

Инкапсуляция

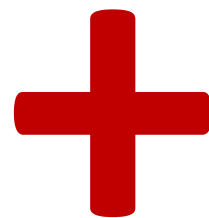
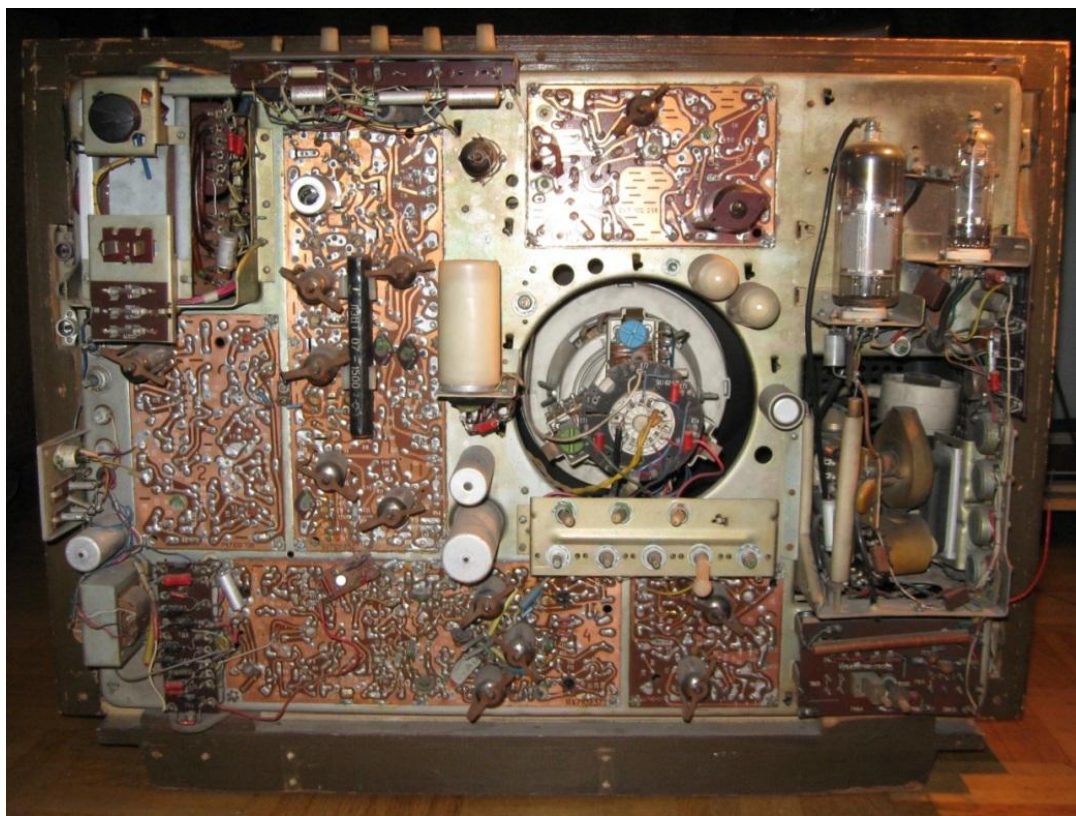
1. Проблема использования объекта
2. Понятие интерфейса. Каким должен быть интерфейс у класса
3. Понятие инкапсуляции. Принцип инкапсуляции. Примеры
4. Реализация инкапсуляции: модификаторы видимости
5. Обозначение инкапсуляции в UML
6. Следствия применения инкапсуляции

Преподаватель:

Ботов Дмитрий Сергеевич

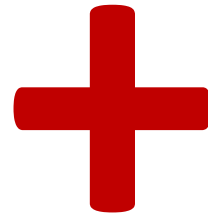
Проблема использования объекта

□ Что внутри объекта?



Проблема использования объекта

- Что внутри объекта?
- А зачем вообще это знать!

