Java Design Patterns

Template Method

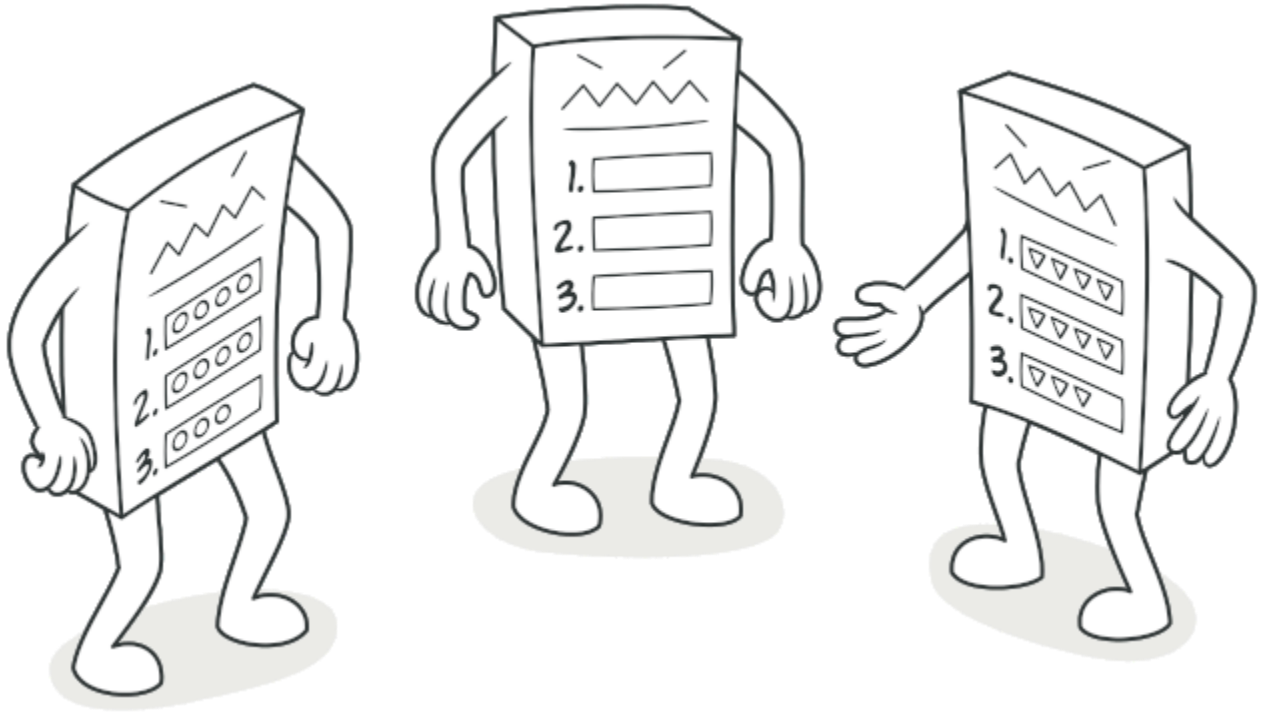
Тема

Template Method

Суть паттерна

Шаблонный метод

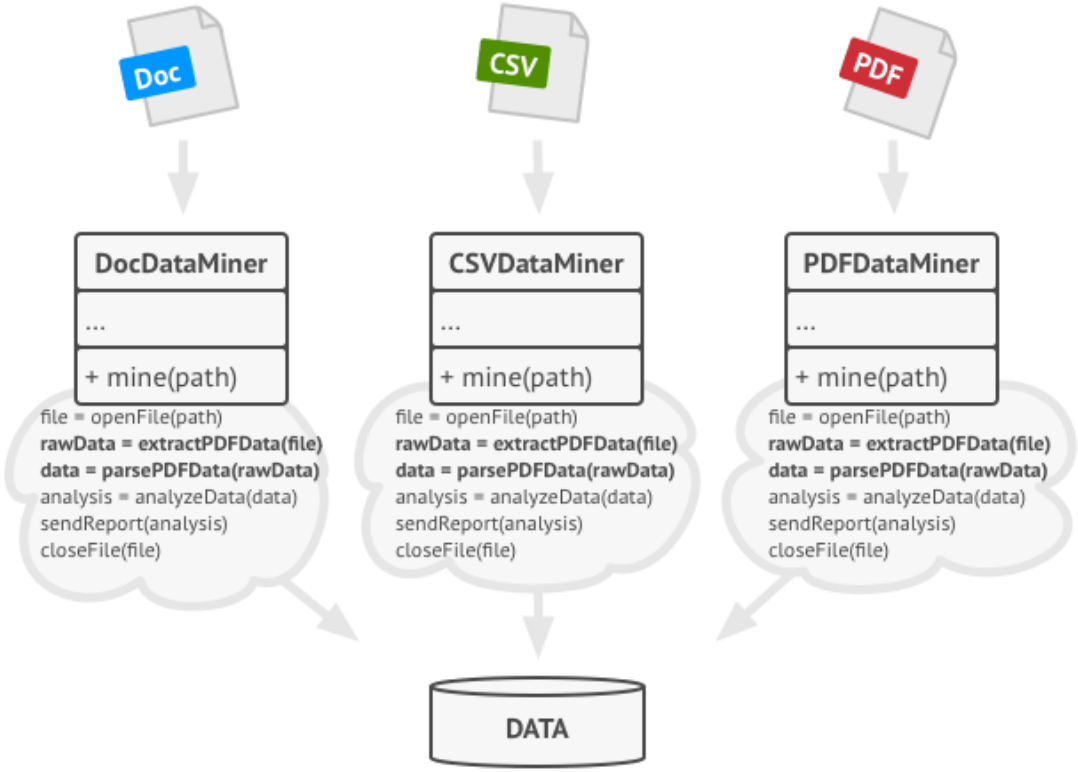
Шаблонный метод — это поведенческий паттерн проектирования, который определяет скелет алгоритма, перекладывая ответственность за некоторые его шаги на подклассы. Паттерн