**Доклад на тему**

«Шаблоны проектирования. Порождающие паттерны.»

1. **Определение**

Порождающие паттерны проектирования представляют собой специальные концепции, спроектированные для эффективного создания объектов и компонентов в гибком и управляемом стиле. Они способствуют повторному использованию кода и упрощают процесс разработки, делая его более гибким и удобным.

1. **Назначение и цели применения**

Основная цель порождающих паттернов - обеспечить гибкость и расширяемость при создании сложных объектов и структур. Они также помогают снизить зависимость между компонентами системы и улучшить тестируемость кода. Эти паттерны также улучшают читаемость кода и его структурирование, разделяя ответственность по созданию объектов от их использования. Такой подход способствует повышению ясности и понимаемости кода, что важно для командной разработки и сопровождения проекта.

1. **Особенности применения**
   1. Создание объектов: Порождающие паттерны используются для создания объектов, причем способ создания может быть абстрагирован от клиентского кода. Это позволяет упростить процесс создания объектов и изолировать детали их конструирования.
   2. Сокрытие деталей создания: Одной из целей порождающих паттернов является сокрытие деталей процесса создания объектов от клиентского кода. Это способствует уменьшению зависимостей между компонентами системы.
   3. Обеспечение гибкости: Порождающие паттерны обеспечивают гибкость в создании объектов. Они позволяют изменять тип создаваемых объектов или их конфигурацию, не затрагивая код, который использует эти объекты.
   4. Улучшение повторного использования: Использование порождающих паттернов способствует повторному использованию кода, поскольку они позволяют создавать объекты с учетом различных условий и требований, не изменяя основной логики программы.
   5. Сокращение зависимостей: Паттерны этой категории помогают уменьшить зависимости между компонентами системы, так как они обеспечивают абстракции для создания объектов и скрывают детали их реализации.
2. **Фабрика**
   1. Описание

Паттерн Фабрика предоставляет интерфейс для создания объектов без указания их конкретных классов.

* 1. Примеры применения

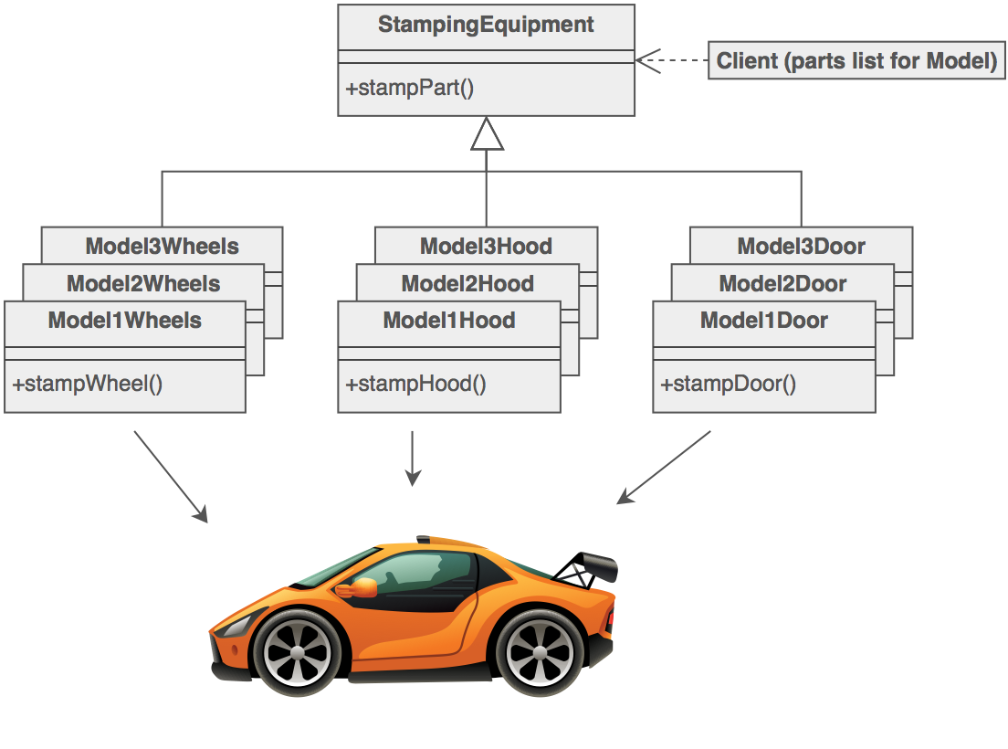
Фабрика может быть использована, когда создание объекта требует сложной логики или зависит от других компонентов системы.

