

Шаблонное проектирование

Выполнила студентка

группы ИВТ-32

Залюбовина Мария

Шаблон проектирования

- Формализованное описание часто встречающейся задачи проектирования, удачное решение данной задачи и рекомендации по применению этого решения в различных ситуациях

Преимущества

- Снижение сложности разработки
- Упрощение коммуникации
- Правильно сформулированный шаблон позволяет пользоваться им снова и снова
- Набор шаблонов помогает разработчику выбрать наиболее подходящий вариант проектирования

Недостатки

- Может привести к усложнению программы
- Необоснованное применение шаблона

Классификация

- Архитектурные паттерны
- Паттерны проектирования
- Паттерны анализа
- Паттерны тестирования
- Паттерны реализации

Архитектурные паттерны

- Описывают структурную схему системы в целом

Паттерны проектирования

- Описывают схемы детализации программных подсистем и отношений между ними.

Паттерны анализа

- Представляют общие схемы организации процесса объектно-ориентированного моделирования.

Паттерны тестирования

- Определяют общие схемы организации процесса тестирования программных систем

Паттерны реализации

- Описывают шаблоны, которые используются при написании программного кода.

ИДИОМЫ

- Шаблоны, описывающие типичные решения на конкретном языке программирования.

ИДИОМЫ

- Инкремент:
`inc(i);`
`i++;`
- Обмен значениями:
`temp = a;`
`a = b;`
`b = temp;`
- Бесконечный цикл:
`while True:`
`do_something()`
`for (;;) {`
`do_something(); }`

Классификация

- **Порождающие шаблоны** – предназначены для создания новых объектов в системе
- **Структурные шаблоны** – решают задачи компоновки системы в виде классов и объектов
- **Шаблоны поведения** – предназначены для распределения обязанностей между объектами в системе

Порождающие шаблоны

- Фабричный метод
- Абстрактная фабрика
- Строитель
- Прототип
- Одиночка
- Пул объектов

Фабрика объектов

```
1  class Warrior
2  {
3      public:
4          virtual void info() = 0;
5          virtual ~Warrior() {}
6  };
7
8  class Infantryman: public Warrior
9  {
10     public:
11         void info() { cout << "Infantryman" << endl; }
12     };
13
14     class Archer: public Warrior
15     {
16         public:
17             void info() { cout << "Archer" << endl; }
18     };
19
20     class Horseman: public Warrior
21     {
22         public:
23             void info() { cout << "Horseman" << endl; }
24     };

```

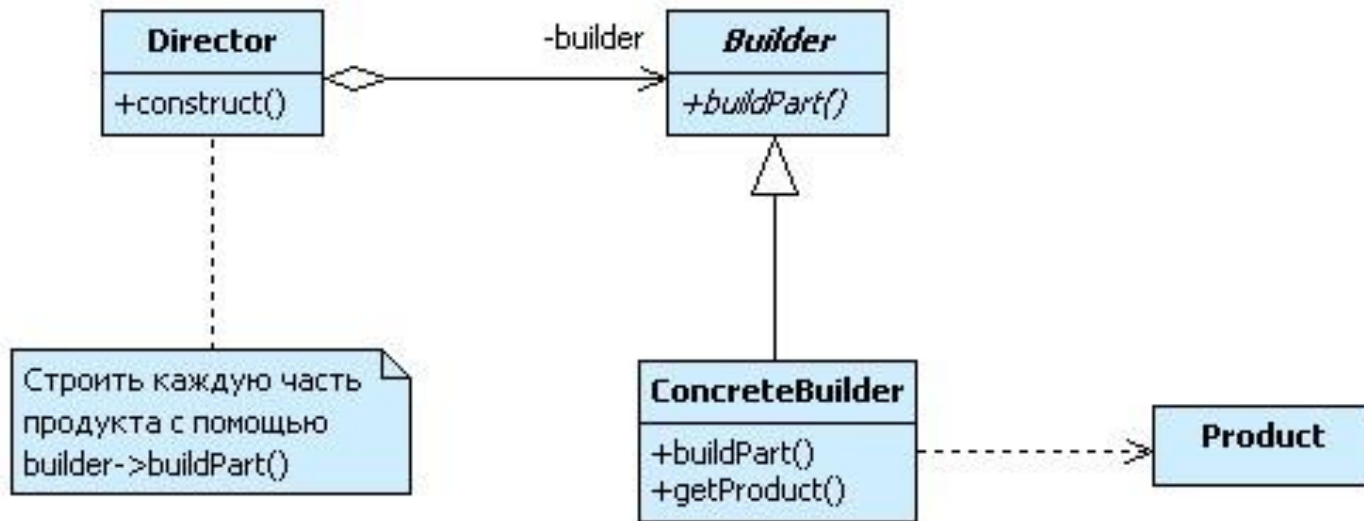
Фабрика объектов

```
1  enum Warrior_ID { Infantryman_ID=0, Archer_ID, Horseman_ID };
2
3  Warrior * createWarrior( Warrior_ID id )
4  {
5      Warrior * p;
6      switch (id)
7      {
8          case Infantryman_ID:
9              p = new Infantryman();
10             break;
11             case Archer_ID:
12                 p = new Archer();
13                 break;
14                 case Horseman_ID:
15                     p = new Horseman();
16                     break;
17                 default:
18                     assert( false);
19             }
20             return p;
21     }
```


