Понять ООП
и начать применять

Понять нельзя применять

Где запятую поставим?

Понять нельзя применять

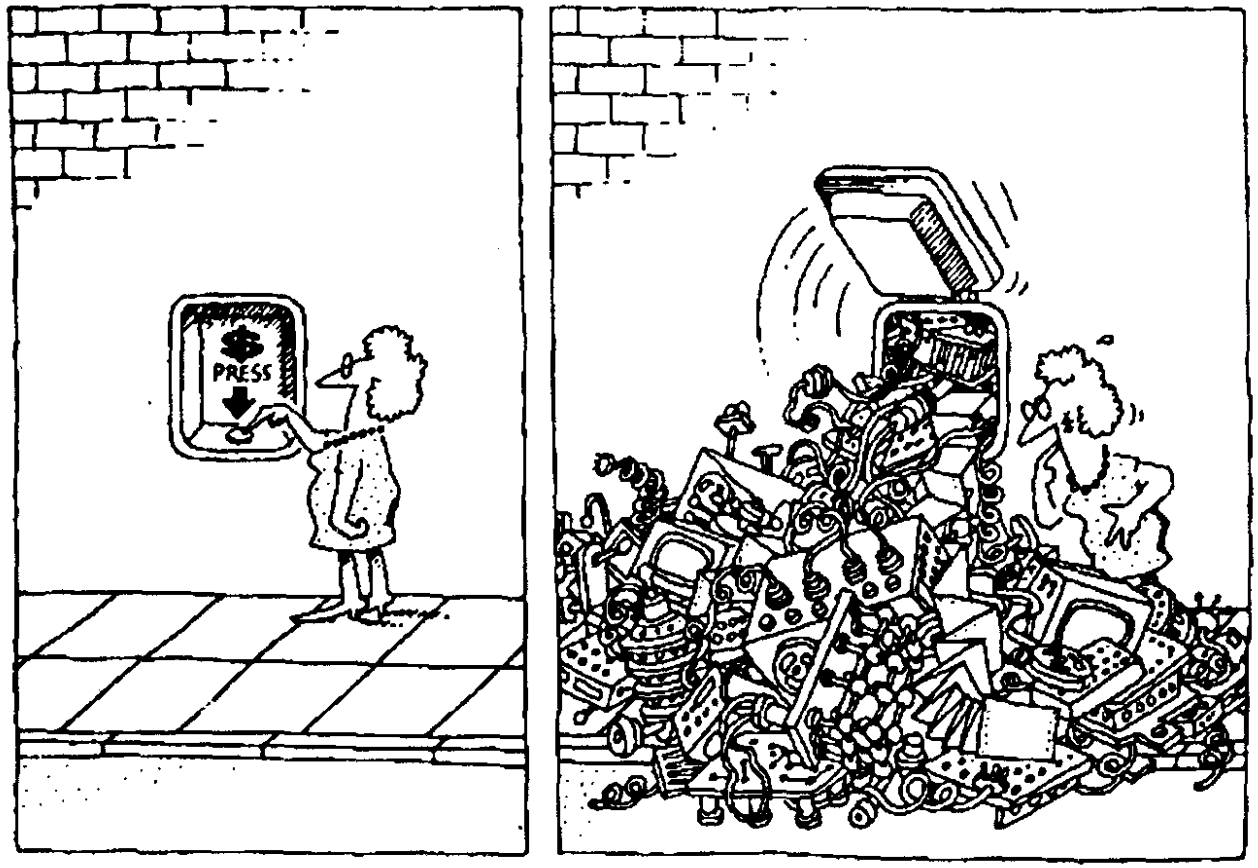
Это не про ООП!

Зачем придумали ООП?

Чтобы справиться со сложностью больших программных систем

Зачем придумали ООП?

Создать иллюзию простоты



Зачем придумали ООП?

Создать иллюзию простоты.

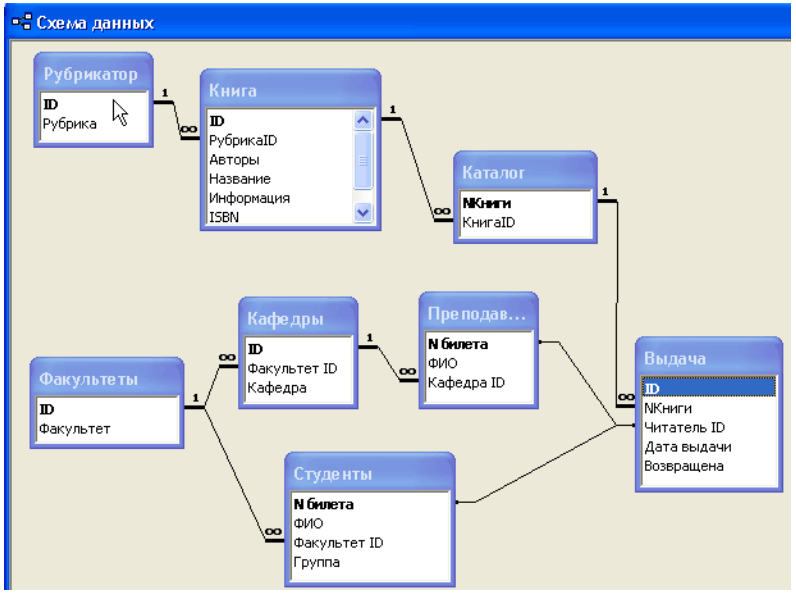
Для кого?

