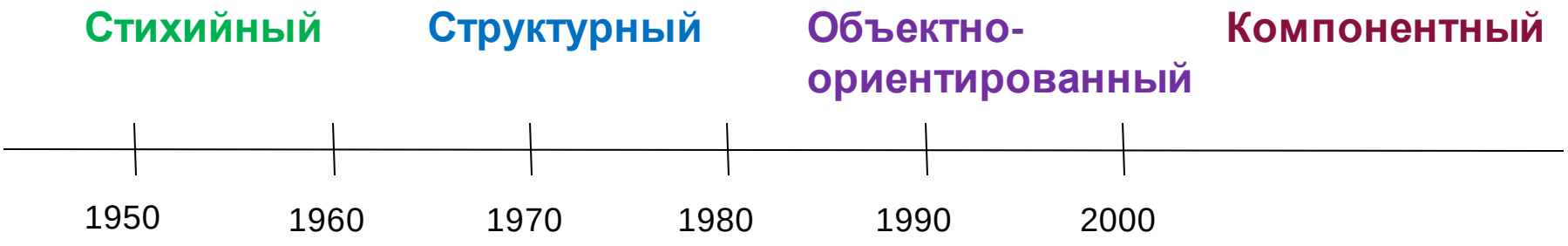
Лекция 5

Введение в объектно-ориентированное программирование

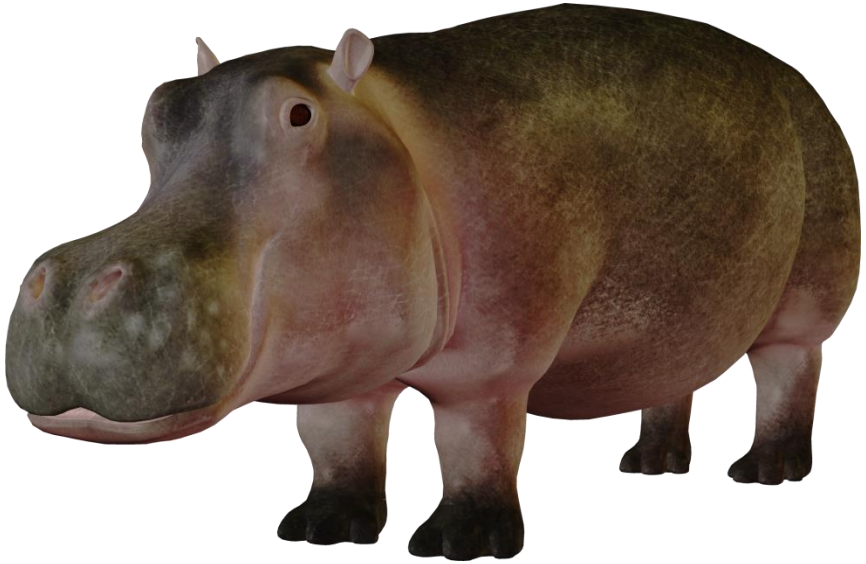
Подходы к программированию



Цель:

Эффективное производство ПО требуемого качества

Задача – съесть бегемота



Методика проектирования ПО

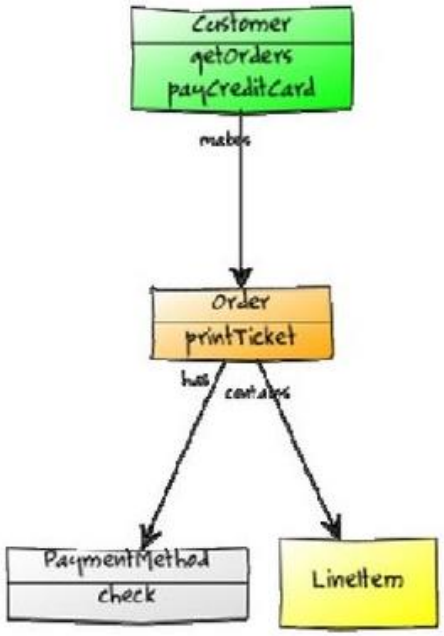
Простые фрагменты

Сложная проблема

Процедурный подход

Программа – алгоритм последовательного вызова проедур изменения данных в памяти.

Объектно-ориентированный



Программа – взаимодействие объектов, компонентов, отсылка и обработка событий.

Алгоритмы + Структуры данных = Программа (Н. Вирт)

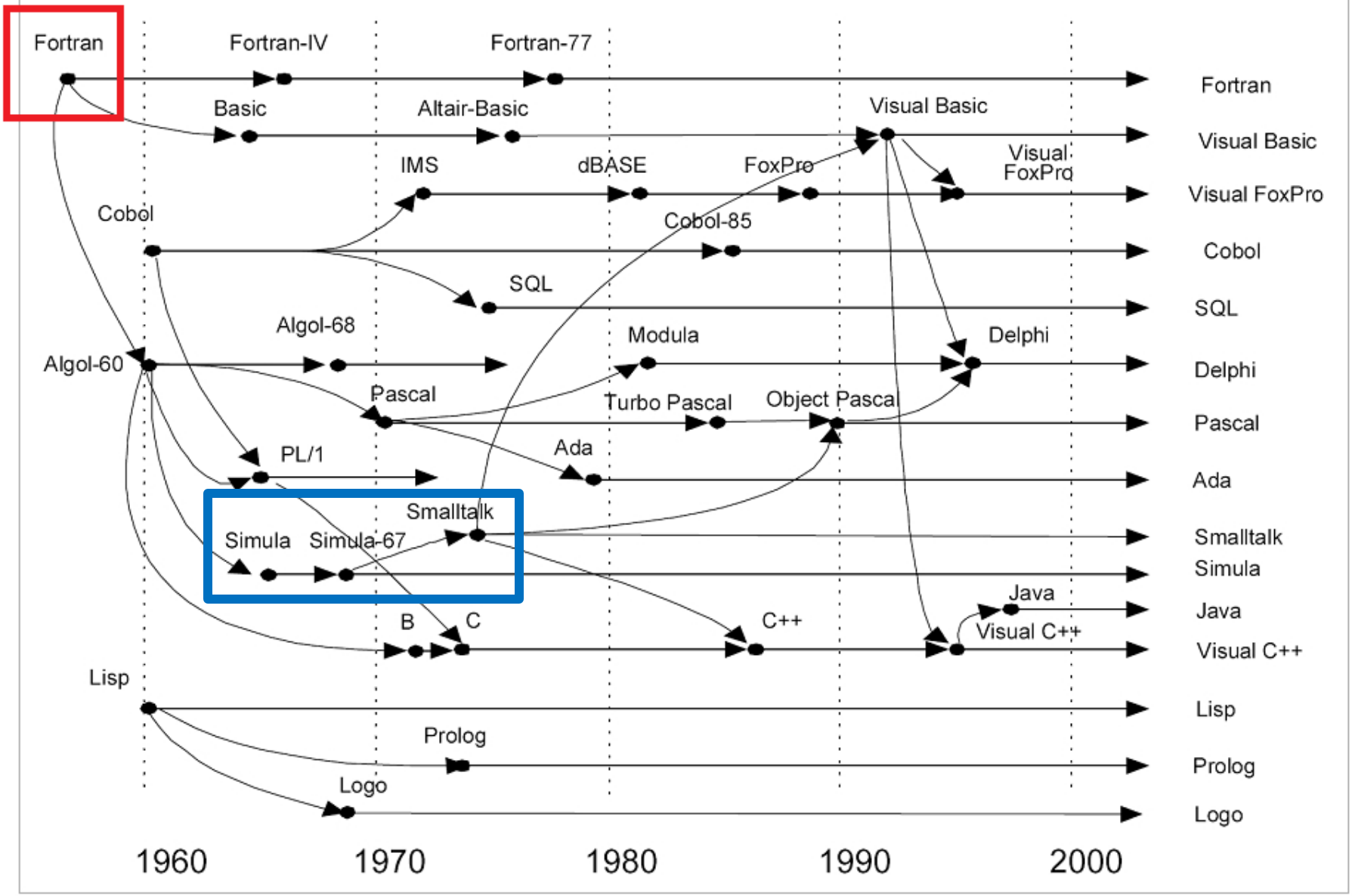
Объекты – «кирпичи» для построения приложения.

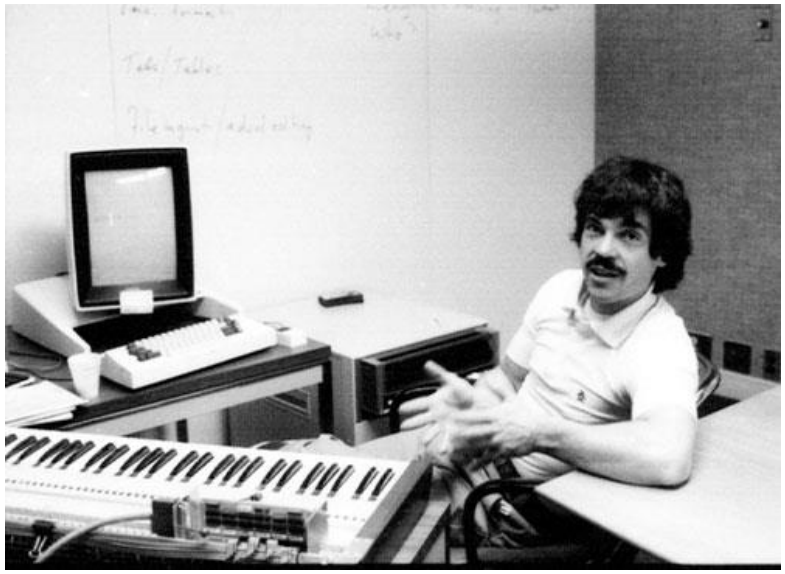
Суть объектно-ориентированного подхода

Попытка связать поведение сущности с её данными и спроецировать объекты реального мира и бизнес-процессов в программный код.

