

# Обектно ориентирано програмиране

---

ВЪВЕДЕНИЕ В ОБЕКТНО ОРИЕНТИРАНОТО  
ПРОГРАМИРАНЕ

# Какво е ООП?

---

**ООП е революционна идея, тотално различна от всичко в програмирането**

**ООП е еволюционна стъпка, следваща естествено по-ранните абстракции в програмирането**

# Език и мислене

---

Езиците, които говорим, директно влияят върху начина, по който гледаме на света. Това се отнася не само до естествените езици, но и до изкуствени езици като езиците за програмиране.

Пример. Ескимосите и снега

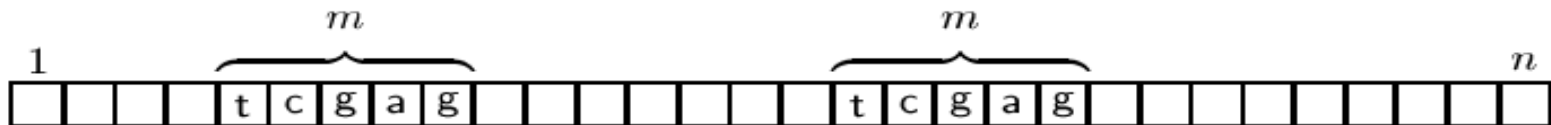
- Езиците на ескимосите имат много думи за различни видове сняг – мокър, пухкав, тежък, заледен и т.н.
- Английски и български езици **не ни принуждават** да разграничаваме различните видове сняг.
- Ескимоските езици ни **водят** да гледаме на света по различен начин.

# Език и мислене ...

---

Пример с езици за програмиране

**Задача.** Анализ на ДНК редица. ДНК се представя като вектор от  $N$  стойности ( $N$  е от порядъка на десетки хиляди). Да се намери дали има повтаряща се последователност с дължина  $M$ .



# Език и мислене ...

---

Програма на FORTRAN

```
      DO 10 I = 1, N-M
      DO 10 J = 1, N-M
      FOUND = .TRUE.
      DO 20 K = 1, M
20    IF X[I+K-1] .NE. X[J+K-1] THEN FOUND = .FALSE.
      IF FOUND THEN ...
10    CONTINUE
```

# Език и мислене ...

---

Програма на APL

Реорганизиране данните

|   |   |   |   |   |   |           |   |    |      |
|---|---|---|---|---|---|-----------|---|----|------|
| A | C | T | C | G | G | positions | 1 | to | M    |
| C | T | C | G | G | A | positions | 2 | to | M +1 |
| T | C | G | G | A | T | positions | 3 | to | M +2 |
| C | G | G | A | T | T | positions | 4 | to | M +3 |
| G | G | A | T | T | C | positions | 5 | to | M +4 |
| G | A | T | T | C | T | positions | 6 | to | M +5 |
| . | . | . |   |   |   |           |   |    |      |
| T | G | G | A | C | C |           |   |    |      |
| G | G | A | C | C | C |           |   |    |      |
| . | . | . |   |   |   |           |   |    |      |

# Език и мислене ...

---

Сортиране на масива

...

T G G A C C

T G G A C C

...

FORTRAN -  $O(M \times N^2)$

APL -  $O(M \times N \log N)$

Езикът ни води да мислим по определен начин

# Език и мислене ...

---

## Хипотеза на Sapir-Whorf

*Възможно е индивид, работещ с един език, да има мисли и да изкаже идеи, които по никакъв начин не могат да се преведат и дори разберат от друг индивид, ползващ друг език.*

## Тезис на Church-Turing

*Всяко изчисление, за което съществува ефективна процедура, може да се реализира чрез машина на Тюринг.*



# Нова парадигма

---

## Paradigm

1. Списък от всички форми на една дума като илюстративен пример
2. Пример или модел

Според Thomas Kuhn [70]:

Множество от теории, стандарти и методи, които заедно представят начин за организиране на знанията.

Революция в науката -> нова парадигма

# Нова парадигма ...

---

Robert Floyd [1979]

**Парадигма в програмирането** е начин на концептуализация какво означава изчисление и как задачите, които трябва да се изпълнят от компютъра, трябва да се структурират и организират.

ООП – нова парадигма

# Въведение в ООП

---

**ООП – начин на гледане към света**

**Пример:** Chris иска да изпрати цветя на приятелката си Robin, която живее в друг град. Chris отива до цветаря Fred и казва какви цветя да се изпратят на Robin и нейния адрес.