* 1. Преимущества

Обеспечивает гибкость и возможность поддержки различных типов создаваемых объектов без изменения существующего кода.

* 1. Недостатки

Может привести к большому количеству кода и увеличенной сложности в случае использования большого количества фабрик.



1. **Абстрактная фабрика**
   1. Описание

Абстрактная фабрика предоставляет интерфейс для создания семейств взаимосвязанных или взаимозависимых объектов.

* 1. Примеры применения

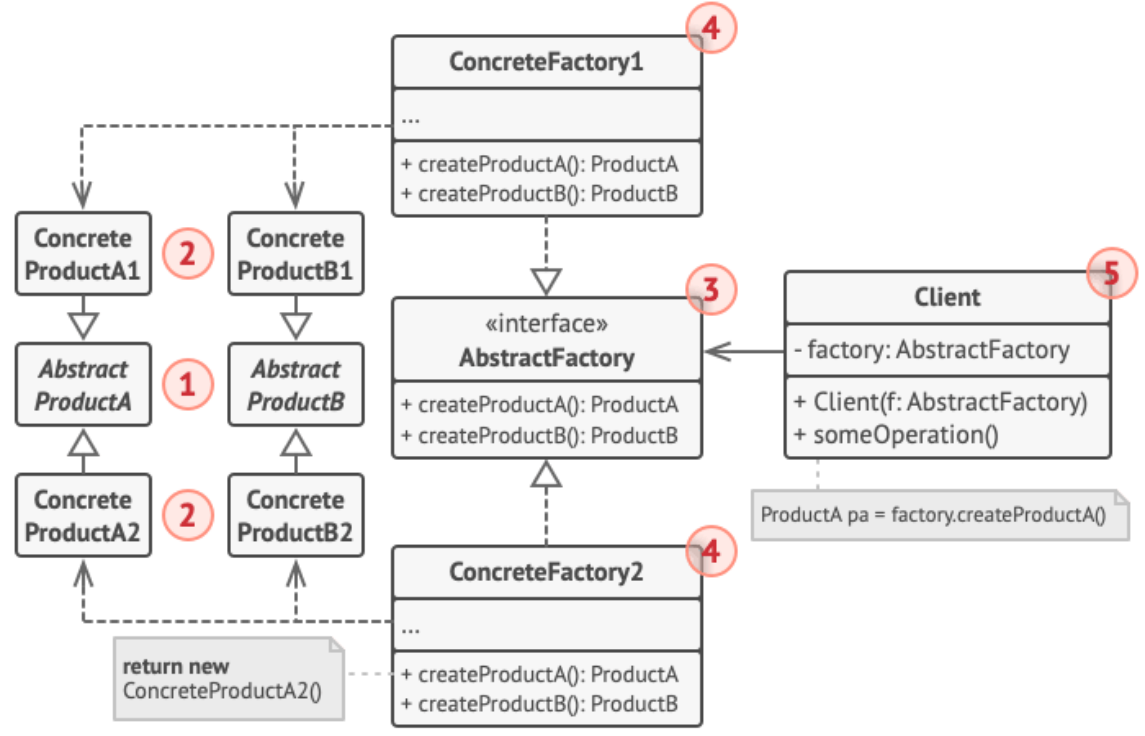
Абстрактная фабрика может быть использована, когда необходимо создать объекты с определенными свойствами, удовлетворяющими определенному контракту.

* 1. Преимущества

Позволяет создавать семейства объектов совместимыми и независимыми от спецификации их конкретных классов.

* 1. Недостатки

Усложняет архитектуру системы за счет увеличения количества абстракций и классов.



1. **Строитель**
   1. Описание

Строитель позволяет создавать сложные объекты, используя один и тот же процесс конструирования, но различные представления и конфигурации.

