

# Лекция 6/Практика 4: Структурные шаблоны

Юрий Литвинов  
y.litvinov@spbu.ru

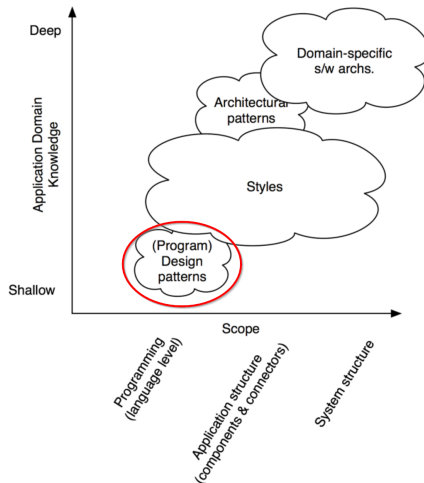
11.04.2023

# Паттерны проектирования

**Шаблон проектирования** — это повторяемая архитектурная конструкция, являющаяся решением некоторой типичной технической проблемы

- ▶ Подходит для класса проблем
- ▶ Обеспечивает переиспользуемость знаний
- ▶ Позволяет унифицировать терминологию
- ▶ В удобной для изучения форме
- ▶ НЕ конкретный рецепт или указания к действию

# Паттерны и архитектурные стили



# Книжка про паттерны

Must read!

Приемы объектно-ориентированного проектирования. Паттерны проектирования

Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес

Design Patterns: Elements of Reusable Object-Oriented Software



# Начнём с примера

## Текстовый редактор

WYSIWYG-редактор, основные вопросы:

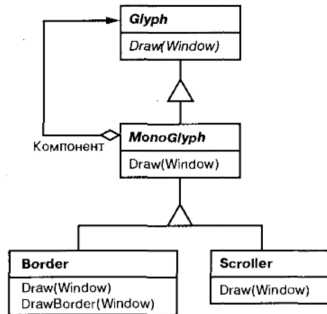
- ▶ Структура документа
- ▶ Форматирование
- ▶ Создание привлекательного интерфейса пользователя
- ▶ Поддержка стандартов внешнего облика программы
- ▶ Операции пользователя, undo/redo
- ▶ Проверка правописания и расстановка переносов

## Усовершенствование UI

- ▶ Хотим сделать рамку вокруг текста и полосы прокрутки, отключаемые по опции
- ▶ Желательно убирать и добавлять элементы оформления так, чтобы другие объекты даже не знали, что они есть
- ▶ Хотим менять во время выполнения — наследование не подойдёт
  - ▶ Наш выбор — композиция
  - ▶ Прозрачное оформление

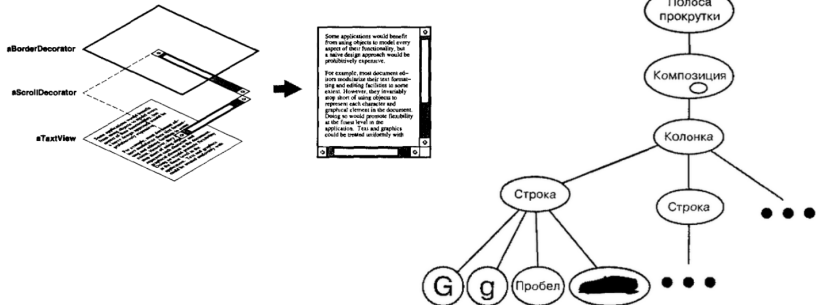
# Моноглиф

- ▶ Абстрактный класс с ровно одним сыном
  - ▶ Вырожденный случай компоновщика
- ▶ “Обрамляет” сына, добавляя новую функциональность



© Э. Гамма и др., Приемы  
объектно-ориентированного  
проектирования

# Структура глифов

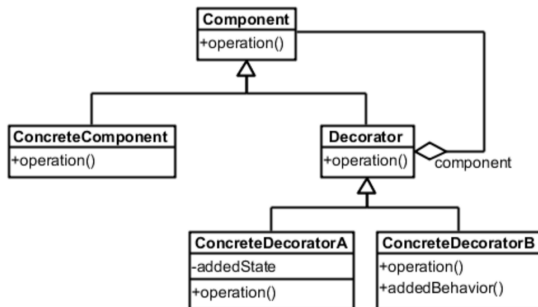


© Э. Гамма и др., Приемы объектно-ориентированного проектирования



# Паттерн “Декоратор”

Decorator

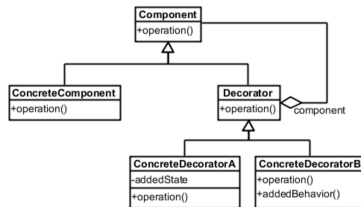


## Декоратор, особенности

- ▶ Динамическое добавление (и удаление) обязанностей объектов
  - ▶ Большая гибкость, чем у наследования
- ▶ Позволяет избежать перегруженных функциональностью базовых классов
- ▶ Много мелких объектов