# Въведение в ООП ...

---

## Агенти и общности

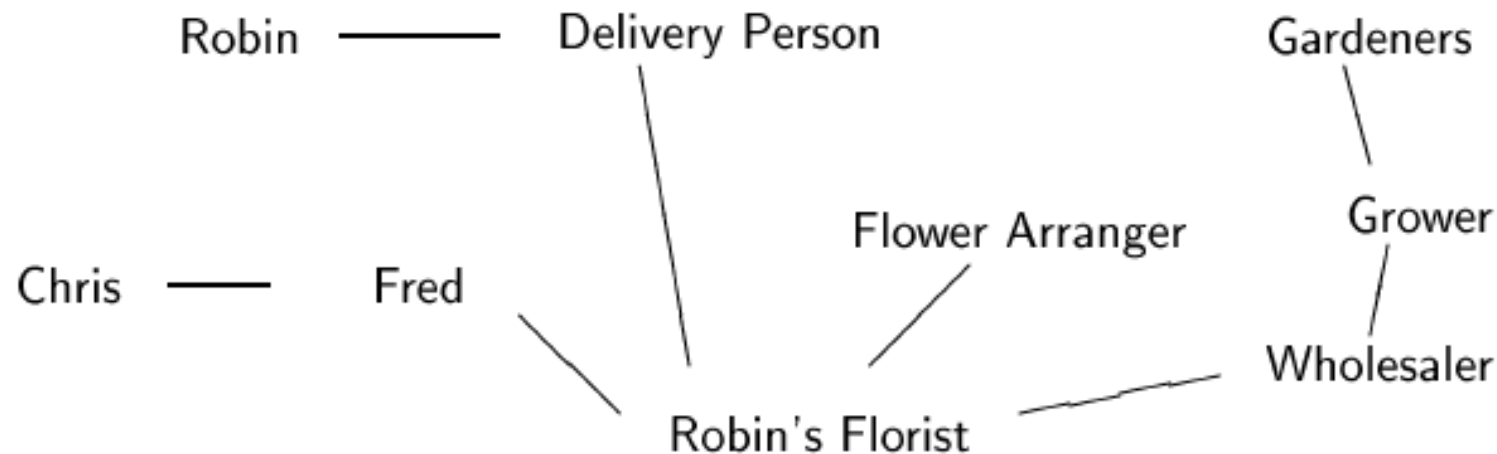
Механизъм за изпращане на цветя:

- Намиране на подходящ **агент** (Fred)
- Предаване на **съобщение**, съдържащо заявката
- **Отговорността** е на Fred
- Fred използва **метод** (алгоритъм или множество от операции)
- **Скриване** на информацията

# Въведение в ООП ...

---

Общност от агенти



# Въведение в ООП ...

---

## I принцип на ООП

Обектно-ориентираната програма е структурирана като **общност** от взаимодействащи си агенти, наречени **обекти**. Всеки обект има роля. Всеки обект предоставя услуга или изпълнява действие, които се използват от другите членове на общността.

# Въведение в ООП ...

---

## Съобщения и методи

Верига от съобщения и действия

## II принцип на ООП

Действие се инициира чрез предаване на **съобщение** към агент (**обект**), отговорен за действието. Съобщението кодира заявка, заедно с допълнителната информация (аргументи), необходима да се изпълни заявката. **Получателят** е обект, до когото е изпратено съобщението. Ако той приеме съобщението, той поема отговорността за изпълнението на посоченото действие. В отговор на едно съобщение, получателят ще изпълни някакъв **метод** за да удовлетвори заявката.

# Въведение в ООП ...

---

- **Скриване на информацията**
- Разлики между извикване на процедура и изпращане на съобщение
  - Съобщението има получател
  - Интерпретация на съобщението – едно и също съобщение може да води до извикването на различни методи в зависимост от получателя
  - **Късно свързване**