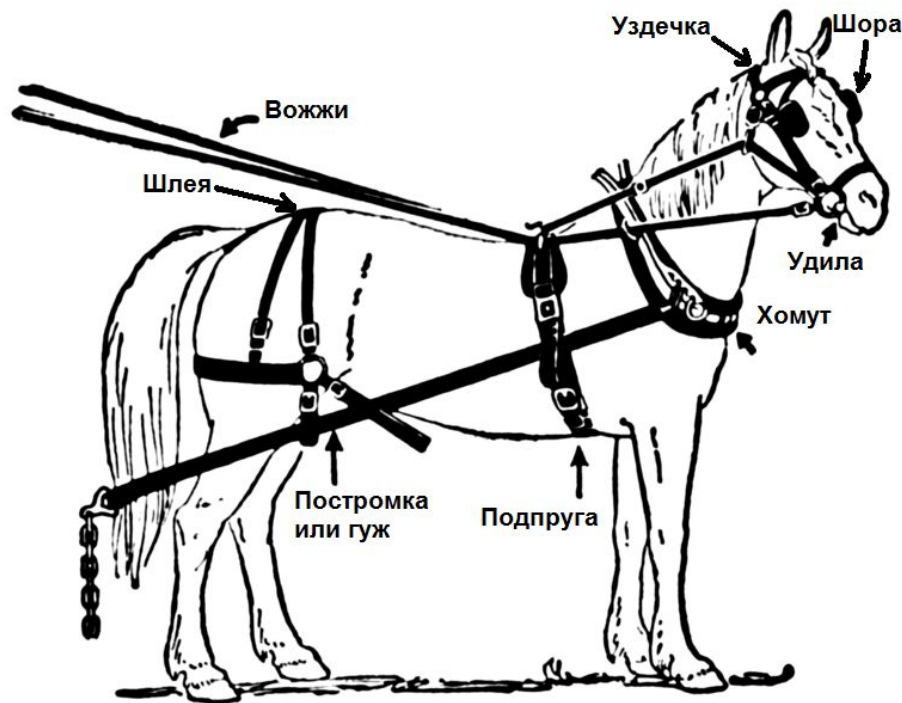


Как нам работать с объектом?

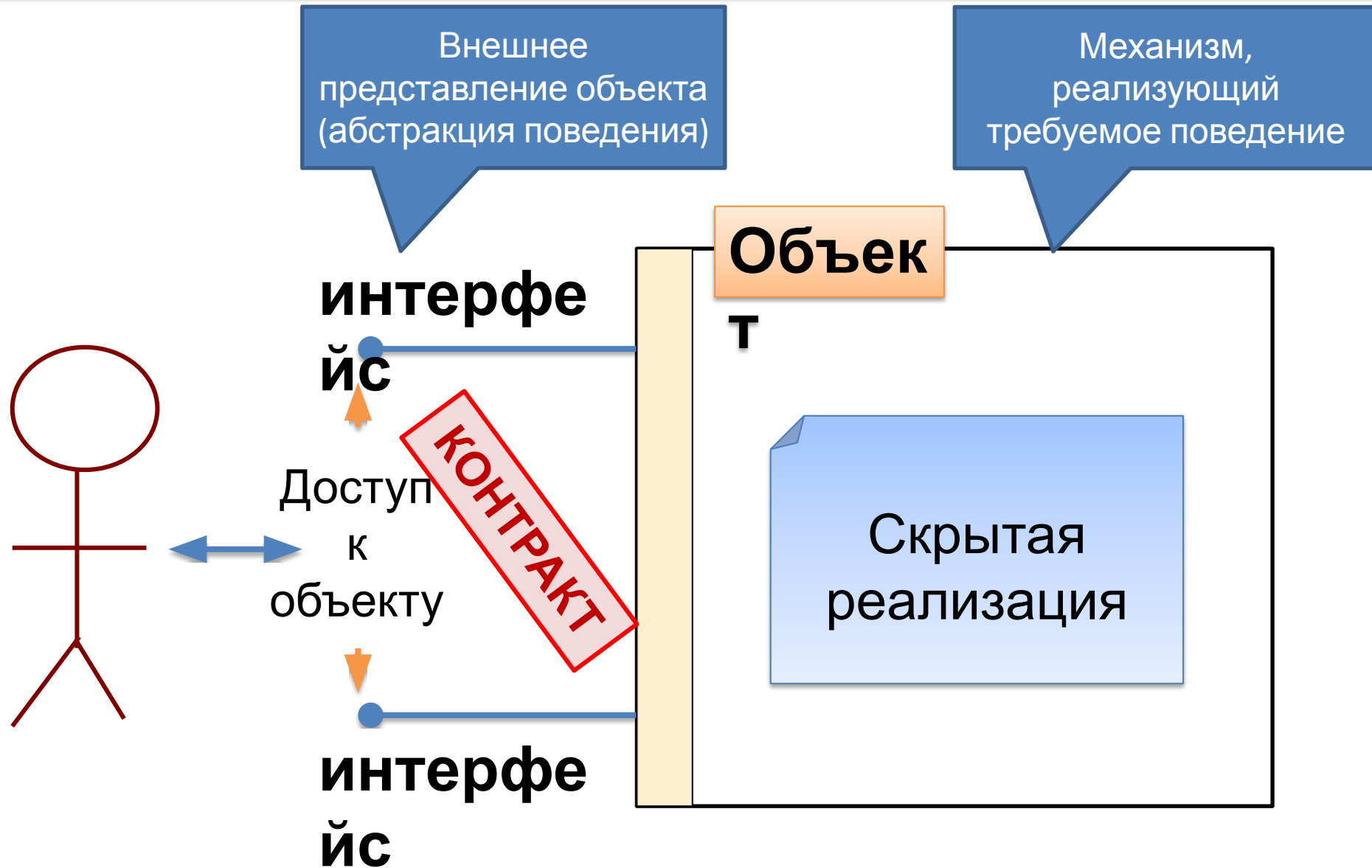
- Использовать **стандартный механизм доступа к объекту - интерфейс**
- Стандартный механизм доступа к объекту должен:
 - ✓ Должен быть универсальным
 - ✓ Должен быть достаточно простым
 - ✓ Должен полностью обеспечивать выполнение **контракта** - спецификации внешнего проявления объекта