Для пользователей
Для разработчиков

ООП и БД. Что общего?

Домашняя библиотека

Номер	Автор	Название	Год	Полка
0001	Беляев А.Р.	Человек-амфибия	1987	5
0002	Кервуд Д.	Бродяги севера	1991	7
0003	Тургенев И.С.	Повести и рассказы	1982	1
0004	Олеша Ю.К.	Избранное	1987	5
0005	Беляев А.Р.	Звезда КЭЦ	1990	5
0006	Тынянов Ю.Н.	Кюхля	1979	1
0007	Толстой Л.Н.	Повести и рассказы	1986	1
0008	Беляев А.Р.	Избранное	1994	7



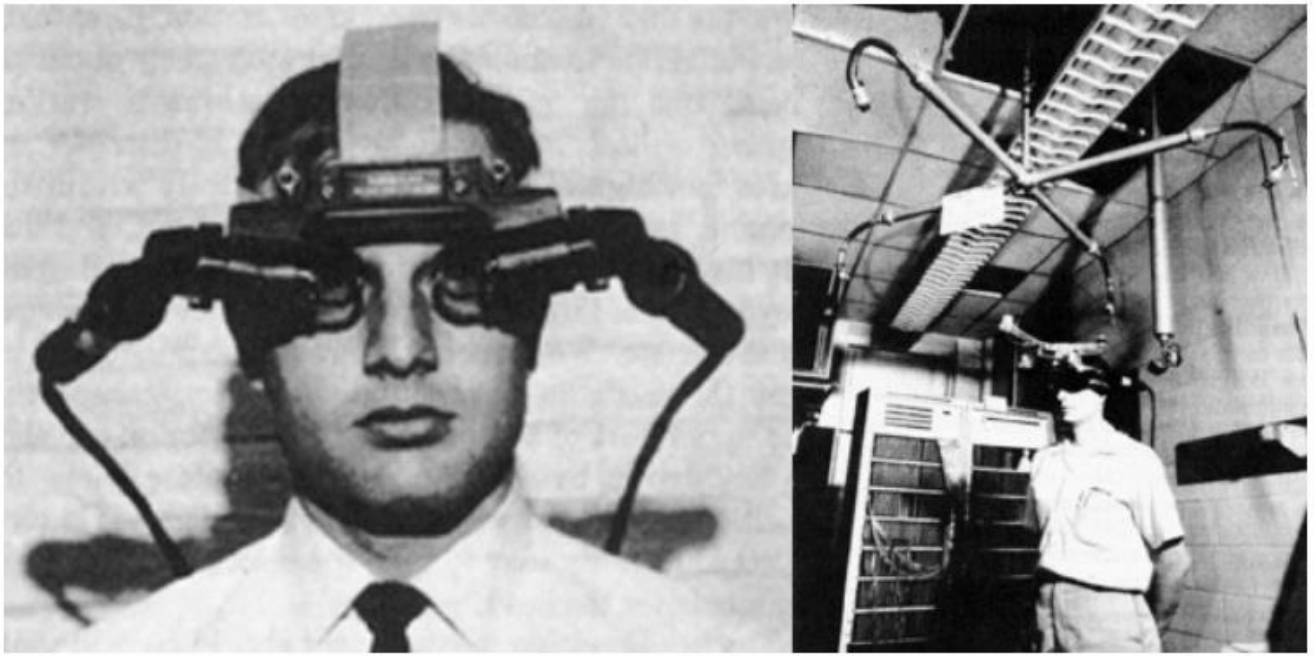
Автомобиль
Корпус
Двигатель
4 колеса
Фары
…
Двигаться вперед()
Набирать скорость()
Снижать скорость()
Двигаться задним ходом()
Поворачивать()
…

И там, и там есть некие сущности с некими характеристиками.

ООП и Сазерленд

Первым компьютерным решением, воплотившим в себе объектный подход, стал программно-аппаратный графический Планшет (Sketchpad), созданный Сазерлендом между 1961 и 1962 годами и описанный в 1963 году.