позволяет подклассам переопределять шаги алгоритма, не меняя его общей структуры.



Проблема

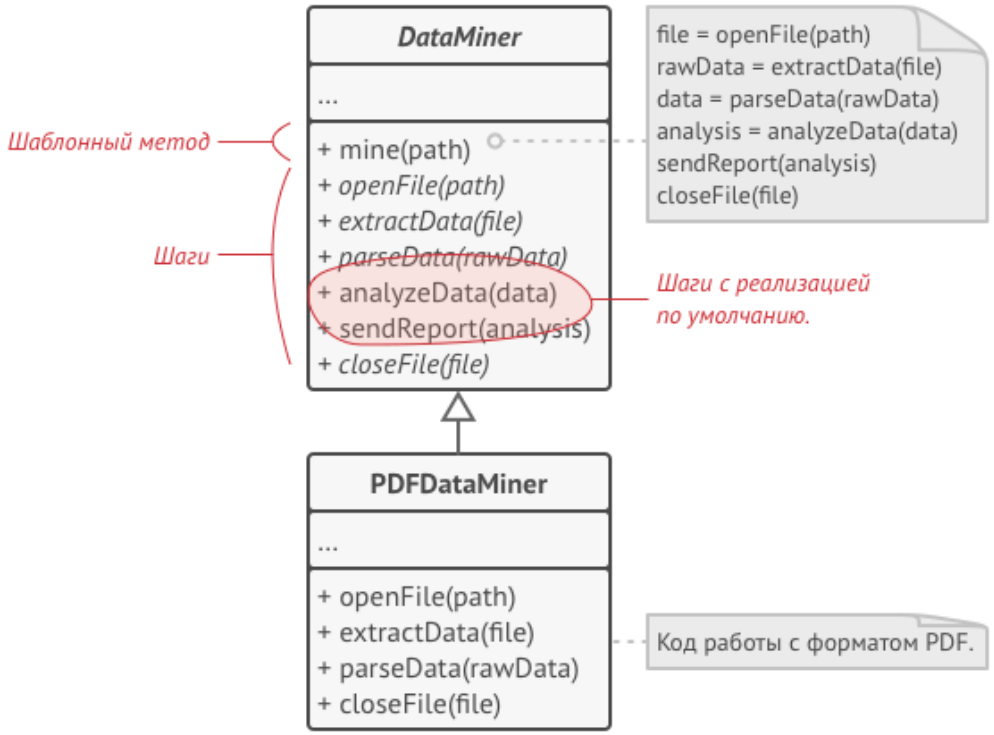
Постановка задачи

Вы пишете программу для дата-майнинга в офисных документах. Пользователи будут загружать в неё документы в разных форматах (PDF, DOC, CSV), а программа должна извлекать из них полезную информацию.



