** lub **Protected**.

Zalążki refleksji

xb is B

Obiekt jest klasy B.

xb in B

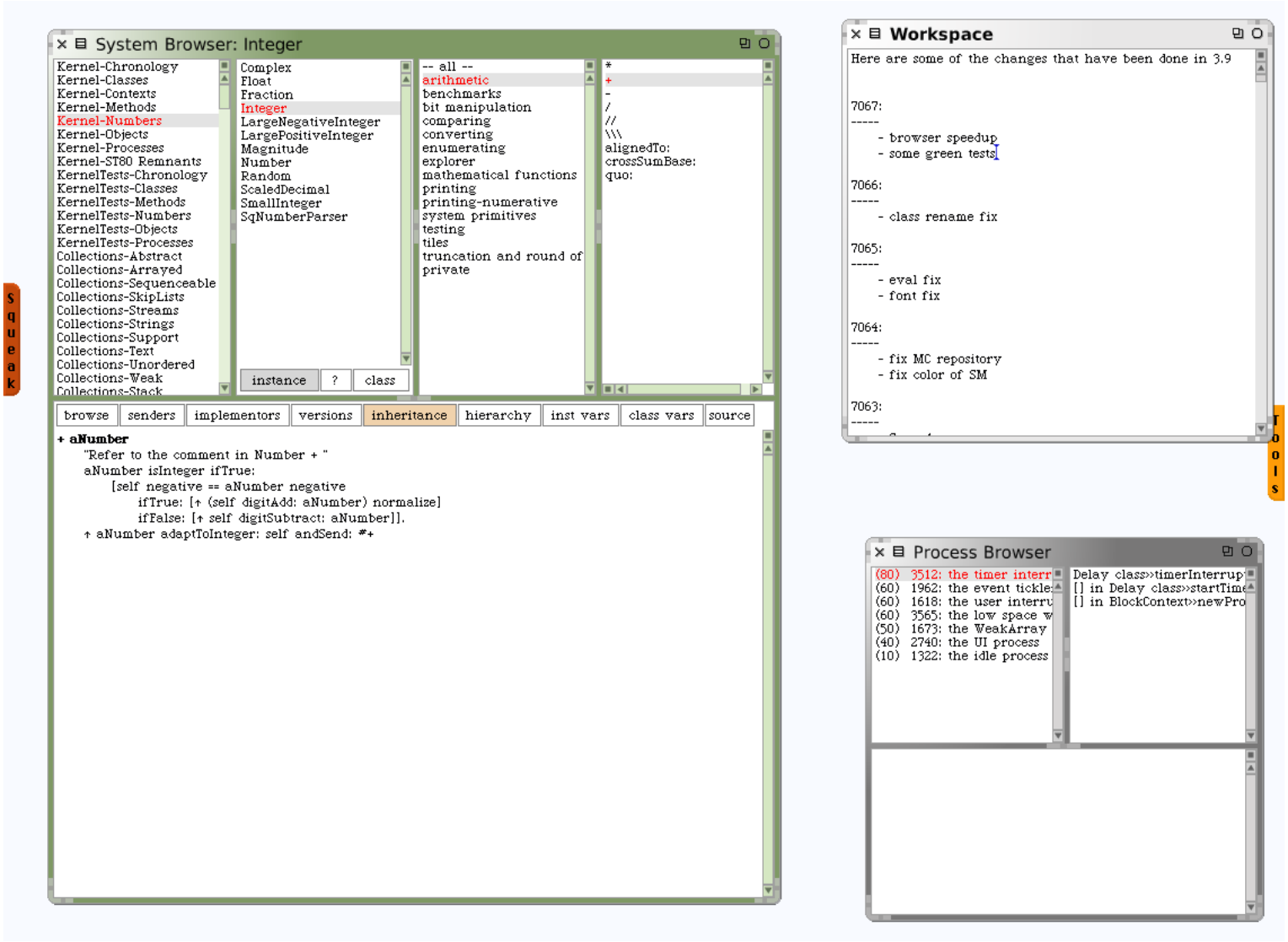
Obiekt jest klasy B lub podklasy B.

Symulacje procesów dyskretnych

Wsparcie dla symulacji (przypomina wątki):

- **Process**: obiekt symulujący działanie jakiegoś rzeczywistego obiektu;
- **Simulation**: kod uruchamiający symulację;
- **Wait**: usypianie;
- **Activate**: budzenie;

