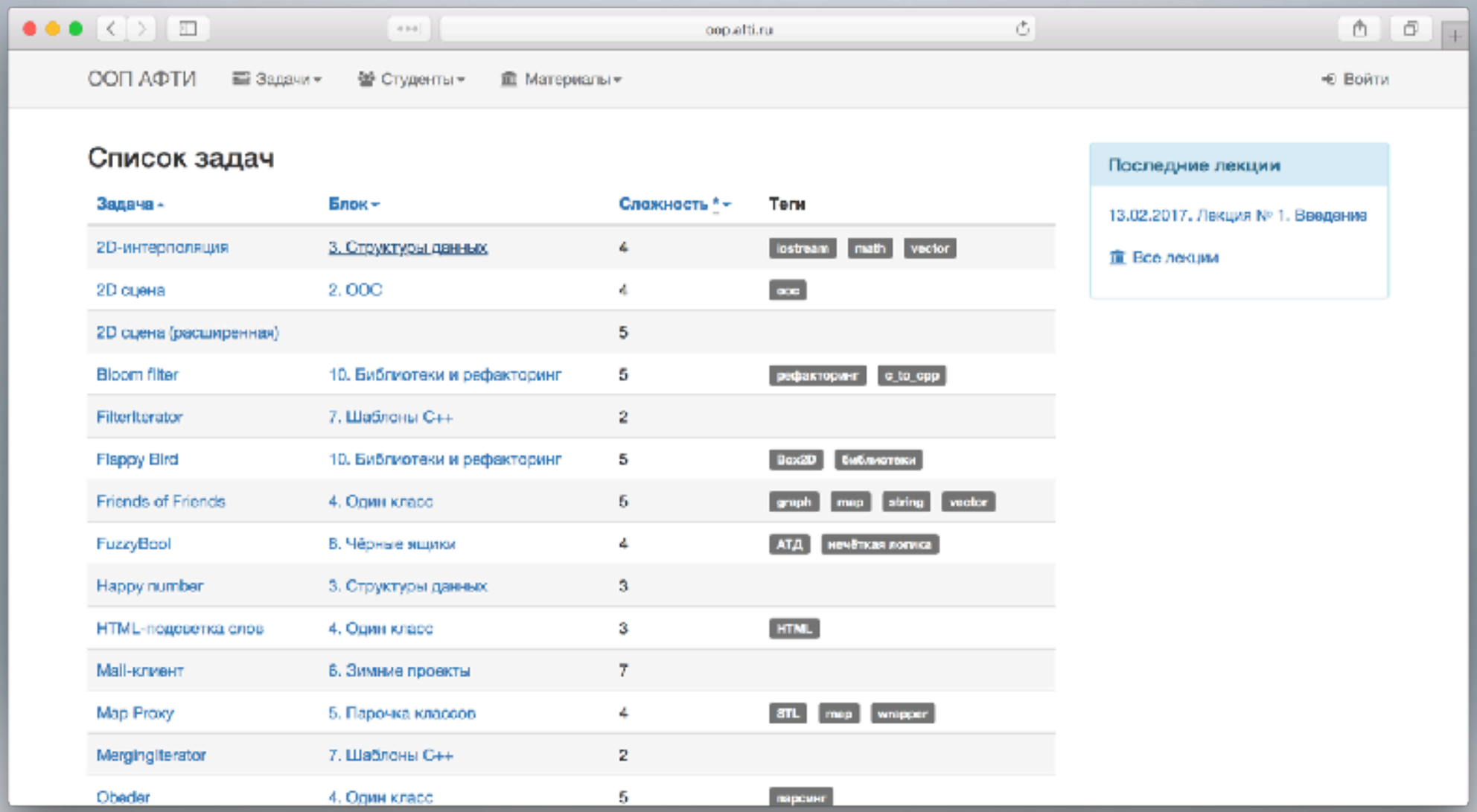
ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Лекция № | / **02**
12.02.2018 г.



http://oop.afti.ru

ЧЕТЫРЕ КИТА ООП

Абстракция

Полиморфизм

Наследование

Инкапсуляция

ИНКАПСУЛЯЦИЯ

- Данные сокрыты.
- Детали реализации сокрыты.
- Наружу торчит только интерфейс.

