



Российский университет дружбы народов
Институт гостиничного бизнеса и туризма

В. Дихтяр

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

(для бакалавров)

Раздел 1. Разработка алгоритмов и программ

Тема 1-5. *Концепция объектно-ориентированного программирования*

Москва 2016

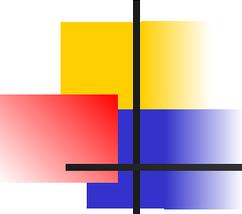
Базовые понятия

- **Объект \hat{O}**
- **свойство \hat{O}**
- **метод обработки**
- **событие**
- **класс $\hat{O} = cls \hat{O}$**

\hat{O} — совокупность свойств определенных сущностей и методов их обработки (*n*-средств).

Свойство — характеристика (параметр) \hat{O} .

$\{\hat{O}\}$ наделены определенными свойствами, которые в совокупности выделяют \hat{O} из множества других \hat{O} .



Пример

- \hat{O} : перечисление свойств:
 \hat{O}_A (*свойство-1, свойство-2, ..., свойство-k*).
- Свойства \hat{O} различных *cls* могут «пересекаться» \Rightarrow
возможны \hat{O} , обладающие одинаковыми свойствами:

\hat{O}_B (...*свойство-n, свойство-m, ... свойство-r, ...*)

\hat{O}_C (...*свойство-n, ..., свойство-r, ...*).



Абстракция (1)

≡ способность отображать \hat{O} внешнего мира в форме абстрактных \hat{S} (структур) в соответствии с решаемой задачей.

Абстрактные структуры, при помощи которых реализуется этот принцип, называются *cls*.

cls ≡ структура, описывающая \hat{O} внешнего мира с помощью двух типов элементов:

состояние \hat{O} описывается полями класса (переменными разного типа), а

поведение \hat{O} — его методами (процедурами и функциями).



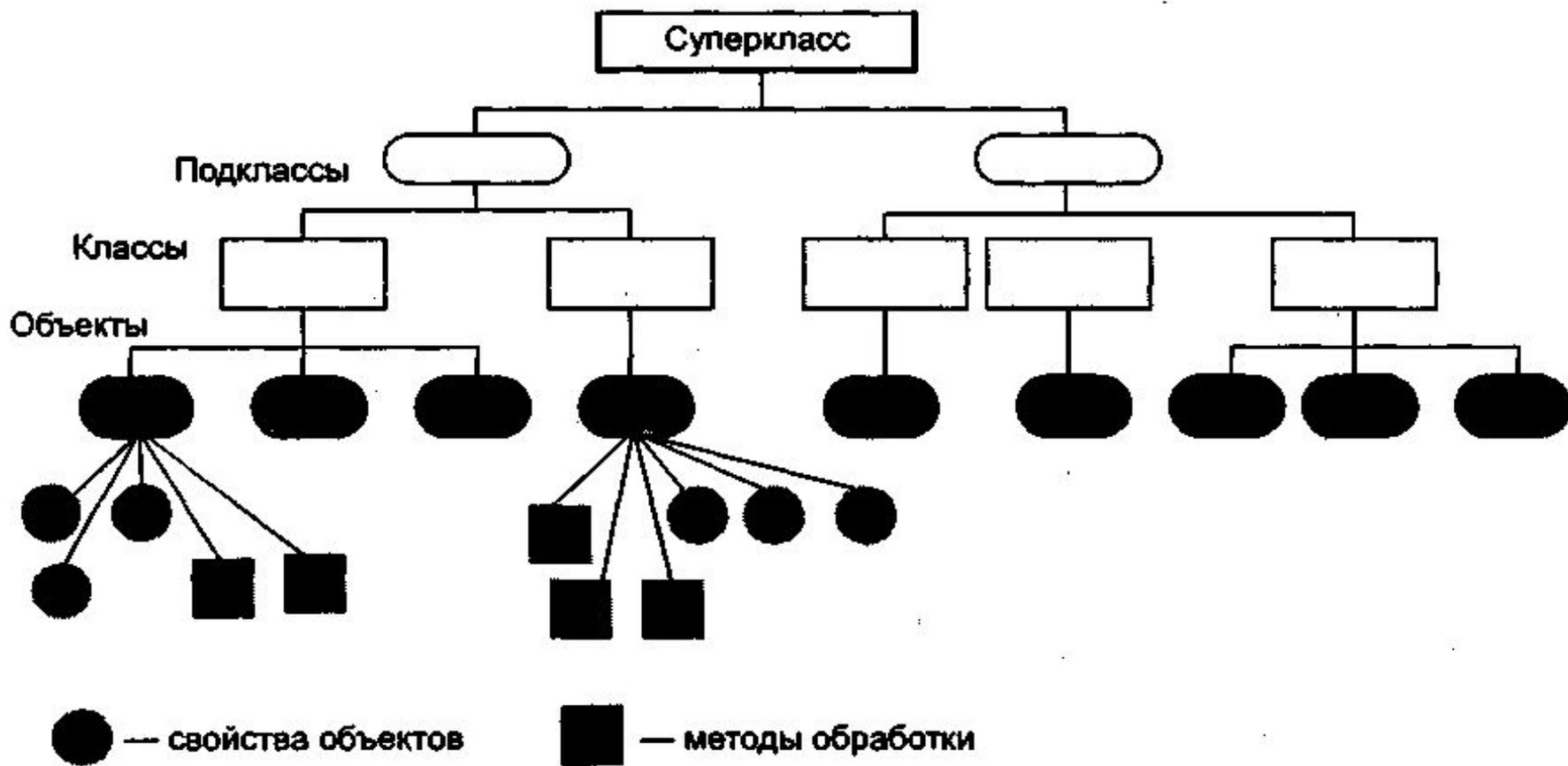
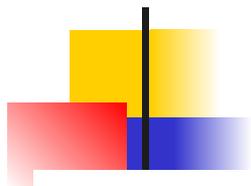
Абстракция (2)