Такой код должно быть проще читать и понимать человеку, так как людям свойственно воспринимать окружающий мир как множество взаимодействующих между собой объектов, поддающихся определенной классификации.

ГЕНЕАЛОГИЧЕСКОЕ ДЕРЕВО ЯЗЫКОВ ПРОГРАММИРОВАНИЯ





Алан Кей (род. в 1946 г.)

- В 1970-х работал в Xerox PARC
- Руководил разработкой Smalltalk

- В Smalltalk заложены основы реализации GUI
- Рассматривался как инструмент программирования для Dynabook - устройства для обучения (концепция ноутбука, планшета)
- Графический интерфейс Star GUI (окна, пиктограммы, меню) - прототип GUI Macintosh и Windows

Объект имеет имя и определяется:

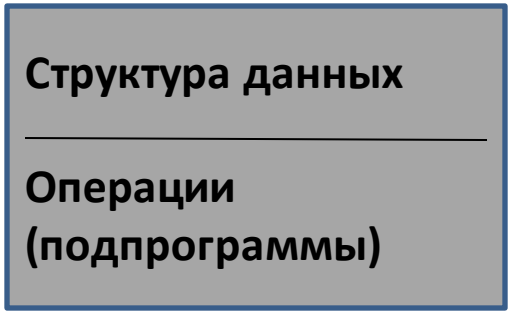
- **Атрибутами/свойствами** (*размер, цвет, вес, …*)
- **Поведениями/действиями** (*бежать, говорить, ...*)

House
- address - numberFloors - numberWindows
+ build() + destroy() + repair()

Human
- age - height - weight
+ speak() + walk() + sleep()

TV
- manufacturer - model - size
+ on() + switchChannels() + off()

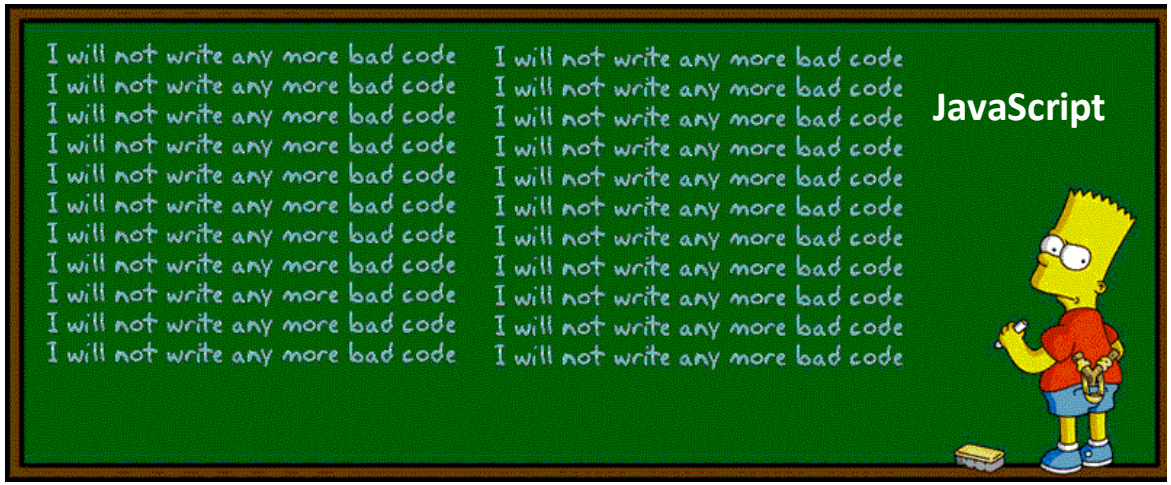
Объект — обладающий именем и физически находящийся в памяти компьютера набор **данных** и **методов**, имеющих доступ к ним.



Объект = Данные + Методы

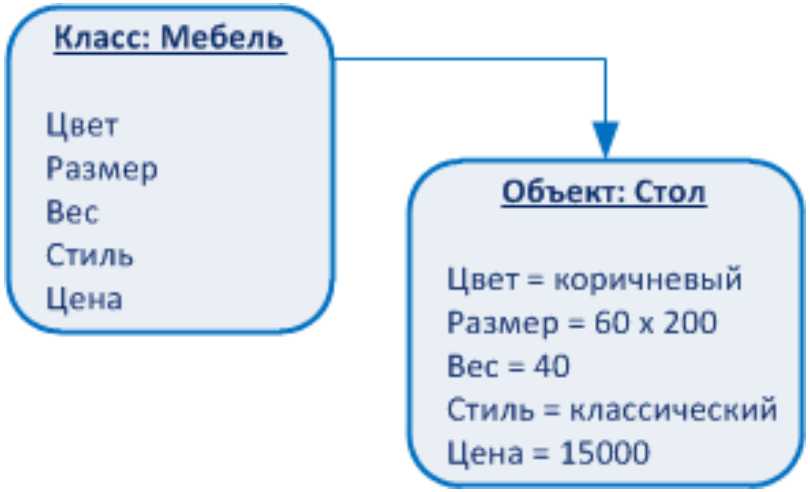
Создание и использование объектов

Объектно-ориентированный API

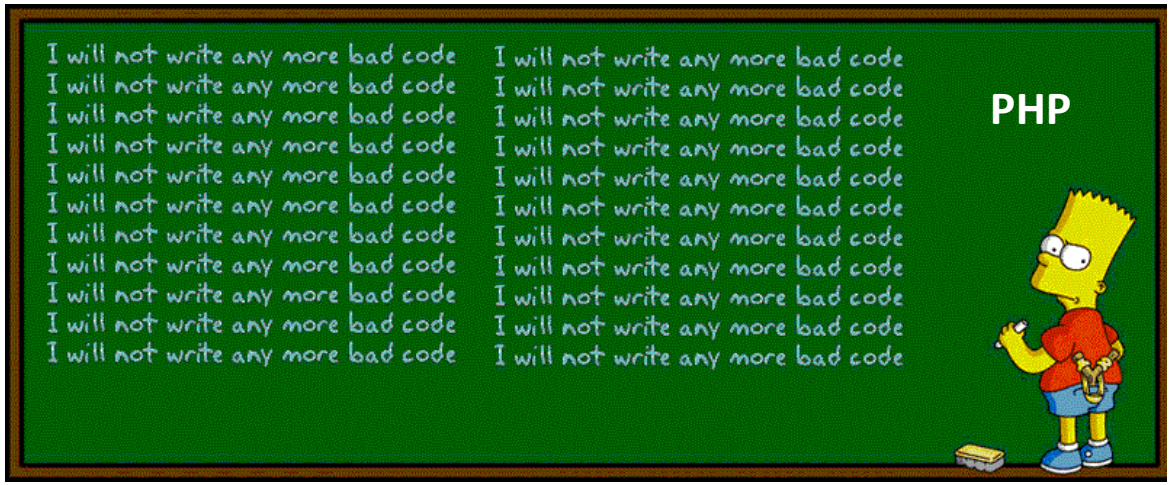


Объектно-ориентированный подход

Класс – шаблон , на основе которого создаются объекты-экземпляры класса (с одним и тем же набором свойств и методов).
Реальный объект должен иметь конкретные значения всех полей.



