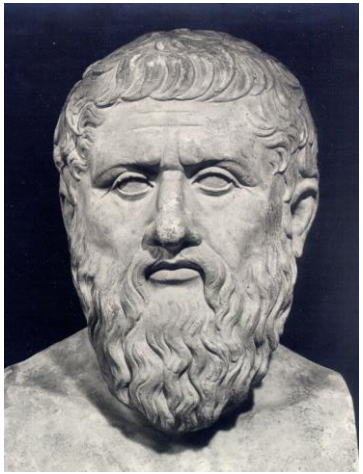
Информатика

Объектно-ориентированное программирование

Принципы на примерах…

Философия: Платон

Во-первых, есть […] **идея**, […] незримая и никак иначе не ощущаемая, но отданная на попечение мысли. Во-вторых, есть *нечто* ***подобное этой идее*** *и носящее то же имя — ощутимое, рождённое, вечно движущееся, возникающее в некоем месте и вновь из него исчезающее* […].
В-третьих, есть […] **пространство**: оно вечно, не приемлет разрушения, дарует обитель всему роду, но само воспринимается вне ощущения, посредством некоего незаконного умозаключения, и поверить в него почти невозможно.



428-348
до н.э.

Kung Fu Panda

*- You can not win! You are just **a** big and fat panda!*

*- No! I am **the** big and fat panda!*

Два приложения

Заказ на разработку двух приложений:

- #1 Система управления договорами
- #2 Текстовая игра

Система управления договорами

Компания оказывает консалтинговые услуги по общим вопросам.
Договор заключается с юр. лицами и физ. лицами. У каждого договора есть предмет, сумма, сроки. Сроки и суммы можно изменять. У договора должен быть статус и ответственный за договор сотрудник. У заказчиков физ. лиц есть ФИО, паспортные данные, прописка; у юр. лиц – наименование, адрес, реквизиты, директор. Договора сохраняются в хранилище с возможностью поиска по заказчикам.

Текстовая игра

Идея текстовой игры для двух игроков. Игроки наносят друг другу удары по очереди. Игроки указывают силу удара от 1 до 9, с увеличением силы возрастает вероятность промахнуться. При успешном ударе у противника уменьшаются очки здоровья (hp). Когда hp одного из игроков становится <=0, игрок проигрывает.

Система управления договорами

Решение понятно:

- типы данных – числа и строки
- хранить можно в файле
- набор договоров – массив

Но есть проблема: что такое договор?

Договор

Набор разнотипных переменных

```
string client; DateTime dueTo; double price;
```

Несколько переменных – несогласованные данные. Нужно хранить вместе

- Нужно хранилище разнотипных данных массив не подходит (однотипные данные)
  - так появились record в Pascal, struct в C
  - но это еще половина проблемы

Недостаток процедурного подхода

Как разработать систему в процедурном стиле?

- Нужно написать методы:
  - создать договор (физ.лицо, предмет)
  - ударить (игрок1, игрок2)

Что за тип физ.лицо или игрок1?

Функции

- Функция (метод) – набор операторов, оператор – действие.
  Функция – тоже действие.
- В русском языке действие – глагол/сказуемое.
- Описание системы на уровне фукнций – описание мира неопределёнными глаголами – несколько ограниченно!

С другой стороны

- Понятны операции и типы данных в обоих примерах
- типы данных

```
string client; DateTime dueTo; double price;
```

- операции

  #1 изменить сумму/строки – присвоить значение

  #2 уменьшение hp - вычитание

Всё вместе

- Нам не нужны новые способы обработки данных
- Нужен новый подход к разработке (новая парадигма)

Анализ требований

- Договор заключается с физ.лицом/юр.лицом
- Сроки меняются
- У договора – статус, сотрудник
- У физ.лица – ФИО, паспорт, прописка
- У юр.лица – наименование, адрес, реквизиты, директор

Анализ требований

- Игроки наносят удары
- Игроки вводят силу удара
- У противника уменьшаются очки здоровья

Данные - существительные

- Договор, срок, сумма, предмет, хранилище, адрес, сотрудник
- Игрок, hp, сила удара

Все – существительные

1. Некоторые – примитивные
2. Некоторые – нет

Не примитивные

Договор, сотрудник, хранилище, игрок…

Не простые!
Вся система строится вокруг них!

- **Данные** хранятся у них
- **Действия** привязаны к ним

Существительное-сущность-объект

Объект

Основная единица (строительный блок) системы

- Содержит в себе данные (статика)
- совершает действия (динамика)

Данные

Статика, состояние объектов

- Какие данные хранятся в объекте?
  - примитивные
  - другие объекты
- Как хранить?
  - объявить переменную
- Данные (переменные) внутри объектов – **поля**

Динамика

Действия, которые совершаются объектами, их поведение

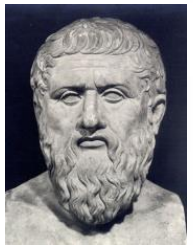
- Процедуры и функции
- Привязаны к объектам – поэтому **методы**
- Методы изменяют значения полей, значит им нужно иметь доступ

Сущности и объекты

Уникальные сущности отличаются от реального количества объектов

- Сущности **– договор, сотрудник, игрок**
- Договором – много, а сотрудников и игроков?

Есть объекты, составляемые по единому подобию, шаблону, каркасу…



Класс

- Пользовательский тип данных
  - Каркас, шаблон, образец, чертёж объекта
- Объект – переменная этого типа
- Объект – экземпляр класса

Итоги и влияние на разработку...

Структурный и процедурный подход

- Структурных подход позволяет строить алгоритмы, но не способствует повторному использованию
- Процедурный подход хорошо описывает процессы/действия, но отстранён от данных
- Вся программа – единый алгоритм

Программирование в малом

- Программу может написать один человек
- Программу можно переписать для
  - исправления/расширения,
  - портирования на новую платформу
- Подходит для небольших систем

Системная сложность

Не алгоритмическая сложность

- Большое количество независимых компонент
- Сложные связи между компонентами
- Одному человеку не разобраться
- Нужно бороться со сложностью, держать сложность «под контролем»

Программирование в большом

- Разрабатывается командой
- Никто не знает всех деталей реализации программы
- Продолжительный срок «жизни» программы
- Нельзя просто так переписать, расширить, исправить
- Требует организации процесса разработки
- Требует использования подходящих методов и инструментов разработки

Требования к системам

- Эффективность (надёжность, производительность)
- Гибкость (изменяемость)
- Расширяемость (добавление сущностей)
- Масштабируемость процесса (добавление людей)
- Тестируемость (возможность проверки)
- Возможность повторного использования
- Сопровождаемость (легко разобраться в программе)

Архитектура ПО

Правила, эвристики и паттерны для

- проектирования системы как нескольких частей
- создание интерфейсов взаимодействия этих частей
- контроля над общей структурой и потоком управления
- взаимодействия с окружением
- соответствующего использования подходов, техник и инструментов разработки

ООП

Объектно-ориентированное программирование

- развитие идей структурного, процедурного и модульного программирования
- поддержка «программирования в большом»
- *ориентировано* на архитектуру ПО

ОО разработка

- Описание системы
- Выделение подсистем
- Разделение подсистем на классы
- Определение взаимодействия
- Проектирование классов
- Тестирование
- Внедрение

Let’s code…

Вопросы?

e-mail: marchenko@it.kfu.ru