Строитель

- В системе могут существовать сложные объекты, создание которых за одну операцию затруднительно или невозможно. Требуется поэтапное построение объектов с контролем результатов выполнения каждого этапа.

Строитель



Одиночка

- Контролирует создание единственного экземпляра некоторого класса и предоставляет доступ к нему.

Одиночка

Singleton
<u>-instance: Singleton</u>
-Singleton() <u>+getInstance(): Singleton</u>

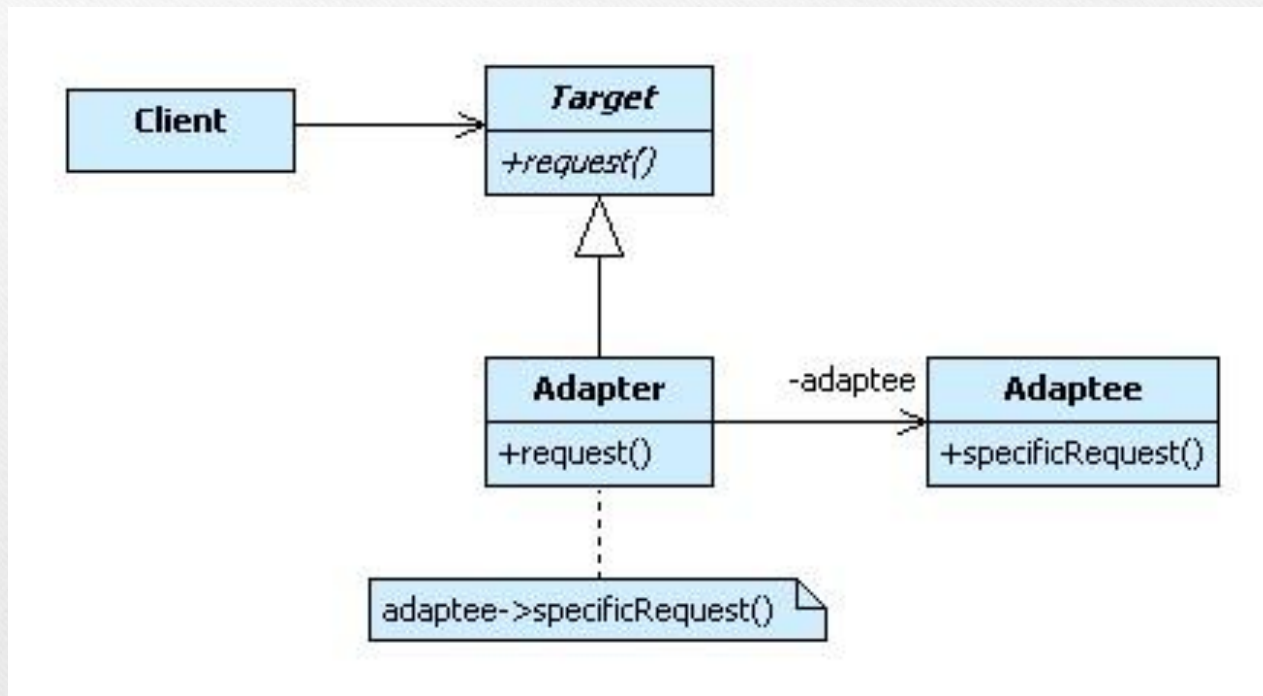
Структурные шаблоны

- Адаптер
- Мост