НАСЛЕДОВАНИЕ

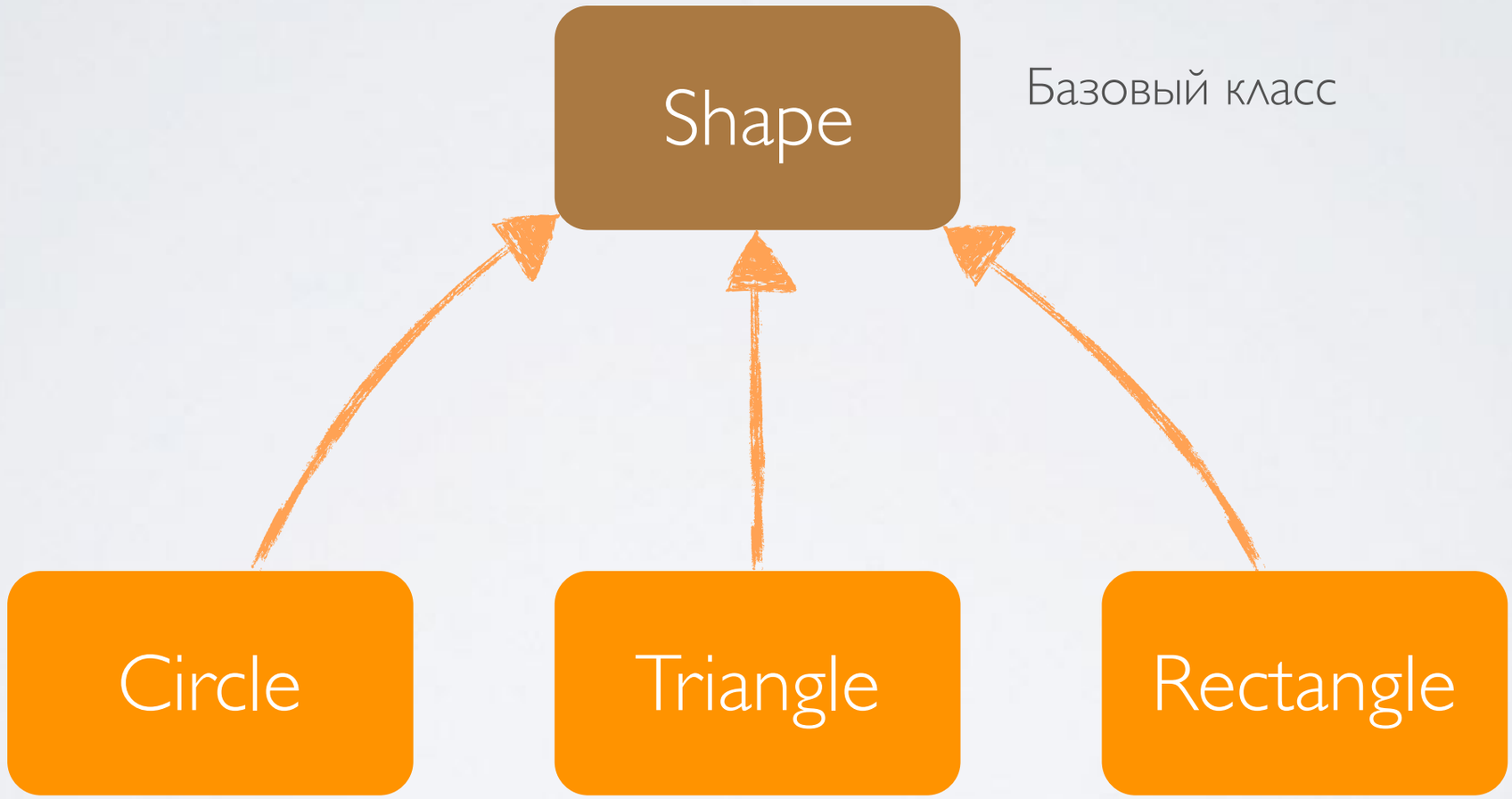
- «**Этот** объект ведет себя так же, как **тот**, только немного по-другому»

- Поведение класса **A** наследуется от класса **B** (с изменениями).

