

Лекция 8. Шаблоны проектирования: поведенческие, структурные и порождающие структуры, другие типы.

Название	Оригинальное название	Описание	Описание в <u>Design Patterns</u>
Основные шаблоны (Fundamental)			
Шаблон делегирования	Delegation pattern	Объект внешне выражает некоторое поведение, но в реальности передаёт ответственность за выполнение этого поведения связанному объекту.	Н/д
Шаблон функционального дизайна	Functional design	Гарантирует, что каждый модуль компьютерной программы имеет только одну обязанность и исполняет её с минимумом побочных эффектов на другие части программы.	Н/д
Неизменяемый интерфейс	Immutable interface	Создание неизменяемого объекта .	Н/д
Интерфейс	Interface	Общий метод для структурирования компьютерных программ для того, чтобы их было проще понять.	Н/д
Интерфейс-маркер	Marker interface	В качестве атрибута (как пометки объектной сущности) применяется наличие или отсутствие реализации интерфейса-маркера. В современных языках программирования вместо этого могут применяться атрибуты или аннотации.	Н/д
Контейнер свойств	Property container	Позволяет добавлять дополнительные свойства для	Н/д

		класса в контейнер (внутри класса), вместо расширения класса новыми свойствами.	
Event Channel	Event channel	Расширяет шаблон Publish/Subscribe, создавая централизованный канал для событий. Использует объект-представитель для подписки и объект-представитель для публикации события в канале. Представитель существует отдельно от реального издателя или подписчика. Подписчик может получать опубликованные события от более чем одного объекта, даже если он зарегистрирован только на одном канале.	Н/д
Порождающие шаблоны (Creational) — шаблоны проектирования, которые абстрагируют процесс инстанцирования. Они позволяют сделать систему независимой от способа создания, композиции и представления объектов. Шаблон, порождающий классы, использует наследование, чтобы изменять инстанцируемый класс, а шаблон, порождающий объекты, делегирует инстанцирование другому объекту.			
Абстрактная фабрика	Abstract factory	Класс, который представляет собой интерфейс для создания компонентов системы.	Да
Строитель	Builder	Класс, который представляет собой интерфейс для создания сложного объекта.	Да
Фабричный метод	Factory method	Определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс инстанцировать.	Да
Отложенная инициализация	Lazy initialization	Объект, инициализируемый во время первого обращения к нему.	Нет
Пул одиночек	Multiton	Гарантирует, что класс имеет	Нет

		поименованные экземпляры объекта и обеспечивает глобальную точку доступа к ним.	
Объектный пул	Object pool	Класс, который представляет собой интерфейс для работы с набором инициализированных и готовых к использованию объектов.	Нет
Прототип	Prototype	Определяет интерфейс создания объекта через клонирование другого объекта вместо создания через конструктор.	Да
Получение ресурса есть инициализация	Resource acquisition is initialization (RAII)	Получение некоторого ресурса совмещается с инициализацией, а освобождение — с уничтожением объекта.	Нет
Одиночка	Singleton	Класс, который может иметь только один экземпляр.	Да
Структурные шаблоны (Structural) определяют различные сложные структуры, которые изменяют интерфейс уже существующих объектов или его реализацию, позволяя облегчить разработку и оптимизировать программу.			
Адаптер	Adapter / Wrapper	Объект, обеспечивающий взаимодействие двух других объектов, один из которых использует, а другой предоставляет несовместимый с первым интерфейс.	Да
Мост	Bridge	Структура, позволяющая изменять интерфейс обращения и интерфейс реализации класса независимо.	Да
Компоновщик	Composite	Объект, который объединяет в себе объекты, подобные ему самому.	Да

Декоратор или Wrapper/Обёртка	Decorator	Класс, расширяющий функциональность другого класса без использования наследования.	Да
Фасад	Facade	Объект, который абстрагирует работу с несколькими классами, объединяя их в единое целое.	Да
Единая точка входа	Front controller	Обеспечивает унифицированный интерфейс для интерфейсов в подсистеме. Front Controller определяет высокоуровневый интерфейс, упрощающий использование подсистемы.	Нет
Приспособленец	Flyweight	Это объект, представляющий себя как уникальный экземпляр в разных местах программы, но по факту не являющийся таковым.	Да
Заместитель	Proxy	Объект, который является посредником между двумя другими объектами, и который реализует/ограничивает доступ к объекту, к которому обращаются через него.	Да
Поведенческие шаблоны (Behavioral) определяют взаимодействие между объектами, увеличивая таким образом его гибкость.			
Цепочка обязанностей	Chain of responsibility	Предназначен для организации в системе уровней ответственности.	Да
Команда , Action, Transaction	Command	Представляет действие. Объект команды заключает в себе само действие и его параметры.	Да
Интерпретатор	Interpreter	Решает часто встречающуюся, но подверженную изменениям, задачу.	Да
Итератор , Cursor	Iterator	Представляет собой объект,	Да

		позволяющий получить последовательный доступ к элементам объекта-агрегата без использования описаний каждого из объектов, входящих в состав агрегации.	
Посредник	Mediator	Обеспечивает взаимодействие множества объектов, формируя при этом слабую связанность и избавляя объекты от необходимости явно ссылаться друг на друга.	Да
Хранитель	Memento	Позволяет не нарушая инкапсуляцию зафиксировать и сохранить внутренние состояния объекта так, чтобы позднее восстановить его в этих состояниях.	Да
Null Object	Null Object	Предотвращает нулевые указатели, предоставляя объект «по умолчанию».	Нет
Наблюдатель или Publish/subscribe[en]	Observer	Определяет зависимость типа «один ко многим» между объектами таким образом, что при изменении состояния одного объекта все зависящие от него оповещаются об этом событии.	Да
Слуга[en]	Servant	Используется для обеспечения общей функциональности группе классов.	Нет
Спецификация	Specification	Служит для связывания бизнес-логики.	Нет
Состояние	State	Используется в тех случаях, когда во время выполнения программы объект должен менять своё поведение в зависимости от своего состояния.	Да

Стратегия	Strategy	Предназначен для определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости.	Да
Шаблонный метод	Template method	Определяет основу алгоритма и позволяет наследникам переопределять некоторые шаги алгоритма, не изменяя его структуру в целом.	Да
Посетитель	Visitor	Описывает операцию, которая выполняется над объектами других классов. При изменении класса Visitor нет необходимости изменять обслуживаемые классы.	Да
Простая политика	Simple Policy		Нет
Event listener			Нет
Одноразовый посетитель[en]	Single-serving visitor	Оптимизирует реализацию шаблона посетитель, который инициализируется, единожды используется, и затем удаляется.	Нет
Иерархический посетитель[en]	Hierarchical visitor	Предоставляет способ обхода всех вершин иерархической структуры данных (напр. древовидной).	Нет