ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Лекция № 2 / **9**
14.11.2017 г.

ПРЕДПОСЫЛКИ

- ООП != C++.
  - Уж точно ООП в 2017 г. != C++17.
- ООП != ООП
  - Разные люди понимают под ООП разное.

КАК РАЗОБРАТЬСЯ?

- Разделяем *идеи* и их *реализацию*.

- Плодотворных идей не так много, но бесконечно широки варианты их реализации.

- Полезно знать, какие идеи существуют. Это позволит узнавать их в существующих реализациях.

ООП.Т=0

- Язык Simula 67 (Kristen Nygaard, Ole-Johan Dahl, Норвегия, 1960-е гг.):
  - База: ALGOL 60.
  - Объекты и классы.
  - Наследование и виртуальные процедуры.
  - Coroutines.
  - Сборка мусора!
- В то время даже не было понятия «объектно-ориентированный».

```
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;
```

Программа на Simula 67

```
Simulation Begin
   Class FittingRoom; Begin
      Ref (Head) door;
      Boolean inUse;
      Procedure request; Begin
         If inUse Then Begin
             Wait (door);
             door.First.Out;
         End;
         inUse:= True;
      End;
      Procedure leave; Begin
         inUse:= False;
         Activate door.First;
      End;
      door:- New Head;
   End;

   Procedure report (message); Text message; Begin
      OutFix (Time, 2, 0); OutText (": " & message); OutImage;
   End;

   Process Class Person (pname); Text pname; Begin
      While True Do Begin
         Hold (Normal (12, 4, u));
         report  (pname & " is requesting the fitting room");
         fittingroom1.request;
         report (pname & " has entered the fitting room");
         Hold (Normal (3, 1, u));
         fittingroom1.leave;
         report (pname & " has left the fitting room");
      End;
   End;

   Integer u;
   Ref (FittingRoom) fittingRoom1;

   fittingRoom1:- New FittingRoom;
   Activate New Person ("Sam");
   Activate New Person ("Sally");
   Activate New Person ("Andy");
   Hold (100);
End;
```

Симуляция событий

ИДЕИ ООП В SIMULA 67

- «Скандинавская школа». Контекст: моделирование событий.
  - Много участников событий, ведущих себя одинаковым образом (*объекты и классы*).
- Общность у разных классов (*наследование*). В Simula это называлось «конкатенация классов»: C_1, … C_k, …, C_N.
- Возможность переопределения атрибутов класса на любом уровне наследования (*виртуальные функции*).

ООП.Т=1

- Alan Kay.
- Автор термина «ООП», концепции графического интерфейса и много другого.
- Создатель языка Smalltalk.

I thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages (so messaging came at the very beginning -- it took a while to see how to do messaging in a programming language efficiently enough to be useful).

Я представлял объекты похожими на биологические клетки и/или отдельные компьютеры в сети, могущие сообщаться только посредством сообщений (так что сообщения были с самого начала, однако понадобилось время, чтобы понять, как реализовать их в языке программирования достаточно эффективно, чтобы это можно было использовать).

I wanted to get rid of data. The B5000 almost did this via its almost unbelievable HW architecture. I realized that the cell/whole-computer metaphor would get rid of data, and that <– would be just another message token (it took me quite a while to think this out because I really thought of all these symbols as names for functions and procedures.

Я хотел избавиться от данных. Компьютер B5000 уже практически позволял это благодаря своей невероятной архитектуре. Я понял, что метафора клетки/отдельного компьютера дала бы эту возможность, и оператор присваивания (<–) был бы просто ещё одним типом сообщения (продумывание всего этого заняло время, ибо я действительно думал о всех этих символах как именах для функций и процедур).

My math background made me realize that each object could have several algebras associated with it, and there could be families of these, and that these would be very very useful. The term "polymorphism" was imposed much later (I think by Peter Wegner) and it isn't quite valid, since it really comes from the nomenclature of functions, and I wanted quite a bit more than functions. I made up a term "genericity" for dealing with generic behaviors in a quasi-algebraic form.

Моё математическое образование позволило понять, что с каждым объектом могло быть связано несколько алгебр, и были возможны их семейства, и что это было бы очень-очень полезно. Термин «полиморфизм» родился гораздо позже (думаю, его предложил Питер Вегнер), и он не очень подходит, потому что происходит из номенклатуры функций, а я хотел гораздо больше, чем просто функции. Я придумал термин «обобщённость» (genericity) для описания общего поведения в квази-алгебраической форме.

I didn't like the way Simula I or Simula 67 did inheritance (though I thought Nygaard and Dahl were just tremendous thinkers and designers). So I decided to leave out inheritance as a built-in feature until I understood it better.

Мне не нравилось, как наследование реализовано в Simula I или Simula 67 (хотя я считал Найгарда и Даля выдающимися мыслителями и разработчиками). Поэтому я не стал делать наследование встроенным в язык до тех пор, пока не разобрался в нём получше.

OOP to me means only messaging, local retention and protection and hiding of state-process, and extreme late-binding of all things. It can be done in Smalltalk and in LISP. There are possibly other systems in which this is possible, but I'm not aware of them.

Для меня ООП означает только передачу сообщений, локальное удержание, защиту и сокрытие состояния-процесса, а также как можно более позднее связывание всего. Это можно реализовать в Smalltalk и LISP. Возможно, есть и другие системы, но мне они неизвестны.

(2003 г.)

Smalltalk

```
2 + 2.      "4"

Date today. "12 November 2017"

DateAndTime today.  "2017-11-12T00:00:00+07:00"

1000 factorial / 999 factorial. "1000"

(1/3) + (4/5). "(17/15)"
```

Сообщение `+`

Сообщение `today`

```
1 class. "SmallInteger"

1 class maxVal class. "SmallInteger"

(1 class maxVal + 1) class. "LargePositiveInteger"

'Hello' at: 1. "$H"

#('hello' 'World') at: 2 put: 'Bold'; yourself. "#('hello' 'Bold')"
```

Сообщение `at:put:`

Объявление переменной

Блок с двумя аргументами

```
|b|
[:x :y| x + y] value:3 value: 5. "8"


b := [:x | x+2].
b value: 12. "14"
```

Ветвление

```
3 > 10
   ifTrue: [Transcript show: 'maybe there''s a bug ....']
   ifFalse: [Transcript show: 'No : 3 is less than 10’].

1 to: 100 by: 3 do: [:i | Transcript show: i asString; cr].

#(11 38 3 -2 10) do: [:each |
     Transcript show: each printString; cr].

#(11 38 3 -2 10) select: [:each | each odd].  "#(11 3)"
```

Циклы

LIVE CODING !!!

https://ci.inria.fr/pharo-contribution/job/UpdatedPharoByExample/lastSuccessfulBuild/artifact/book-result/FirstApplication/FirstApplication.html

ПОЧЕМУ SMALLTALK НЕ ВЗЛЕТЕЛ

- http://wiki.c2.com/?WhyIsSmalltalkDead

- Дорого (лицензия стоила тысячи $). Сложно зарабатывать независимым разработчикам (нужна платформа).

- Медленно.

- Не было большой компании, которая бы проталкивала язык (как Microsoft — C++ и Sun Microsystems — Java).

- Небольшая популярность и относительная замкнутость быстро привела к технологической отсталости.