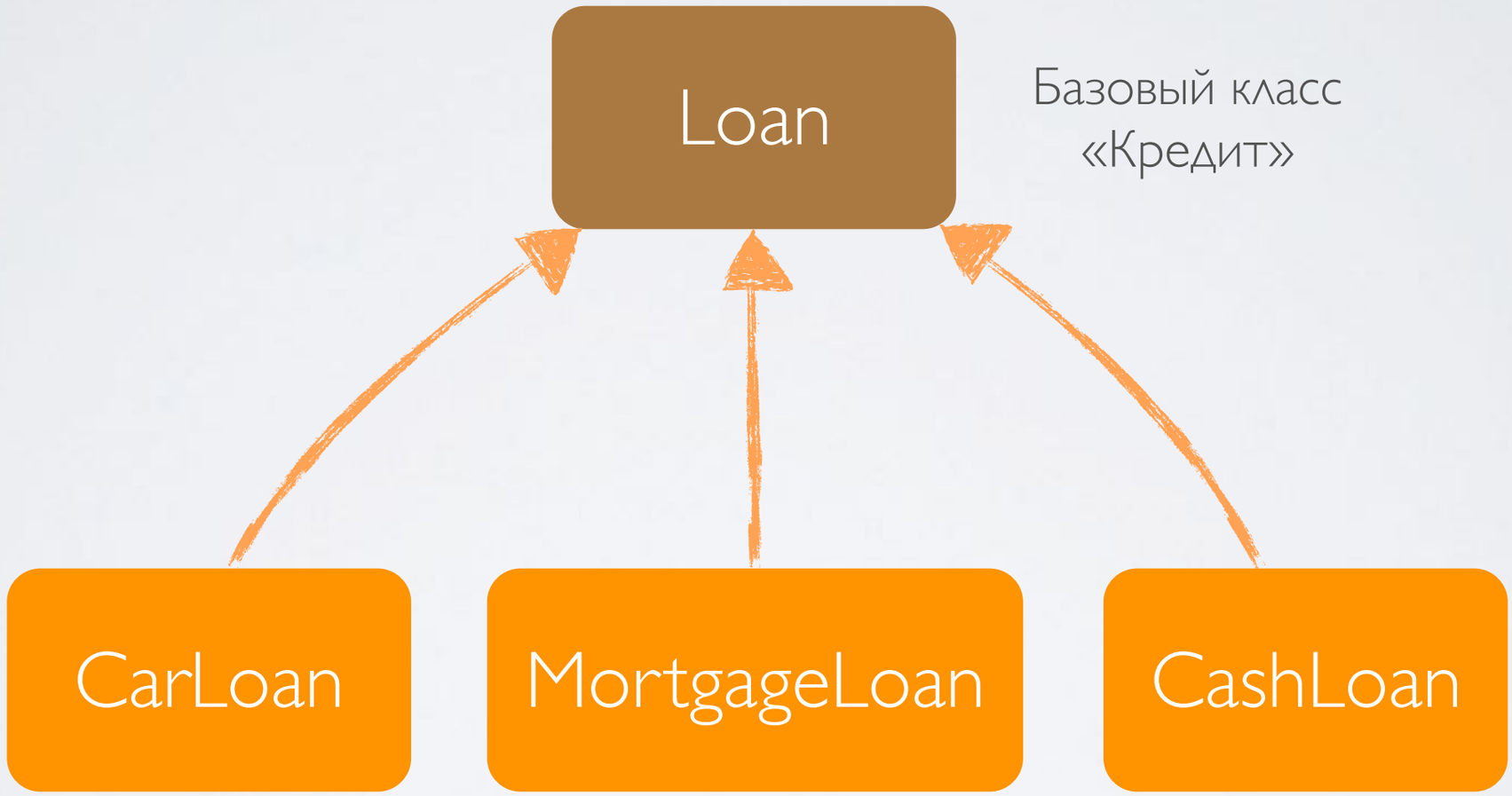
- **B** — базовый класс (суперкласс).

- **A** — производный класс (подкласс).