Описание класса и создание объекта

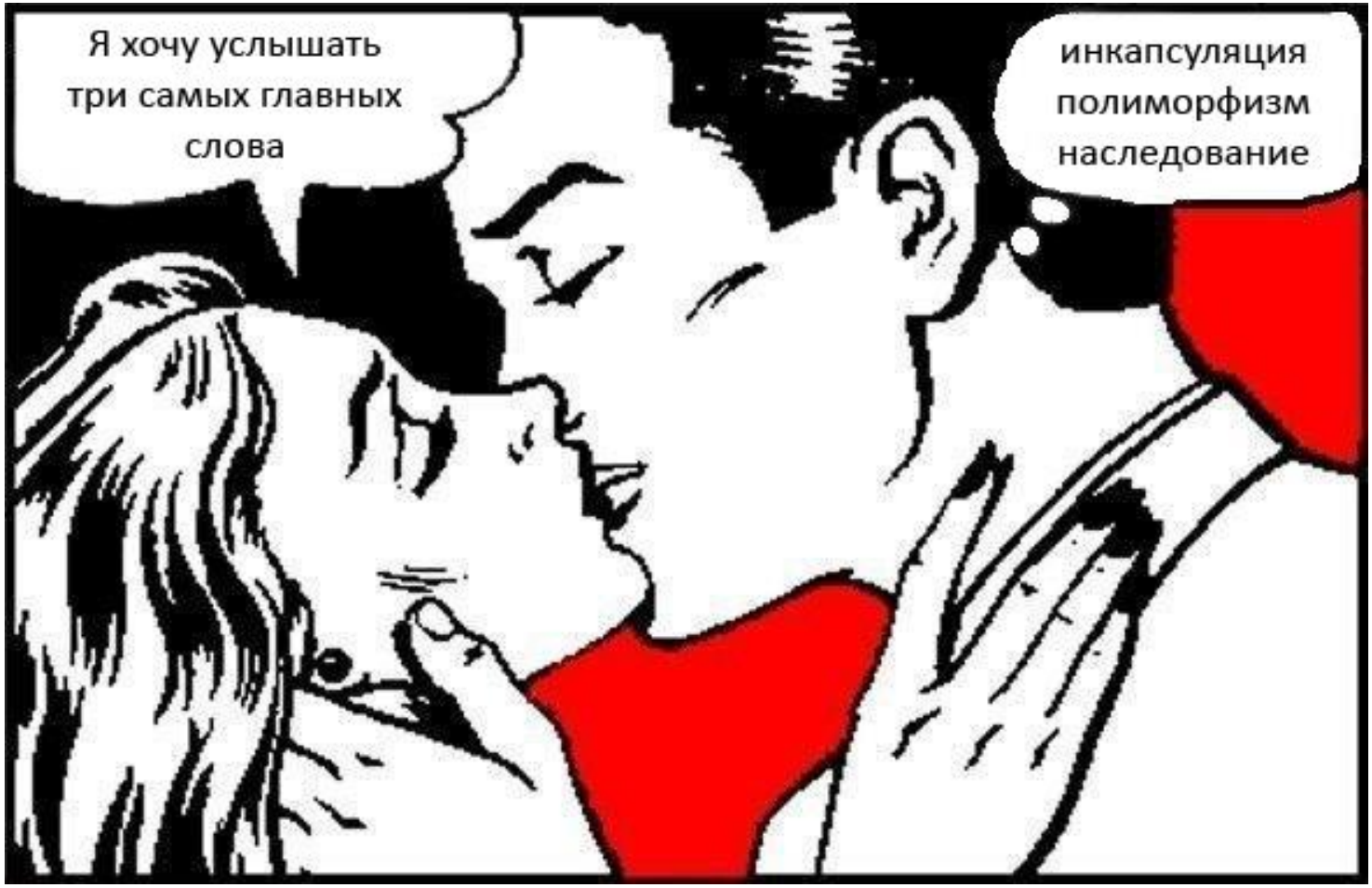


Конструкторы

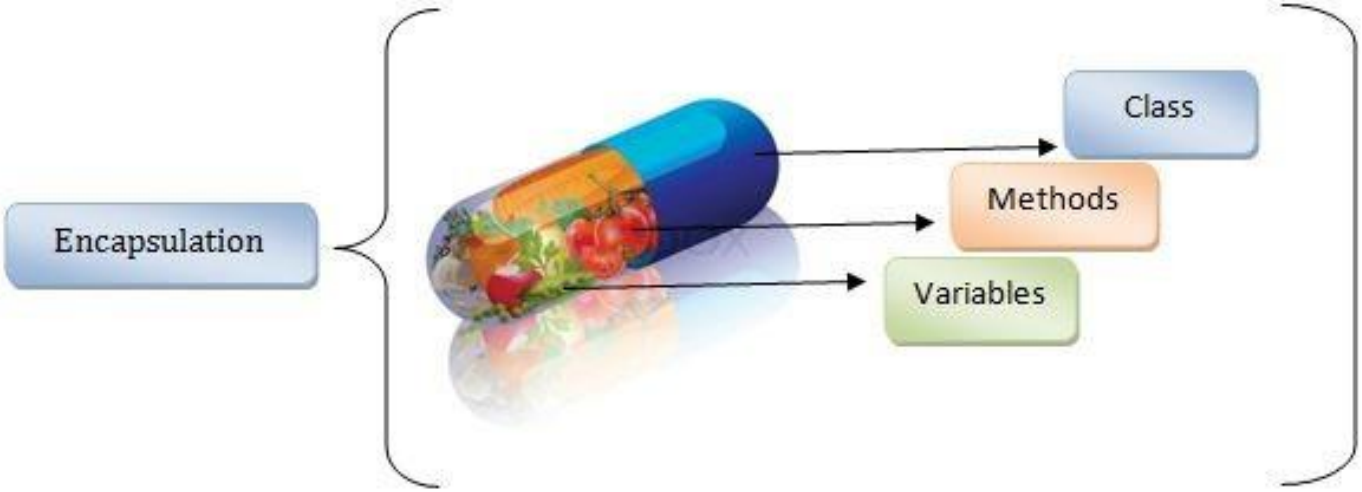
Основное назначение – инициализация атрибутов объекта.

- PHP: *функция __construct().*
- C++, Java, C#: *совпадают с именем класса.*
- Visual Basic: *оператор* New.
- Objective-C: *оператор* Init.
- Object Pascal: *оператор* Create.

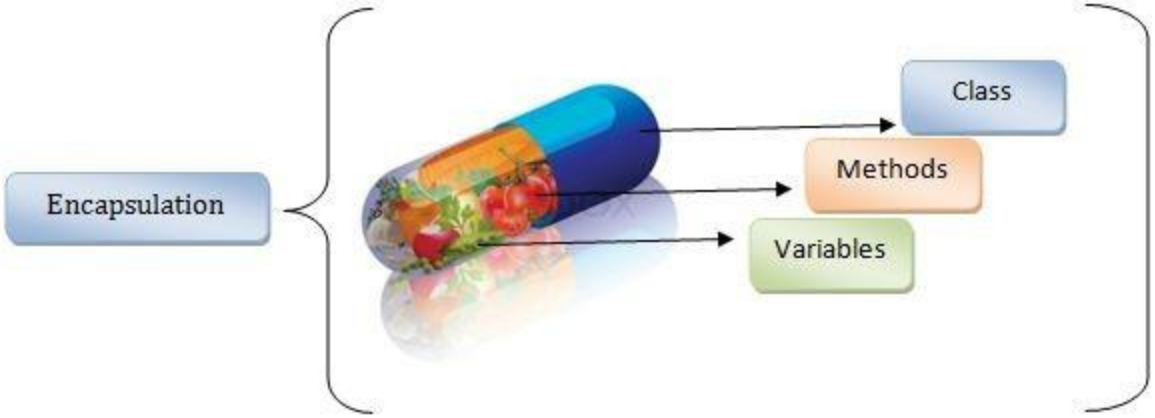
Основные принципы ООП



Инкапсуляция и сокрытие данных



Инкапсуляция - упаковка данных и функций в один компонент.



Сокрытие данных – пользователь класса может работать только с его интерфейсной частью и не имеет доступа к реализации функциональности класса.

Инкапсуляция (программирование)

Материал из Википедии — свободной энциклопедии [править | править код]

Текущая версия страницы пока не проверялась опытными участниками и может значительно отличаться от версии, проверенной 21 сентября 2016; проверки требуют 44 правки.

У этого термина существуют и другие значения, см. Инкапсуляция.

Инкапсуляция (англ. *encapsulation*, от лат. *in capsula*) — в информатике упаковка данных и функций в единый компонент.

Любая программная сущность, обладающая нетривиальным состоянием, должна быть превращена в замкнутую систему, которую можно только перевести из одного корректного состояния в другое.

Принцип инкапсуляции и сокрытия данных

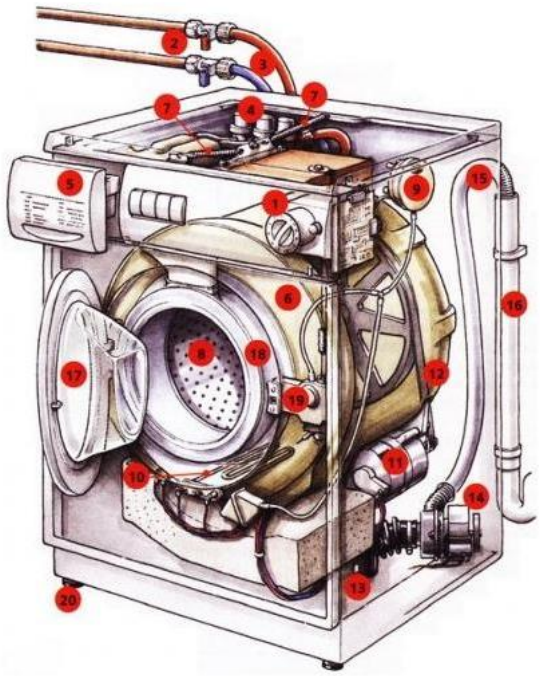
Сокрытие реализации класса и отделение его внутреннего представления от внешнего интерфейса.

Интерфейс

Реализация

Принцип инкапсуляции и сокрытия данных

Сокрытие реализации класса и отделение его внутреннего представления от внешнего интерфейса.



Интерфейс **Реализация**

Принцип инкапсуляции и сокрытия данных в ООП

Доступ к данным класса возможен только посредством методов этого класса.

Свойство объекта (property) – совокупность атрибута (поля) объекта и методов его чтения/записи.

Преимущества:

- Локализация данных, интегрирование их с подпрограммами обработки.
- Практически независимая разработка отдельных частей (объектов) программы.

Принцип инкапсуляции - реализация в ЯП

C#

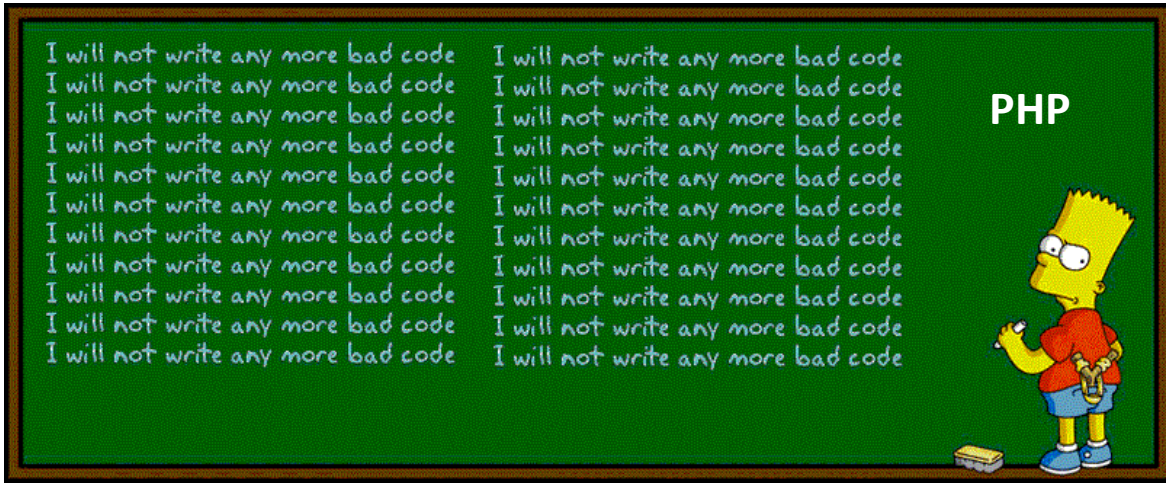
```
class TimePeriod
{
  private double seconds;
  public double Hours
  {
    get { return seconds / 3600; }
    set { seconds = value * 3600; }
   }
 }
class Program
{
static void Main()
{
TimePeriod t = new TimePeriod();
t.Hours = 24;
System.Console.WriteLine("Time in hours: " + t.Hours);
}
}
```

