Объектная модель: семь компонентов

Основы объектно-ориентированного подхода

Александра Волосова,
к.т.н., доцент кафедры
ИУ5

Презентация разработана в рамках гранта «Грант на обучение студентов по образовательным программам высшего образования для топ-специалистов в сфере информационных технологий. Договор № 70-2025-000850 с АНО «Аналитический центр при Правительстве Российской Федерации»

Основные компоненты объектной модели

1. Абстрагирование
2. Инкапсуляция
3. Модульность
4. Иерархия

Без любого из них модель не будет объектно-ориентированной

Дополнительные компоненты

5. Типизация
6. Параллелизм
7. Сохраняемость

Полезны, но не обязательны
Обогащают объектную модель

Абстрагирование: определение

- Выделение существенных характеристик объекта
- Игнорирование несущественных деталей
- Определение концептуальных границ
- Упрощенное описание системы
- Барьер между смыслом и реализацией

Принципы хорошей абстракции

- Принцип минимальной связанности
- Принцип наименьшего удивления
- Контрактная модель программирования
- Предусловия и постусловия операций
- Инварианты как логические условия
- Четкое разделение между интерфейсом и реализацией

Типы абстракций

- Абстракция сущности (объекты предметной области)
- Абстракция поведения (операции)
- Абстракция виртуальной машины (группы операций)
- Произвольная абстракция (несвязанные операции)

Цель: абстракции сущности

Клиент-серверная модель

- Клиент: объект, использующий ресурсы другого
- Сервер: объект, предоставляющий услуги
- Операции как услуги сервера
- Протокол: все возможные способы взаимодействия
- Статические и динамические аспекты

Пример абстракции: TemperatureSensor

```
class TemperatureSensor {
  public:
    TemperatureSensor(Location);
    ~TemperatureSensor();
    void calibrate(Temperature
actualTemperature);
    Temperature currentTemperature() const;
  private:
    ...
};
```

- Абстракция датчика температуры
- Интерфейс: конструктор, калибровка, чтение
- Скрытая реализация

Пример: ActiveTemperatureSensor

```
class ActiveTemperatureSensor {
  public:
    ActiveTemperatureSensor(Location,
      void (*f)(Location, Temperature));
    ~ActiveTemperatureSensor();
    void establishSetpoint(Temperature setpoint,
      Temperature delta);
    Temperature currentTemperature() const;
  private:
    ...
};
```

- Активный датчик с обратным вызовом
- Функция обратного вызова как параметр
- Автоматическое оповещение при отклонениях

Пример: GrowingPlan (план выращивания)

- Абстракция плана выращивания культур
- Таблица: время → условия (температура, освещение)
- Координация действий во времени
- Разделение планирования и исполнения
- Интерфейс для установки и запроса плана

Код: GrowingPlan на C++

```
class GrowingPlan {
  public:
    GrowingPlan(char* name);
    virtual ~GrowingPlan();
    void clear();
    virtual void establish(Day, Hour, const
Condition&);
    const char* name() const;
    const Condition& desiredConditions(Day, Hour)
const;
  protected:
    ...
};
```

- Виртуальные функции для переопределения
- Защищенные члены для подклассов
- Инкапсуляция реализации

Вспомогательные типы данных

```
typedef unsigned int Day;
typedef unsigned int Hour;
enum Lights {OFF, ON};
typedef float pH;
typedef float Concentration;

struct Condition {
  Temperature temperature;
  Lights lighting;
  pH acidity;
  Concentration concentration;
};
```

- Приближение к словарю предметной области
- Структура вместо класса для простого объединения
- Повышение читаемости кода

Инкапсуляция: определение

- Разделение интерфейса и реализации
- Сокрытие внутреннего устройства объекта
- «Никакая часть системы не должна зависеть от внутреннего устройства другой» (Ингалс)
- Тайны абстракции (Бритон и Парнас)
- Локализация возможных изменений

Пример инкапсуляции: класс Heater

```
class Heater {
  public:
    Heater(Location);
    ~Heater();
    void turnOn();
    void turnOff();
    Boolean isOn() const;
  private:
    const Location repLocation;
    Boolean repIsOn;
    SerialPort* repPort;
};
```

- Публичный интерфейс: включение/выключение
- Приватная реализация: состояние и порт
- Изменение реализации без изменения интерфейса

Реализация методов Heater

```
void Heater::turnOn() {
  if (!repIsOn) {
    repPort->write("**");
    repPort->write(repLocation);
    repPort->write(1);
    repIsOn = TRUE;
  }
}
```

- Проверка состояния перед действием
- Последовательная передача команд
- Изменение внутреннего состояния
- Сокрытие протокола общения с аппаратурой

Управление доступом в разных языках

- Smalltalk: защита на этапе компиляции
- Object Pascal: отсутствие инкапсуляции представления
- CLOS: атрибуты :reader, :writer, :accessor
- C++: public, private, protected, friend
- Ada: разделение спецификации и тела пакета

Модульность: определение

- Разделение программы на компилируемые отдельно части
- Связи между модулями — их представления друг о друге
- Физические контейнеры для классов и объектов
- Отделение интерфейса от реализации
- Управление зависимостями

Модульность и ее принципы

- Физическое разделение программы на компилируемые отдельно части
- Физические контейнеры для классов и объектов
- Принцип сокрытия изменяемых частей системы
- Использование только стабильных связей между модулями
- Баланс между сокрытием информации и её видимостью

Поддержка модульности в языках

- C++: файлы .h и .cpp
- Object Pascal: units с интерфейсом и реализацией
- Ada: packages (спецификация и тело)
- Smalltalk: нет модулей, только классы
- Java: packages и модульная система

Пример модуля на C++

```
// gplan.h
#ifndef _GPLAN_H
#define _GPLAN_H 1

#include "gtypes.h"
#include "except.h"
#include "actions.h"

class GrowingPlan { ... };
class FruitGrowingPlan { ... };
class GrainGrowingPlan { ... };

#endif
```

- Защита от множественного включения
- Импорт зависимостей
- Объявление интерфейсов классов
- Физическое разделение кода

Принципы модульности

- Скрывать изменяемые части системы
- Использовать только стабильные межмодульные связи
- Баланс между сокрытием информации и видимостью
- Избегать чрезмерного дробления
- Минимизировать перекомпиляцию

- Объектная модель состоит из семи компонентов

Основные элементы*:

1. **Абстрагирование**
2. **Инкапсуляция**
3. **Модульность**
4. **Иерархия**

Дополнительные элементы:

5. **Типизация**
6. **Параллелизм**
7. **Сохраняемость**