Java Design Patterns

State

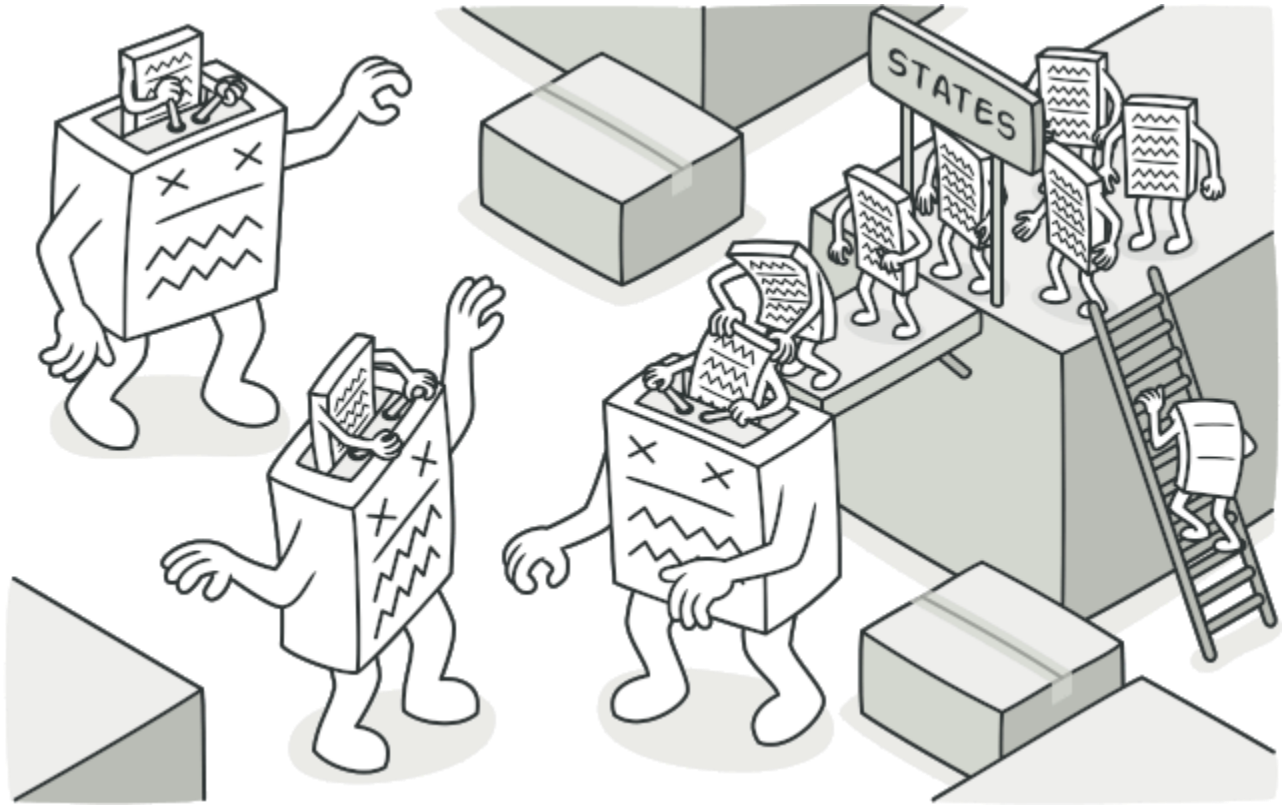
Тема

State

Суть паттерна

Состояние

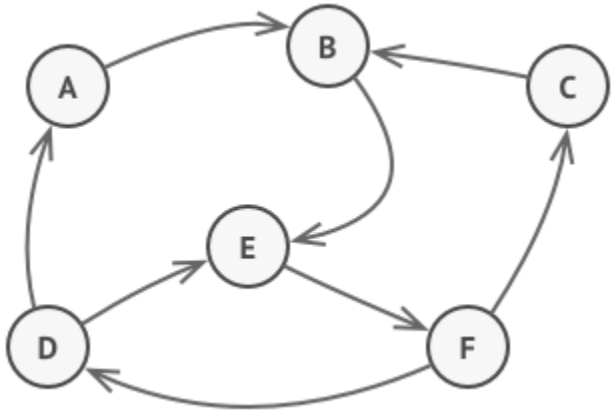
Состояние — это поведенческий паттерн проектирования, который позволяет объектам менять поведение в зависимости от своего состояния. Извне создаётся впечатление, что изменился класс объекта.