- Компоновщик
- Декоратор
- Фасад
- Приспособленец
- Заместитель

Адаптер

- Представляет собой программную обертку над уже существующими классами и предназначен для преобразования их интерфейсов к виду, пригодному для последующего использования в новом программном проекте.

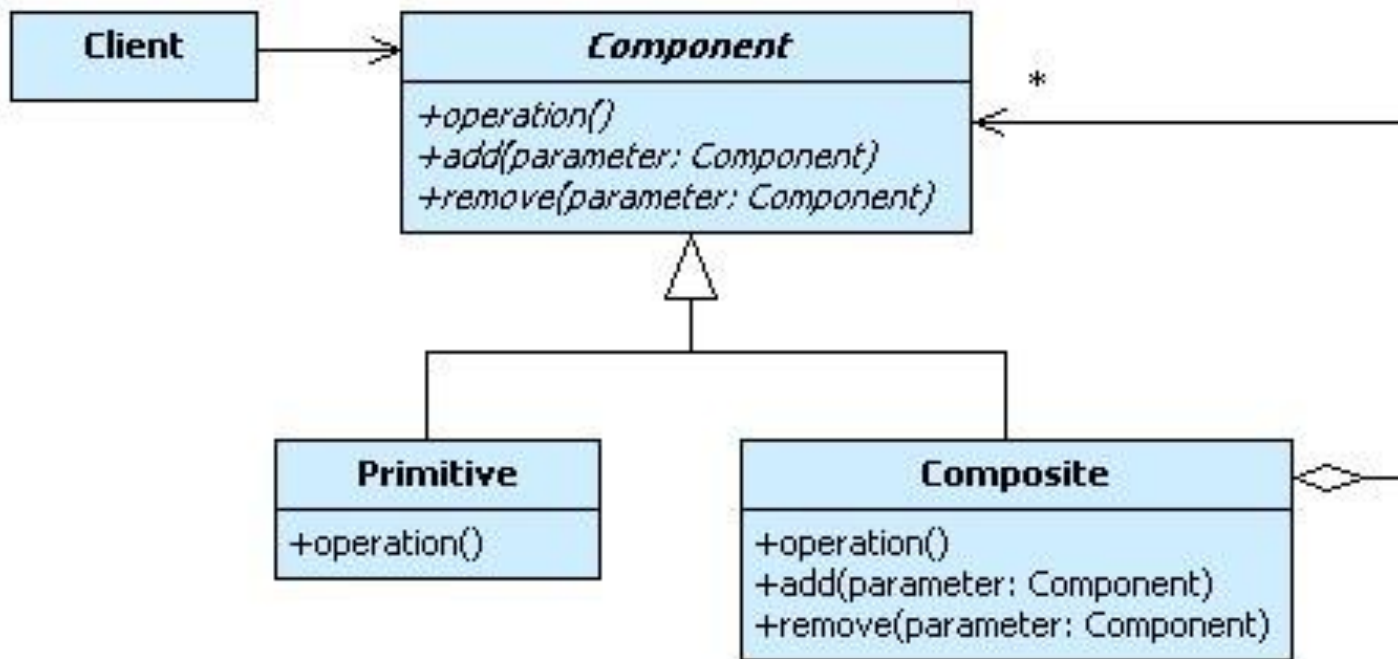
Адаптер



КОМПОНОВЩИК

- Используется для расширения функциональности объектов. Являясь гибкой альтернативой порождению классов, паттерн Decorator динамически добавляет объекту новые обязанности.

