**Паттерны**

**Паттерны проектирования** или шаблоны проектирования — это повторяющиеся решения типичных задач, возникающих при проектировании программных систем. Они представляют собой описания хорошо зарекомендовавших себя архитектурных решений, которые применяются для решения определённых проблем в разработке программного обеспечения.

Основная цель паттернов — предложить разработчикам проверенные решения, которые повышают гибкость, удобочитаемость и поддержку кода.

1)Шаблон **фасада** — это [шаблон проектирования программного обеспечения](https://en.wikipedia.org/wiki/Software_design_pattern) , обычно используемый в [объектно-ориентированном программировании](https://en.wikipedia.org/wiki/Object-oriented_programming) . Аналогично [фасаду](https://en.wikipedia.org/wiki/Fa%C3%A7ade) в архитектуре, это [объект](https://en.wikipedia.org/wiki/Object_(computer_science)) , который служит в качестве интерфейса, маскирующего более сложный базовый или структурный код.

**Обоснование:**

Использование «Фасада» сделает работу с системой управления зоопарком проще и эффективнее, поскольку будет скрывать детали внутренней работы множества подсистем, предоставляя унифицированный интерфейс.

2)Шаблон **строителя** — это [шаблон проектирования](https://en.wikipedia.org/wiki/Software_design_pattern) , который обеспечивает гибкое решение различных проблем создания объектов в [объектно-ориентированном программировании](https://en.wikipedia.org/wiki/Object-oriented_programming) . Шаблон строителя [отделяет](https://en.wikipedia.org/wiki/Separation_of_concerns) создание сложного объекта от его представления.

**Обоснование:**

Паттерн «Строитель» упрощает создание сложных объектов с разными вариациями. В случае зоопарка это особенно полезно при необходимости создавать сложные объекты вроде вольеров или других конструкций, которые должны соответствовать разным видам животных.

3) Шаблон **команды** — это [поведенческий](https://en.wikipedia.org/wiki/Behavioral_pattern)[шаблон проектирования](https://en.wikipedia.org/wiki/Design_pattern_(computer_science)) , в котором объект используется для [инкапсуляции](https://en.wikipedia.org/wiki/Information_hiding) всей информации, необходимой для выполнения действия или запуска события в более позднее время. Эта информация включает имя метода, объект, которому принадлежит метод, и значения для параметров метода.

**Обоснование:**

«Команда» позволяет легко управлять действиями, которые должны выполняться по расписанию или при необходимости, что особенно важно для зоопарка с его сложными операциями и процессами.

4) Шаблон **цепочки ответственности** представляет собой [поведенческий](https://en.wikipedia.org/wiki/Behavioral_pattern)[шаблон проектирования](https://en.wikipedia.org/wiki/Design_pattern_(computer_science)) , состоящий из источника [объектов команд](https://en.wikipedia.org/wiki/Command_pattern) и ряда **объектов обработки** . [[ 1 ]](https://en.wikipedia.org/wiki/Chain-of-responsibility_pattern#cite_note-1) Каждый объект обработки содержит логику, которая определяет типы объектов команд, которые он может обрабатывать; остальные передаются следующему объекту обработки в цепочке. Также существует механизм для добавления новых объектов обработки в конец этой цепочки.

**Обоснование:**

«Цепочка обязанностей» упрощает передачу запросов и распределение задач между различными службами в зоопарке, улучшая обработку событий и заявок.

5) Шаблон **метода фабрики** — это [шаблон проектирования](https://en.wikipedia.org/wiki/Software_design_pattern) , который использует методы фабрики для решения проблемы [создания объектов](https://en.wikipedia.org/wiki/Object_creation) без указания их точных [классов](https://en.wikipedia.org/wiki/Class_(computer_programming)) . Вместо вызова [конструктора](https://en.wikipedia.org/wiki/Constructor_(object-oriented_programming)) это достигается путем вызова метода фабрики для создания объекта. Методы фабрики могут быть указаны в [интерфейсе](https://en.wikipedia.org/wiki/Interface_(object-oriented_programming)) и реализованы подклассами или реализованы в базовом классе и, при необходимости, [переопределены](https://en.wikipedia.org/wiki/Method_overriding) подклассами

**Обоснование:**

«Фабричный метод» упрощает создание объектов, давая гибкость при добавлении новых типов животных в систему.