# Въведение в ООП ...

---

## Класове и екземпляри

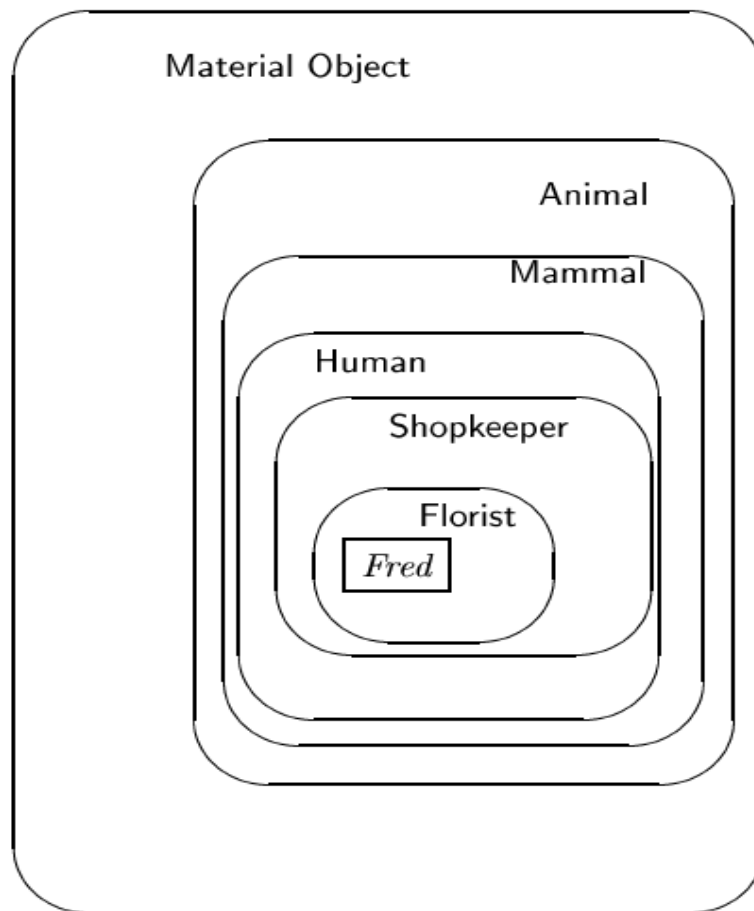
### III принцип на ООП

Всички обекти са **екземпляри** на **клас**. Кой метод ще се активира в отговор на едно съобщение се определя от класа на получателя. Всички обекти от даден клас използват един и същ метод в отговор на подобни съобщения.

# Въведение в ООП ...

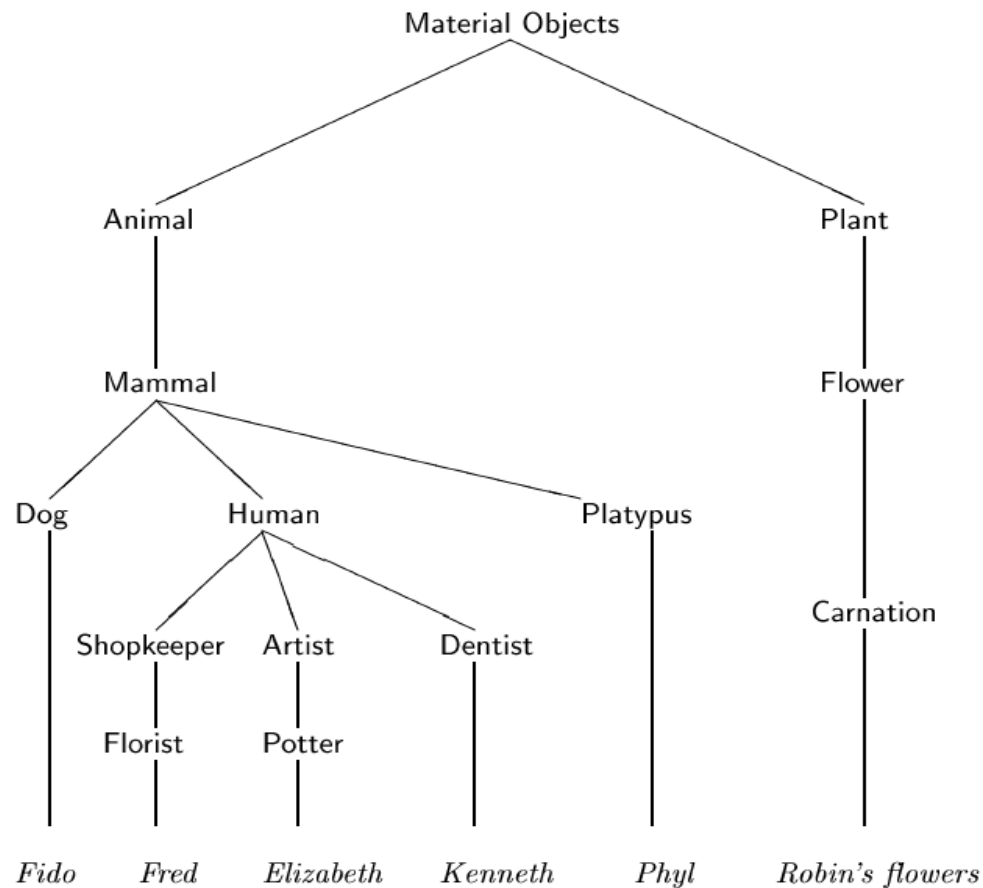
---

Йерархии на класовете –  
наследяване



# Въведение в ООП ...

---



# Въведение в ООП ...

---

## IV принцип на ООП

Класовете могат да се организират в йерархична структура. **Класът – наследник** (подклас) **наследява** свойствата от **класа – родител**. **Абстрактен клас** е клас, който се използва само за създаване на подкласове, т. е. той няма преки екземпляри.