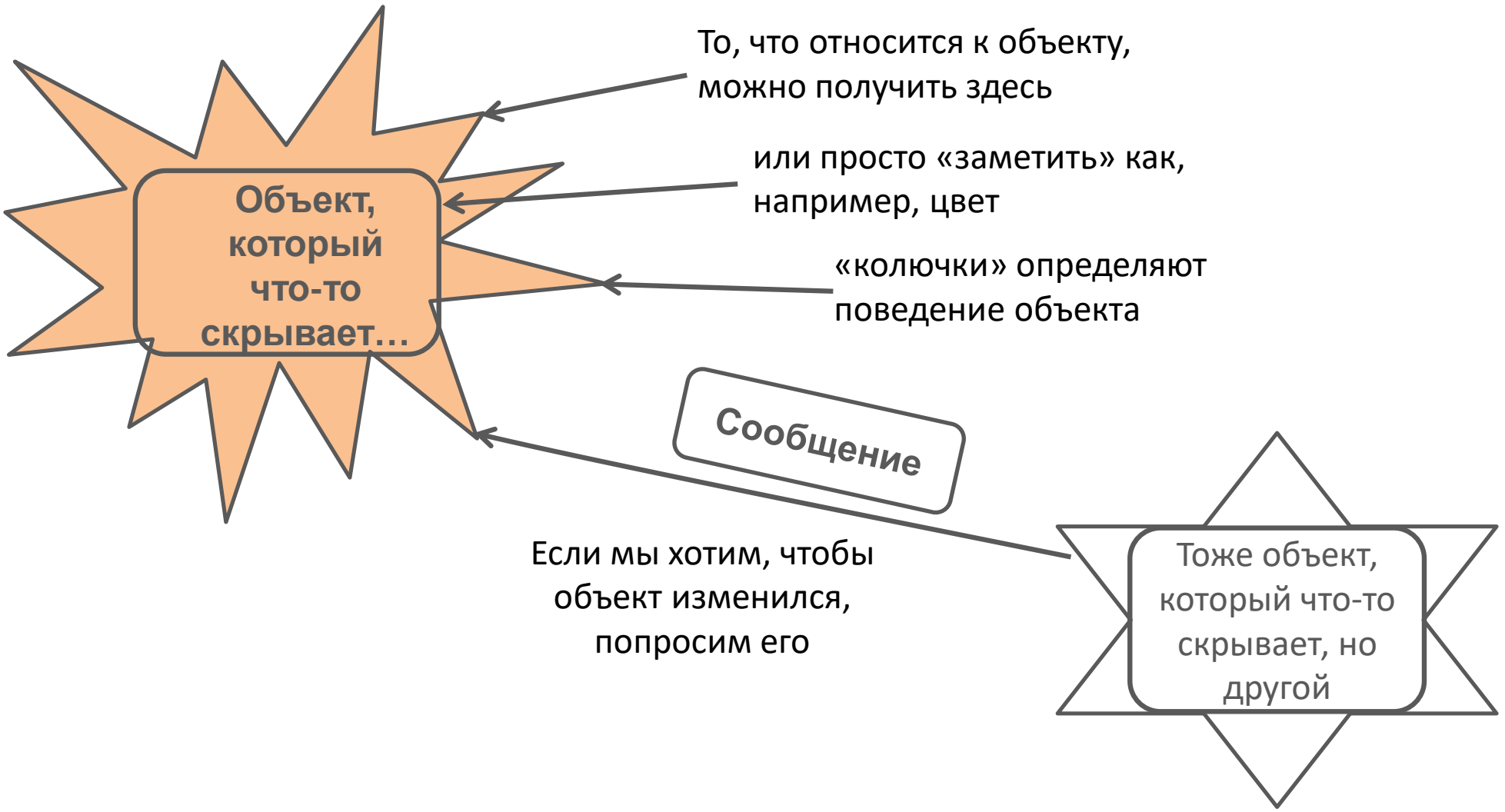
Сазерленд и VR



«Дамоклов меч» Сазерленда и Спрула

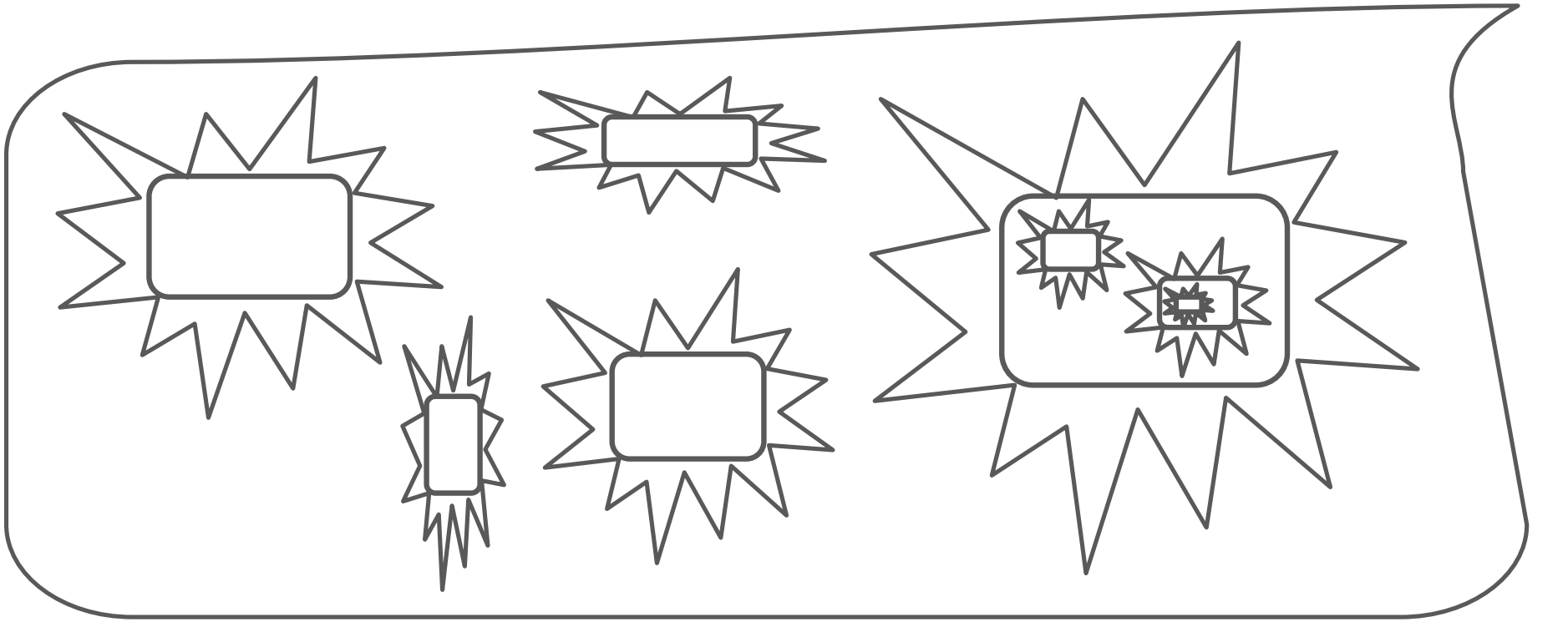
Так задумывалось ООП

Алан Кей: «Всё является объектом».



Программа с точки зрения ООП

Набор **объектов**, имеющих **состояние и поведение**.
Объекты взаимодействуют посредством **сообщений**.



Как сделать так, чтобы это было устойчивым и управляемым?

Основные принципы ООП по Гради Бучу

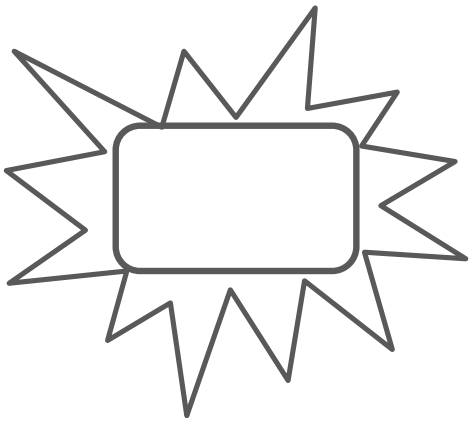