Принцип инкапсуляции в ООП

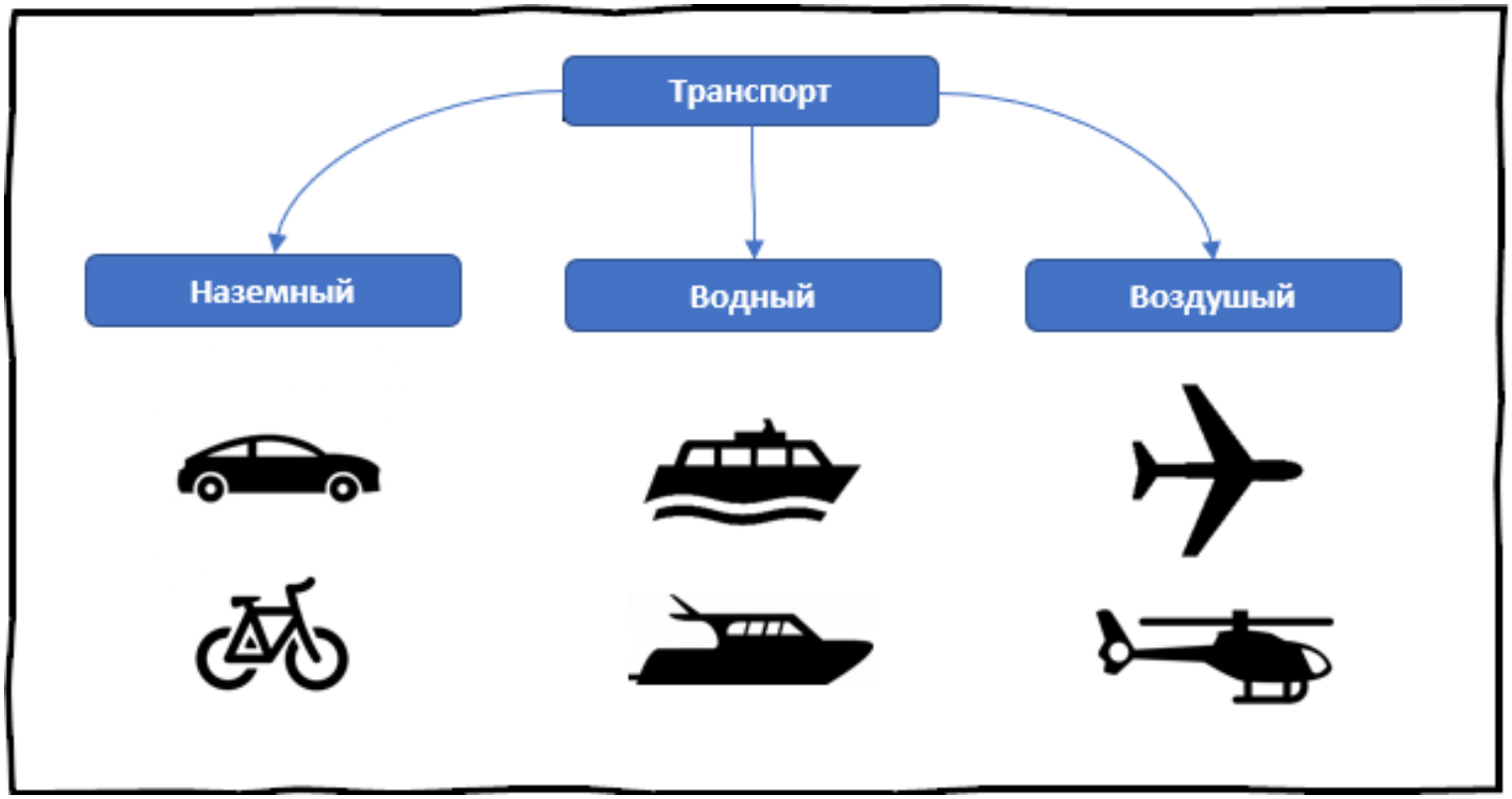
Области видимости полей класса в PHP:

public — доступны отовсюду

private — доступны только из методов данного класса



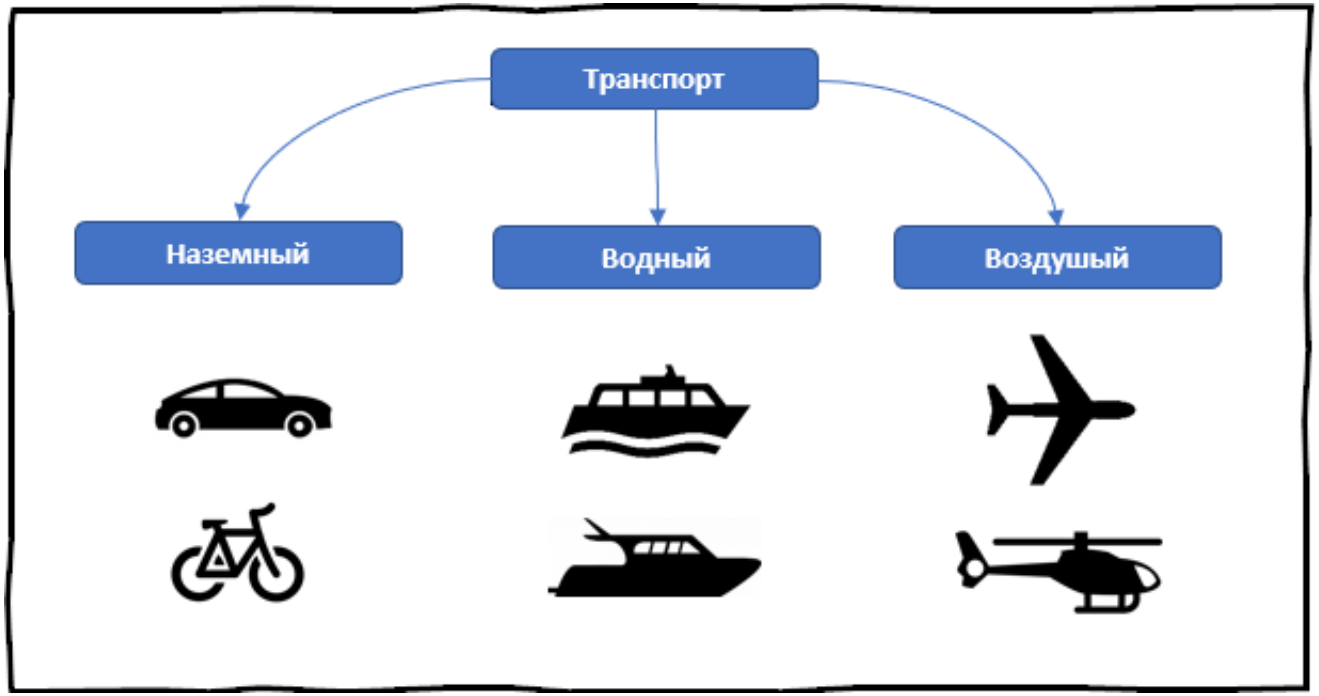
Наследование



Наследование (иерархия) - возможность порождать один класс от другого с сохранением всех свойств и методов класса-предка, добавляя, при необходимости, новые свойства и методы.

Принцип наследования/расширения

Получение по наследству атрибутов и поведений от родительских классов.



Отношение **“является”**

Принцип наследования/расширения

Если вас *почти* устраивает какой-то класс, вы можете создать потомка и переопределить или добавить какую-то часть его функциональности.

В C++ поддерживается **множественное наследование**, когда дочерний класс может иметь несколько родительских. Языки Java, C#, Object Pascal, PHP этого не поддерживают.

Принцип наследования/расширения

PHP

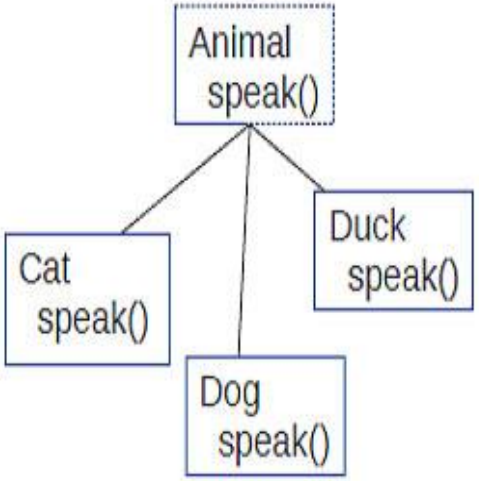
Область видимости **protected** — поле доступно в классе, котором оно определено и во всех его дочерних классах.

Классы могут наследоваться от **абстрактного класса**, который не может иметь собственных экземпляров.

В потомке можно:

- Перекрыть метод родителя.
- Вызвать одноименный метод родителя через **parent::*method()***

Полиморфизм

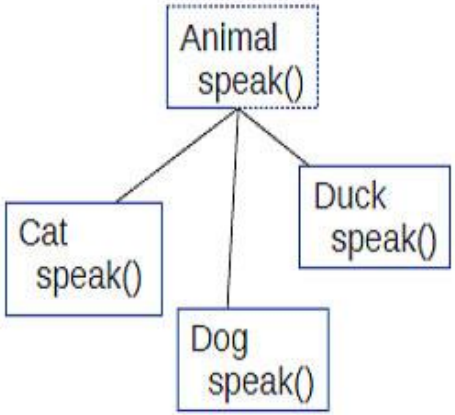


Полиморфизм. Функции с одним и тем же именем соответствует разный программный код в зависимости от того, объект какого класса используется при вызове этой функции.

