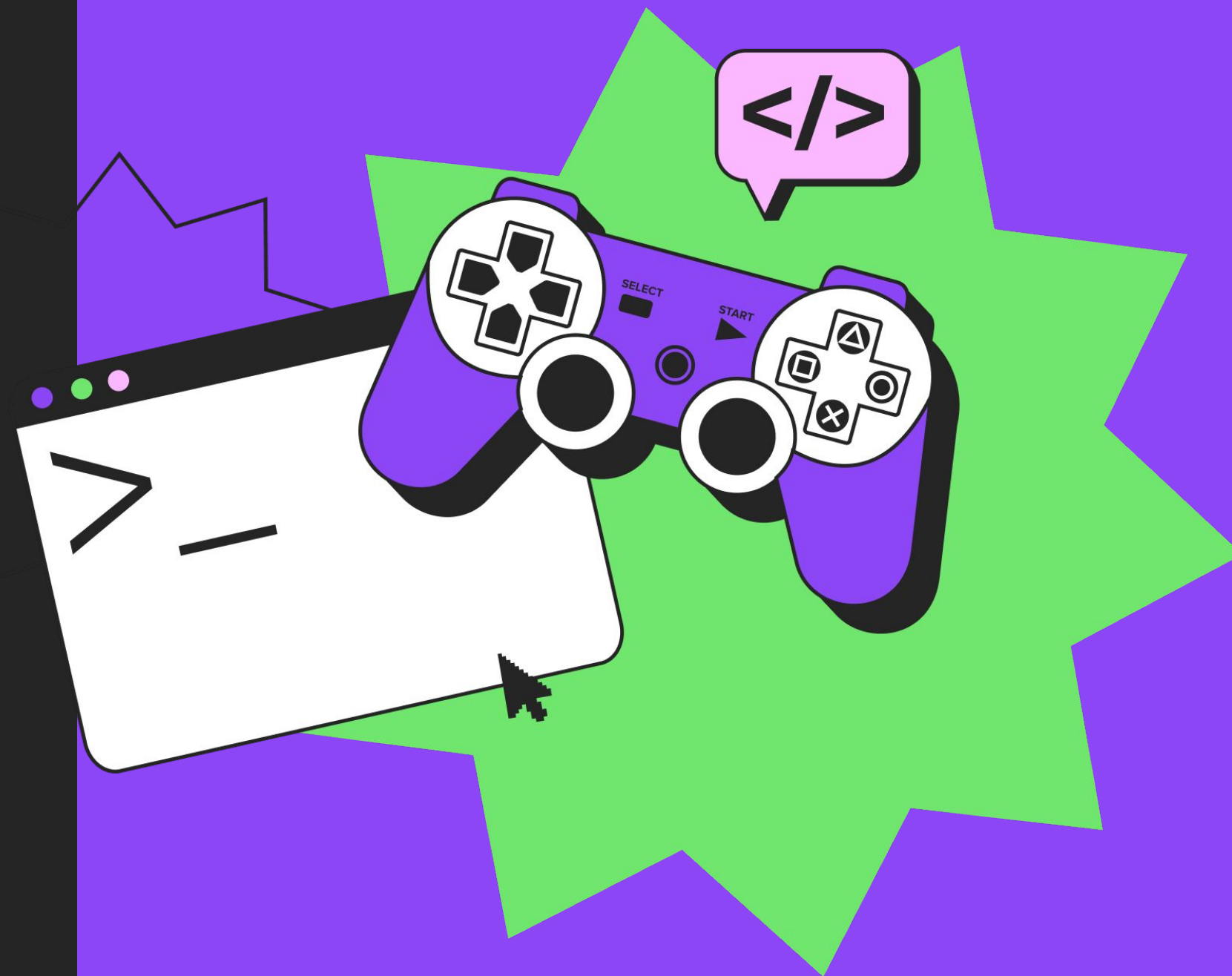


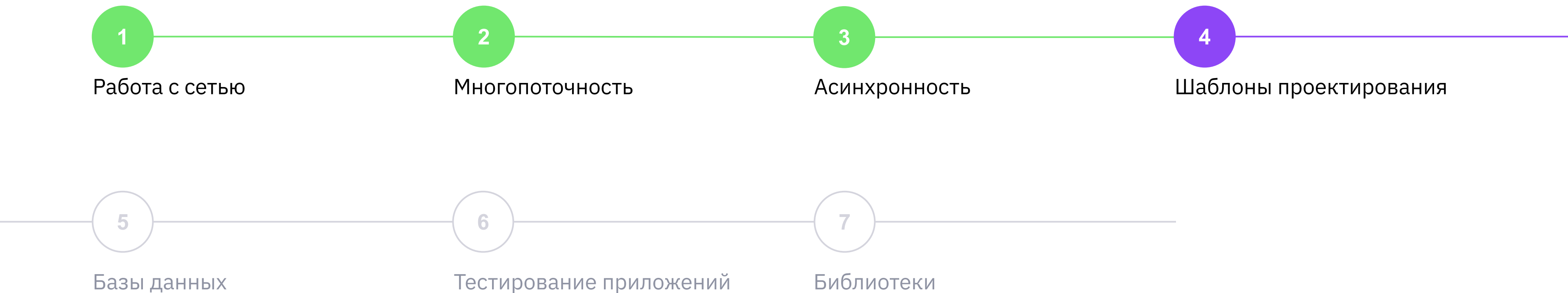
Разработка сетевого приложения на C#

Урок 4
Шаблоны проектирования





План курса





Содержание урока

- Шаблон проектирования
- GOF
- Шаблоны проектирования
- Антипаттерны



Шаблон проектирования

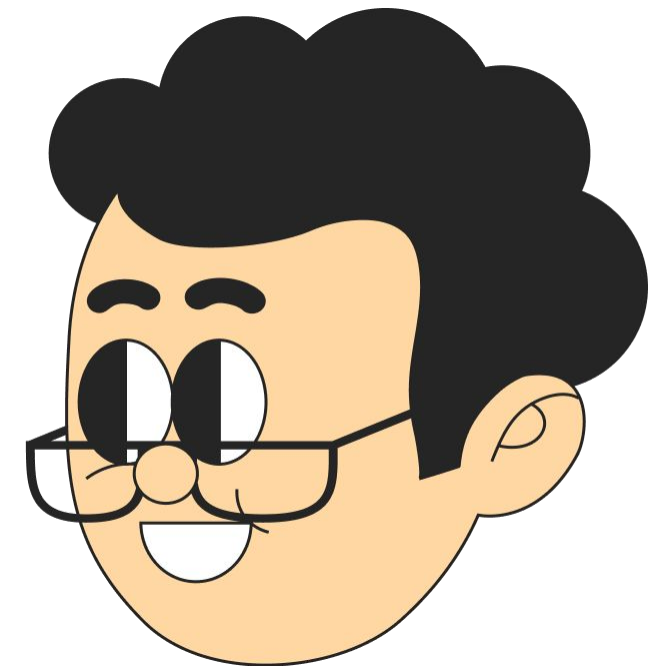
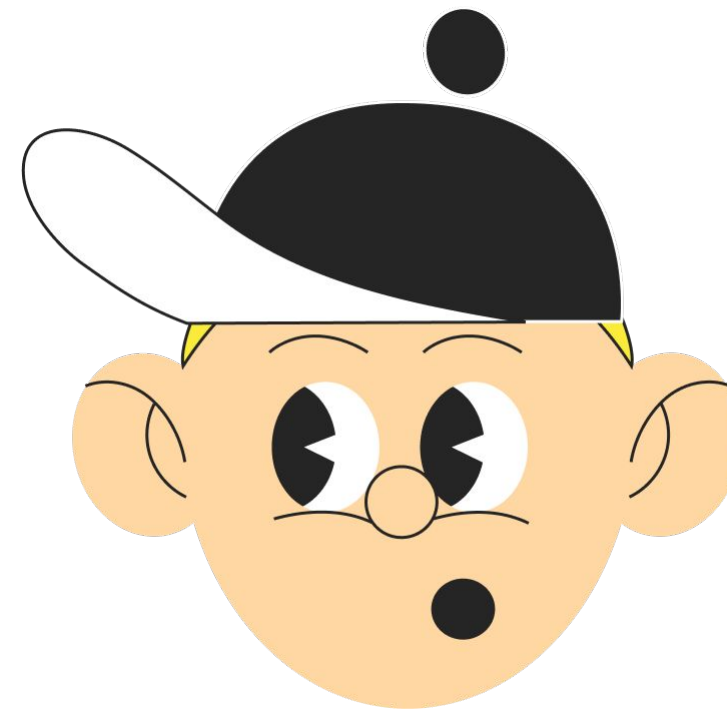
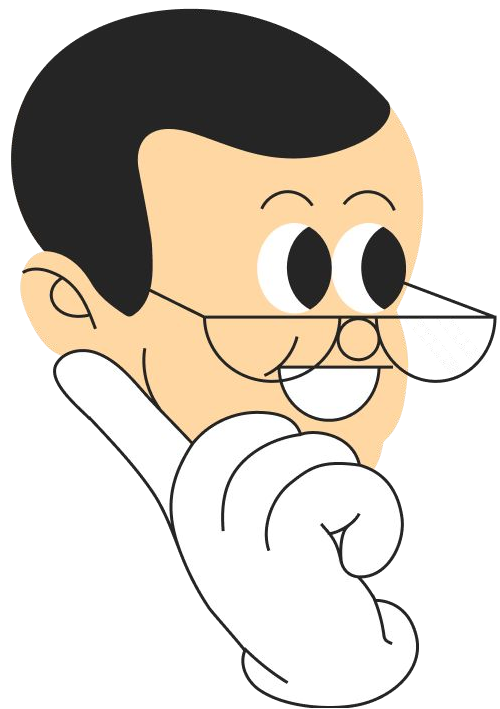
Шаблон проектирования – это повторно используемый и протестированный подход к решению типичных проблем при проектировании программных систем. Это некая абстракция, описывающая общую структуру и взаимодействие компонентов в системе.