- **Абстрагирование**
- **Ограничение доступа**
- **Модульность**
- **Иерархия**

Устойчивость и управляемость простыми словами

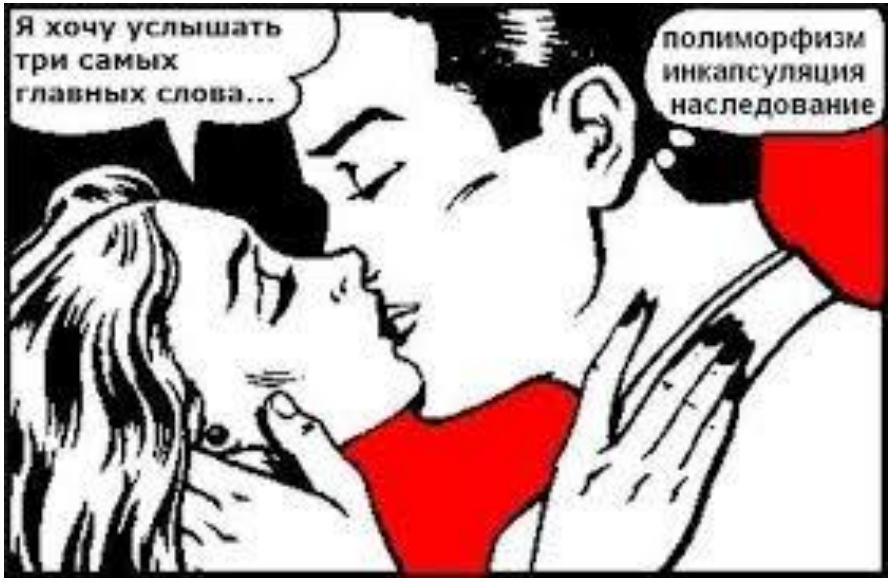
1. Ответственность объектов четко разделяется: за каждое действие отвечает определённый объект. И он один. Снаружи один.

2. Межобъектное взаимодействие определяется однозначно.
Через **интерфейсы**.

3. Внутренняя структура объекта полностью изолирована от внешней среды (**инкапсулирована**).