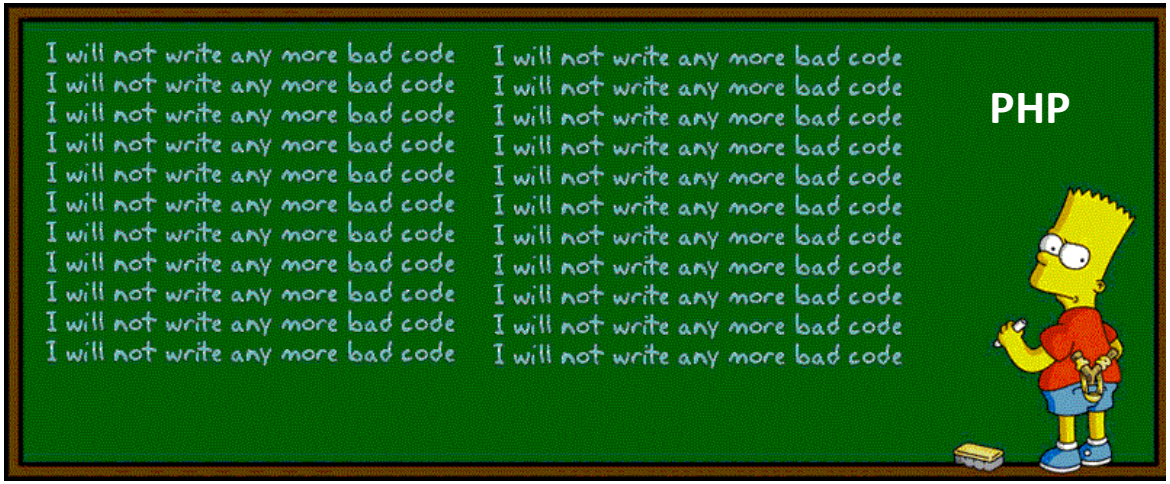
Полиморфизм = "многообразие форм"

Полиморфизм - это один интерфейс для множества реализаций

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



Принцип полиморфизма



Принцип полиморфизма (позднее связывание)

Объект подкласса (потомка) может использоваться всюду, где используется объект суперкласса (предка).

При добавлении к иерархии классов нового подкласса не нужно менять написанный код.

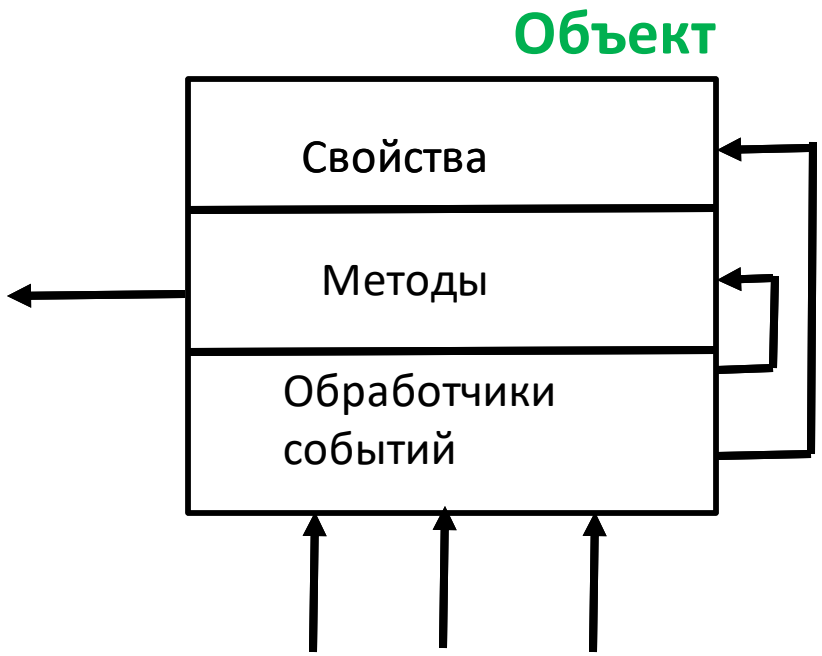
«Позднее связывание» позволяет определять версию полиморфного (виртуального) метода во время выполнения программы.

События и сообщения

Объекты могут:

- Взаимодействовать друг с другом посредством **сообщений**.
- Реагировать на определенные **события** (обрабатывать их), возникающие вследствие действий пользователя или других объектов.

Хорошо подходит для графического интерфейса (обработчики событий нажатия на кнопку и т.п.)

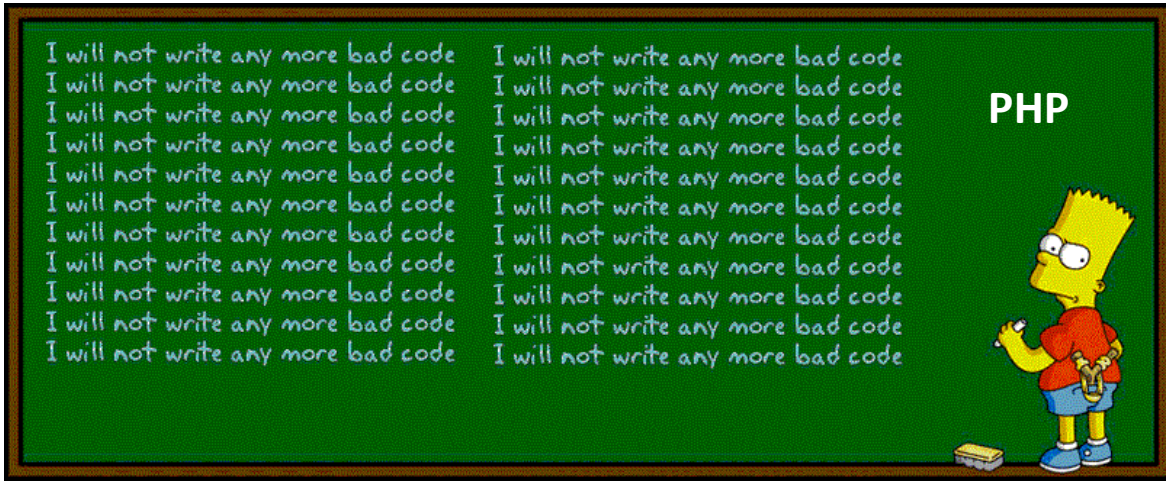


Тип переменной – множество отношений между данным типом и другими типами.

Статические свойства и методы

Определяются в контексте класса, а не объекта.

- Доступны из любой точки программы.
- Доступны каждому экземпляру данного класса.
- Не нужно создавать экземпляры класса только ради вызова простой функции.



Интерфейсы

В интерфейсе декларируется функциональность, которую должен реализовать каждый класс, реализующий этот интерфейс.

Что здесь?

Любая реализация интерфейса
«Вилка для электрической розетки»

Интерфейсы

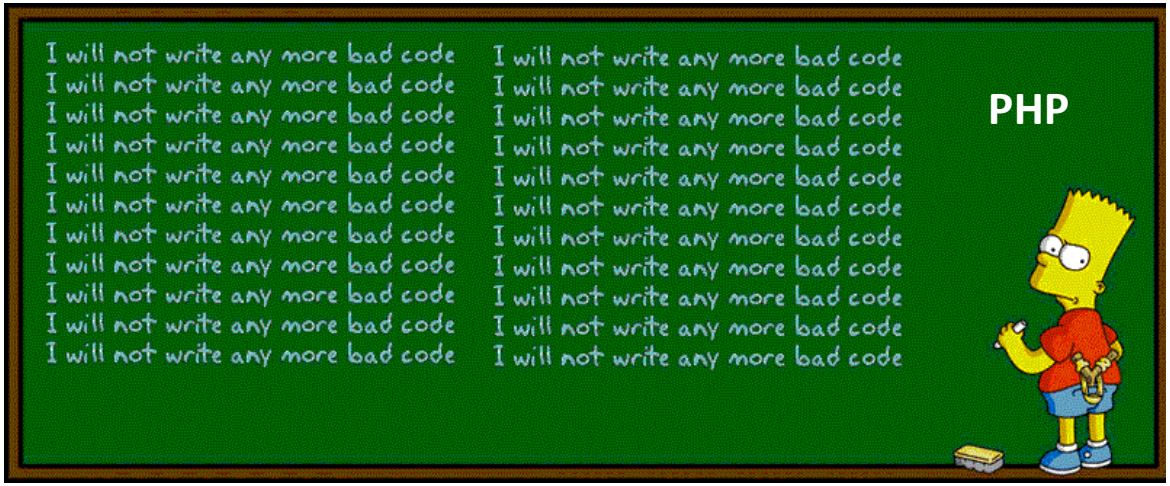
Аналог абстрактного класса, содержащего только абстрактные методы. Интерфейсы — это облегченные классы, которые диктуют нам условия.

Интерфейсы можно наследовать друг от друга.

Класс может реализовывать несколько интерфейсов (наследоваться от нескольких классов нельзя).

Класс = тип данных + шаблон для создания экземпляров
Интерфейс = тип данных

Интерфейсы



Как правильно использовать ООП ?