* 1. Примеры применения

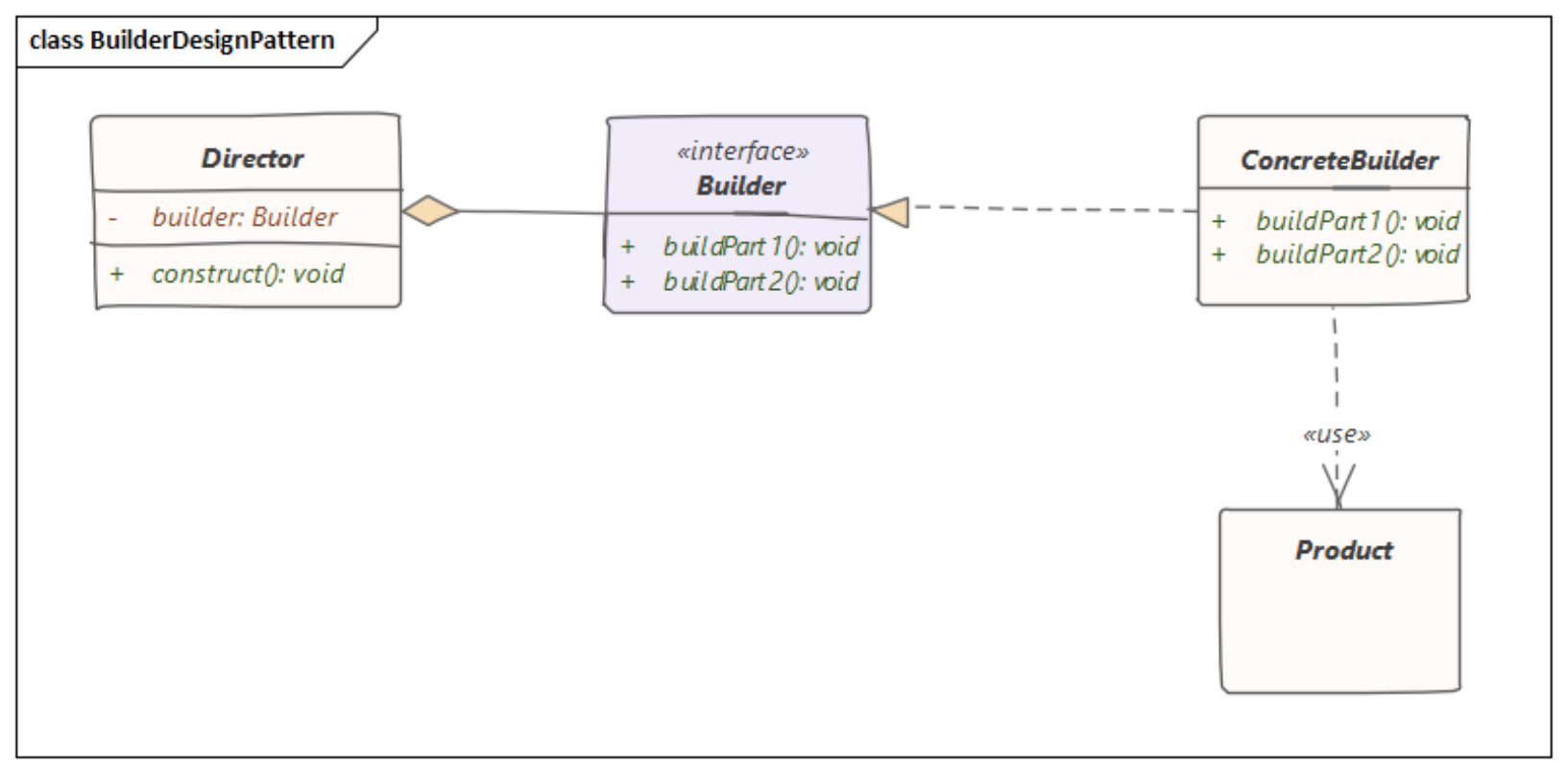
Строитель может быть полезен, когда требуется постепенно создавать и настраивать объект с множеством опций и параметров.

* 1. Преимущества

Облегчает процесс создания сложных объектов путем выделения шагов конструирования и их последовательного выполнения.

* 1. Недостатки

Может привести к избыточности создания классов-строителей, если их число и сложность становятся неуправляемыми.



1. **Прототип**
   1. Описание

Прототип позволяет создавать объекты, базируясь на уже существующих объектах-прототипах, без явного указания их классов.