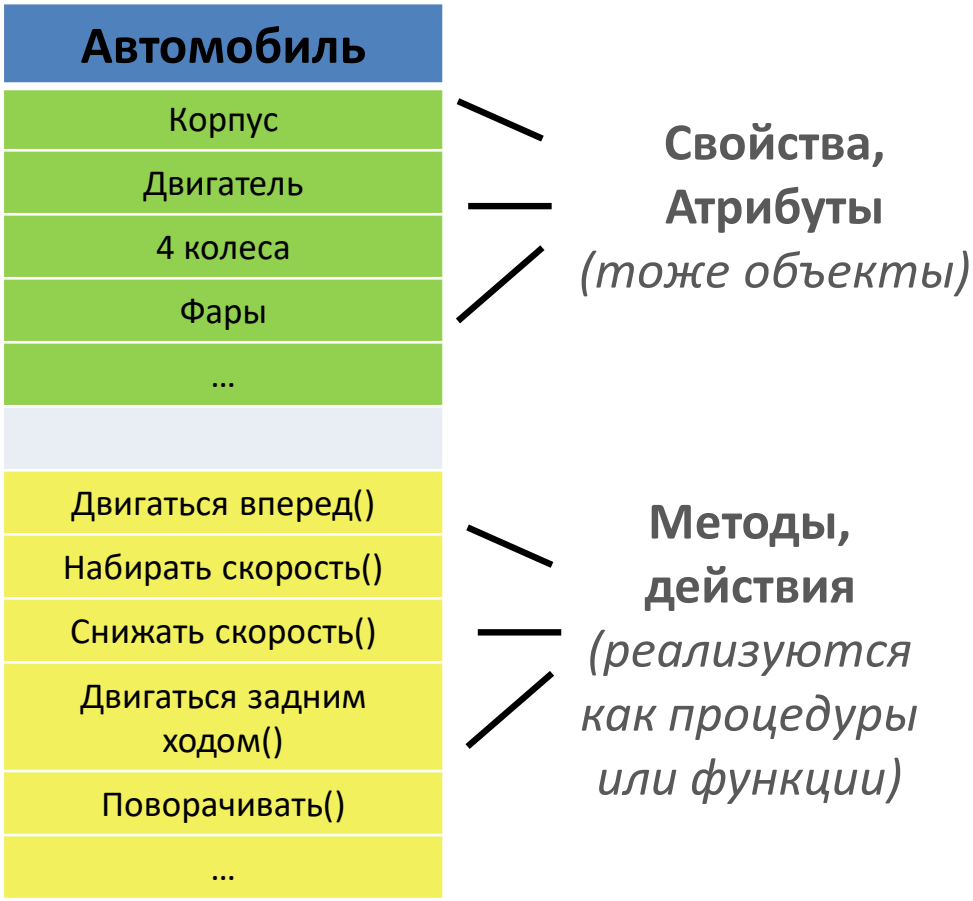
Еще умные слова про ООП



И еще умные слова:

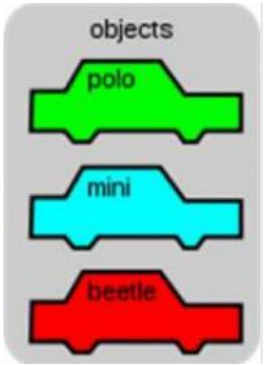
абстракция

композиция

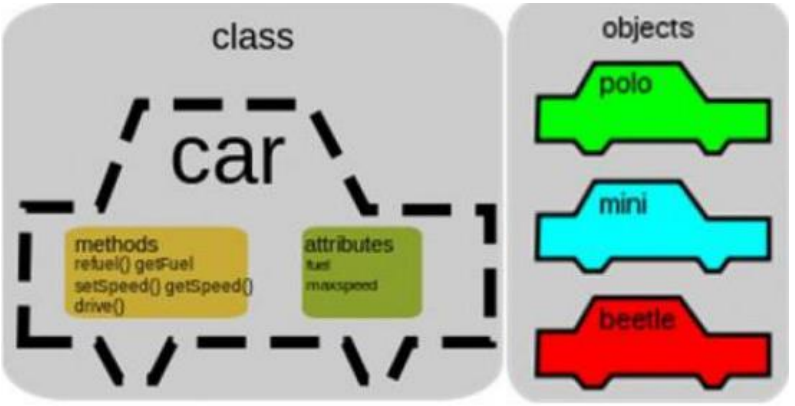


Про классы-то забыли!

Объекты могут быть однотипными.



Почему бы информацию о них не сгруппировать?



Класс – это способ описания объекта. Он определяет:

- состояние объекта (через свойства),
- поведение объекта, зависящее от этого состояния,
- и правила взаимодействия с этим объектом.

С точки зрения структуры программы класс - сложный тип данных.

Абстрагирование

Выделяем значимые характеристики объекта и сними работаем, *незначимые скрываем*.

Автомобиль

руль,
педали,
указатель поворота,
…
химический состав краски кузова ,
удельная теплоёмкость лампочки подсветки номеров
…