Хорошее ООП

- снижает сложность
- привносит модульность
- повышает совместимость
- локализует изменения
- абстрагирует от реализации
- упрощает код
- легко тестируется

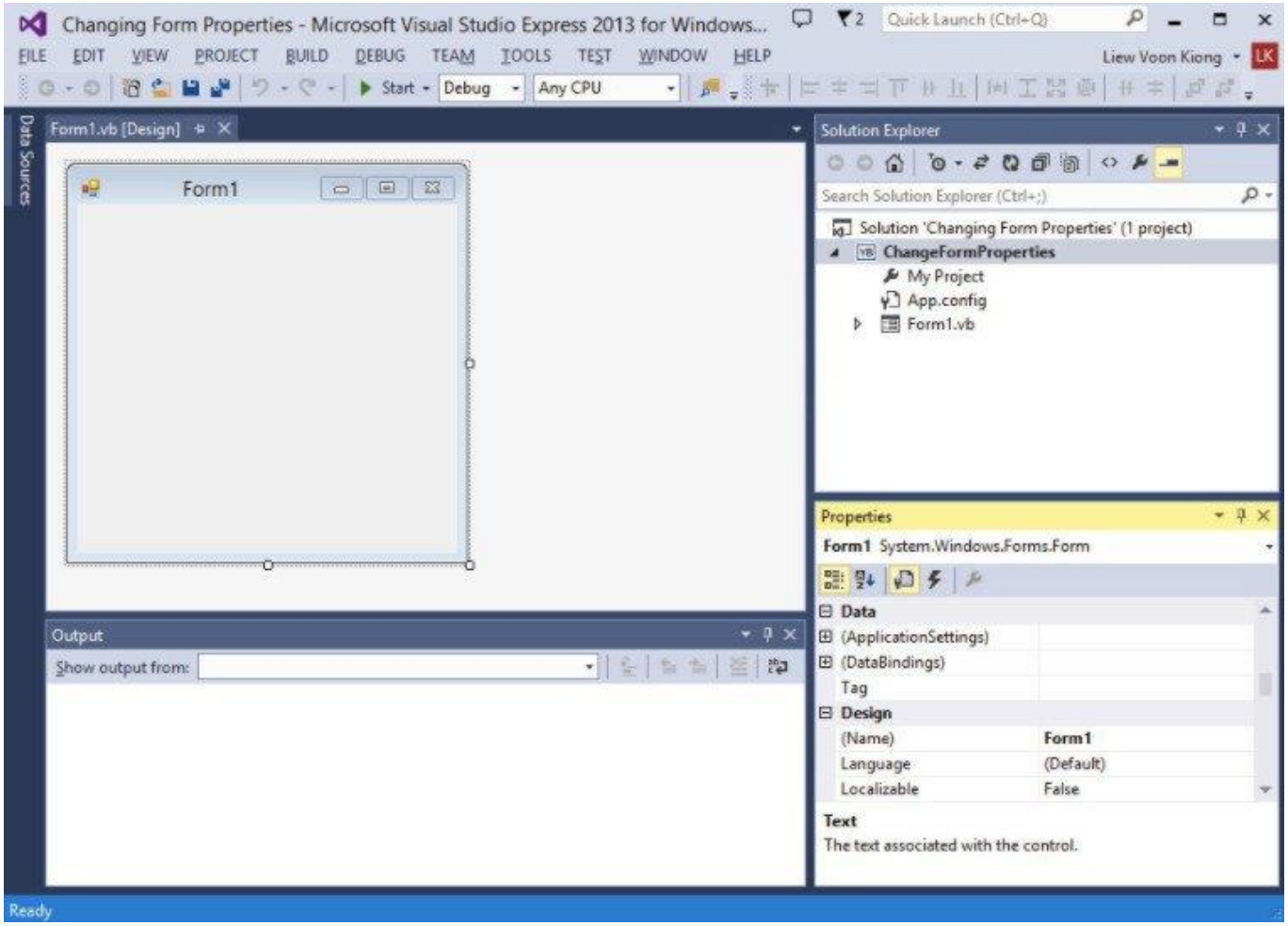
Плохое ООП

- привносит сложность
- сохраняет процедурный подход
- причиняет неудобство

Визуальное проектирование приложений

Используются визуальные объекты-компоненты.

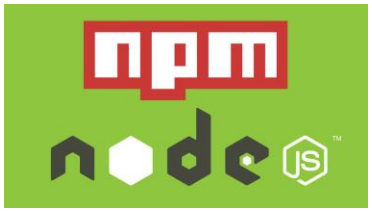
Результат – приложение с автоматически генерируемым кодом.



Повторное использование кода

- Выделение фрагментов кода в модули
- Повторное использование модулей
- Создание централизованных хранилищ модулей
- Инструменты для подключения модулей из хранилищ

Повторное использование объектов



Повторное использование объектов

Как сделать объекты-модули **стандартными**, чтобы их использование **не зависело от языка программирования и платформы**, на которой работает программа?

Стандартизация

Все камни разные – строить стены тяжело

Стандартизация

Все камни разные – строить стены тяжело

1. На всей земле был один язык и одно наречие.
2. Двинувшись с востока, они нашли в земле Сеннаар равнину и поселились там.
3. И сказали друг другу: наделаем кирпичей и обожжем огнем. **И стали у них кирпичи вместо камней, а земляная смола вместо извести.**
(Книга Бытия 11:1-3)

Компонентный подход к программированию

Построение программ из физически отдельных компонентов, которые взаимодействуют между собой через стандартизированные интерфейсы.

Объекты-компоненты можно распространять без исходных кодов и использовать в любом языке программирования.

Конвейерное соединение утилит

ОС Unix (начало 1970-х):

- Текст как универсальный формат представления информации
- Все есть файл
- Командный язык оболочки для программирования на базе инструментальных средств (softwaretools)

Одна команда = одна функция
find, cat, sed, grep, awk, …

Конвейерное соединение утилит

Одна команда = одна функция
find, cat, sed, grep, awk, …

Командный язык для поддержки программных конвейеров

```
find /tmp -type f -name '*' -mtime +7 -print0 | xargs -0 rm -f
```

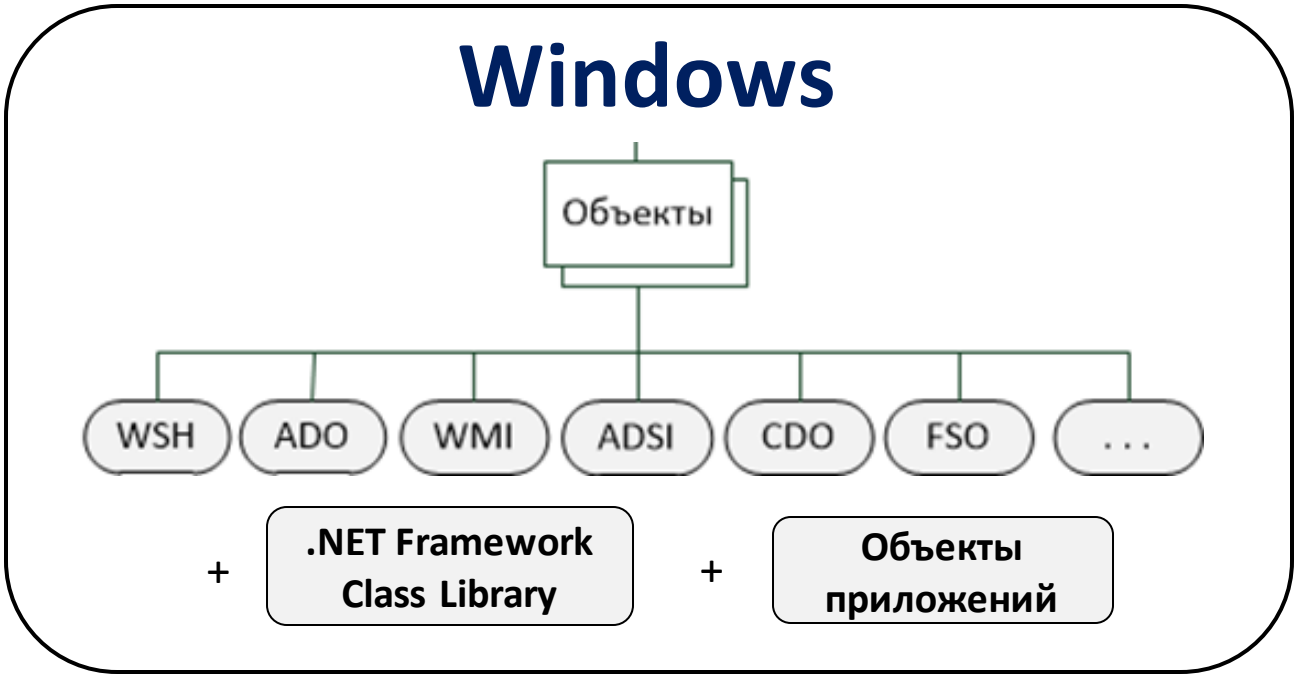
Программный конвейер – пример модульного проектирования (мощная стратегия управления сложностью).

Прохождение по конвейеру <u>потока символов</u> хорошо подходит для Unix, где все настройки хранятся в виде текста.

