Основное назначение паттернов

|  | Паттерн | Уровень | F | Назначение |
| --- | --- | --- | --- | --- |
|  | **Порождающие** | | | |
|  | Abstract Factory  Абстрактная фабрика  Kit (Набор инструментов) | Объект | 5 | Порождение семейств взаимодействующих продуктов.  Предоставляет клиенту интерфейс (набор методов) для создания семейств взаимосвязанных или взаимозависимых объектов-продуктов, при этом скрывает от клиента информацию о конкретных классах создаваемых объектов-продуктов  *Рекомендуемые случаи применения*:   * Требуется создавать объекты-продукты разных типов и налаживать между ними взаимодействие, при этом образуя семейства из этих объектов-продуктов. Входящие в семейство объекты-продукты обязательно должны использоваться вместе. * Требуется построить подсистему (модуль или компонент) таким образом, чтобы ее внутреннее устройство (состояние и/или поведение) настраивалось при ее создании. При этом чтобы ни процесс, ни результат построения подсистемы не был зависим от способа создания в ней объектов, их композиции (составления и соединения объектов) и представления (настройки внутреннего состояния объектов). * Подсистема или система должна настраиваться (конфигурироваться) через использование одного из семейств объектов-продуктов, порождаемых одним объектом-фабрикой;   *Преимущества*:   * Сокрытие работы с конкретными классами продуктов. Фабрика скрывает от клиента детали реализации конкретных классов и процесс создания экземпляров этих классов. Конкретные классы-продуктов известны только конкретным фабрикам и в коде клиента они не используются. Клиент управляет экземплярами конкретных классов только через их абстрактные интерфейсы. * Позволяет легко заменять семейства используемых продуктов. Экземпляр класса конкретной фабрики создается в приложении в одном месте и только один раз, что позволяет в дальнейшем проще подменять фабрики. Для того чтобы изменить семейство используемых продуктов, нужно просто создать новый экземпляр класса-фабрики, тогда заменится сразу все семейство. * Обеспечение совместного использования продуктов. Позволяет легко контролировать взаимодействие между объектами-продуктами, которые спроектированы для совместного использования и входят в одно семейство. |
|  | Builder  Строитель |  |  | Пошаговое построение сложных объектов.  *Рекомендуемые случаи применения*:   * Алгоритм пошагового создания сложного объекта-продукта не должен зависеть от того, из каких частей состоит объект-продукт и как эти части стыкуются между собой; * Процесс создания продукта должен обеспечивать возможность получения различных вариаций создаваемого продукта.   *Преимущества*:   * Позволяет изменять состав продукта. Абстрактный класс Builder предоставляет директору набор абстрактных методов (абстрактный интерфейс) для управления построением продукта. За абстрактным интерфейсом Builder скрывает внутреннюю структуру создаваемого продукта и процесс его построения. Поскольку построение продукта производится согласно абстрактному интерфейсу, то для изменения структуры продукта достаточно создать новую разновидность строителя; * Скрывает код, реализующий конструирование и представление. Паттерн Builder улучшает модульность, скрывая способы построения и представления сложных объектов-продуктов. Клиенты ничего не знают о классах, входящих в состав внутренней структуры продукта, использование этих классов отсутствует в интерфейсе строителя. Конкретные строители ConcreteBuilder содержат код, необходимый для создания и сборки конкретного вида продукта. Код пишется только один раз и разные директоры могут использовать его повторно для построения различных вариантов продукта из одних и тех же частей комбинируя эти части. * Предоставляет полный контроль над процессом построения продукта. В отличие от других порождающих паттернов, которые сразу конструируют весь объект-продукт полностью, строитель строит продукт шаг за шагом под управлением директора. И только тогда, когда построение продукта завершено, директор или клиент забирают его у строителя. Поэтому интерфейс строителя в большей степени отражает процесс пошагового конструирования продукта, нежели другие порождающие паттерны. Это позволяет обеспечить полный контроль над процессом конструирования, а значит, и над внутренней структурой (комбинацией частей) готового продукта.   *Полезные приемы реализации*:   * Интерфейс строителя. Строители конструируют продукты шаг за шагом. Интерфейс класса Builder должен быть достаточно общим, чтобы обеспечить создание продуктов при любой реализации конкретных строителей. Иногда может потребоваться доступ к частям уже сконструированного, готового продукта и такую возможность желательно предусмотреть. * Отсутствие общего базового абстрактного класса для продуктов. Чаще всего продукты имеют настолько разный состав, что создание для них общего базового класса ничего не дает. |
|  | Factory Method  Фабричный метод |  |  |  |
|  | Prototype  Прототип |  |  |  |
|  | Singleton  Одиночка |  |  |  |
|  | **Структурные** | | | |
|  | Adapter  Адаптер |  |  |  |
|  | Bridge  Мост |  |  |  |
|  | Composite  Компоновщик |  |  |  |
|  | Decorator  Декоратор |  |  |  |
|  | Facade  Фасад |  |  |  |
|  | Flyweight  Приспособленец |  |  |  |
|  | Proxy  Заместитель |  |  | Предоставляет объект заместитель |
|  | **Поведенческие** | | | |
|  | Chain of Responsibility  Цепочка обязанностей  Цепочка зависимостей |  |  | Создает цепочки из обработчиков запросов |
|  | Command  Команда |  |  | Представление запроса в виде объекта |
|  | Interpreter  Интерпретатор |  |  | Формирует объектно-ориентрованное представление регулярной грамматики |
|  | Iterate |  |  | Предоставляет объект-итератор для доступа к элементам коллекции |
|  | Mediator  Посредник |  |  | Предоставляет объект-посредник для замены множества связей между объектами в единый алгоритм в объекте-посреднике. |
|  | Memento  Хранитель |  |  | Предоставляет объект для хранения состояний |
|  | Observer  Наблюдатель |  |  | Описывает технику издатель-подписчик |
|  | State  Состояние |  |  | Описывает способы построения конечных автоматов |
|  | Strategy  Стратегия |  |  | Описывает способы подмены алгоритмов. |
|  | Template Method | Класс |  | Формирует структура алгоритма |
|  | Visitor |  |  | Организует обход набора элементов с разнородными интерфейсами. |