Методы класса

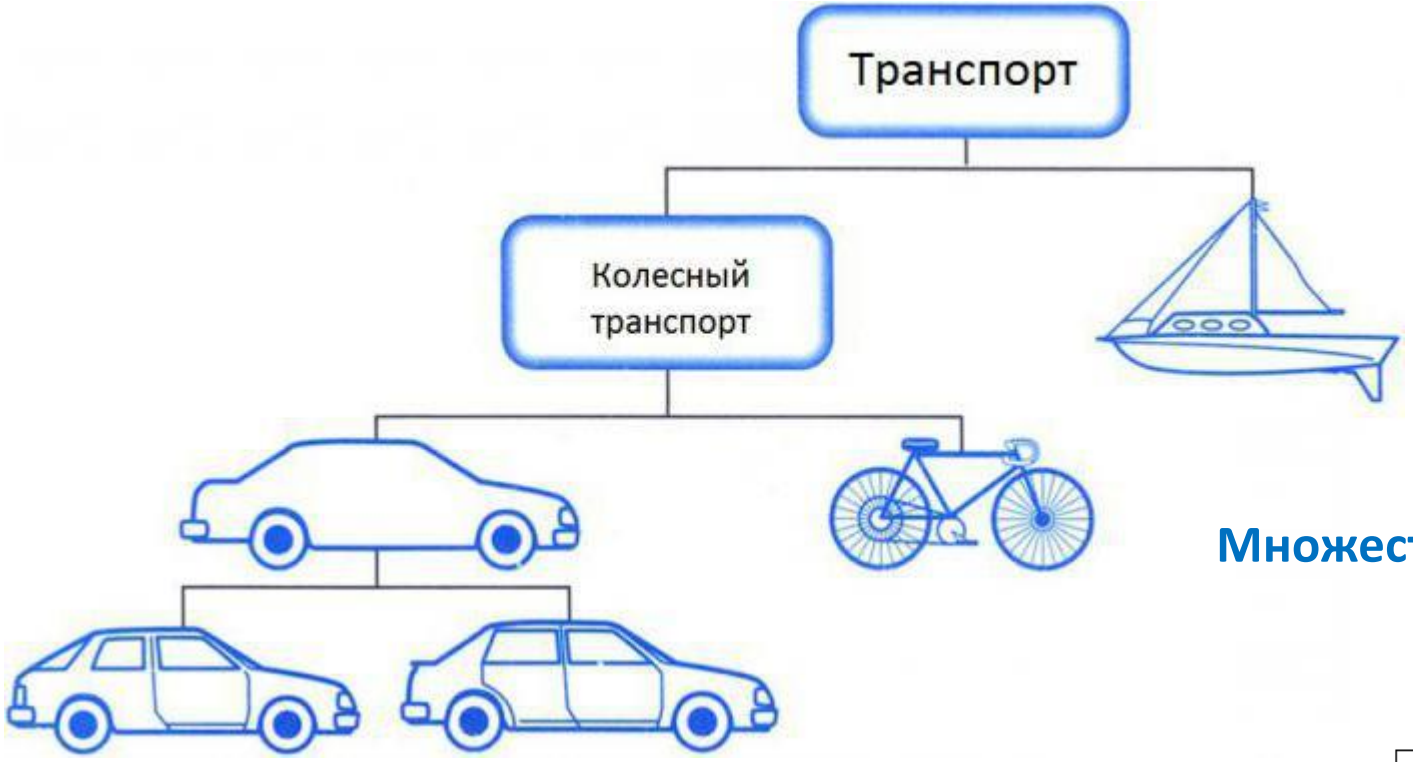
private

protected

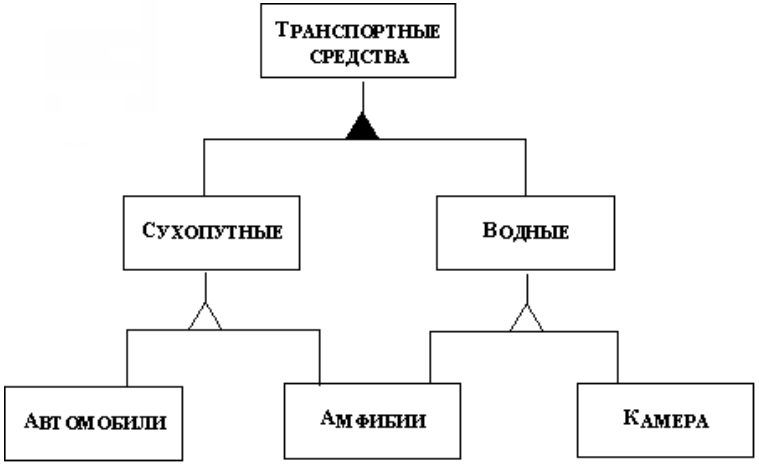
public

Наследование

Глупо, чтобы изобрести мопед, снова изобретать велосипед.



Множественное наследование



Полиморфизм

Полиморфизм – это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Полиморфизм. Еще пример

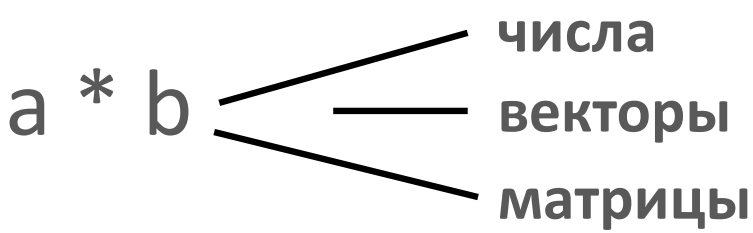
Складываем

Числа: 5 + 6

Векторы: V (1, 2, 3) + V (1, 2, 3)