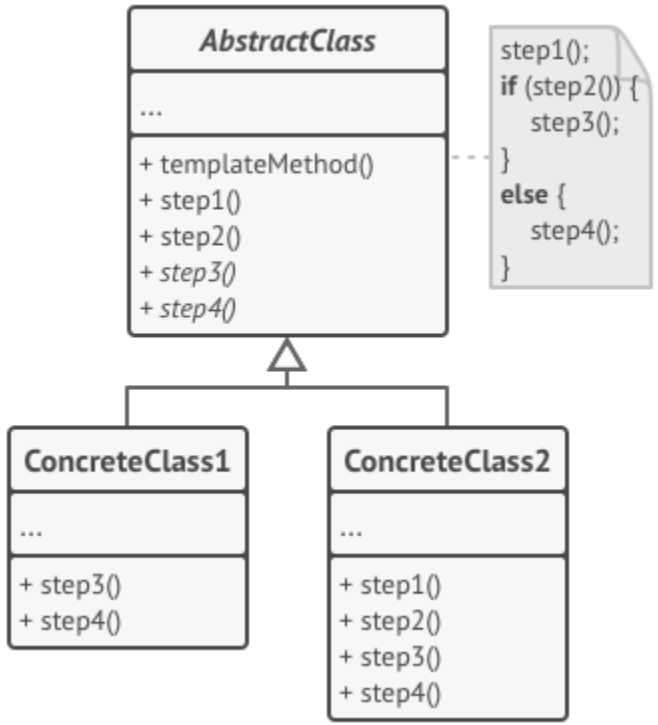
Решение задачи

Паттерн Шаблонный метод предлагает разбить алгоритм на последовательность шагов, описать шаги в отдельных методах и вызывать их в одном «шаблонном» методе друг за другом.



Структура паттерна

1. Абстрактный класс определяет шаги алгоритма и содержит шаблонный метод, состоящий из вызовов этих шагов. Шаги могут быть как абстрактными, так и содержать реализацию по умолчанию.
2. Конкретный класс переопределяет некоторые (или все) шаги алгоритма. Конкретные классы не переопределяют сам шаблонный метод.



Применимость

Применение паттерна

1. Когда подклассы должны расширять базовый алгоритм, не меняя его структуры.
2. Когда у вас есть несколько классов, делающих одно и то же с незначительными отличиями. Если вы редактируете один класс, то приходится вносить такие же правки и в остальные классы.

Шаги реализации

Алгоритм реализации паттерна

1. Изучите алгоритм и подумайте, можно ли его разбить на шаги. Прикиньте, какие шаги будут стандартными для всех вариаций алгоритма, а какие — изменчивыми.
2. Создайте абстрактный базовый класс. Определите в нём шаблонный метод. Этот метод должен состоять из вызовов шагов алгоритма. Имеет смысл сделать шаблонный метод финальным, чтобы подклассы не могли переопределить его (если ваш язык программирования это позволяет).
3. Добавьте в абстрактный класс методы для каждого из шагов алгоритма. Вы можете сделать эти методы абстрактными или добавить какую-то реализацию по умолчанию. В первом случае, все подклассы *должны* будут реализовать эти методы, а во втором — только если реализация шага в подклассе отличается от стандартной версии.
4. Подумайте о введении в алгоритм хуков. Чаще всего, хуки располагают между основными шагами алгоритма, а также до и после всех шагов.
5. Создайте конкретные классы, унаследовав их от абстрактного класса. Реализуйте в них все недостающие шаги и хуки.

Преимущества и недостатки

Плюсы и недостатки

Плюсы:

- Облегчает повторное использование кода.

Минусы:

- Вы жёстко ограничены скелетом существующего алгоритма.
- Вы можете нарушить *принцип подстановки Барбары Лисков*, изменяя базовое поведение одного из шагов алгоритма через подкласс.
- С ростом количества шагов, шаблонный метод становится слишком сложно поддерживать.

Отношение с другими паттернами

- Фабричный метод можно рассматривать как частный случай Шаблонного метода. Кроме того, *Фабричный метод* нередко бывает частью большого класса с *Шаблонными методами*.
- Шаблонный метод использует наследование, чтобы расширять части алгоритма. Стратегия использует делегирование, чтобы изменять выполняемые алгоритмы на лету. *Шаблонный метод* работает на уровне классов. *Стратегия* позволяет менять логику отдельных объектов.

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