# Въведение в ООП ...

---

## Свързване на методи, подтискане и изключения

Пример - птичечовката

Търсенето на метод, който да се изпълни в отговор на едно съобщение започва от класа на получателя. Ако в него няма подходящ метод, тогава се търси в класа – родител и т.н.

Ако съществуват методи с едно и също име нагоре в йерархията, казваме че изпълненият метод **подтиска** наследения метод.

# Въведение в ООП ...

---

## Основни характеристики на ООП (Alan Kay):

1. Всичко е *обект*.
2. Изчислението се извършва от обекти, комуникиращи помежду си, давайки заявки други обекти да извършат действия. Обектите комуникират чрез изпращане и получаване на *съобщения*. Съобщение е заявка за действие заедно с аргументите, необходими за изпълнението на задачата.
3. Всеки обект има собствена *памет*, която се състои от други обекти.

# Въведение в ООП ...

---

4. Всеки обект е *екземпляр* на *клас*. Класът представлява просто групиране на подобни обекти.
5. Класът е склад за *поведението*, асоциирано с обектите. Всички обекти, които са екземпляри на един и същ клас, могат да изпълняват едни и същи действия.
6. Класовете са организирани в дървовидна структура, наречена *наследствена йерархия*. Паметта и поведението, асоциирани с екземплярите на един клас, автоматично са достъпни до всеки клас по-надолу в тази дървовидна структура.

# Въведение в ООП ...

---

**ООП като следваща стъпка в еволюцията на езиците за програмиране**

- Еволюцията в програмирането – непрекъснатата борба със сложността
- Нелинейно поведение на сложността
- Примери



# Въведение в ООП ...

---

## Абстрактни механизми

### 1. Процедури

Процедурите и функциите – един от първите абстрактни механизми. Позволяват задачите да бъдат изпълнявани многократно или с леки вариации, да се съберат на едно място и да се използват многократно, вместо кодът да се дублира многократно.

Дават първата възможност за *скриване на информацията*. Другите програмисти не трябва да знаят детайлите по реализацията, а само интерфейса.

# Въведение в ООП ...

---

Пример

```
int datastack[100];
int datatop = 0;
void init()
{
    datatop = 0;
}
void push(int val)
{
    datastack[datatop++] = val;
}
```

# Въведение в ООП ...

---

```
int top()
{
    return datastack[datatop - 1];
}

int pop()
{
    return datastack[--datatop];
}

bool isEmpty()
{
    return datatop == 0;
}
```

Недостатъци

# Въведение в ООП ...

---

## 2. Блокове

```
begin
  var
    datastack : array [ 1 .. 100 ] of integer;
    datatop : integer;
  procedure init;
    ...
  procedure push(val : integer);
    ...
  function pop : integer;
    ...
  ...
end;
```

# Въведение в ООП ...

---

## 3. Модули

Модулите позволяват пространството на имената да се раздели на 2 части – public и private.

David Parnas дава следните 2 принципа

- На евентуалния потребител трябва да му се даде цялата информация нужна за да използва модула и нищо повече.
- На разработчика – цялата нужна информация за да реализира модула и нищо повече.

При военните – ако няма нужда да знаеш някаква информация, то нямаш достъп до нея.

Недостатък – само 1 стек (само 1 комплексно число)

# Въведение в ООП ...

---

## 4. Абстрактни типове данни

АТД са типове данни, дефинирани от програмиста, които могат да се обработват по подобен начин на системно дефинираните типове.

- Имат множество от допустими стойности
- Примитивни операции над тези стойности

Модулите – имплементационна техника за АТД

# Въведение в ООП ...

---

За да построим АТД трябва да можем:

1. Да експортираме дефиниция на тип
2. Да предоставим множество от операции за обработка на екземпляри на типа
3. Да защитим данните, асоциирани с типа, така че да могат да се обработват само чрез предоставените процедури
4. Да създаваме много екземпляри на типа

Модулите – само 2 и 3

Пакетите в CLU и ADA – опит в тази насока

# Въведение в ООП ...

---

## 5. ооп

Техниките на ООП решават 1-4 за АТД и добавя няколко нови идеи към понятието АТД

- Предаване на съобщения
- Интерпретация на съобщения
- Наследяване