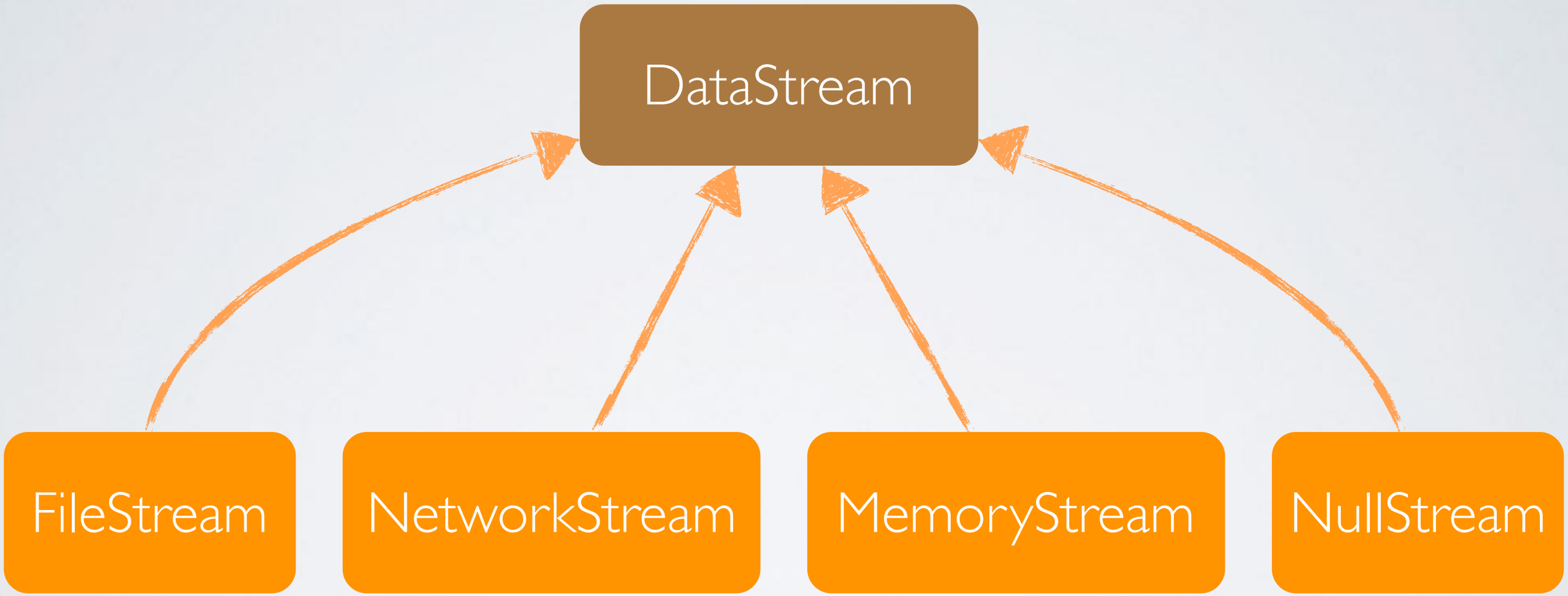
ГЕОМЕТРИЧЕСКИЕ ФИГУРЫ



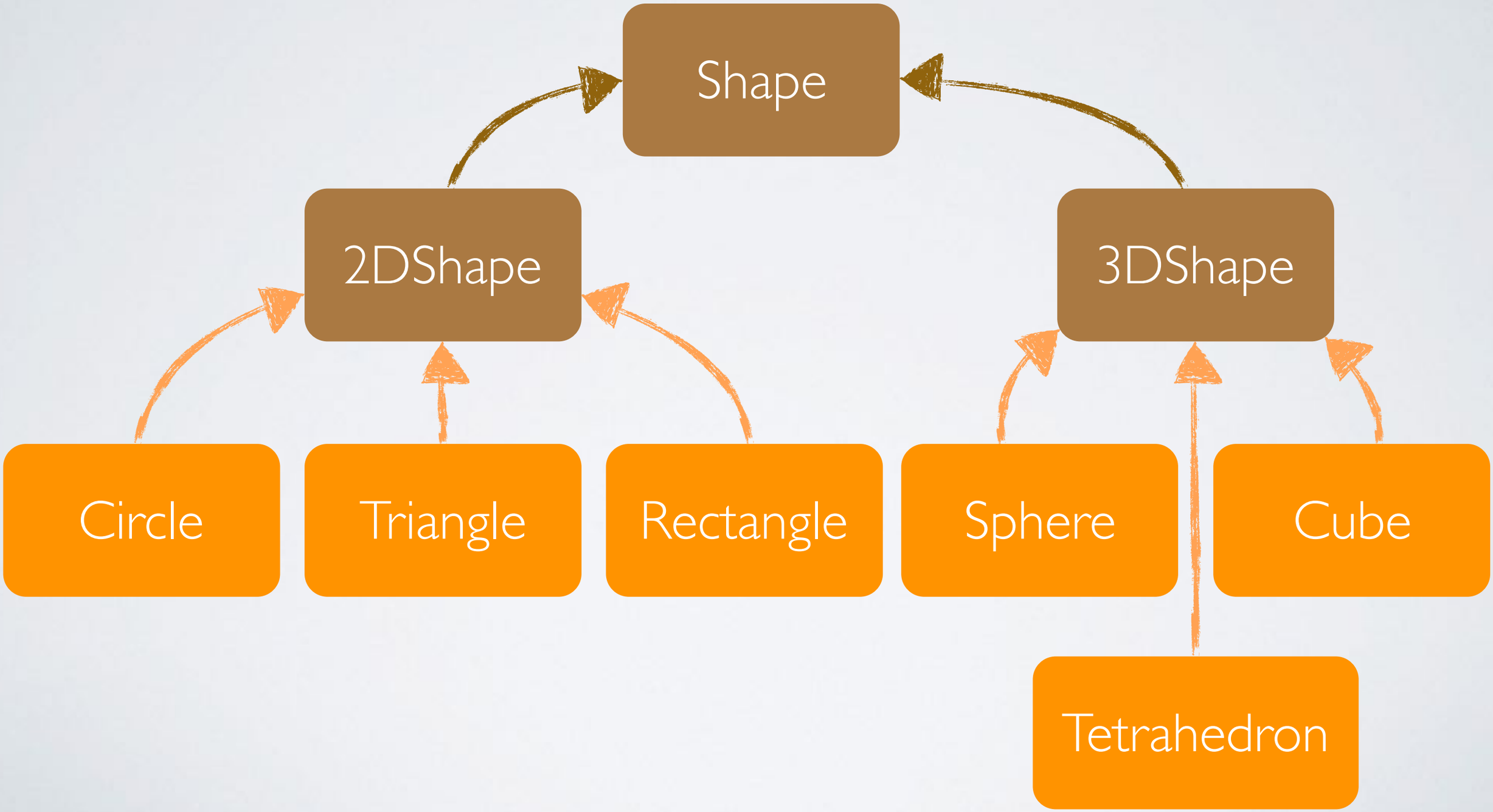
БАНКОВСКИЕ КРЕДИТЫ



ПЕРЕДАЧА ДАННЫХ



ИЕРАРХИЯ НАСЛЕДОВАНИЯ



ПРИНЦИПЫ НАСЛЕДОВАНИЯ

- Принцип «**A** является **B**» (*is-a*). Круг (**Circle**) является плоской фигурой (**2DShape**), которая является просто фигурой (**Shape**). Обратное неверно!

- Нельзя путать с «**A** содержит **B**» (*has-a*). Это не наследование, а композиция!