Матрицы…

Умножаем



Снаружи это выглядит одинаково, а то, как сделано внутри - скрыто

Полиморфизм. Еще пример

Полиморфизм

(C) Класс Animal

```
void voice() {
    System.out.print('Голос');
};
```

extends extends extends

(C) Класс Cat

```
@Override
void voice() {
    System.out.print('Мяу!');
};
```

(C) Класс Dog

```
@Override
void voice() {
    System.out.print('Гав!');
};
```

(C) Класс Cow

```
@Override
void voice() {
    System.out.print('Муу!');
};
```

C# - объектно-ориентированный

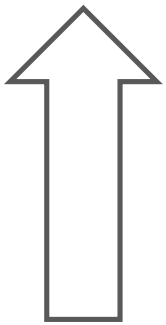
Все, что выделаете, должно быть разложено на объекты, классы, свойства, методы + наследование, инкапсуляция, полиморфизм.

```
ссылок: 0
public class MyClass
{

    // СВОЙСТВА
    public int a;
    public double b;

    // МЕТОДЫ
    ссылок: 0
    void Start()...

    ссылок: 0
    void Do ()...

    ссылок: 0
    void Finish()...
}
```

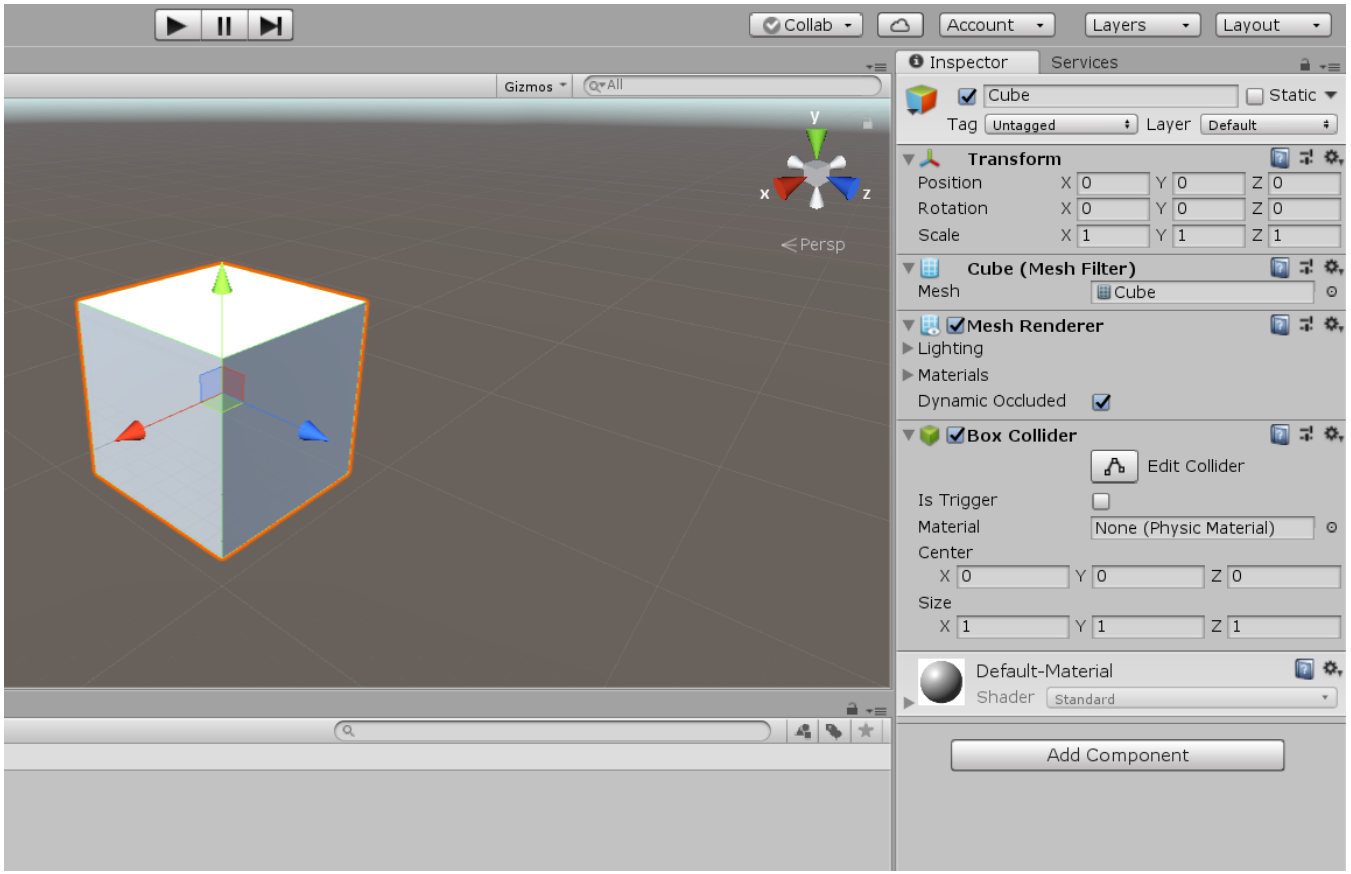
```
ссылок: 0
public class MyClass
{

    // СВОЙСТВА
    public int a;
    public double b;

    // МЕТОДЫ
    ссылок: 0
    void Start()
    {
        // тут пишем, что делаем
    }

    ссылок: 0
    void Do ()
    {
        // тут пишем, что делаем
    }

    ссылок: 0
    void Finish()
    {
        // тут пишем, что делаем
    }
}
```