Определение интерфейса системы

Интерфейс (interface) — совокупность средств, методов и правил взаимодействия между элементами системы

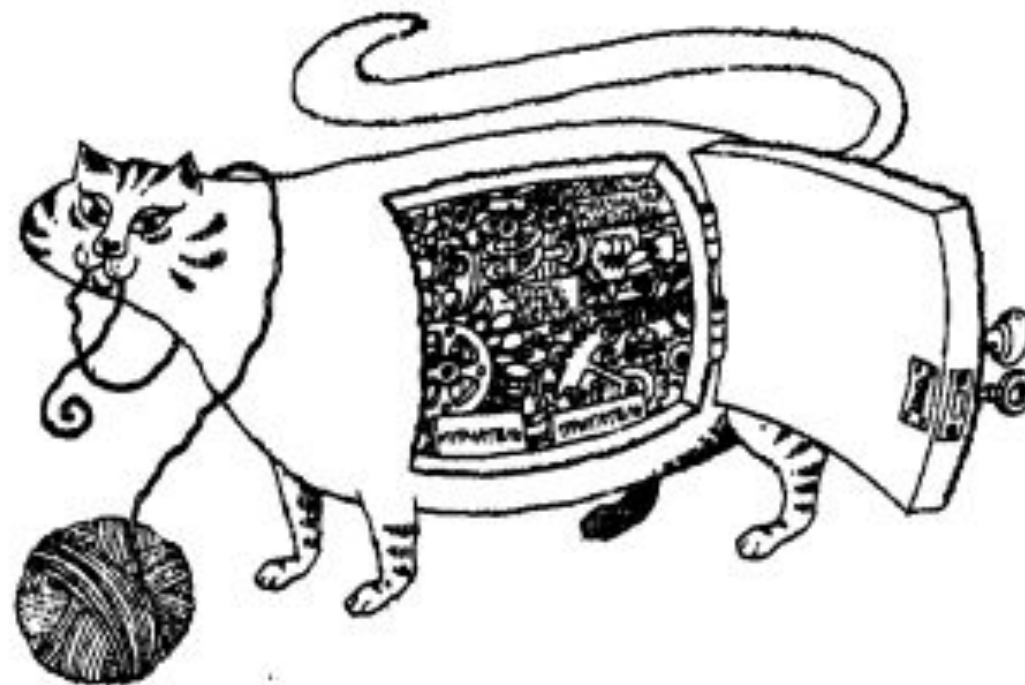


Интерфейс и его реализация



Понятие инкапсуляции

Инкапсуляция –
сокрытие внутреннего
устройства объекта
(реализации)