Plan wykładu


Squeak
System Browser: Integer
Kernel-Chronology
Kernel-Classes
Kernel-Contexts
Kernel-Methods
Kernel-Numbers
Kernel-Objects
Kernel-Processes
Kernel-ST80 Remnants
KernelTests-Chronology
KernelTests-Classes
KernelTests-Methods
KernelTests-Numbers
KernelTests-Objects
KernelTests-Processes
Collections-Abstract
Collections-Arrayed
Collections-Sequenceable
Collections-SkipLists
Collections-Streams
Collections-Strings
Collections-Support
Collections-Text
Collections-Unordered
Collections-Weak
Collections-Stack
Complex
Float
Fraction
Integer
LargeNegativeInteger
LargePositiveInteger
Magnitude
Number
Random
ScaledDecimal
SmallInteger
SqNumberParser
instance
?
class
-- all --
arithmetic
benchmarks
bit manipulation
comparing
converting
enumerating
explorer
mathematical functions
printing
printing-numerative
system primitives
testing
tiles
truncation and round of
private
*
+
-
/
//
\\\
alignedTo:
crossSumBase:
quo:
browse
senders
implementors
versions
inheritance
hierarchy
inst vars
class vars
source
+ aNumber
"Refer to the comment in Number + "
aNumber isInteger ifTrue:
[self negative == aNumber negative
ifTrue: [↑ (self digitAdd: aNumber) normalize]
ifFalse: [↑ self digitSubtract: aNumber]].
↑ aNumber adaptToInteger: self andSend: #+
Workspace
Here are some of the changes that have been done in 3.9
7067:
- browser speedup
- some green tests
7066:
- class rename fix
7065:
- eval fix
- font fix
7064:
- fix MC repository
- fix color of SM
7063:
Tools
Process Browser
(80) 3512: the timer interr
(60) 1962: the event tickle
(60) 1618: the user interru
(60) 3565: the low space w
(50) 1673: the WeakArray
(40) 2740: the UI process
(10) 1322: the idle process
Delay class»timerInterrup
[] in Delay class»startTim
[] in BlockContext»newPro

Cele projektu

Zaprojektowanie od podstaw czystego języka obiektowego: wszystko ma być obiektem.

Kilka faktów

- Laboratorium Xerox Palo Alto Research Center
- okres powstania: lata 70;
- wersja referencyjna: Smalltalk'80;
- twórcy: Alan Kay, Dan Ingalls

Wszystko jest obiektem

- klasy są obiektami;
- liczby i wartości boolowskie są obiektami;
- tablice są obiektami;
- metody są obiektami;

Bardzo prosta składnia

```
3 * 5 -2
3 neg
tablica at: 3 put:'trzy'
tablica at: 3
```

Tworzenie nowych obiektów

```
r1 := Rectange new.
r2 := Rectangle width: 3 height: 4.
```

'Instrukcje' warunkowe

```
(array size) > 5
    ifTrue: [ "To duża tablica" printString ]
    ifFalse: [ "Nieduża tablica" printString ]
```

'Instrukcje' pętli

```
aList do: [ | element | Transcript show: element ]
```

'Instrukcje' pętli

```
aList do: [ | element | Transcript show: element ]

[ i > 0 ] whileTrue: [ i := i + 1 ]
```

Środowisko programistyczne

Prawdziwe GUI!

- Przeglądarka klas;
- Uruchamianie programów;
- Zaimplementowane w Smalltalku (MVC).

Inne cechy

- implementacja maszyny wirtualnej;
- przenośność;
- wbudowany debugger;
- implementacja Smalltalka w Smalltalku;
- inspektor klas i obiektów.

Wpływ języka

- Większość współczesnych języków obiektowych, w tym: Objective-C
- popularny język w programowaniu zwinnym i szybkim prototypowaniu;

Objective-C

Objective-C (1984 r.): język C ze Smalltalkiem

```
[myColor changeColorToRed:5.0 green:2.0 blue:6.0];
```

Swift

Swift (2014 r): Objective-C bez bagażu C

- wywołanie metod z kropką;
- wiele z programowania funkcyjnego (inferencja typów, typy opcjonalne).

Plan wykładu

Loglan 82: historia

1982 rok: prof. Antoni Kreczmar, prof. Andrzej Salwicki, Uniwersytet Warszawski

Przykład

```
unit zespolone: class(re, im: real)
  var modul: real;
  unit plus: function(z: zespolone) : zespolone;
    begin
      result := new zespolne(re+z.re, im + z.im))
    end plus
  begin
    modul := sqrt(re*re + im*im)
  end zespolone

z1 := new zespolone(1.0, 2.0);
```

Cele i koncepcje

- efektywne zarządzanie pamięcią;
- wątki i procesy;
- natywny protokół współpracy między kilkoma instancjami programów w Loglan'82

Plan wykładu

Założenia

Brak klas

Obiekty są tworzone *ex nihilo* lub jako klony innych obiektów, a następnie modyfikowane.

Założenia

Brak klas

Obiekty są tworzone *ex nihilo* lub jako klony innych obiektów, a następnie modyfikowane.

Dziedziczenie

Klasa ma referencję do tzw. prototypu („nadklasy") i może korzystać z jego metod.