* 1. Примеры применения

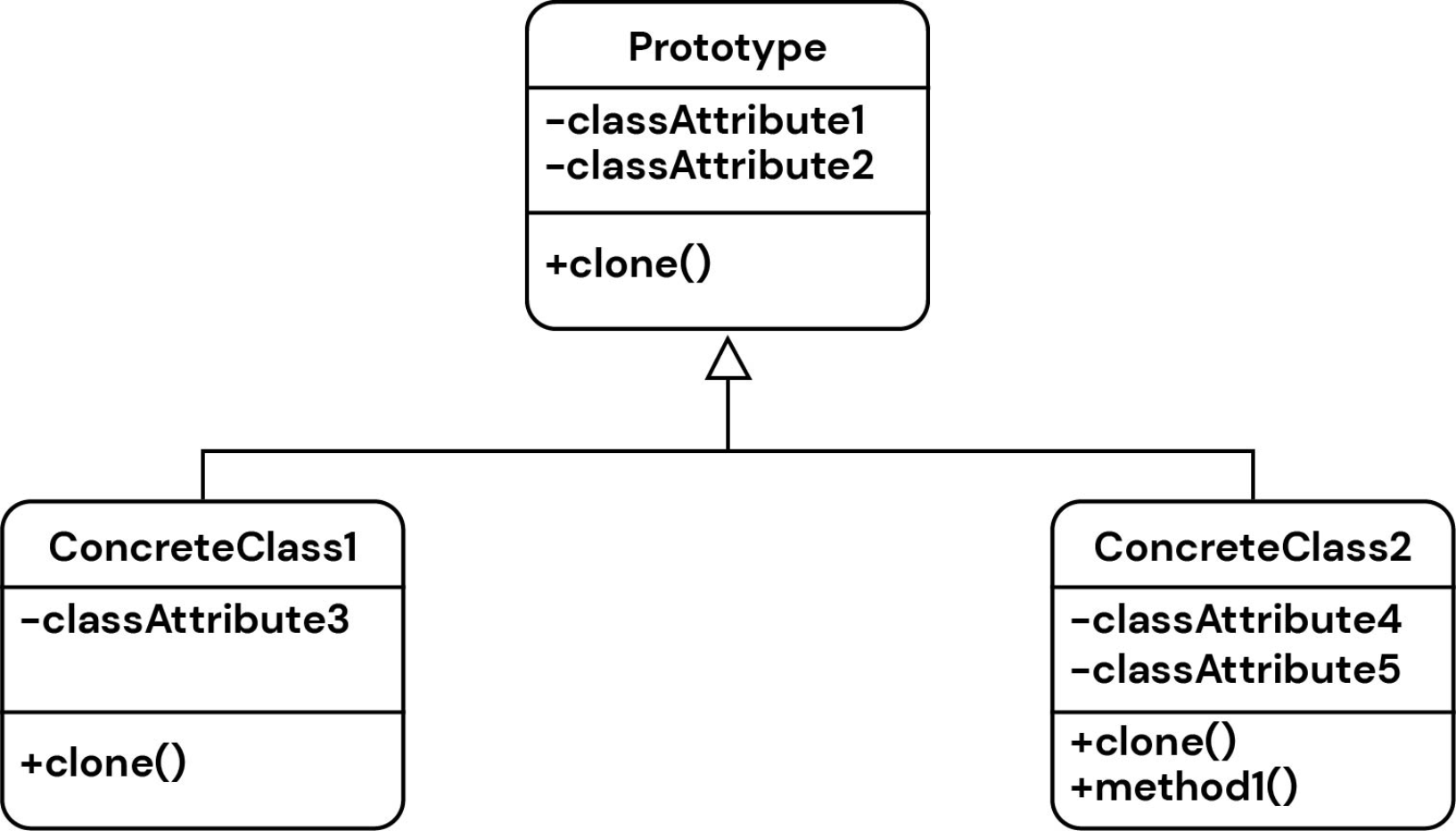
Прототип может быть полезен, когда требуется создать объекты с уже установленными свойствами или параметрами.

* 1. Преимущества

Позволяет создавать объекты, не зависящие от конкретных классов и использующие уже существующие прототипы.

* 1. Недостатки

Для некоторых объектов создание точных копий может быть трудным или даже невозможным.