Инкапсуляция скрывает детали
реализации объекта

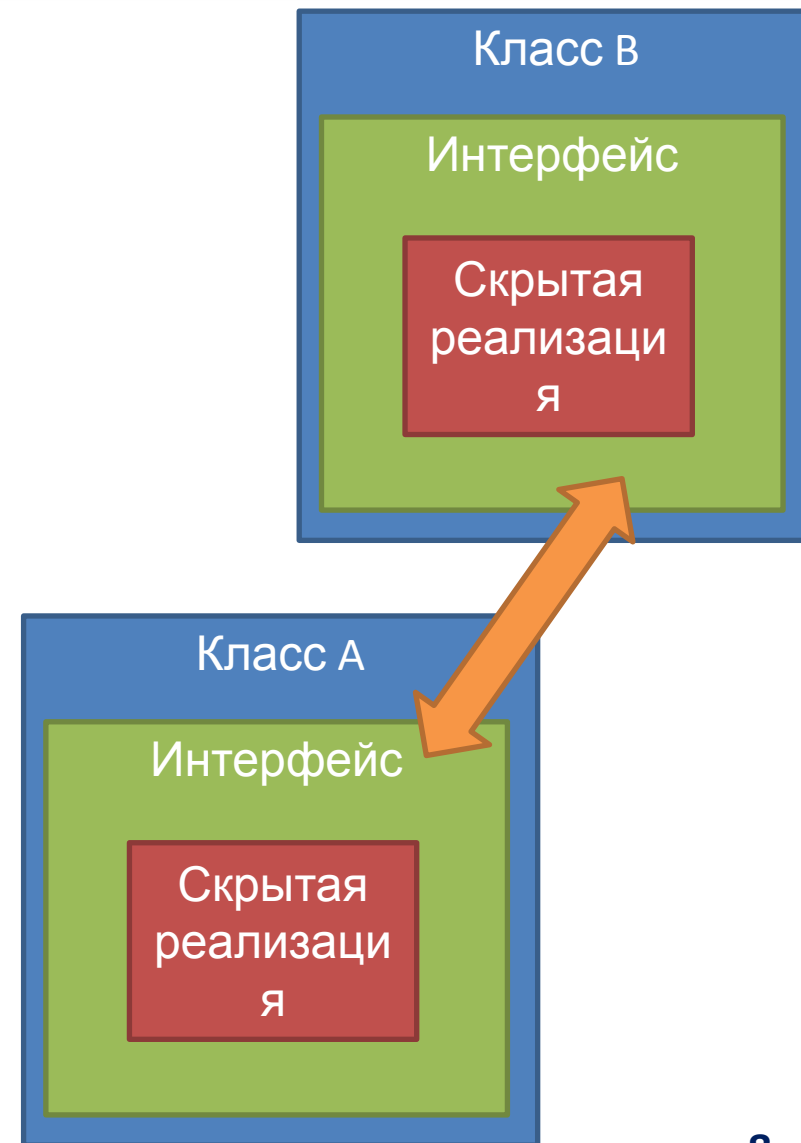
Принцип инкапсуляции

Принцип инкапсуляции: обеспечить **независимость** внутренней реализации объекта от остальных частей системы

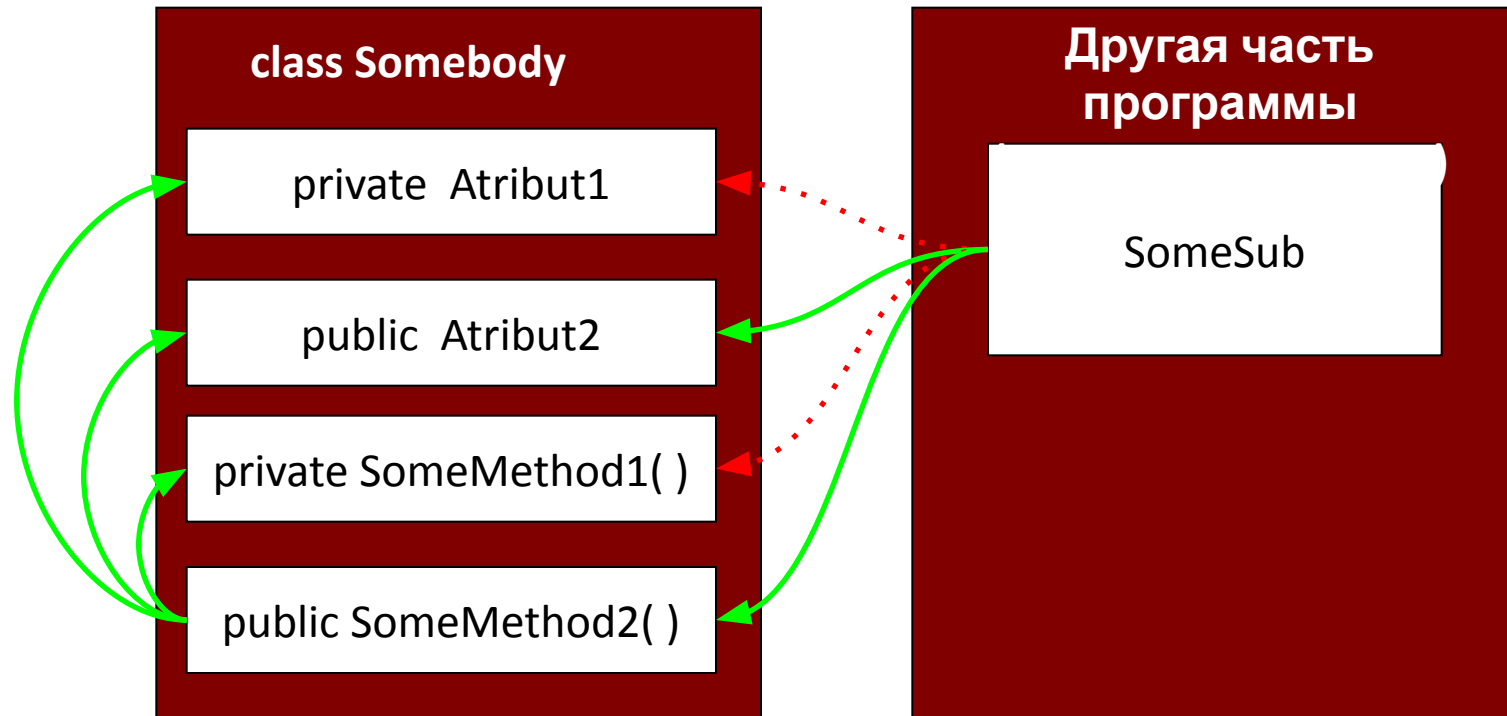
Как?