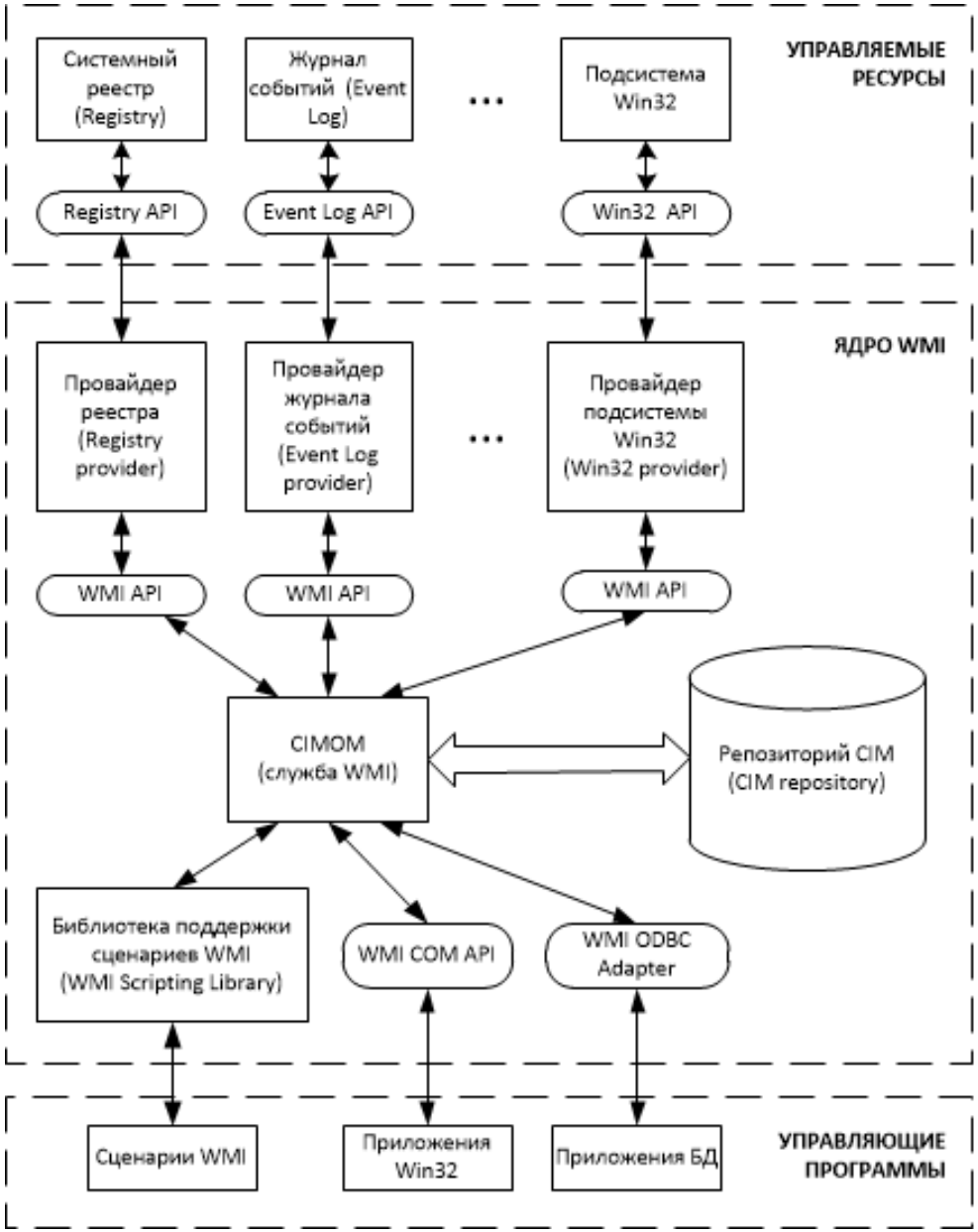
Объекты внутри операционки

ОС Windows (1990-е) - API-ориентированная

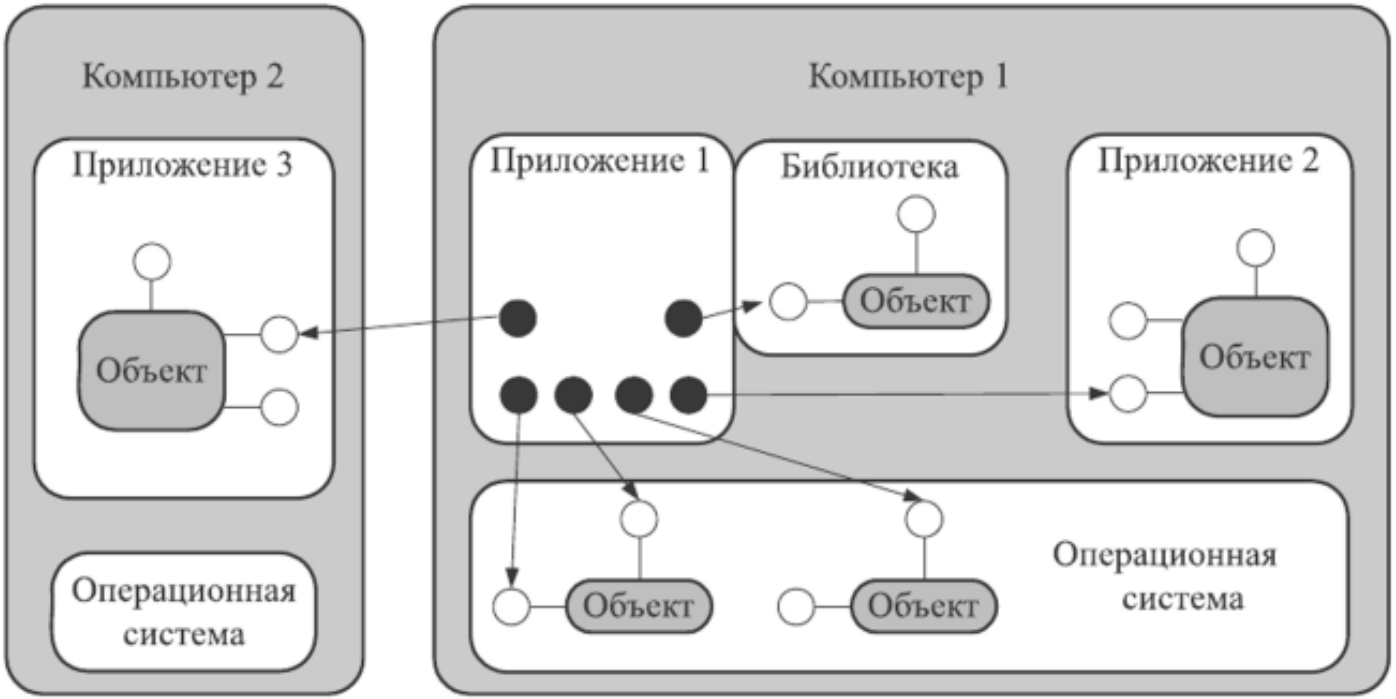
Управляемые элементы сгруппированы во внутренние объекты.



