НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Паттерны (шаблоны) проектирования

Поведенческие паттерны

Eugeny Berkunsky, Computer Science dept.,
National University of Shipbuilding
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Поведенческие паттерны

Поведенческие шаблоны (*behavioral patterns*) —шаблоны проектирования, определяющие алгоритмы и способы реализации взаимодействия различных объектов и классов.

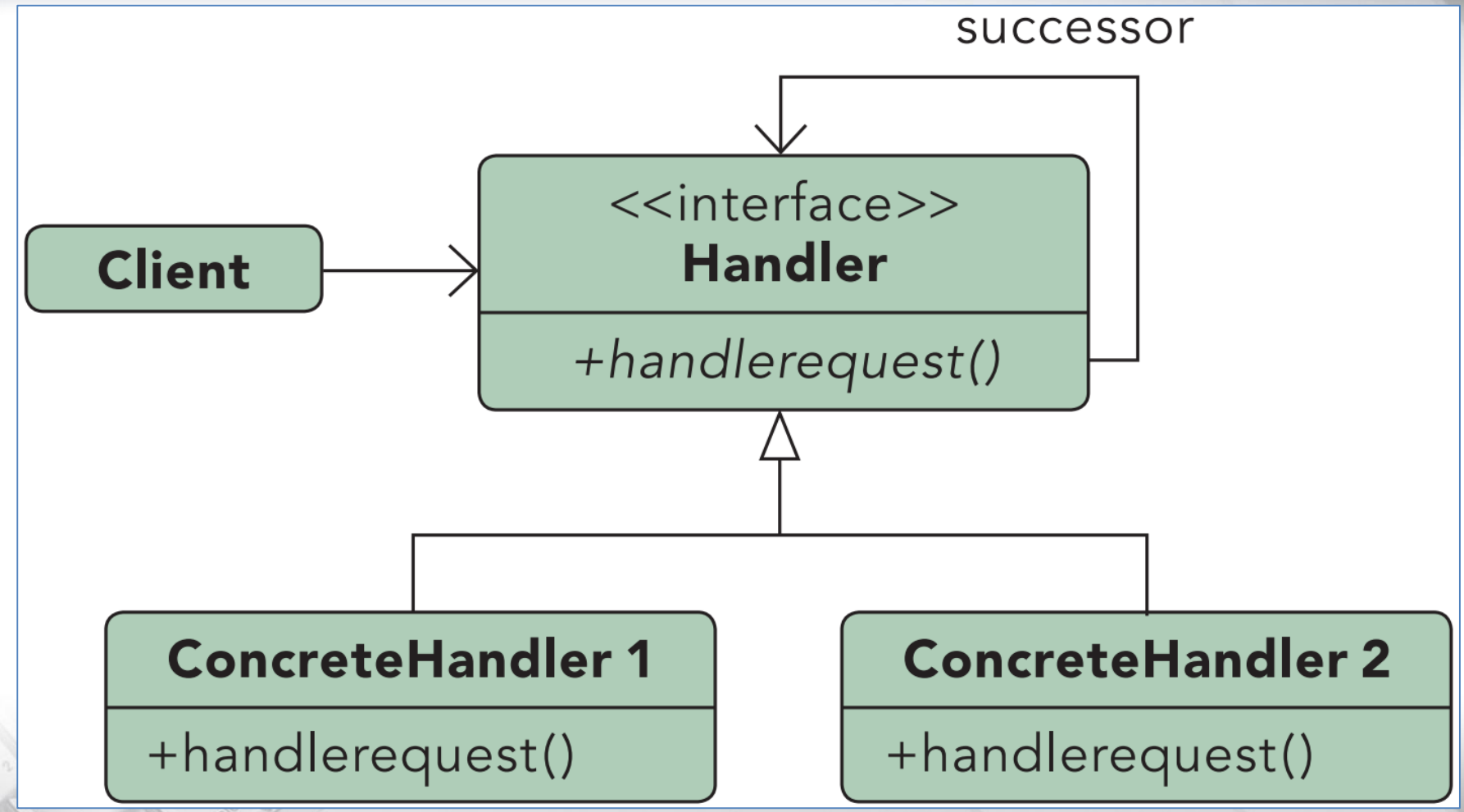
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Поведенческие паттерны

- Цепочка ответственности – Chain of Responsibility
- Команда – Command (Action, Transaction)
- Интерпретатор – Interpreter
- Итератор – Iterator (Cursor)
- Посредник – Mediator
- Напоминание – Memento (Хранитель, Token)
- Null Object (Null object)
- Наблюдатель – Observer (Dependents, Publish-Subscribe, Listener)
- Состояние – State (Objects for States)
- Стратегия – Strategy
- Шаблонный метод – Template Method
- Посетитель – Visitor
- Слуга – Servant