Unity и ООП

Компонентно-ориентированное программирование

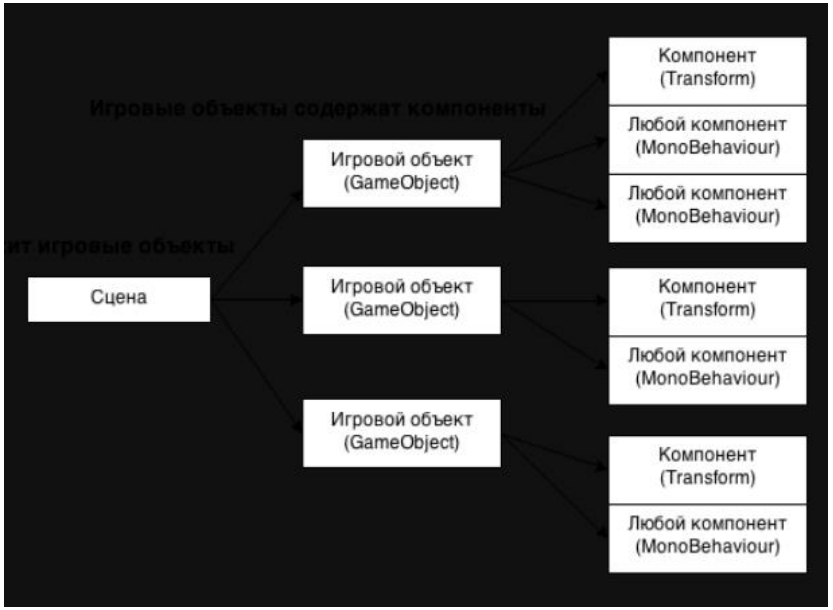


ООП

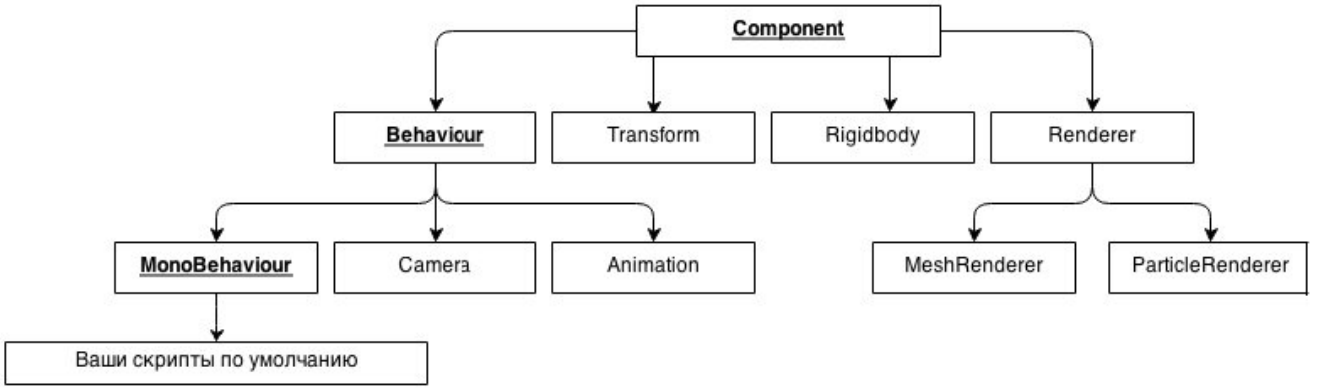


Источник: https://devtribe.ru/p/unity/components

Компонентно-ориентированное программирование (в Unity)



Иерархия наследования стандартных компонентов.



Namespace

Пространство имён — некоторое множество каким-либо образом взаимосвязанных имён или терминов. Во избежание путаницы, именам в одном пространстве имён не дают более одного значения.

Хороший пример:

А пространстве имён улиц любого города названия улиц, как правило, не повторяются.

А про почитать?

Тут очень просто и про машинки:
часть 1: https://habr.com/ru/post/87119/
часть 2: https://habr.com/ru/post/87205/

Тут кратко и для тех, кто в теме:
https://tproger.ru/translations/oop-principles-cheatsheet/

Это – если, очень хочется заморочиться:
много против: https://habr.com/ru/post/147927/
много за : https://habr.com/ru/post/149096/

Тут – если хочется фактов:
история ООП, как она есть, с именами и годами:
http://chernykh.net/content/view/994/1077/

Хмм… ООП вообще не про это:
https://habr.com/ru/company/ruvds/blog/428582/

Тут, если хочется что-то от гуру почитать:
Гради Буч «Объектно-ориентированный анализ и проектирование с примерами приложений»

Контакты

Мазнина Юлия,
VR/AR-квантум
детский технопарк «Кванториум» Магнитогорск

сот. тел. 8-912-807-18-84
Telegram / Viber
e-mail: maznina81@yandex.ru, maznina.ckm@gmail.com
VKontakte: https://vk.com/ymaznina
skype: maznina_ua