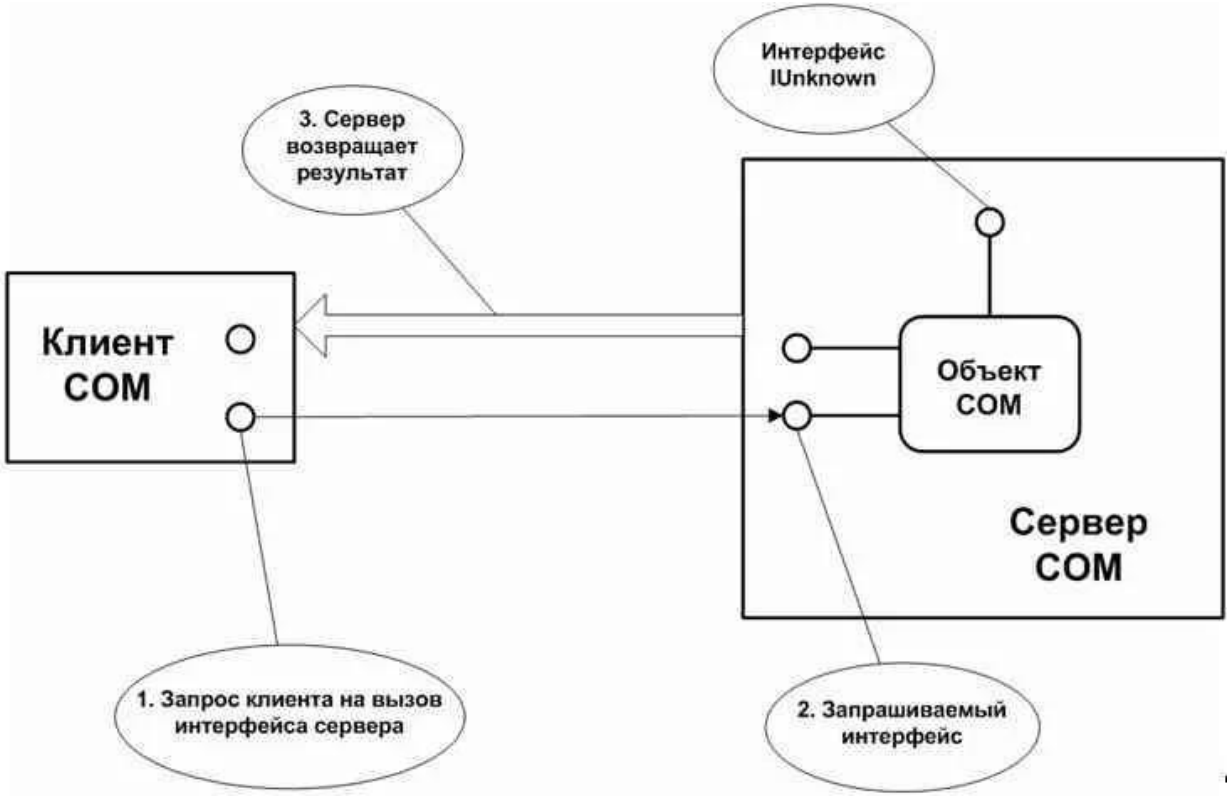
Архитектура WMI



COM-объекты

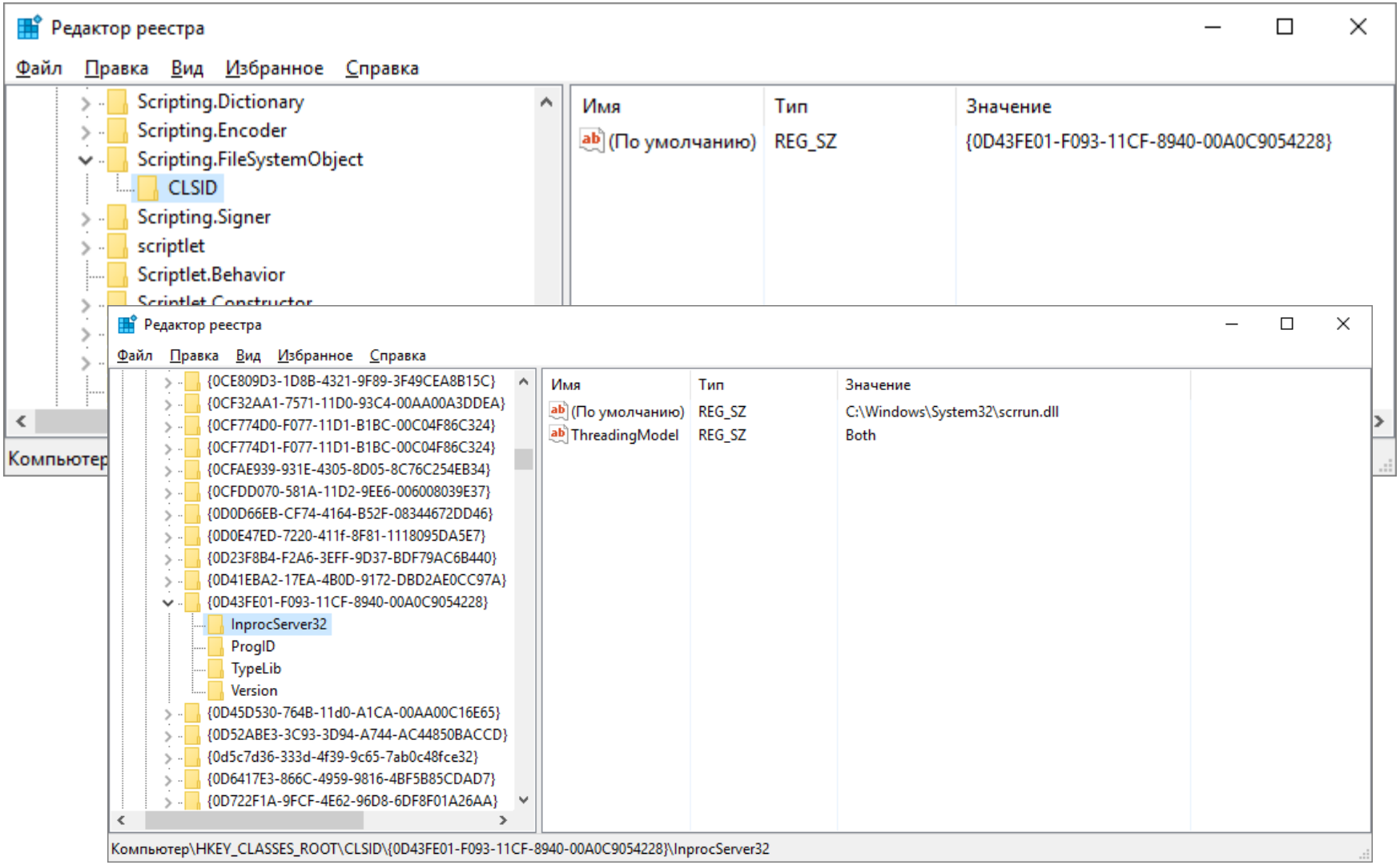

Компьютер 2
Приложение 3
Объект
Операционная система
Компьютер 1
Приложение 1
Библиотека
Объект
Приложение 2
Объект
Объект
Объект
Операционная система

COM-объекты


Интерфейс
IUnknown
3. Сервер
возвращает
результат
Клиент
COM
Объект
COM
Сервер
COM
1. Запрос клиента на вызов
интерфейса сервера
2. Запрашиваемый
интерфейс

COM-объекты в системе

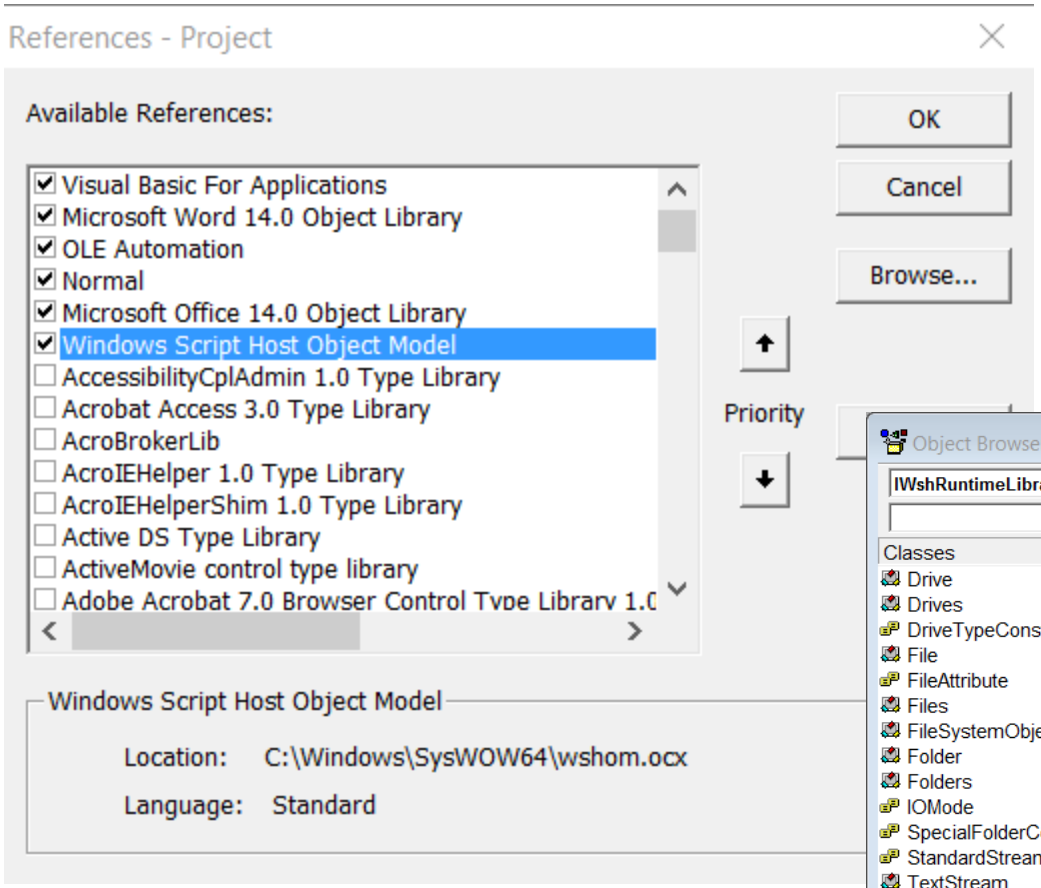
HKEY_CLASSES_ROOT
HKEY_CLASSES_ROOT\CLSID

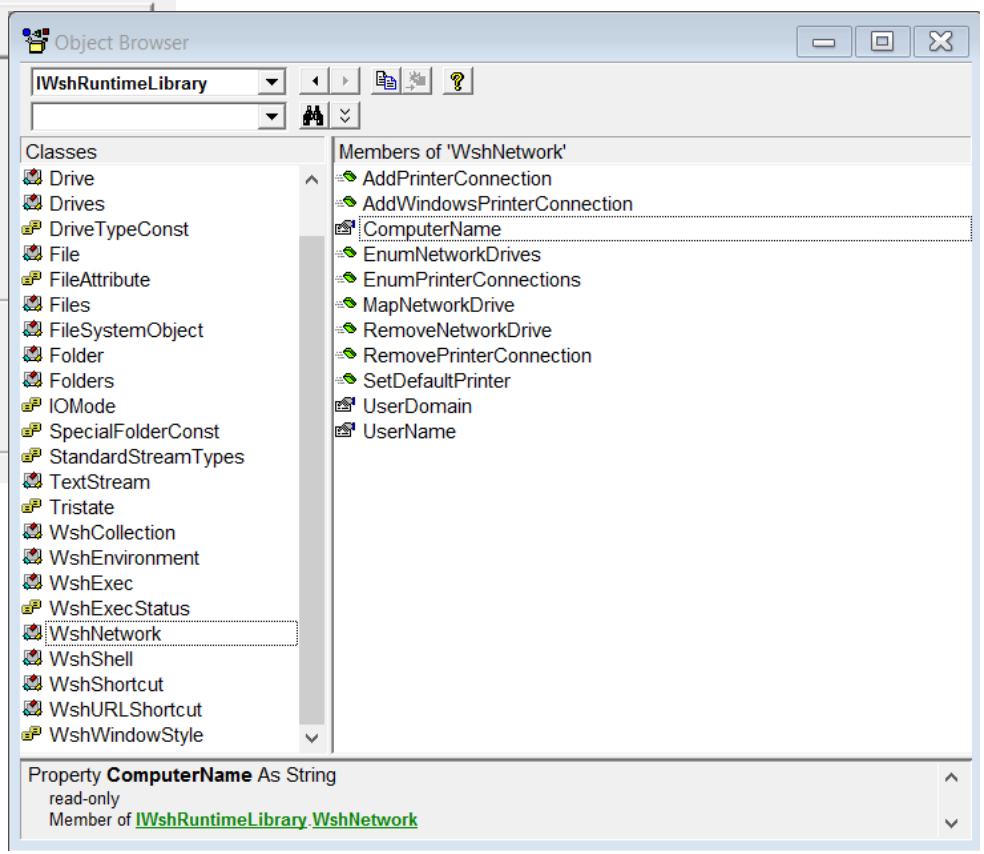


COM-объекты в системе

MS Office, Visual Basic

1. Tools\References
2. View\Object Browser





Объекты внутри Windows

Windows Script Host	• Работа со стандартными потоками ввода/вывода • Управление дочерними процессами • Работа с локальной сетью • Просмотр и изменение переменных среды • Доступ к специальным папкам Windows • Работа с ярлыками • Работа с системным реестром
Scripting.FileSystemObject	• Управление файловой системой (создание, копирование, удаление файлов/папок, …) • Чтение/запись текстовых файлов
Windows Management Instrumentation (WMI)	Обеспечивают программный интерфейс управления всеми компонентами операционной модели.

Объекты внутри Windows

Active Directory Scripting Interface (ADSI)	Работа с каталогами пользователей в доменных сетях и сетях на базе рабочих групп (Active Directory, Microsoft Exchange Server, Microsoft IIS, любой LDAP-каталог)
Collaboration Data Object (CDO)	Работа с сообщениями электронной почты и почтовыми серверами
Active Database Object (ADO)	Работа с базами данных
Приложения Microsoft Office	Управление офисными приложениями и документами
Сторонние приложения	