- За счет изоляции контрактных обязательств (интерфейса) от их реализации

«Никакая часть сложной системы не должна зависеть от внутреннего устройства какой-либо другой части»



Модификаторы видимости



- `public` – модификатор открытого доступа
- `private` – модификатор закрытого доступа

← Доступ
возможен
◄... Доступ
невозможен

Инкапсуляция: реализация

Обычно скрываются:

- поля классов
(если они не представляют собой уже инкапсулированные объекты)
- реализация методов (тело метода)
- методы, которые непосредственно связаны с внутренней реализацией

В языках программирования реализуется на основе ограничения доступа к атрибутам и операциям путем использования **модификаторов ВИДИМОСТИ**

Обозначение модификаторов видимости в UML

- Для документирования ограничений по доступу в UML атрибуты и операции обозначаются символами слева от имен
 - «+» (открытый доступ/public)
 - «-» (только из операций этого же класса/private)
 - «#» (только из операций этого же класса и классов создаваемых на его основе/protected)

Пример инкапсуляции

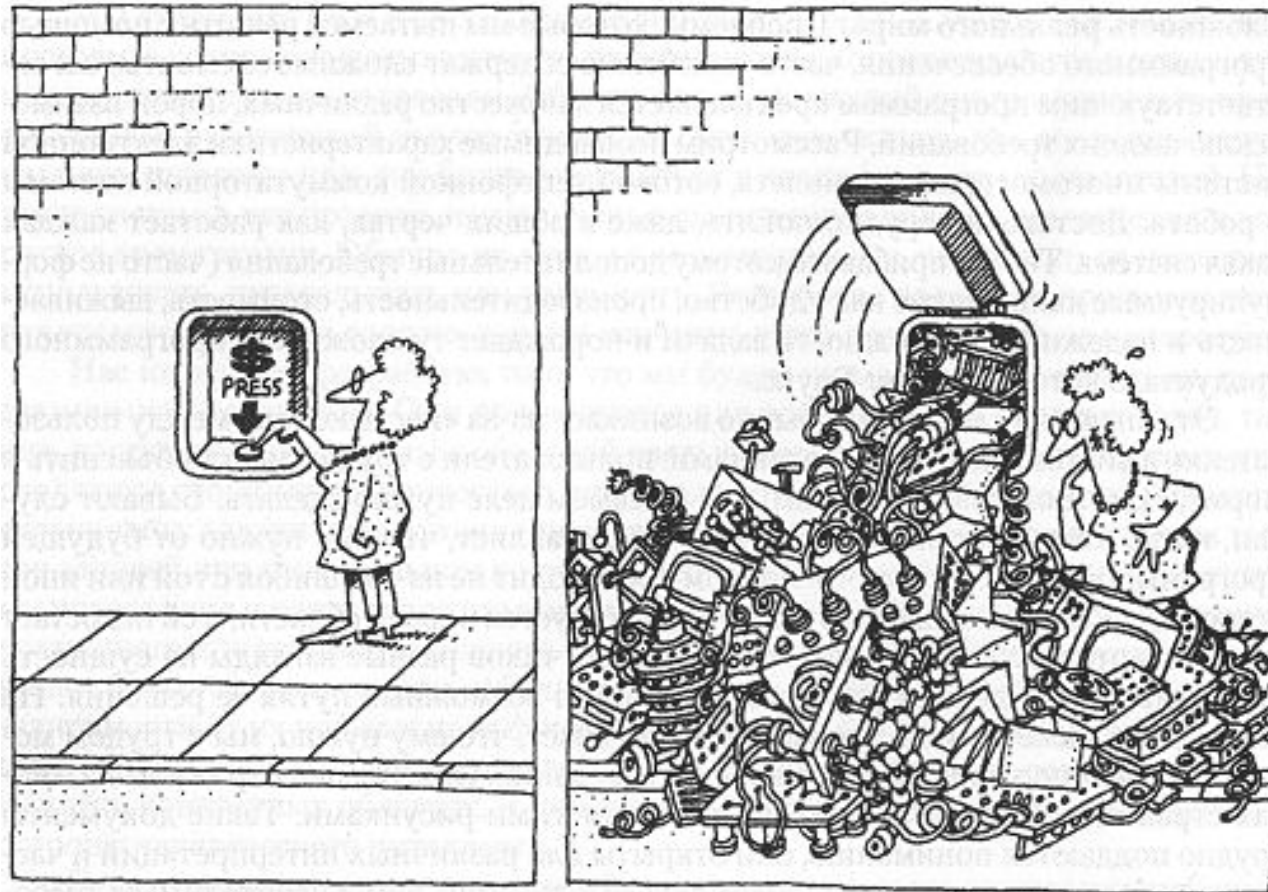
Пластиковая карта VISA
- Номер счета: длинное целое - Пин-код: целое - Сумма: финансовый
- Дешифрация(пин) + Авторизация(пин) + Остаток() + Снятие(сумма) + Зачисление()

Пример инкапсуляции

Телевизор
-НастройкиКаналов -ДешифраторСигнала -Преобразователь
-Самодиагностика() +Включить() +Выключить() -ДекодерСигнала() +ПереключениеКанала()

Следствие 1: простота использования

Обеспечивает создание **иллюзии простоты** при использовании за счет скрытия «сложных» деталей реализации

