Объектно-ориентированное программирование

Object-oriented programming

II. Возникновение понятия “объект”

Emergence of objects

"A language is considered *object-based* if it directly supports **data abstraction** and **classes**. An *object-oriented* language is one that is object-based but also provides support for **inheritance** and **polymorphism**."

***G. Booch** et al.*

"Object-Oriented Analysis, Design and Implementation"

Критерии к языкам программирования

```
i := 0
while i != f(i) do
    i := g(i)
```

- Вычислимость (computability, частичные ф-ции, “проблема остановки”)
- Статический анализ (compile-time/run-time)
- Производительность (efficiency, i.e. влияние управления памятью)
- Выразительность (expressiveness)

https://stanford-cs242.github.io/f19/

Lisp (**lis**t **p**rocessor)

"A Lisp programmer knows the value of everything, but the cost of nothing."

A. Perlis

- Любые частичные рекурсивные функции могут быть выражены в языке (Turing-complete)
- Использует λ-исчисление в качестве базовой модели
- Не имеет утверждений, только выражения (следовательно - нет побочных эффектов)
- Код и данные в программе взаимозаменяемы, так как представлены списками

https://www.cs.tufts.edu/~nr/cs257/archive/john-mccarthy/recursive.pdf

Lisp (**lis**t **p**rocessor)

Для выражения **λx.(x² + y)**:

```
(lambda (x) (+ (square x) y))
```

```
(define find (lambda (x y)
    (cond ((equal y nil) nil)
          ((equal x (car y)) x)
    (true (find x (cdr y)))
)))
```

Simula

- симуляции, основанные на событиях (event-based)
- события обрабатываются в “очередях”
  - новые события попадают в очередь
  - первое событие обрабатывается (симулируется)
  - все события, сгенерированные этой симуляцией, помещаются в конец очереди
  - цикл продолжается до тех пор, пока очередь не опустеет
- очереди должны быть обобщенными структурами данных
- идея “процесса”: классы, ссылки и сопрограммы (coroutines)

https://www.cs.tufts.edu/comp/150FP/archive/kristen-nygaard/hopl-simula.pdf

Simula

“We needed subclasses of processes with actions, not only of pure data records.”

K. Nygaard

```
class Point; real x, y;
begin
    boolean procedure equals(p); ref(Point) p;
        if p =/= none then
        equals := abs(x - p.x) + abs(y - p.y) < 0.00001;
    real procedure distance(p); ref(Point) p;
        if p == none then error
        else distance := sqrt((x - p.x)**2 + (y - p.y)**2);
end
```

Smalltalk

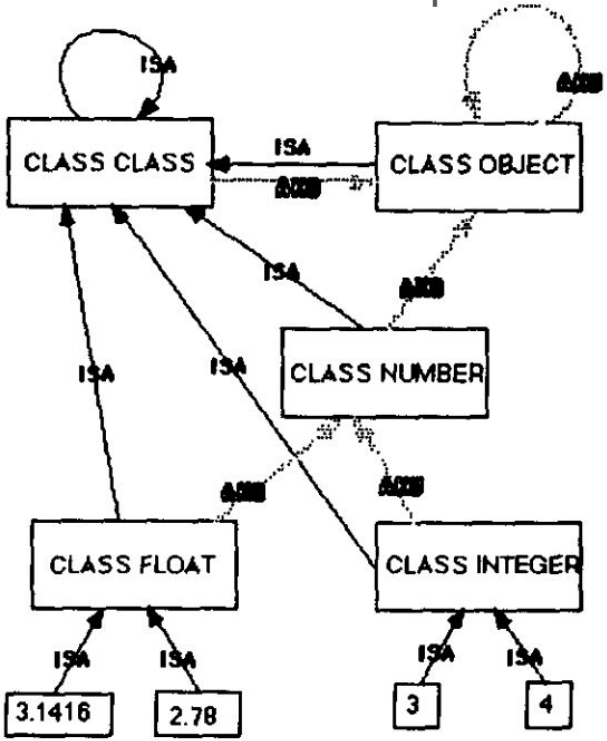
"What I got from Simula was that you could now *replace bindings and assignment with* ***goals***. The last thing you wanted any programmer to do is mess with **internal state** even if presented figuratively. Instead, the **objects** should be presented as sites of *higher level behaviors* more appropriate for use as dynamic components."