Проблема

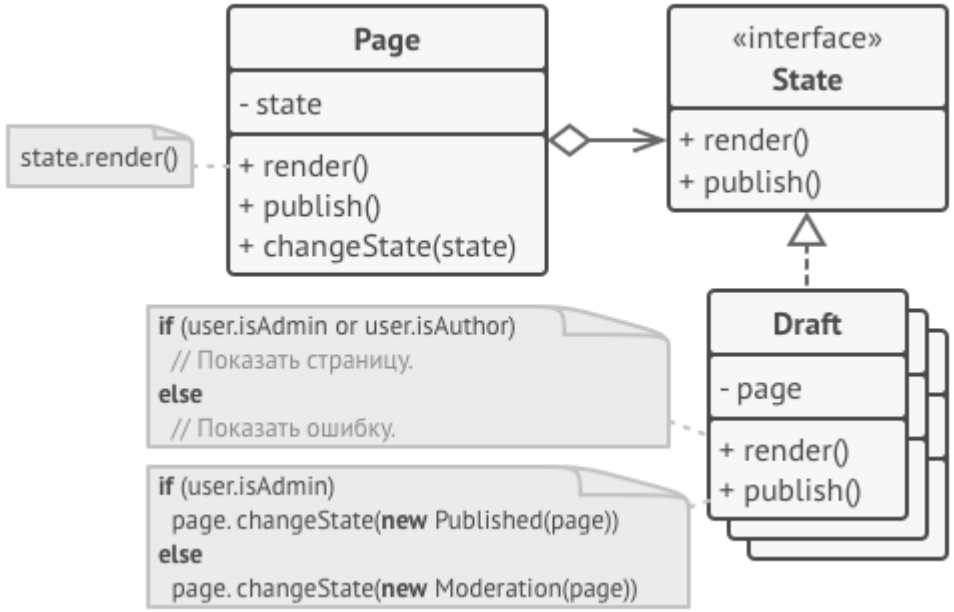
Постановка задачи

Паттерн Состояние невозможно рассматривать в отрыве от концепции машины состояний (также известной как *стейт-машина* или *конечный автомат*).



Решение задачи

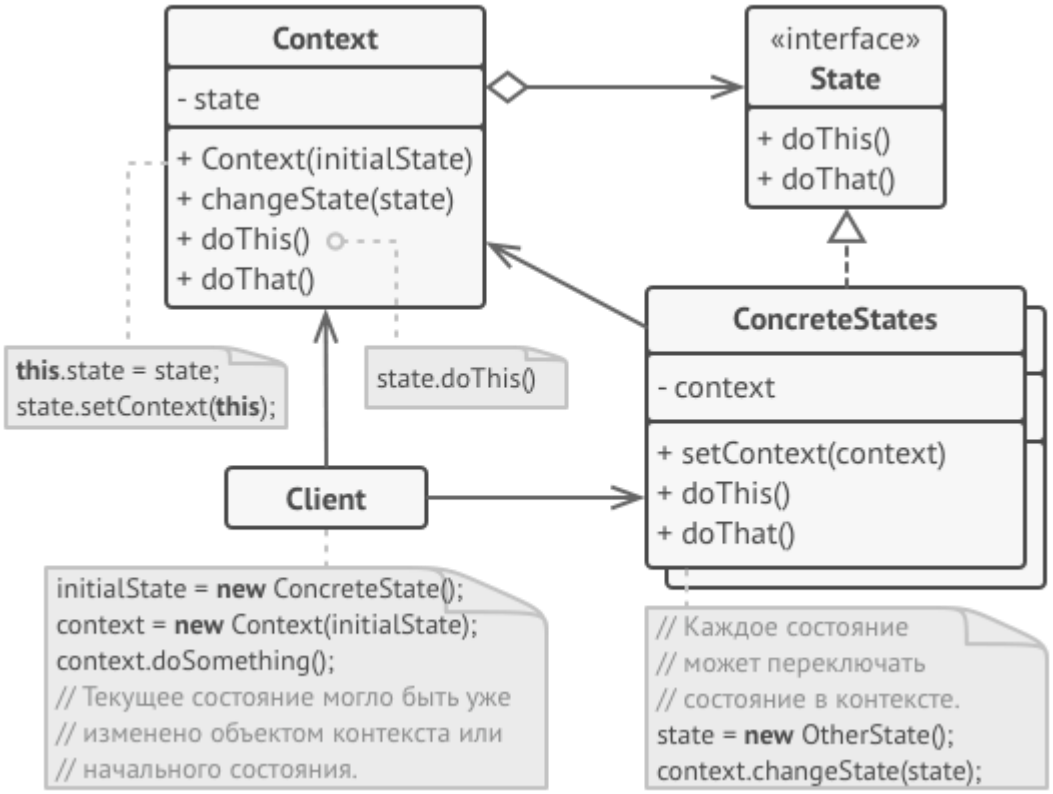
Паттерн Состояние предлагает создать отдельные классы для каждого состояния, в котором может пребывать контекстный объект, а затем вынести туда поведения, соответствующие этим состоянию.