1. **Одиночка**
   1. Описание

Одиночка гарантирует, что класс имеет только один экземпляр, и предоставляет глобальную точку доступа к нему.

* 1. Примеры применения

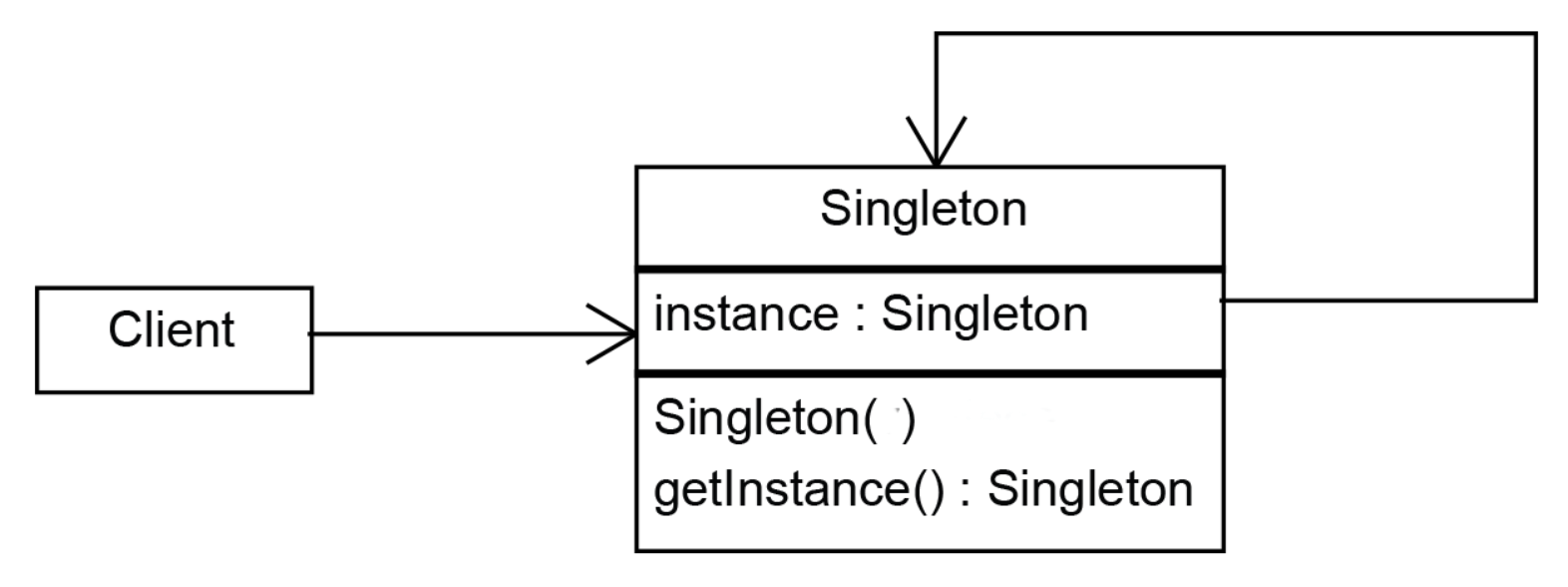
Одиночка полезен, когда требуется обеспечить доступ к единственному экземпляру класса или создать разделяемый ресурс.

* 1. Преимущества

Гарантирует наличие только одного экземпляра класса и предоставляет глобальную точку доступа.

* 1. Недостатки

Использование одиночки может создавать жесткую связь между компонентами системы, что затрудняет их переиспользование и изменение.



1. **Подведём итоги:**
2. Абстрактная фабрика: предоставляет интерфейс для создания семейств связанных или зависимых объектов без указания их конкретных классов.
3. Строитель: позволяет создавать сложные объекты пошагово. Он разделяет процесс создания объекта на несколько этапов, что позволяет использовать один и тот же процесс создания для разных представлений.
4. Фабричный метод: предоставляет интерфейс для создания объектов, но позволяет подклассам выбирать класс создаваемого экземпляра.
5. Прототип: позволяет создавать объекты путем клонирования уже существующих объектов.
6. Одиночка: гарантирует, что класс имеет только один экземпляр, и предоставляет глобальную точку доступа к этому экземпляру.