Шаблоны проектирования помогают разработчикам создавать эффективные, надежные и легко поддерживаемые решения, используя лучшие практики и опыт, накопленные в области разработки.

GOF

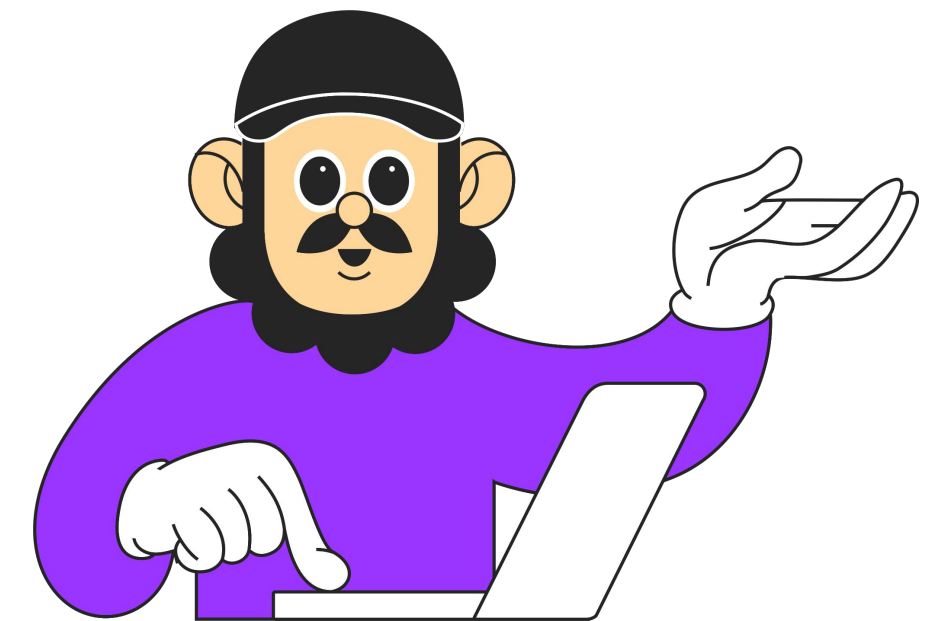
“Gang of four” или банда четырёх – группа из четырех авторитетных авторов, которые в 90-х годах прошлого столетия опубликовали книгу "Design Patterns: Elements of Reusable Object-Oriented Software" (Шаблоны проектирования: Элементы многократного использования).