# “Декоратор” (Decorator), детали реализации

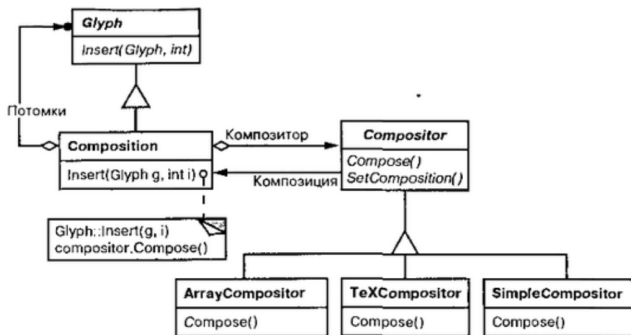
- ▶ Интерфейс декоратора должен соответствовать интерфейсу декорируемого объекта
  - ▶ Иначе получится “Адаптер”
- ▶ Если конкретный декоратор один, абстрактный класс можно не делать
- ▶ Component должен быть по возможности небольшим (в идеале, интерфейсом)
  - ▶ Иначе лучше паттерн “Стратегия”
  - ▶ Или самодельный аналог, например, список “расширений”, которые вызываются декорируемым объектом вручную перед операцией или после неё



## Форматирование текста

- ▶ Задача — разбиение текста на строки, колонки и т.д.
- ▶ Высокоуровневые параметры форматирования
  - ▶ Ширина полей, размер отступа, межстрочный интервал и т.д.
- ▶ Компромисс между качеством и скоростью работы
- ▶ Инкапсуляция алгоритма

# Compositor и Composition

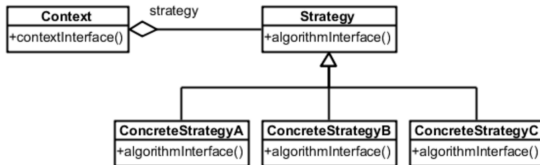


© Э. Гамма и др., Приемы объектно-ориентированного проектирования

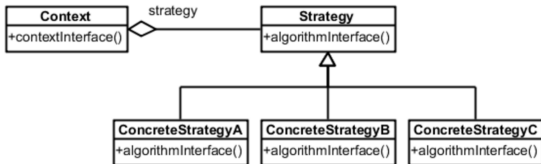
# Паттерн “Стратегия”

## Strategy

- ▶ Назначение — инкапсуляция алгоритма в объект
- ▶ Самое важное — спроектировать интерфейсы стратегии и контекста
  - ▶ Так, чтобы не менять их для каждой стратегии
- ▶ Применяется, если
  - ▶ Имеется много родственных классов с разным поведением
  - ▶ Нужно иметь несколько вариантов алгоритма
  - ▶ В алгоритме есть данные, про которые клиенту знать не надо
  - ▶ В коде много условных операторов



# “Стратегия” (Strategy), детали реализации



- ▶ Передача контекста вычислений в стратегию
  - ▶ Как параметры метода — уменьшает связность, но некоторые параметры могут быть стратегии не нужны
  - ▶ Передавать сам контекст в качестве аргумента — в **Context** интерфейс для доступа к данным

## “Стратегия” (Strategy), детали реализации (2)

- ▶ Стратегия может быть параметром шаблона
  - ▶ Если не надо её менять на лету
  - ▶ Не надо абстрактного класса и нет оверхеда на вызов виртуальных методов
- ▶ Стратегия по умолчанию
  - ▶ Или просто поведение по умолчанию, если стратегия не установлена
- ▶ Объект-стратегия может быть приспособленцем



## Задачи на остаток пары

Уточнить модель компьютерной игры Roguelike с предыдущего занятия:

1. Используя шаблон “Стратегия” для поддержки различных поведений мобов
  - ▶ Агрессивное поведение, атакуют игрока, как только его видят
  - ▶ Пассивное поведение, просто стоят на месте
  - ▶ Трусливое поведение, стараются держаться на расстоянии от игрока
2. Используя шаблон “Декоратор” для поддержки временных эффектов, накладываемых на мобов и игрока
  - ▶ Эффект конфузии, заставляющий персонажа двигаться в случайном направлении
  - ▶ Возможность добавить другие похожие эффекты

Выложить модифицированные диаграммы как решение задания в Teams