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Цепочка ответственности
Chain of Responsibility

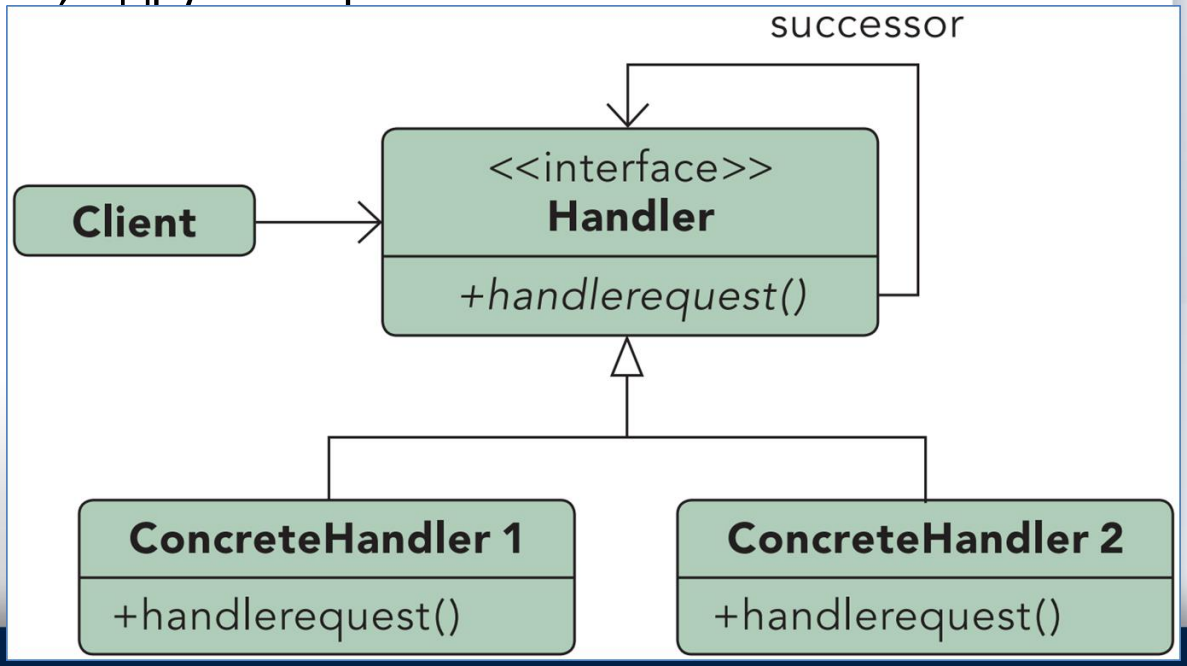


НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

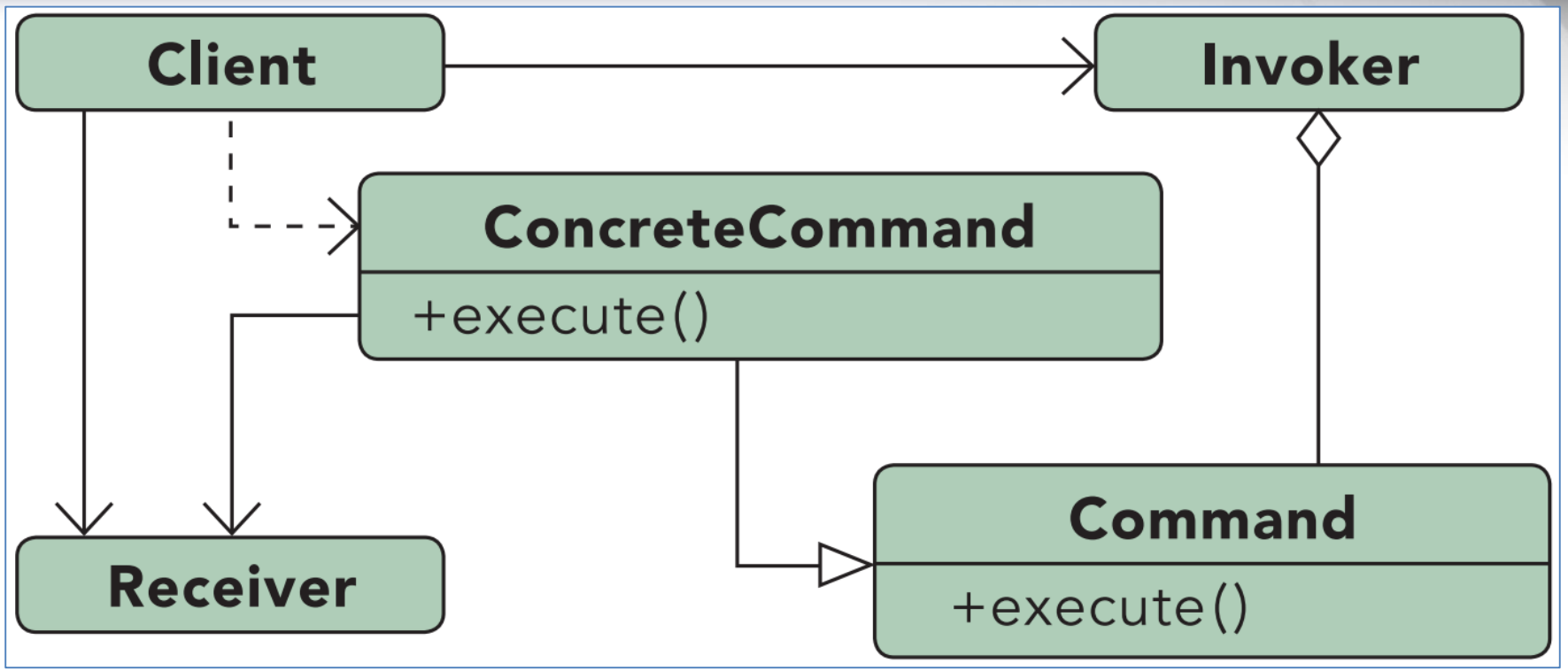
Цепочка ответственности Chain of Responsibility

Стоит использовать, если:

- в разрабатываемой системе имеется группа объектов, которые могут обрабатывать сообщения определенного типа;