Realizacje koncepcji

- Self (inspirowany Smalltalkiem)
- JavaScript
- ActionScript (Flash Player)

JavaScript

- Nie mylić z językiem JAVA!
- obecnie JavaScript to implementacja standardu ECMAScript

Tworzenie obiektów ex nihilo

Konstruktor

```
function ObiektOsoba(imie, nazwisko) {
  this.imie = imie;
  this.nazwisko = nazwisko;
  function info() {
    alert("imie " + this.imie);
  }
  this.info = info;
}

var informatyk = new ObiektOsoba("Alan", "Turing");
```

Tworzenie obiektów w locie

```
var informatyk = {
    imie: 'Alan',
    nazwisko: 'Turing',
    info: function() { alert('info'); }
}
```

Prototypowanie (dziedziczenie)

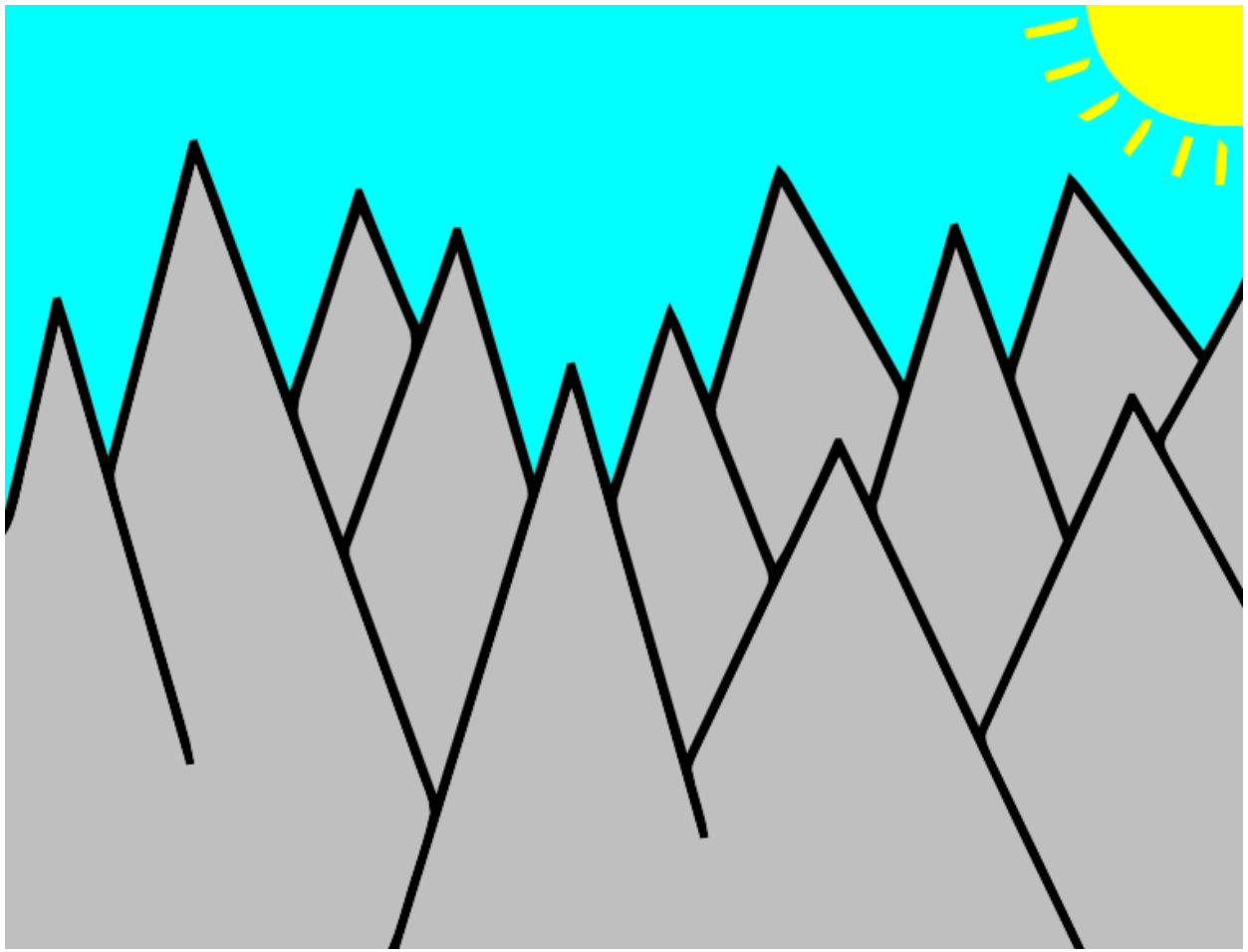
```
function Student(kierunek) {
  this.info = function() {
    alert("Kierunek " + this.kierunek);
  }
}

Student.prototype = Osoba;
```

Pochodne JavaScript

TypeScript, Dart, CoffeeScript czy ClosureScript

Plan wykładu

PYTANIA?



KONIEC