Структура

Структура паттерна

1. Контекст хранит ссылку на объект состояния и делегирует ему работу, зависящую от внутреннего состояния. Контекст работает с этим объектом через общий интерфейс состояний.
2. Состояние описывает общий интерфейс для всех конкретных состояний.
3. Конкретные состояния реализуют поведения, связанные с определённым состоянием контекста. Иногда приходится создавать целые иерархии классов состояний, чтобы обобщить дублирующий код.
4. И контекст, и объекты конкретных состояний могут решать, когда и какое следующее состояние будет выбрано. Чтобы переключить состояние, нужно подать другой объект-состояние в контекст.



Применимость

Применение паттерна

1. Когда у вас есть объект, поведение которого кардинально меняется в зависимости от внутреннего состояния. Причём типов состояний много и их код часто меняется.
2. Когда код класса содержит множество больших, похожих друг на друга, условных операторов, которые выбирают поведения в зависимости от текущих значений полей класса.
3. Когда вы сознательно используете табличную машину состояний, построенную на условных операторах, но вынуждены мириться с дублированием кода для похожих состояний и переходов.

Шаги реализации

Алгоритм реализации паттерна

1. Определитесь с классом, который будет играть роль контекста. Это может быть как существующий класс, в котором уже есть зависимость от состояния, так и новый класс, если код состояний размазан по нескольким классам.
2. Создайте интерфейс состояний. Он должен описывать методы, общие для всех состояний, обнаруженных в контексте. Заметьте, что не всё поведение контекста нужно переносить в состояние, а только то, которое зависит от состояний.
3. Для каждого фактического состояния, создайте класс, реализующий интерфейс состояния. Переместите весь код, связанный с конкретным состоянием в нужный класс. В конце концов, все методы интерфейса состояния должны быть реализованы.
4. Создайте в контексте поле для хранения объектов-состояний, а также публичный метод для изменения значения этого поля.
5. Старые методы контекста, в которых находился зависимый от состояния код, замените на вызовы соответствующих методов объекта-состояния.
6. В зависимости от бизнес-логики, разместите код, который переключает состояние контекста либо внутри контекста, либо внутри классов конкретных состояний.

Преимущества и недостатки

Плюсы и недостатки

Плюсы:

- Избавляет от множества больших условных операторов машины состояний.
- Концентрирует в одном месте код, связанный с определённым состоянием.
- Упрощает код контекста.

Минусы:

- Может неоправданно усложнить код, если состояний мало и они редко меняются.

Отношение с другими паттернами

- Мост, Стратегия и Состояние (а также слегка и Адаптер) имеют схожие структуры классов — все они построены на принципе «композиции», то есть делегирования работы другим объектам. Тем не менее, они отличаются тем, что решают разные проблемы. Помните, что паттерны — это не только рецепт построения кода определённым образом, но и описание проблем, которые привели к данному решению.
- Состояние можно рассматривать как надстройку над Стратегией. Оба паттерна используют композицию, чтобы менять поведение основного объекта, делегируя работу вложенным объектам-помощникам. Однако в *Стратегии* эти объекты не знают друг о друге и никак не связаны. В *Состоянии* сами конкретные состояния могут переключать контекст.

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association