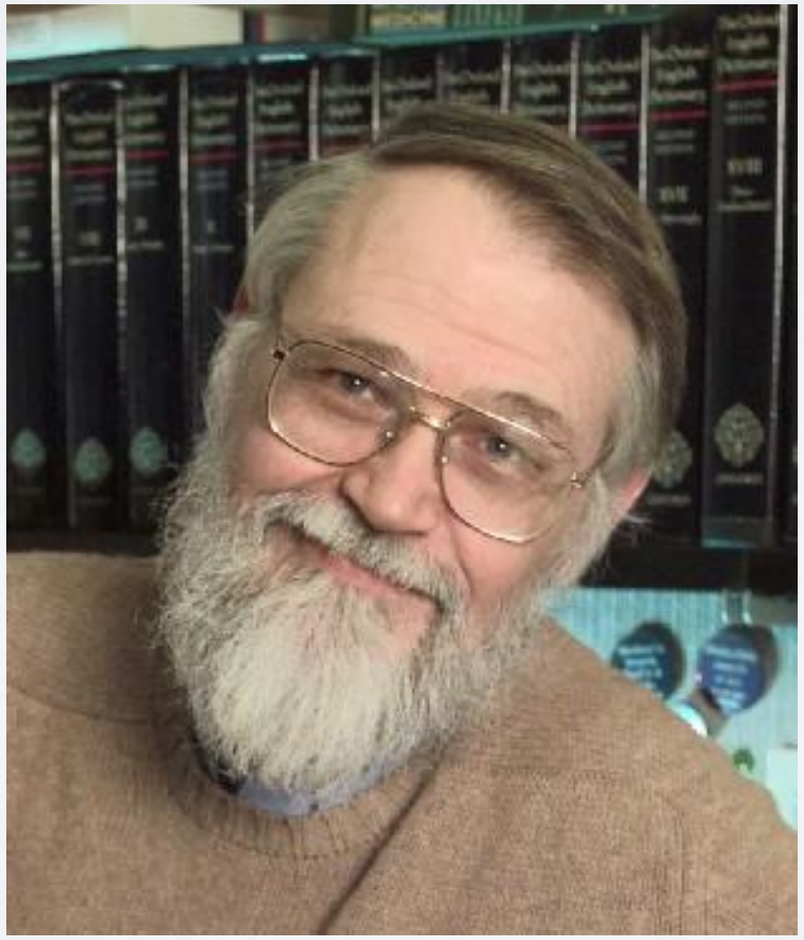
НЕКОТОРЫЕ ПРОБЛЕМЫ

- Square **is a** Rectangle? (Circle **is a** Ellipse?)
- Будет ли удачным ходом сделать класс **Square** наследником **Rectangle**?
- Ответ: **нет**, если объект может изменяться. Методы **stretchWidth** и **stretchHeight** не годятся для **Square**.
- Ответ: **да**, если объект не будет изменяться после создания.

ValueObject

OBJECT-ORIENTED C (OOC)

- Указатели на функции
- Функции с переменным количеством параметров



УКАЗАТЕЛИ НА ФУНКЦИИ

```c
#include <stdio.h>

void main()
{
    int (*func_ptr)();                          /* Объявление указателя */
    func_ptr = printf;                          /* Присваивание */
    (*func_ptr)("Printf is here!\n");           /* Вызов функции */
}
```

```c
void qsort(void *base, size_t nitems,
           size_t item_size,
           int (*compar)(void *p1, void *p2));
```

```c
void *bsearch(void *key, void *base,
           size_t nitems, size_t item_size,
           int (*compar)(void *p1, void *p2));
```

- **base** — указатель на начало массива.
- **nitems** — количество сортируемых элементов.
- **item_size** — размер одного элемента в байтах.
- **compar** — функция сравнения. Возвращает **−1**, **0** или **1**. **p1** и **p2** — указатели на сравниваемые элементы.

```c
#include <stdio.h>
#include <stdlib.h>

char *names[] = {
    "Vasya",
    "Petya",
    "Semyon Semenych"
};


int name_compare(void *p1, void *p2) {
    char *n1 = *(char **)p1;
    char *n2 = *(char **)p2;
    return strcmp(n1, n2);
}

int main()
{
    int i;
    qsort(names,                                    // void *base
          sizeof (names) / sizeof (char *),         // size_t n_items
          sizeof (char *),                          // size_t item_size
          name_compare);                            // int (*compar)(void *p1, void *p2)

    for (i = 0; i < 3; ++i)
        puts(names[i]);
}
```

ПАМЯТЬ В С-ПРОГРАММЕ

Скомпилированный код

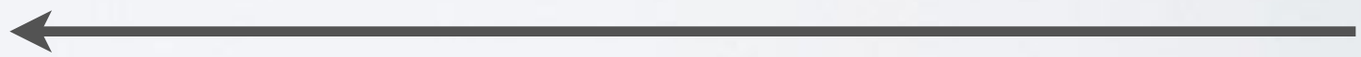
Только чтение
Размер фиксирован

Глобальная память (вкл. static)