Singleton/ Одиночка

Шаблон проектирования "Одиночка" (Singleton) относится к категории порождающих шаблонов и используется для обеспечения того, чтобы у класса был только один экземпляр, и предоставляет глобальную точку доступа к этому экземпляру.





Prototype/ Прототип

Паттерн "Прототип" (Prototype) – это шаблон проектирования, который позволяет создавать новые объекты путем клонирования существующих объектов, избегая необходимости создания объектов с нуля.





Factory method/ Фабричный метод

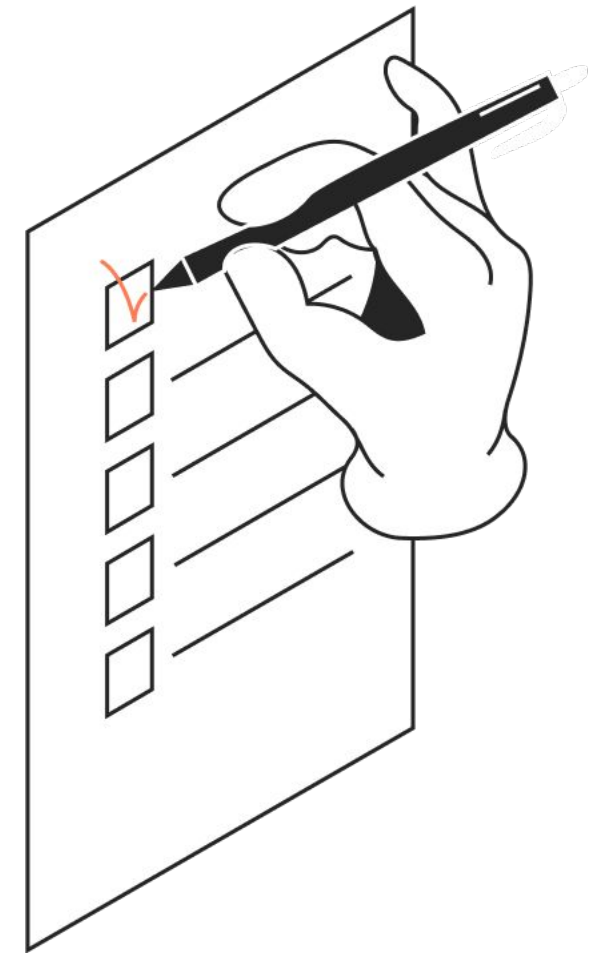
Паттерн "Фабричный метод" (Factory Method) – это шаблон проектирования, который определяет интерфейс для создания объектов, но позволяет подклассам выбирать классы для создания. Таким образом, он делегирует создание объектов подклассам.





Abstract factory/ Абстрактная фабрика

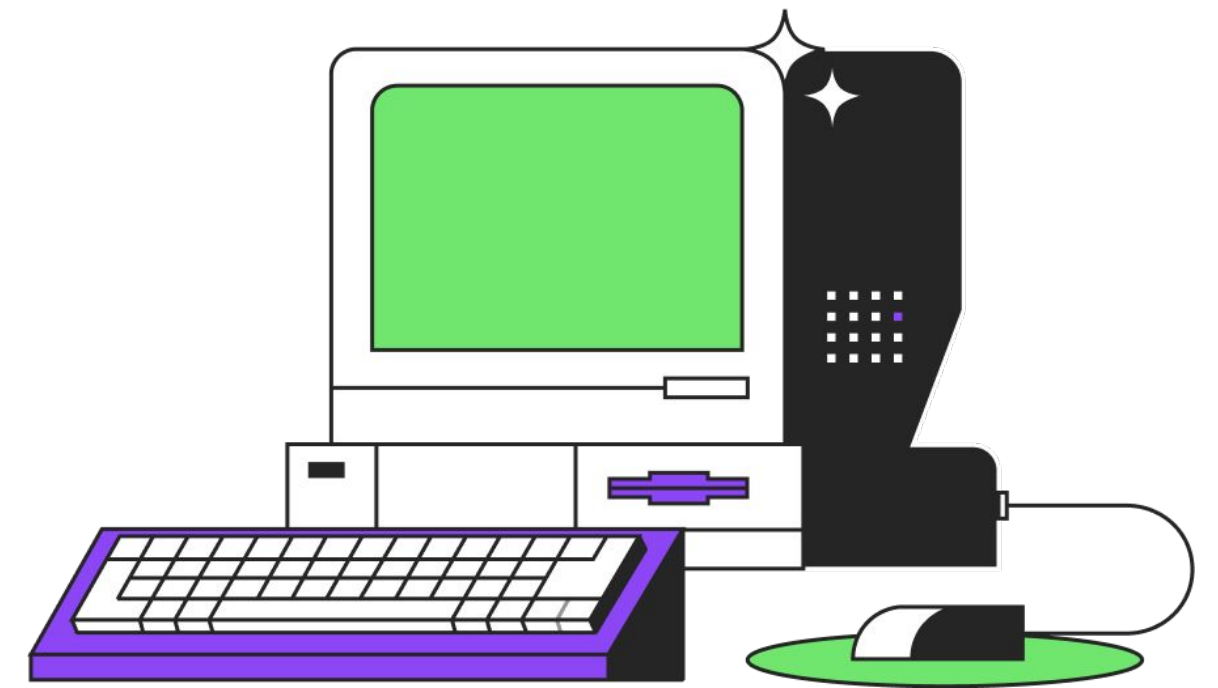
Абстрактная фабрика (Abstract Factory) – это шаблон проектирования, который предоставляет интерфейс для создания семейств взаимосвязанных объектов, не указывая их конкретных классов.