КОМПОНОВЩИК



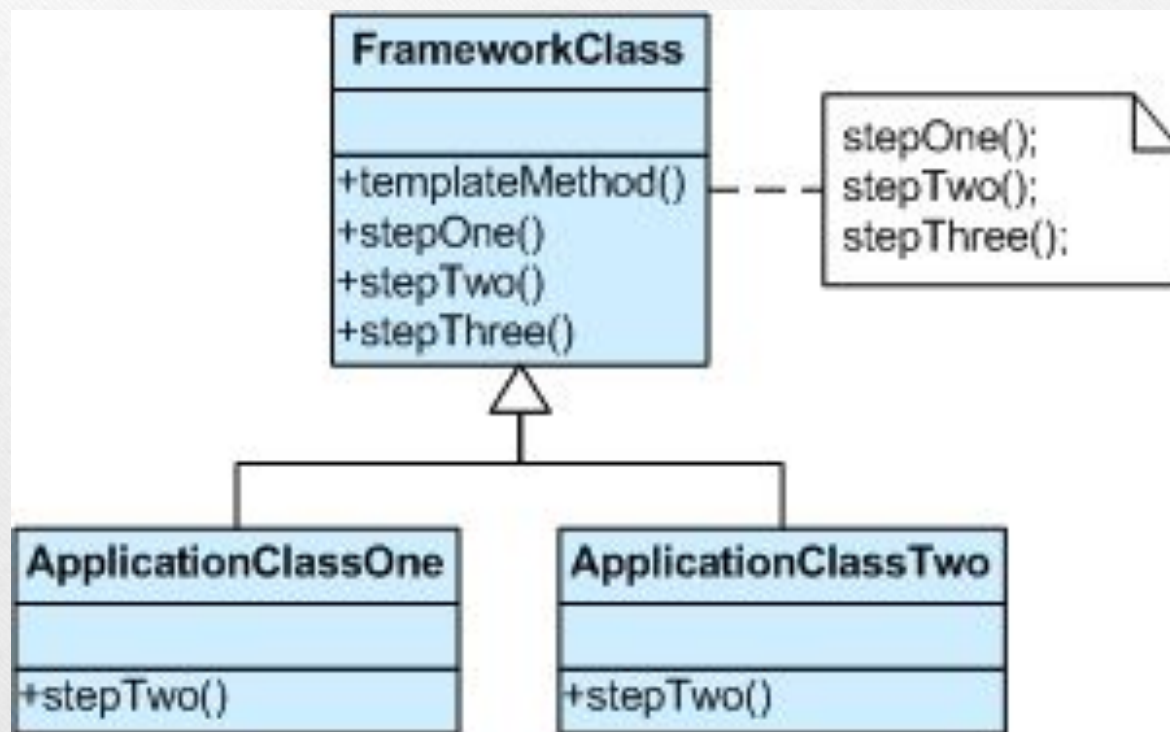
Шаблоны поведения

- Цепочка обязанностей
- Команда
- Итератор
- Интерпретатор
- Посредник
- Хранитель
- Наблюдатель
- Состояние
- Стратегия
- Шаблонный метод
- Посетитель

Шаблонный метод

- Определяет основу алгоритма и позволяет подклассам изменить некоторые шаги этого алгоритма без изменения его общей структуры.

Шаблонный метод



Антипаттерны

- Golden hammer
- Hard code
- Magic numbers
- Programming by permutation
- Blind faith

Литература

- Мартин Р. // Принципы, паттерны и методики гибкой разработки на языке С#. – СПб.: Символ-Плюс, 2011.
- <http://cpp-reference.ru/patterns>