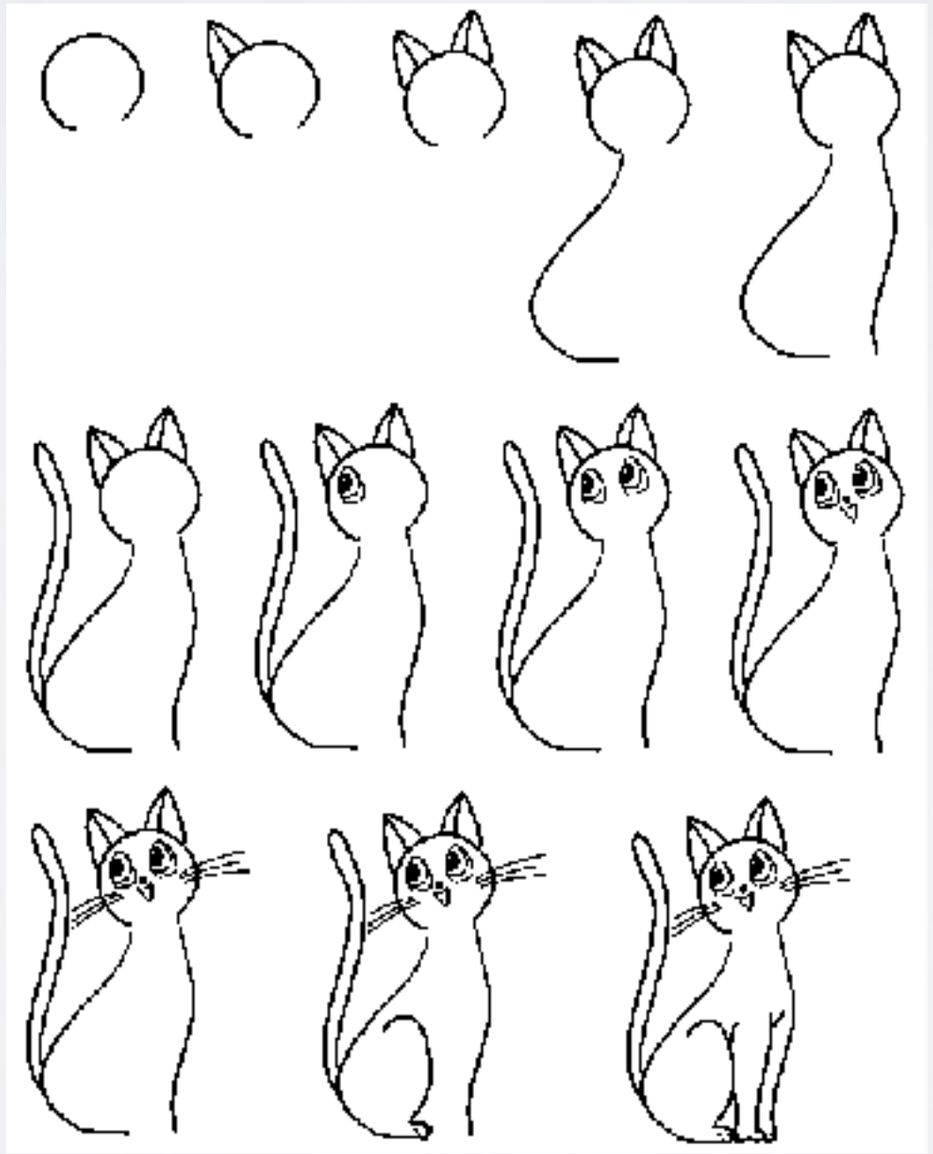
ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Лекция № I / **13**
30.04.2018 г.

ПРАКТИЧЕСКОЕ РУКОВОДСТВО



I. ПЛАНИРОВАНИЕ

- Отделяем то, что будем делать сейчас и в ближайшее время, от планов на будущее.

- Лучше всего браться за нечто *простое, но цельное*.