Builder/ Строитель

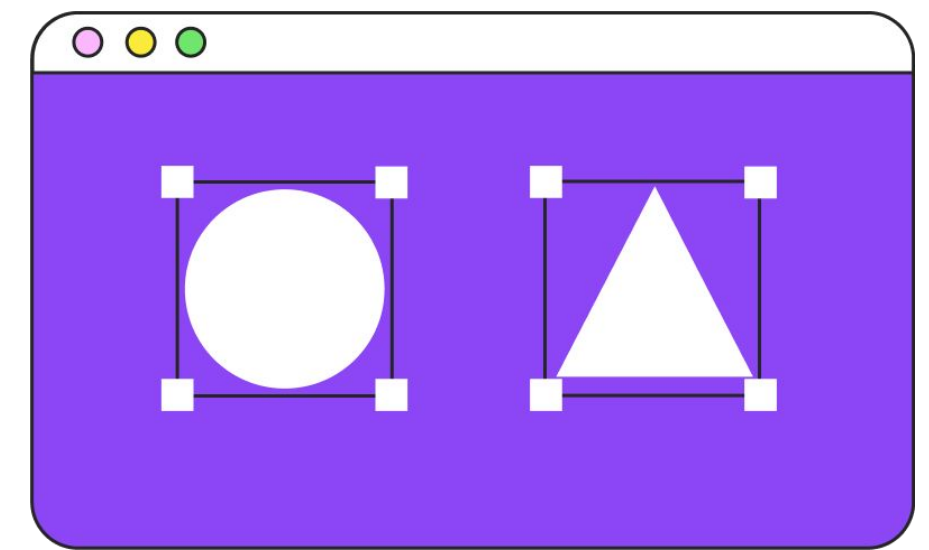
Паттерн Строитель позволяет создавать сложные объекты пошагово. Он позволяет разделить процесс создания объекта от его представления, что позволяет создавать разные представления одного и того же объекта.





Adapter/ Адаптер

Шаблон "Адаптер" (Adapter) – это структурный шаблон проектирования, который позволяет объектам с несовместимыми интерфейсами работать вместе. Адаптер предоставляет промежуточный интерфейс, который преобразует интерфейс одного класса в интерфейс, ожидаемый другим классом.





Bridge/ Мост

Данный шаблон позволяет отделить абстракцию от реализации, чтобы они могли изменяться независимо друг от друга. Этот шаблон особенно полезен, когда у вас есть несколько способов вариантов реализации для какой-либо абстракции, и вы хотите, чтобы они могли меняться независимо друг от друга.





Composite/ Компоновщик

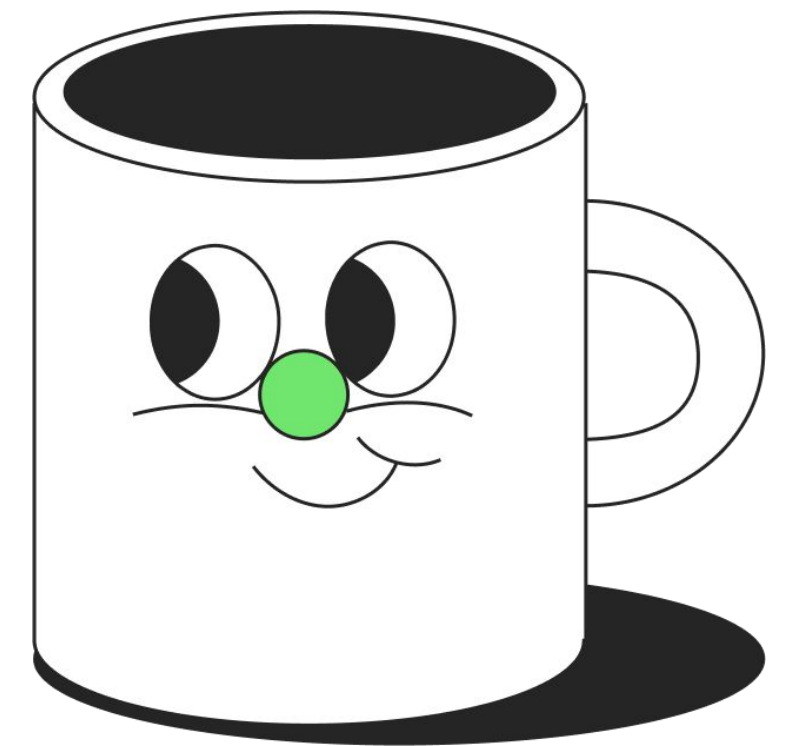
Позволяет создавать иерархии объектов в виде древовидных структур, где как отдельные объекты, так и их композиты (группы объектов) могут обрабатываться одинаково. Этот шаблон позволяет клиентам единообразно обращаться к отдельным объектам и составным структурам.





