1.4. Каталог паттернов проектирования

Каталог содержит 23 паттерна. Ниже для удобства перечислены их имена и назначение. В скобках после названия каждого паттерна указан номер страницы, откуда начинается его подробное описание.

Abstract Factory (абстрактная фабрика) (93)
Предоставляет интерфейс для создания семейств, связанных между собой, или независимых объектов, конкретные классы которых неизвестны.

Adapter (адаптер) (141)
Преобразует интерфейс класса в некоторый другой интерфейс, ожидаемый клиентами. Обеспечивает совместную работу классов, которая была бы невозможна без данного паттерна из-за несовместимости интерфейсов.

Bridge (мост) (152)
Отделяет абстракцию от реализации, благодаря чему появляется возможность независимо изменять то и другое.

Builder (строитель) (103)
Отделяет конструирование сложного объекта от его представления, позволяя использовать один и тот же процесс конструирования для создания различных представлений.

Chain of Responsibility (цепочка обязанностей) (217)
Можно избежать жесткой зависимости отправителя запроса от его получателя, при этом запросом начинает обрабатываться один из нескольких объектов. Объекты-получатели связываются в цепочку, и запрос передается по цепочке, пока какой-то объект его не обработает.

Command (команда) (226)
Инкапсулирует запрос в виде объекта, позволяя тем самым параметризовывать клиентов типом запроса, устанавливать очередность запросов, протоколировать их и поддерживать отмену выполнения операций.

Composite (компоновщик) (162)
Группирует объекты в древовидные структуры для представления иерархий типа «часть-целое». Позволяет клиентам работать с единичными объектами так же, как с группами объектов.

Decorator (декоратор) (173)
Динамически возлагает на объект новые функции. Декораторы применяются для расширения имеющейся функциональности и являются гибкой альтернативой порождению подклассов.

Facade (фасад) (183)
Предоставляет унифицированный интерфейс к множеству интерфейсов в некоторой подсистеме. Определяет интерфейс более высокого уровня, облегчающий работу с подсистемой.

Factory Method (фабричный метод) (111)
Определяет интерфейс для создания объектов, при этом выбранный класс инстанцируется подклассами.

Flyweight (приспособленец) (191)
Использует разделение для эффективной поддержки большого числа мелких объектов.

Interpreter (интерпретатор) (236)
Для заданного языка определяет представление его грамматики, а также интерпретатор предложений языка, использующий это представление.

Iterator (итератор) (173)
Дает возможность последовательно обойти все элементы составного объекта, не раскрывая его внутреннего представления.

Mediator (посредник) (263)
Определяет объект, в котором инкапсулировано знание о том, как взаимодействуют объекты из некоторого множества. Способствует уменьшению числа связей между объектами, позволяя им работать без явных ссылок друг на друга. Это, в свою очередь, дает возможность независимо изменять схему взаимодействия.

Memento (хранитель) (272)
Позволяет, не нарушая инкапсуляции, получить и сохранить во внешней памяти внутреннее состояние объекта, чтобы позже объект можно было восстановить точно в таком же состоянии.

Observer (наблюдатель) (281)
Определяет между объектами зависимость типа один-ко-многим, так что при изменении состоянии одного объекта все зависящие от него получают извещение и автоматически обновляются.

Prototype (прототип) (121)
Описывает виды создаваемых объектов с помощью прототипа и создает новые объекты путем его копирования.

Proxy (заместитель) (203)
Подменяет другой объект для контроля доступа к нему.

Singleton (одиночка) (130)
Гарантирует, что некоторый класс может иметь только один экземпляр, и предоставляет глобальную точку доступа к нему.

State (состояние) (291)
Позволяет объекту варьировать свое поведение при изменении внутреннего состояния. При этом создается впечатление, что поменялся класс объекта.

Strategy (стратегия) (300)
Определяет семейство алгоритмов, инкапсулируя их все и позволяя подставлять один вместо другого. Можно менять алгоритм независимо от клиента, который им пользуется.

Template Method (шаблонный метод) (309)
Определяет скелет алгоритма, перекладывая ответственность за некоторые его шаги на подклассы. Позволяет подклассам переопределять шаги алгоритма, не меняя его общей структуры.

Visitor (посетитель) (314)
Представляет операцию, которую надо выполнить над элементами объекта. Позволяет определить новую операцию, не меняя классы элементов, к которым он применяется.

1.5. Организация каталога

Паттерны проектирования различаются степенью детализации и уровнем абстракции и должны быть каким-то образом организованы. В данном разделе описывается классификация, позволяющая ссылаться на семейства взаимосвязанных паттернов. Она поможет быстрее освоить паттерны, описываемые в каталоге, и укажет направление поиска новых.

Мы будем классифицировать паттерны по двум критериям (табл. 1.1). Первый – *цель* – отражает назначение паттерна. В связи с этим выделяются порождающие паттерны, структурные паттерны и паттерны поведения. Первые связаны с процессом создания объектов. Вторые имеют отношение к композиции объектов и классов. Паттерны поведения характеризуют то, как классы или объекты взаимодействуют между собой.

Таблица 1.1. Пространство паттернов проектирования

Цель / Уровень	Порождающие паттерны	Структурные паттерны	Паттерны поведения
Класс	Фабричный метод	Адаптер (класса)	Интерпретатор
			Шаблонный метод
Объект	Абстрактная фабрика	Адаптер (объекта)	Итератор
	Одиночка	Декоратор	Команда
	Прототип	Заместитель	Наблюдатель
	Строитель	Компоновщик	Посетитель
		Мост	Посредник
		Приспособленец	Состояние
		Фасад	Стратегия
			Хранитель
			Цепочка обязанностей

Второй критерий – *уровень* – говорит о том, к чему обычно применяется паттерн: к объектам или классам. Паттерны уровня классов описывают отношения между классами и их подклассами. Такие отношения выражаются с помощью наследования, поэтому они статичны, то есть зафиксированы на этапе компиляции. Паттерны уровня объектов описывают отношения между объектами, которые