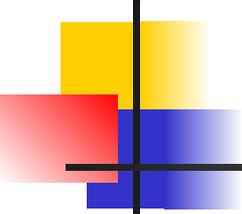
cls - абстрактные описания структур \mathcal{D} , но сами \mathcal{D} они не содержат.

\mathcal{D} появляются тогда, когда по описаниям *cls* выделяется необходимое пространство и в нем создаются экземпляры *cls*, или \hat{O} .

Тогда для каждого поля *cls* отводится необходимая область памяти и в эту область можно поместить нужное значение.

Связь основных понятий ООР





Методика объектно-ориентированного проектирования

Объектно-ориентированные τ и методики проектирования \mathcal{N} -продуктов (обеспечение выполнения принципов \hat{O} -подхода):

- инкапсуляция (замыкание) свойств \check{D} и \mathcal{N} в \hat{O} ;
- наследование;
- полиморфизм

Инкапсуляция = сочетание $\hat{S}\check{D}$ с методами их обработки в абстрактных типах $\check{D} - cls\hat{O}$

Полиморфизм — способность \hat{O} реагировать на запрос сообразно своему типу (одно и то же имя метода может использоваться для различных $cls\hat{O}$)



Свойства методик объектно-ориентированного проектирования

- \hat{O} описывается как Θ некоторой сущности реального мира;
- \hat{O} , для которых определены места хранения, рассматриваются во взаимосвязи, и применительно к ним создаются n -модули \check{S} .



Объектно-ориентированный анализ

- идентификация \hat{O} и их свойств;
- перечень Q , выполняемых над каждым \hat{O} , в зависимости от его состояния;
- связи между \hat{O} для образования *cls*;
- требования к интерфейсу с \hat{O} .