Decorator/ Декоратор

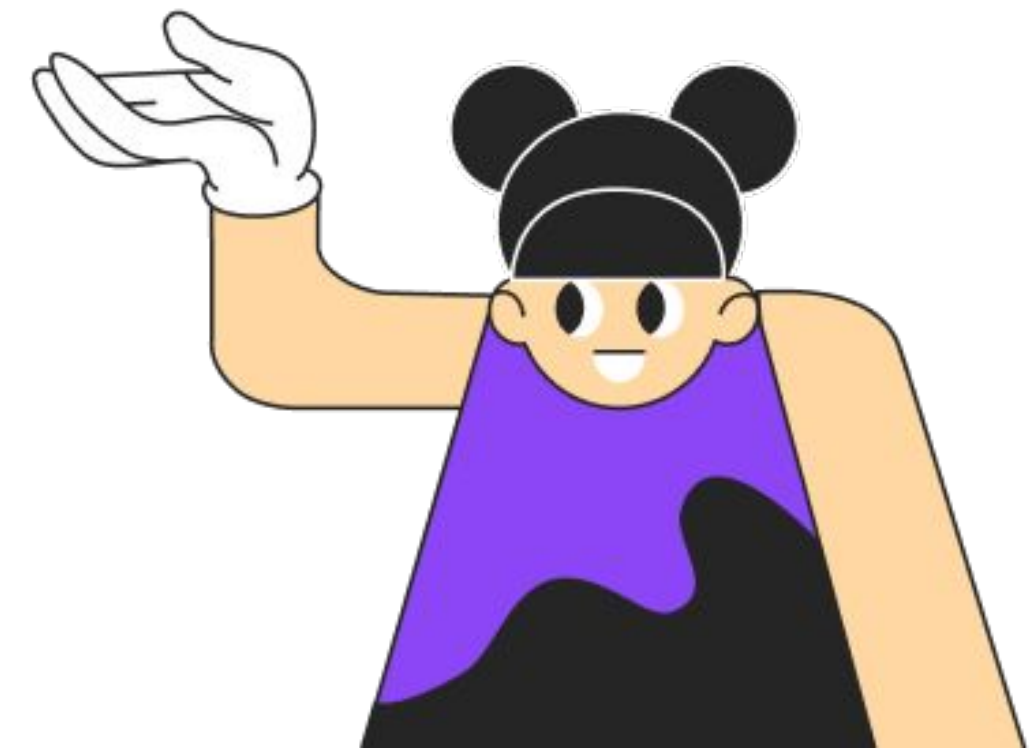
Шаблон позволяет добавлять новую функциональность объектам, не изменяя их структуру. Декоратор предоставляет гибкую альтернативу наследованию для расширения функциональности классов.





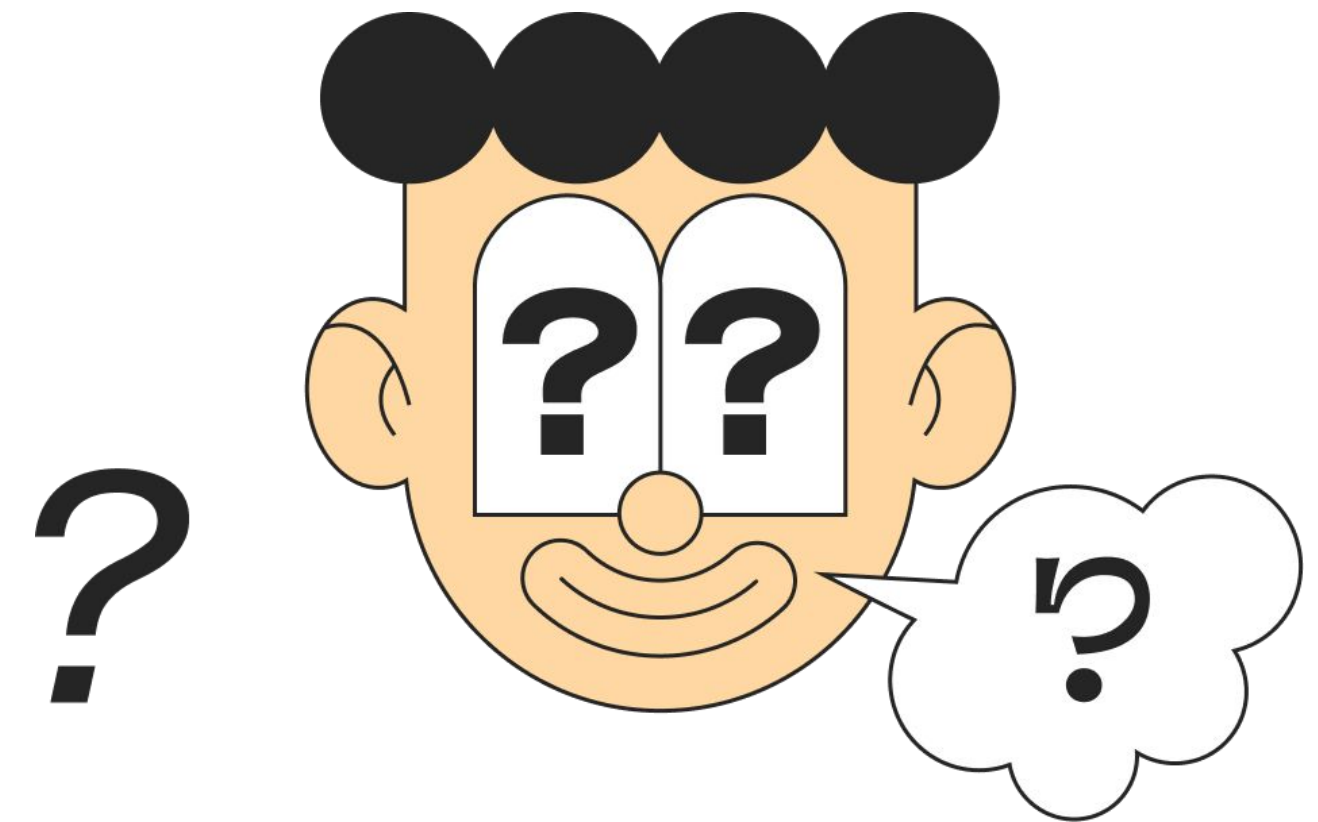
Facade/ Фасад

Фасад предоставляет унифицированный интерфейс для группы интерфейсов в подсистеме. Фасад упрощает взаимодействие клиента с комплексной системой, предоставляя более удобный и понятный интерфейс.



Flyweight/ Приспособленец

Шаблон позволяет эффективно разделить объекты на общие и индивидуальные части. Приспособленцы разделяются между несколькими объектами для экономии памяти и ресурсов. Зачастую приспособленцы применяются вместе с шаблоном фабрики, и тогда они наиболее эффективны.