Чтение / запись
Размер фиксирован
Время жизни переменных: максимально

Свободный стек

Занятый стек



Чтение / запись
Размер динамический, обычно ограничен сверху
(задается в свойствах линкера)
Время жизни переменных = время жизни фрейма

СТЕКОВЫЕ ФРЕЙМЫ

```
void MyFunction3(int x, int y, int z)
{
  int a, int b, int c;
  a = 10;
  ...
  return;
}
```

```
_MyFunction3:
  push ebp
  mov ebp, esp
  sub esp, 12 ; sizeof(a) + sizeof(b) + sizeof(c)
  ; x = [ebp + 8], y = [ebp + 12], z = [ebp + 16]
  ; a = [ebp - 4] = [esp + 8], b = [ebp - 8] = [esp + 4],
  ; c = [ebp - 12] = [esp]
  mov [ebp -  4], 10
  ...
  mov esp, ebp
  pop ebp
  ret 12 ; sizeof(x) + sizeof(y) + sizeof(z)
```

ebp+16	Аргумент **z**
ebp+12	Аргумент **y**
ebp+8	Аргумент **x**
ebp+4	Адрес возврата
ebp	Старый **ebp**
ebp−4	Переменная **a**
ebp−8	Переменная **b**
esp = ebp−12	Переменная **c**
esp + 4	Свободная ячейка
....	Свободная ячейка

```
MyFunction3(10, 5, 2);
```

→

```
push 2
push 5
push 10
call _MyFunction3
```

ПЕРЕМЕННОЕ ЧИСЛО АРГУМЕНТОВ

```
#include <stdarg.h>

double average(int n, ...)
{
    va_list myList;
    va_start(myList, n);

    // ...

    va_end(myList);
}

average(5, 1.0, 2.0, 3.0, 4.0, 5.0);
```

ebp+44	Аргумент 5.0
ebp+36	Аргумент 4.0
ebp+28	Аргумент 3.0
ebp+20	Аргумент 2.0
ebp+12	Аргумент 1.0
ebp+8	Аргумент n
ebp+4	Адрес возврата
ebp	Старый ebp
ebp−4	Переменная myList
....	

```c
#include <stdarg.h>

double average(int n, ...)
{
    va_list myList;
    va_start(myList, n);                          // Инициализация

    int numbersAdded = 0;
    double sum = 0;

    while (numbersAdded < n) {
        double number = va_arg(myList, double); // Очередное число
        sum += number;
        numbersAdded++;
    }

    va_end(myList);

    return sum / numbersAdded;                    // Вычисляем среднее
}
```

ОБЪЕКТНАЯ СИСТЕМА НА С

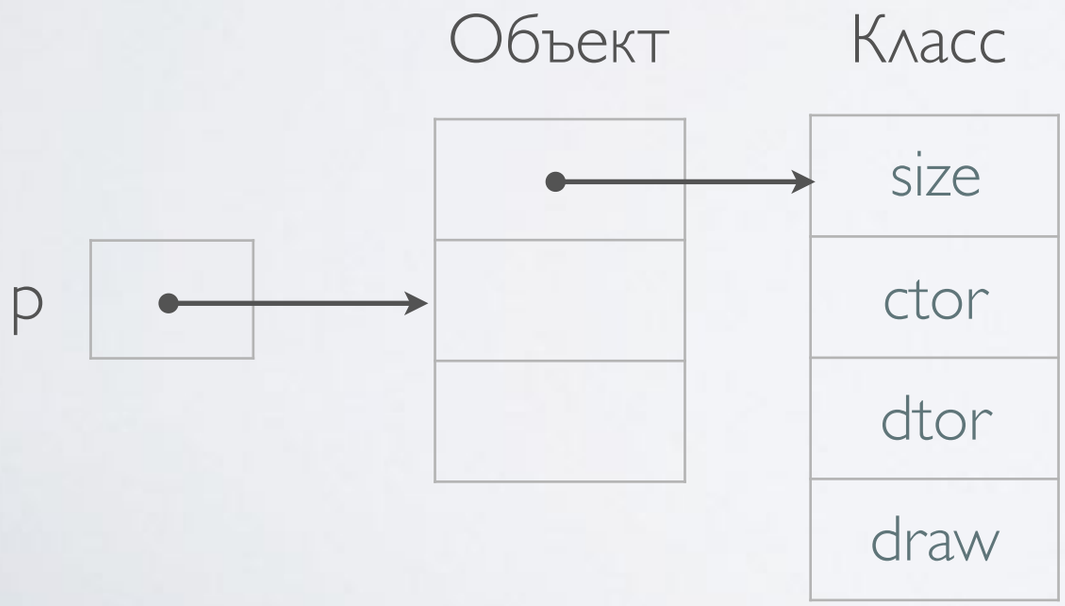
```
// new.h
struct Class {
    size_t size;
    void *(*ctor)(void *self, va_list *app);
    void *(*dtor)(void *self);
    void (*draw)(const void *self);
};

void *new(const void *class, ...);
void delete(void *item);
void draw(const void *self);
```

```
// new.c
void *new(const void *_class, ...)
{
    const struct Class *class = _class;
    void *p = calloc(1, class->size);
    assert(p);
    *(const struct Class **)p = class;

    if (class->ctor) {
        va_list ap;
        va_start(ap, _class);
        p = class->ctor(p, &ap);
        va_end(ap);
    }
    return p;
}

// продолжение следует ...
```