A. Kay

https://dl.acm.org/doi/pdf/10.1145/234286.1057828

Smalltalk

Все является объектом, даже классы и “литералы”.



leon.brinzan@iis.utm.md

Smalltalk

```
5 mod 3
```

```
mod ⇒ (⇑ SELF - (☞ b ← :.) * SELF / b)
```


leon.brinzan@iis.utm.md

Smalltalk

```
5 (mod 3)
```

объект — сообщение для объекта

```
mod 3
```

объект — сообщение для объекта

```
3
```

объект

```
mod ⇒ ((b ← :.) is number ⇒ (⇑ SELF - b * SELF / b)
                error ('non-numeric operand'))
```

Smalltalk

```
to if exp
        ((☞exp ← :) ⇒ (◅ then ⇒ (☞exp ← :.  ◅else ⇒ (☝. exp) exp)
                                error ☞(no then ))
          ◅ then ⇒ (☝. ◅else ⇒ (☞exp ← :) false)
          error ☞(no then)) !

☞val ← if a > 10 then 4 else (if a < 10 then (-4) else 0)!
```

```
to while Cond Exp
        (☞Cond ← ☝.
          ◅ do.
          ☞Exp ← ☝.
           repeat (apply Boolean to Cond ⇒ (Exp eval) done))!

☞str ← stream!
while (kbck and  ((☞t ← kbd) ≠ 13))
    do (str ← t)!
```

Smalltalk

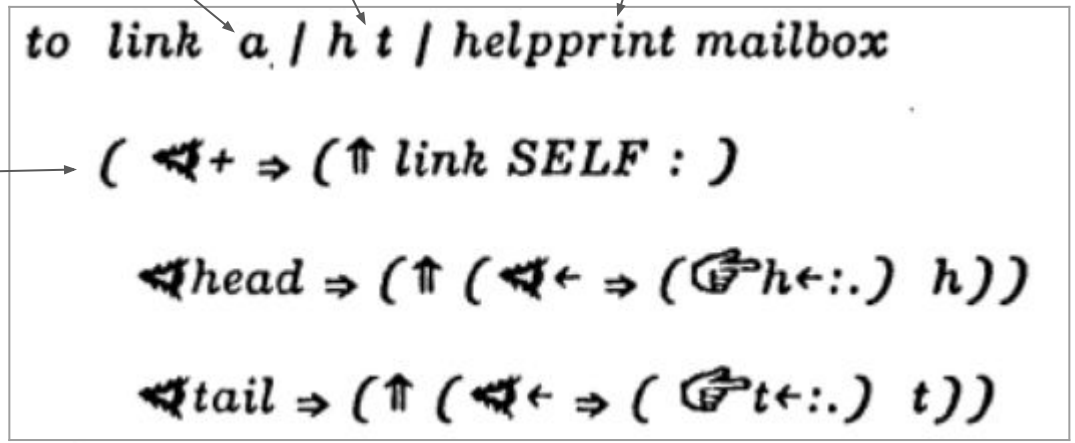
Структура класса:

временные переменные

переменные экземпляра

переменные класса

таблица методов и ответов на сообщения



Что значит “объектно-ориентированный”?

- инкапсуляция
- полиморфизм
- наследование

leon.brinzan@iis.utm.md

Что значит “объектно-ориентированный”?

- ~~инкапсуляция~~ (абстрактные типы данных)
- ~~полиморфизм~~ (алгебраическое свойство системы типов)
- наследование? (не все “OO”-языки реализуют наследование)
- делегация? (делегирование сообщений другим объектам)
- виртуальные функции (dynamic look-up)
- обобщение через абстракцию (скрытая реализация, открытый интерфейс)
- подмена типов (subtyping)

Что значит “объектно-ориентированный”?

“Can you define a new kind of integer, put your new integers into rectangles, ask the system to blacken a rectangle, and have everything work?”

D. Ingalls