- 90 % сил — сделать хорошо то, что «есть»
  10 % сил — «зарубки» на будущее.

МЕТАФОРИЧЕСКИ...

```
// На данном этапе важно уловить суть решаемой задачи и основные интерфейсы
System system = new System();

// А теперь добавляем фичи так, что System не пришлось сильно переделывать
system += feature1();
system += feature2();

// и т.д.
```

- **Принцип контактирующих сферических коней в вакууме.**

- Мы стремимся сделать каждый модуль по возможности максимально независимым от других.
- Независимость упрощает разработку, отладку и тестирование.
- …Но создаётся модуль для того, чтобы взаимодействовать с другими, и для этого он должен иметь удобный интерфейс.

2. КЛАССЫ И ОБЪЕКТЫ

- Выделяем основополагающие объекты.
- *…если не получается: берем описание проекта и смотрим на самые часто встречающиеся существительные.*
- Смотрим на связи (наследование, композиция, связь через *другой* объект, …)
- Ищем понятную *простую, но цельную* подсистему, с которой и начинаем, потом ищем еще одну…

- **Каково время жизни объектов данного класса?** (эвристика: 90 % объектов имеют очень малое время жизни).

- **Тяжёлый или лёгкий интерфейс?** Копируем данные или используем указатели/ссылки? (эвристика: 90 % объектов — лёгкий интерфейс).

- **Как удобнее всего будет вызывать методы?** Так и нужно их запроектировать.

3. ЯДРО И ПРОТОТИП

- Признаки «ядерности»:
  - Ключевые алгоритмы.
  - Наиболее широко используемые абстракции.
  - Нет аддитивности. Убираем модуль — все рушится.
- Признаки периферийности:
  - Частные случаи («поддержка 100 форматов файлов»).
  - Есть аддитивность. Убрали модуль — общий функционал чуть-чуть уменьшился.

На первом этапе нужно вкладываться
в ядро и довольствоваться
простой периферией!

Ядро (прототип) +
простенькая периферия =
первая версия системы!

ЗАДАЧА ПРО ПАРСИНГ CSV

- Модули:
  - Парсер строк (из `string` в `vector<string>`).
  - Лёгкий интерфейс `Row`.
  - Поддержка строки с заголовками.

ЗАТЫЧКИ

- При объявленном полном **интерфейсе** класса реализация является:
  - пустой;
  - неполной (частный случай);
  - медленной;
  - приближенной;
  - …но зато пишется мгновенно!

4. ДАЛЬНЕЙШЕЕ РАЗВИТИЕ

- Переход от взрывного роста к эволюционному.
- Любые изменения не должны ухудшать работоспособность системы, качество кода и документации, нарушать конвенции.
- **Либо** мы находим способ интегрировать новое, **либо** делаем отдельную систему.

ВЫЯВЛЕНИЕ «БЯКИ»

- Реализация «бяки» плоха. **переписать!**
- Реализация «бяки» сама по себе хороша, но при включении в систему нарушаются принципы ООП. **рефакторинг!**
- «Бяка» ставит под сомнение какой-то фундаментальный принцип или ограничение, заложенное в ядро. **проектирование заново с учетом переделок**

САМОДОКУМЕНТИРУЕМОСТЬ

- Работа сознания человека:

  имя → ассоциативные поля → понимание → действие.

- Имя переменной: смысл ее значения.

- Имя функции/метода: суть ее поведения.

- Имя класса: смысл в общем контексте.

- **Если смысл меняется, должно меняться и имя!**

МОДИФИЦИРУЕМОСТЬ

- 100 % кода должно быть модифицируемым.
- Какой-то код придется выбрасывать в любом случае, нужно относиться к этому спокойно.
- «Код на выброс» — не боимся экспериментировать.


anapakurort.info

Планирование

Выявление классов

Прототип

«Долизывание»

КОНЕЦ ТРИНАДЦАТОЙ ЛЕКЦИИ