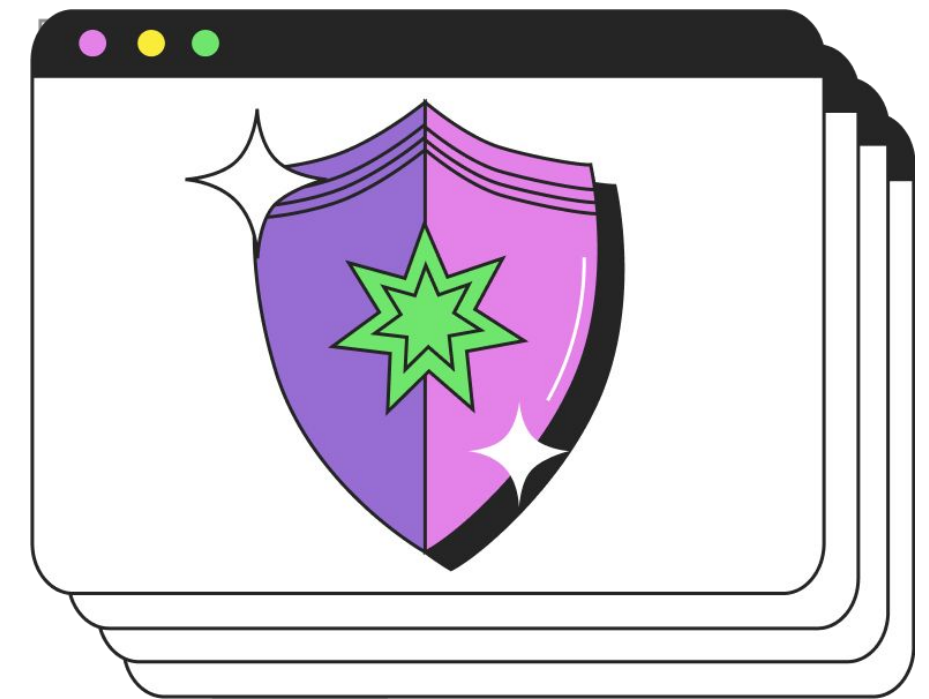
Proxy/ Прокси

Прокси контролирует доступ к оригинальному объекту, позволяя выполнять дополнительные действия до или после доступа к нему.



Chain of responsibility/ Цепочка ответственности

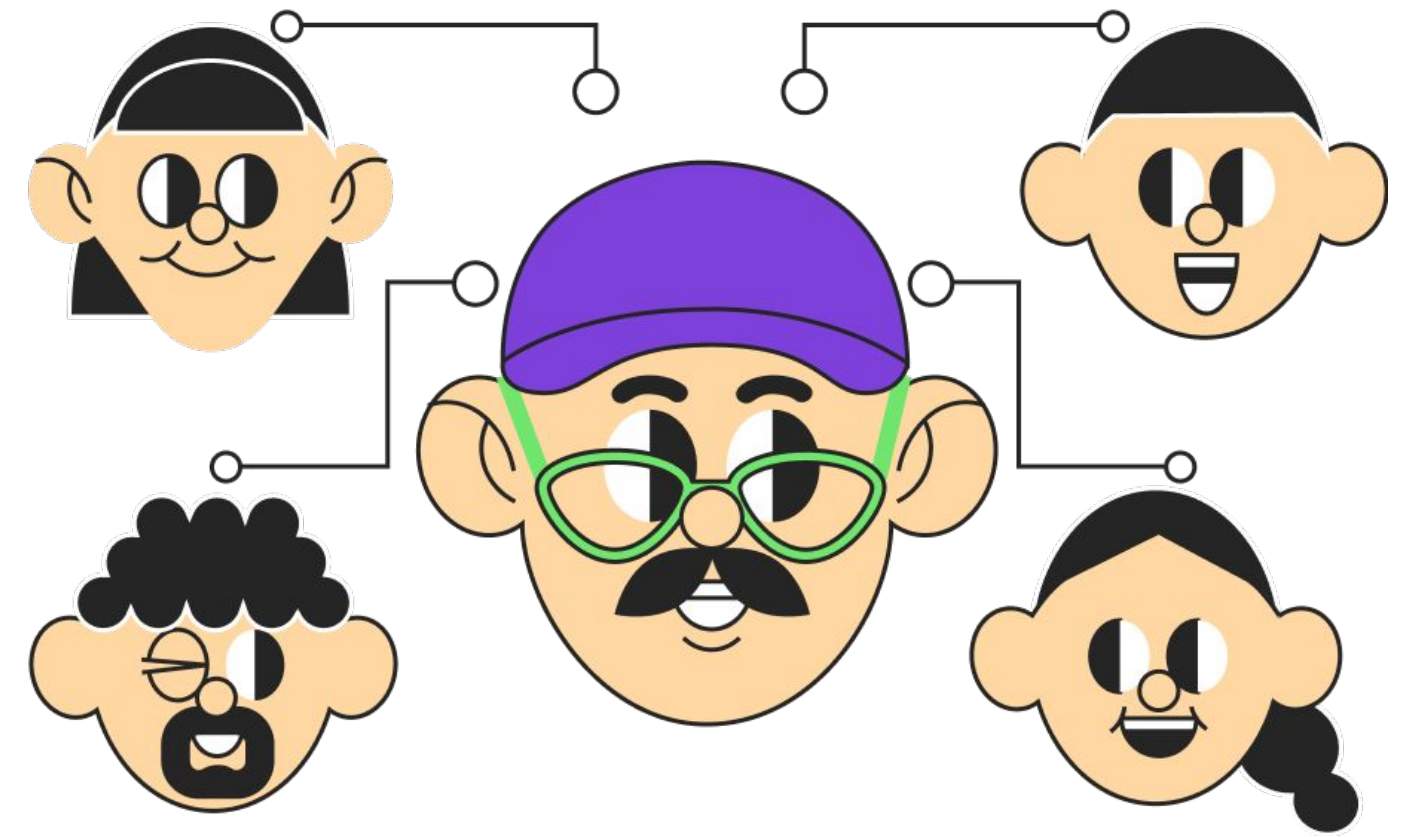
Шаблон позволяет создать цепочку объектов, способных обрабатывать запросы последовательно. Каждый объект в цепочке может решить, может ли он обработать запрос, и либо обработать его, либо передать дальше по цепочке.





Command/ Команда

Шаблон инкапсулирует запрос в виде объекта, позволяя параметризовать клиентов с разными запросами, ставить запросы в очередь, а также поддерживать отмену операций.





