

Programowanie obiektowe

Wykład 15.

Marcin Młotkowski

14 czerwca 2024

Plan wykładu

- 1 Simula 67
 - Koncepcje Tony'ego Hoare, 1966
 - SIMULA 67
- 2 C++
- 3 Smalltalk
- 4 Polacy nie gęsi
 - Loglan'82
 - Nemerle
- 5 Języki prototypowe
- 6 Co dalej
- 7 Zakończenie

Plan wykładu

- 1 Simula 67
 - Koncepcje Tony'ego Hoare, 1966
 - SIMULA 67
- 2 C++
- 3 Smalltalk
- 4 Polacy nie gęsi
 - Loglan'82
 - Nemerle
- 5 Języki prototypowe
- 6 Co dalej
- 7 Zakończenie

Simula 67
C++
Smalltalk
Polacy nie gęsi
Języki prototypowe
Co dalej
Zakończenie

Koncepcje Tony'ego Hoare, 1966
SIMULA 67

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

Rekordy są tworzone dynamicznie i usuwane za pomocą odśmiecania.

Inne koncepty

Referencja

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

Tworzenie rekordów

Rekordy są tworzone dynamicznie i usuwane za pomocą odśmiecania.

Podklasy

Deklarowanie klas rekordów poprzez rozszerzanie zadeklarowanych wcześniej klas.

Inne pomysły Hoare

Wprowadzenie pojęcia pustej referencji:

In order to enable references to represent partial functional relationships, i.e. ones which do not necessarily yield a value, a special reference value null is introduced. This value fails to refer to a record, and any attempt to use it to refer to a record leads to an undefined result.

Inne pomysły Hoare

Wprowadzenie pojęcia pustej referencji:

In order to enable references to represent partial functional relationships, i.e. ones which do not necessarily yield a value, a special reference value null is introduced. This value fails to refer to a record, and any attempt to use it to refer to a record leads to an undefined result.

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

QCon, London, 2009

Źródło: Relacja na YouTube

Powstanie języka SIMULA

Zespół

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

Główne cele projektu

Stworzenie języka symulującego dyskretnie 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**.

Załączki 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

- 1 Simula 67
 - Koncepcje Tony'ego Hoare, 1966
 - SIMULA 67
- 2 C++
- 3 Smalltalk
- 4 Polacy nie gęsi
 - Loglan'82
 - Nemerle
- 5 Języki prototypowe
- 6 Co dalej
- 7 Zakończenie

Simula 67
C++
Smalltalk
Polacy nie gęsi
Języki prototypowe
Co dalej
Zakończenie

Twórca

Bjarne Stroustrup
używał Simuli przy pisaniu doktoratu (1979 r.) z obliczeń
rozproszonych.

Wnioski

Doświadczenia Stroustrupa z pracy z językiem Simula:

- + bardzo przydatne cechy w implementacji dużych programów;
- zbyt wolny, by był praktyczny.

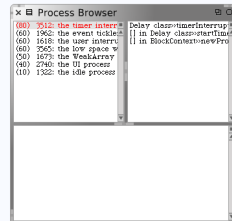
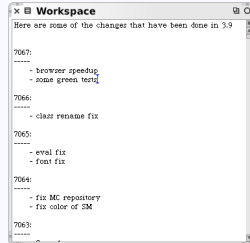
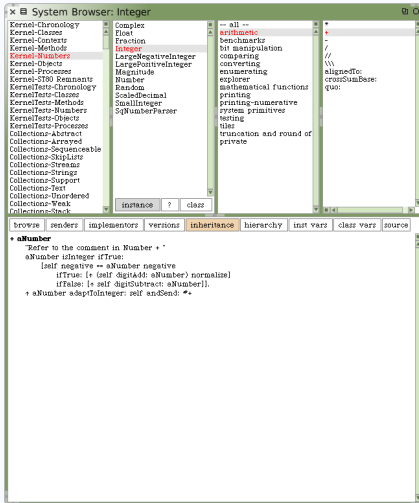
(Bardzo) krótka historia

- początkowo (1978-1982) był to *język C z klasami* i kompilator Cfront z C++ do C;
- 1998 r.: C++98;
- ...
- 2023 r.: C++23;
- 12 czerwca 2024: 2. miejsce w popularności (wg. rankingu TIOBE).