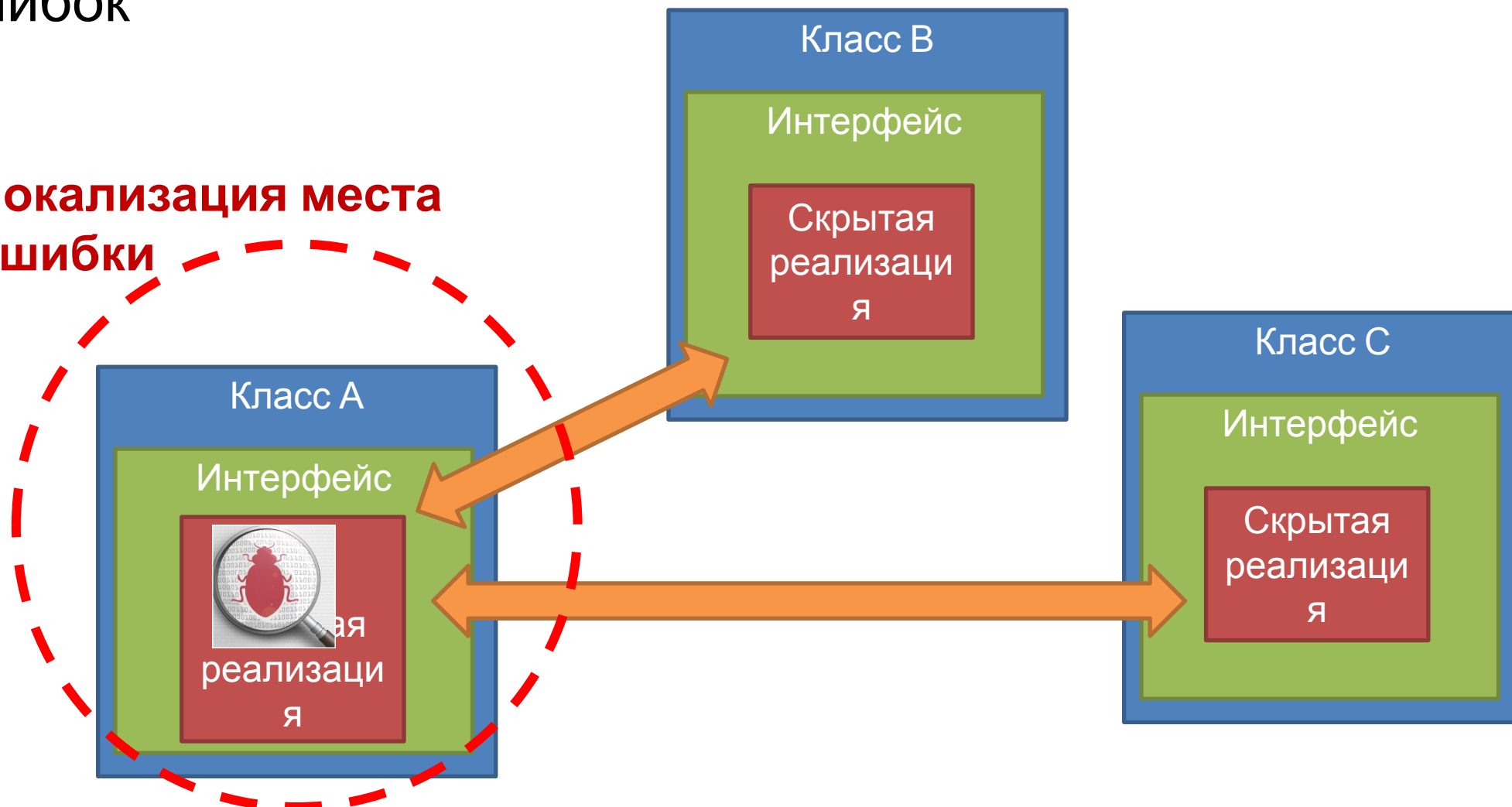


Следствие 2: защита внешнего кода от ошибок

Обеспечивает защиту внутреннего устройства объекта от ошибок



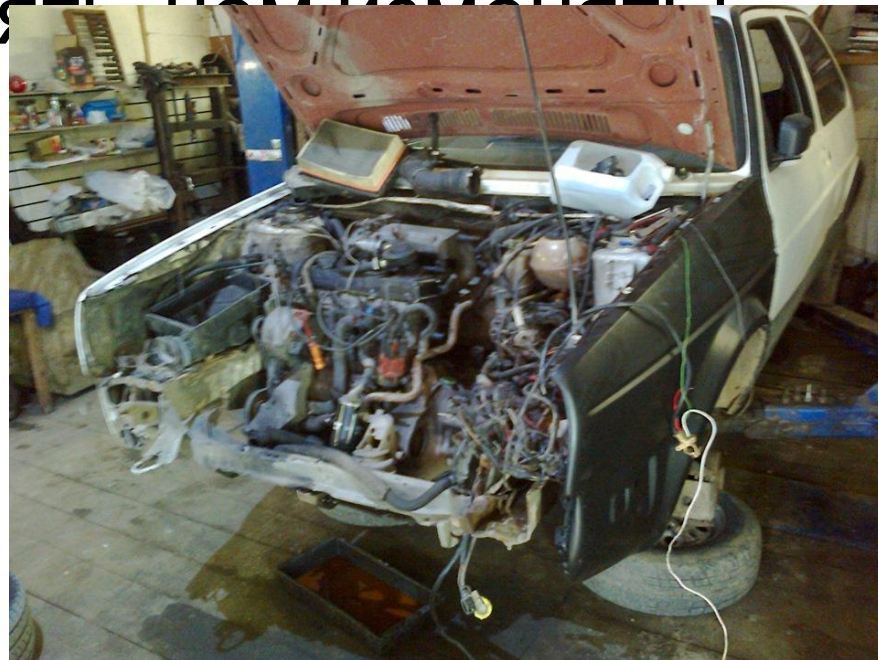
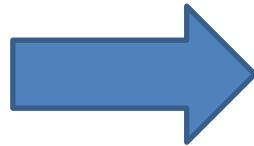
Локализация места ошибки



Следствие 3: легкое изменение системы

При инкапсуляции мы можем **легко изменять внутреннюю реализацию классов** (модулей) системы (**гибкость**)