Interpreter/ Интерпретатор

Шаблон используется для интерпретации и оценки языковых грамматик или выражений. Он позволяет создавать язык, который позволяет интерпретировать заданные выражения и выполнять соответствующие действия.





Iterator/ Итератор

Шаблон предоставляет способ последовательного доступа к элементам коллекции без раскрытия деталей его внутренней реализации. Этот шаблон позволяет клиентам обходить элементы коллекции, независимо от того, какая структура используется для хранения данных.

```
public interface IEnumerator
{
    object Current { get; }

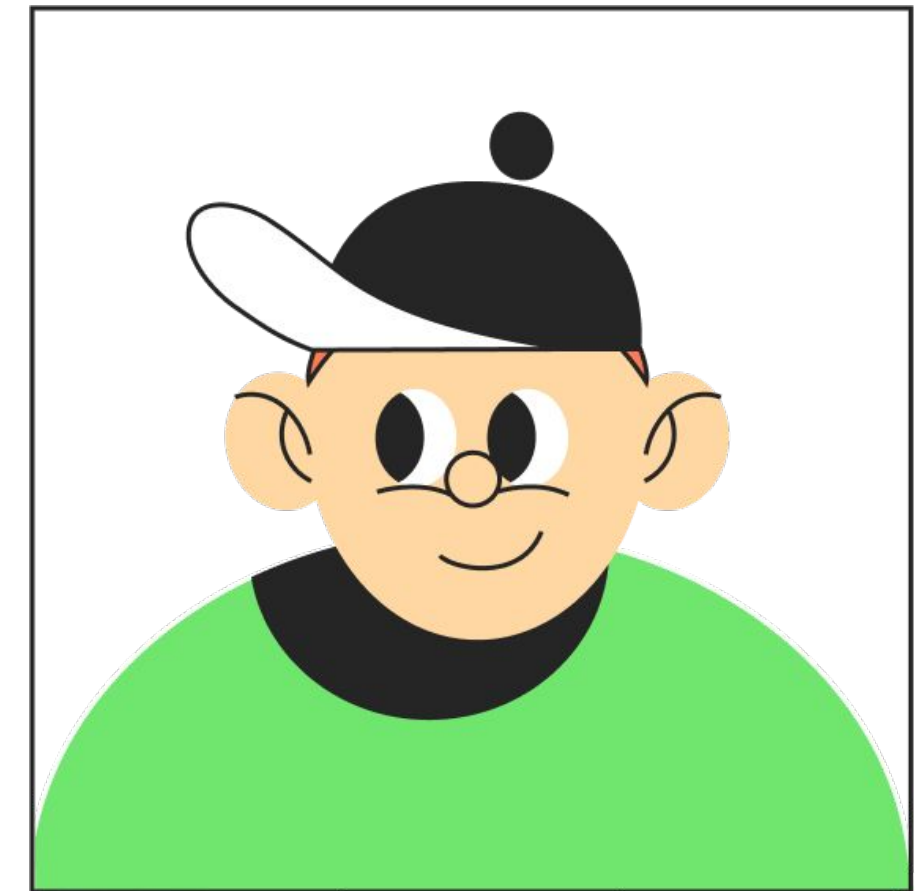
    bool MoveNext ();

    void Reset ();
}
```



Mediator/ Медиатор

Шаблон обеспечивает централизованную связь между различными компонентами (коллегами) без необходимости прямого взаимодействия между ними. Медиатор управляет коммуникацией между компонентами, снижая их зависимость друг от друга.





Memento/ Хранитель

Шаблон позволяет сохранять и восстанавливать внутреннее состояние объекта без раскрытия деталей его реализации. Этот шаблон позволяет создавать "снимки" состояния объекта, которые могут быть восстановлены позже.





Observer/ Наблюдатель

Шаблон позволяет объектам подписываться на изменения состояния других объектов и автоматически получать уведомления о таких изменениях. Этот шаблон обеспечивает связь между объектами, при этом один объект (называемый "субъектом") уведомляет своих наблюдателей о своих изменениях.





State/ Состояние

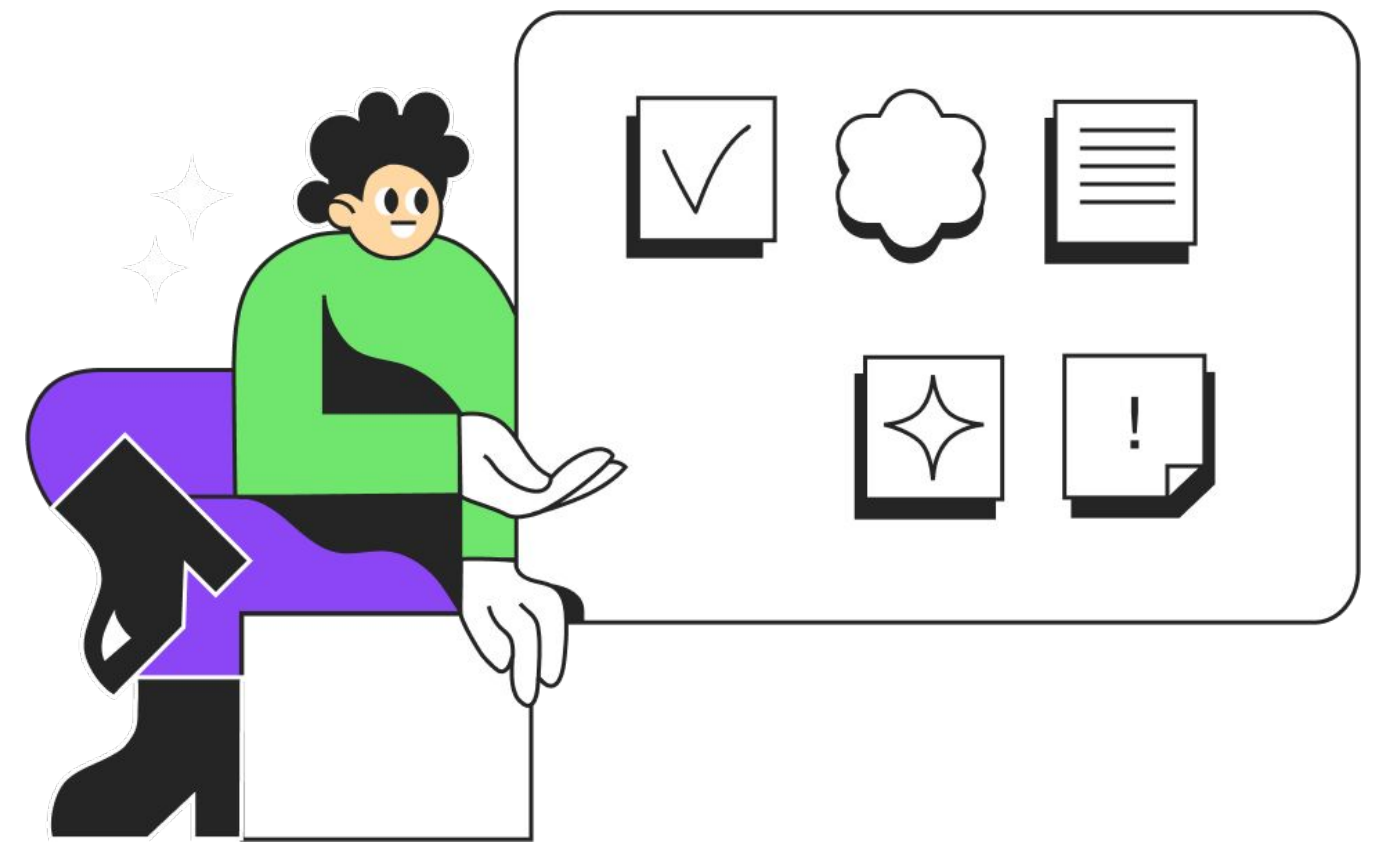
Шаблон позволяет объектам менять свое поведение в зависимости от внутреннего состояния. Шаблон позволяет создавать классы для каждого состояния объекта и делегировать выполнение операций в зависимости от текущего состояния.