Plan wykładu

- 1 Simula 67
 - Koncepcje Tony'ego Hoare, 1966
 - SIMULA 67
- 2 C++
- 3 **Smalltalk**
- 4 Polacy nie gęsi
 - Loglan'82
 - Nemerle
- 5 Języki prototypowe
- 6 Co dalej
- 7 Zakończenie

Simula 67
C++
Smalltalk
Polacy nie gęsi
Języki prototypowe
Co dalej
Zakończenie



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 := Rectangle 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

Wstawka Smalltalkowa w kodzie C:

```
[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

- 1 Simula 67
 - Koncepcje Tony'ego Hoare, 1966
 - SIMULA 67
- 2 C++
- 3 Smalltalk
- 4 **Polacy nie gęsi**
 - Loglan'82
 - Nemerle
- 5 Języki prototypowe
- 6 Co dalej
- 7 Zakończenie

Simula 67
C++
Smalltalk
Polacy nie gęsi
Języki prototypowe
Co dalej
Zakończenie

Loglan'82
Nemerle

Loglan 82: historia

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

Przykład

```
unit zespolone: class(re, im: real)
  var modul: real;
  unit plus: function(z: zespolone) : zespolone;
    begin
      result := new zespolone(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

Powstanie języka Nemerle

2003: **Michał Moskal**, Paweł Olszta, Kamil Skalski.

Cechy języka:

- wspiera różne paradygmaty programowania: obiektowy, funkcjonalny, makrodefinicje;
- typy algebraiczne, rekonstrukcja typów, dopasowywanie wzorców;
- kompilacja do .NET.

Przykład

```
class Osoba
{
    public Name : string {get; set ;}

    public SayHello() : void
    {
        Console.WriteLine($"Hello $Name");
    }
}
```

Plan wykładu

- 1 Simula 67
 - Koncepcje Tony'ego Hoare, 1966
 - SIMULA 67
- 2 C++
- 3 Smalltalk
- 4 Polacy nie gęsi
 - Loglan'82
 - Nemerle
- 5 Języki prototypowe
- 6 Co dalej
- 7 Zakończenie

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/współdzielenie kodu

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;
```

Simula 67
C++
Smalltalk
Polacy nie gęsi
Języki prototypowe
Co dalej
Zakończenie

Pochodne JavaScript

TypeScript, Dart, CoffeeScript czy ClosureScript

Plan wykładu

- 1 Simula 67
 - Koncepcje Tony'ego Hoare, 1966
 - SIMULA 67
- 2 C++
- 3 Smalltalk
- 4 Polacy nie gęsi
 - Loglan'82
 - Nemerle
- 5 Języki prototypowe
- 6 Co dalej**
- 7 Zakończenie

Simula 67
C++
Smalltalk
Polacy nie gęsi
Języki prototypowe
Co dalej
Zakończenie

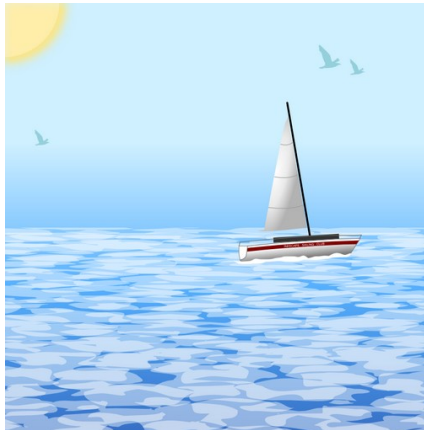
Teraźniejszość

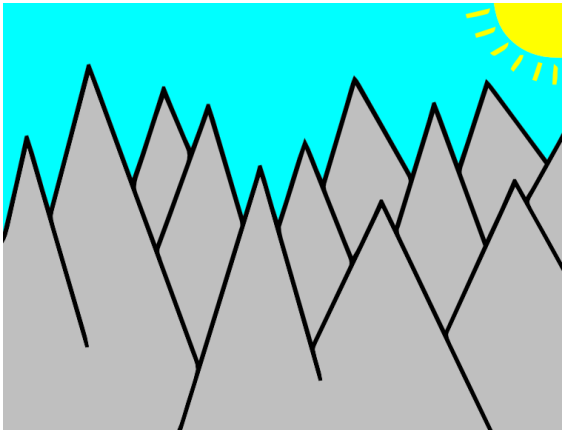
► [Link](#)

Plan wykładu

- 1 Simula 67
 - Koncepcje Tony'ego Hoare, 1966
 - SIMULA 67
- 2 C++
- 3 Smalltalk
- 4 Polacy nie gęsi
 - Loglan'82
 - Nemerle
- 5 Języki prototypowe
- 6 Co dalej
- 7 Zakończenie

PYTANIA?





KONIEC