Интерфейс проще расширять, чем изменить



Пример: изменение алгоритмов поиска Google

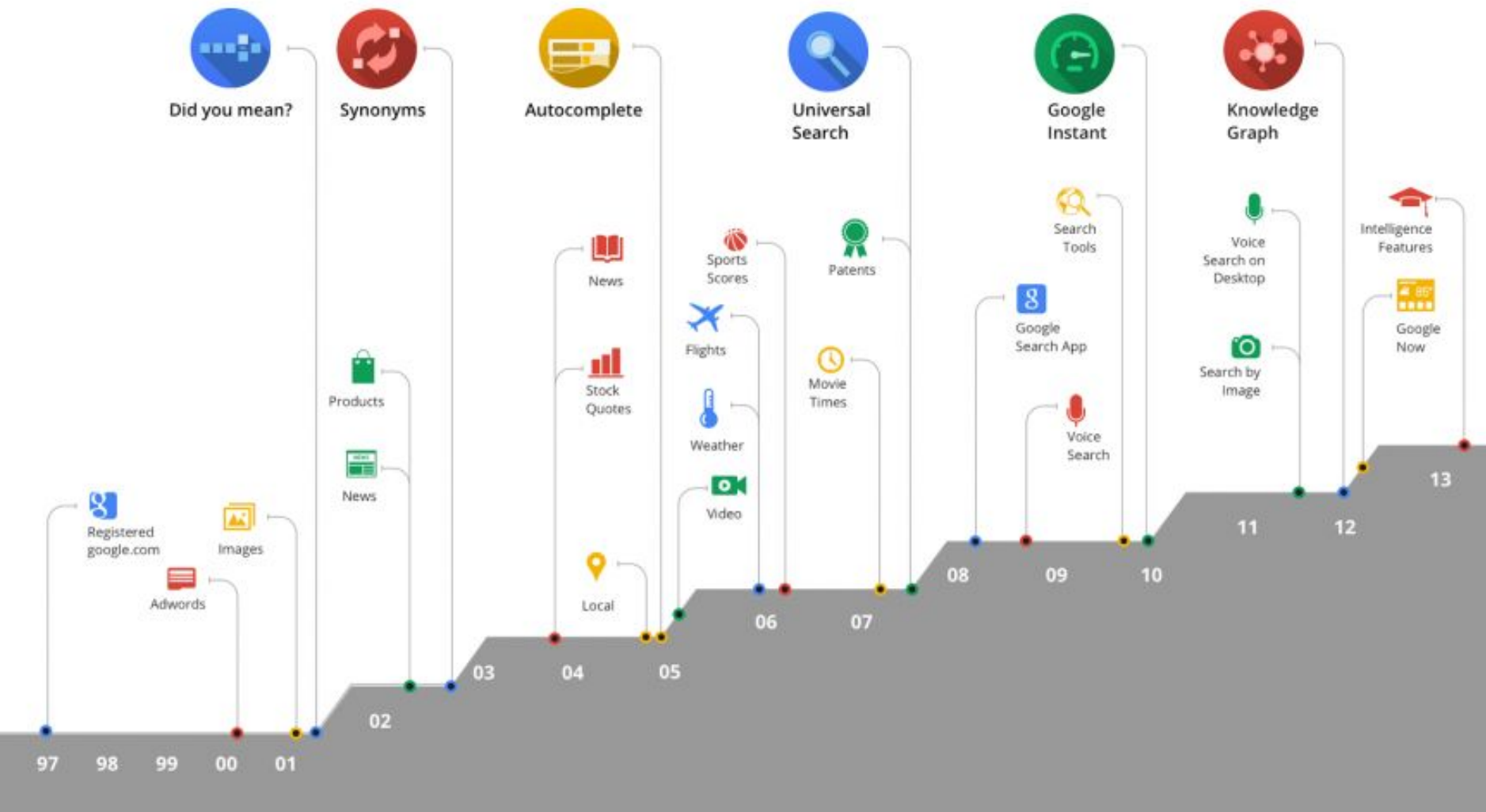
Google Search Timeline

Google

Интерфе
йс
Google

Search bar with "Say 'Ok Google'" and microphone icon

Buttons: "Google Search" and "I'm Feeling Lucky"



Резюме: рассмотренные вопросы

- Почему знание о внутреннем устройстве объекта является проблемой при его использовании?
- Что такое интерфейс? Каким должен быть интерфейс у класса? Как связано с интерфейсом понятие контракта?
- В чем заключается принцип инкапсуляции?
- Как реализуется отделение интерфейса от реализации?
- Как обозначаются модификаторы видимости в языке UML?
- Какие следствия дает грамотное применение инкапсуляции?