4 этапа объектно-ориентированного проектирования

- разработка диаграммы аппаратных средств системы обработки \hat{D} , показывающей процессоры, внешние устройства, вычислительные сети и их соединения;
- разработка структуры cls , описывающей связь между cls и \hat{O} ;
- разработка диаграмм \hat{O} , показывающих взаимосвязи с другими \hat{O} ;
- разработка внутренней \hat{S} (n -продукта)

Концепция

объектно-ориентированного программирования

Основные принципы:

- абстракция,
- наследование,
- инкапсуляция,
- полиморфизм.



Наследование

≡ свойство *cls* порождать другие *cls*ы таким образом, что в порожденном *cls* (*cls*-потомке) содержатся все поля и все методы *cls* - родителя (базового *cls*), а также дополнительно собственные поля и методы.

Пример 1

Измерительные приборы: термометр, барометр и весы.

Чтобы описать эти приборы внутри *n₃*, создается три *cls*

| <i>cls</i> «термометр» | <i>cls</i> «весы» | <i>cls</i> «барометр» | |
|------------------------|-------------------|-----------------------|--------------|
| Инв. номер | Инв. номер | Инв. номер | Поля классов |
| Название | Название | Название | |
| Цвет | Цвет | Цвет | |
| Исполнение | Исполнение | Исполнение | |
| Температура | Вес | Давление | |
| Изменить температуру | Изменить вес | Изменить давление | |

Метод классов



Пример 1

Методы *cls* представляют реализацию некоторых действий - описывают поведение *cls*. Например, метод *cls* «термометр», который носит название «измерить температуру», призван помещать значение температуры в поле с именем «температура».

| Экземпляр класса «термометр» | | Экземпляр класса класс «весы» | | Экземпляр класса класс «барометр» | |
|------------------------------|---------|-------------------------------|-----------|-----------------------------------|-----------|
| Инв. номер | 1 | Инв. номер | 3 | Инв. номер | 3 |
| Название | TRM | Название | VSV | Название | DRM |
| Цвет | белый | Цвет | серый | Цвет | хром |
| Исполнение | оконный | Исполнение | напольный | Исполнение | настенный |
| Температура | 10 | Вес | 2 | Давление | 230 |
| Изменить температуру | | Изменить вес | | Изменить давление | |

Пример 2

У разных *cls* есть одинаковые поля: Инв. номер, Название, Цвет, Исполнение.
Наследование: класс Измеритель, в котором повторяющиеся поля,
а затем *cls* Термометр, Весы и Барометр





Инкапсуляция (1)

≡ способность *cls* скрывать от внешнего мира детали внутренней реализации.

Области видимости обозначаются в структуре *cls* специальными служебными словами:

private - все поля и методы, объявленные в этой области видимости, невидимы для внешнего мира,

public - все, объявленное в этой зоне, для внешнего мира открыто.



Инкапсуляция (2)

У *cls* появляется защитный корпус, предохраняющий от излишней перегрузки *n* объявленными именами и методами.

Наружу *cls* экспонирует только то, что составляет его интерфейс и предназначено для внешнего взаимодействия.

Все остальные поля и методы он скрывает внутри, помещая их в область видимости *private*.



Полиморфизм

≡ способность \hat{O} ов выполнять одну и ту же команду разными способами.

Свойства:

- способности трактовать \hat{O} *cls*-наследника как тип базового *cls*
- способности *cls* иметь виртуальные методы.

Особенность виртуального метода - надежная связь с созданным экземпляром *cls*.



Пример

Если добавить в каждый из *cls*ов (в класс Измеритель и каждый из его наследников) метод Сигналить, а в реализации этого метода описали бы для каждого *cls* свой способ подавать сигнал (для термометра — мигать, для весов — издавать звук сирены, а для барометра — свистеть), то мы могли бы всем *О*ам, потомкам класса Измеритель, дать команду: измерители, сигналить!

И каждый из них подал бы сигнал: термометры бы замигали, весы издали звук сирены, а барометры засвистели. Это и есть результат работы виртуальных методов.