Strategy/ Стратегия

Шаблон позволяет определить семейство алгоритмов, инкапсулировать их в отдельные классы и делать их взаимозаменяемыми. Это позволяет изменять алгоритмы независимо от клиентов, которые используют эти алгоритмы.





Template method/ Шаблонный метод

Шаблон определяет скелет алгоритма в базовом классе и позволяет подклассам переопределять определенные шаги этого алгоритма без изменения его структуры. Таким образом, он обеспечивает общую структуру алгоритма, но делегирует конкретную реализацию дочерним классам.





Visitor/ Посетитель

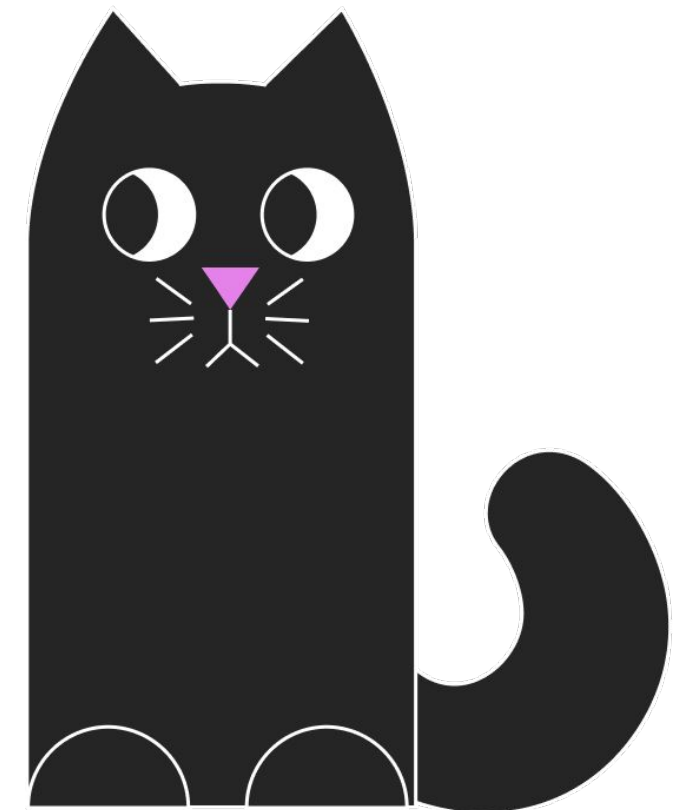
Шаблон позволяет добавлять новые операции к объектам, не изменяя их классы. Он позволяет вынести операции из классов объектов в отдельные классы (посетители), тем самым разделяя сущности объекта и операции над ним.





Антипаттерны

Антипаттерны – это практики в разработке программного обеспечения, которые на первый взгляд кажутся разумными, но на самом деле приводят к плохому качеству кода, сложностям в поддержке и другим проблемам. Они являются своего рода "антитезой" хороших практик и принципов проектирования.





Антипаттерны

Вот некоторые из распространенных антипаттернов:


- **Синглтонизм (Singletonitis):** Использование слишком большого количества единственных экземпляров классов (Singleton), что создает множество сложных зависимостей и проблем при тестировании.
- **Полтергейст (Poltergeist):** Классы, которые слишком активно взаимодействуют с другими классами, обладая слишком большой функциональностью, что делает их сложными для понимания и обслуживания.
- **Плитка (Big Ball of Mud):** Система, в которой отсутствует ясная архитектура, код является неразделенным и сложно поддерживаемым.
- **Клонирование и прилипание (Cut-and-Paste Programming):** Применение метода копирования и вставки кода, ведущее к дублированию и ненадежности.
- **Необходимость повторения (Reinvent the Wheel):** Попытка переписать функциональность, которая уже существует в стандартных так и в сторонних библиотеках или фреймворках.
- **Неопределенность (Lava Flow):** Ненужный или устаревший код, который не удаляется, а накапливается в проекте.



Подведение итогов

На этой лекции вы:

- Разобрались с тем, что такое шаблон проектирования
- Узнали про GOF
- Освоили основные шаблоны проектирования
- Сложили представление об антипаттернах

Спасибо 
за внимание