struct Class

```c
// new.c
void delete(void *self)
{
    const struct Class **cp = self;

    if (self && *cp && (*cp)->dtor)
        self = (*cp)->dtor(self);

    free(self);
}
```

```c
// new.h
struct Class {
    size_t size;
    void *(*ctor)(void *self,
                  va_list *app);
    void *(*dtor)(void *self);
    void (*draw)(const void *self);
};

void *new(const void *class, ...);

void delete(void *item);

void draw(const void *self);
```

```c
void draw(const void *self)
{
    const struct Class *const *cp = self;
    assert(self && *cp && (*cp)->draw);
    (*cp)->draw(self);
}
```

```c
// point.c

void move(void *_self, int dx, int dy) {
    struct Point *self = _self;
    self->x += dx;
    self->y += dy;
}

static void *Point_ctor(void *_self, va_list *app) {
    struct Point *self = _self;
    self->x = va_arg(*app, int);
    self->y = va_arg(*app, int);
    return self;
}

static void Point_draw(const void *_self) {
    const struct Point *self = _self;

    printf("\".\" at %d,%d\n", self->x, self->y);
}

static const struct Class _Point = {
    sizeof(struct Point),       // size
    Point_ctor,                 // ctor
    0,                          // dtor
    Point_draw                  // draw
};

const void *Point = &_Point;
```

Нединамический метод

Конструктор Point

```c
// point.h
struct Point {
    const void *class;
    int x, y; /* координаты */
};

extern const void *Point;

void move(void *_self,
          int dx,
          int dy);
```

Деструктора нет

```c
// circle.h
struct Circle {
    const struct Point _;
    int rad;
};
```

«Круг — это такая жирная точка
с радиусом **rad**»

```c
// circle.c
static void *Circle_ctor(void *_self, va_list *app) {
    struct Circle *self = ((const struct Class *)Point)->ctor(_self, app);
    self->rad = va_arg(*app, int);
    return self;
}

#define x(p) (((const struct Point *)(p)) -> x)
#define y(p) (((const struct Point *)(p)) -> y)

static void Circle_draw(const void * _self) {
    const struct Circle *self = _self;
    printf("circle at %d,%d rad %d\n", x(self), y(self), self->rad);
}

static const struct Class _Circle = {
    sizeof(struct Circle), Circle_ctor, 0, Circle_draw
};

const void *Circle = &_Circle;
```

```c
#include "point.h"
#include "circle.h"
#include "new.h"

int main(int argc, char **argv)
{
    void *p;
    while (*++argv) {
        switch (**argv) {
        case 'p':
            p = new(Point, 1, 2);
            break;
        case 'c':
            p = new(Circle, 1, 2, 3);
            break;
        default:
            continue;
        }

        draw(p);
        move(p, 10, 20);
        draw(p);
        delete(p);
    }

    return 0;
}
```

```
$ circles p c
"." at 1,2
"." at 11,22
circle at 1,2 rad 3
circle at 11,22 rad 3
```

КОНЕЦ ВТОРОЙ ЛЕКЦИИ