- все сообщения должны быть обработаны хотя бы одним объектом системы;
- сообщения в системе обрабатываются по схеме «обработай сам либо перешли другому», то есть одни сообщения обрабатываются на том уровне, где они получены, а другие пересылаются объектам иного уровня.



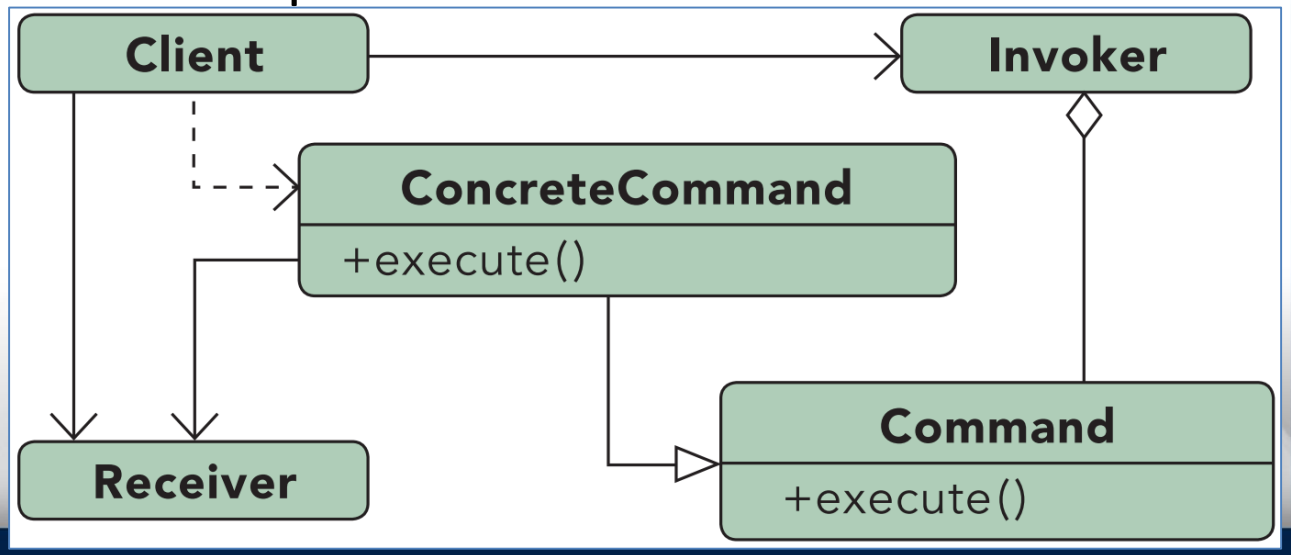
Команда / Command



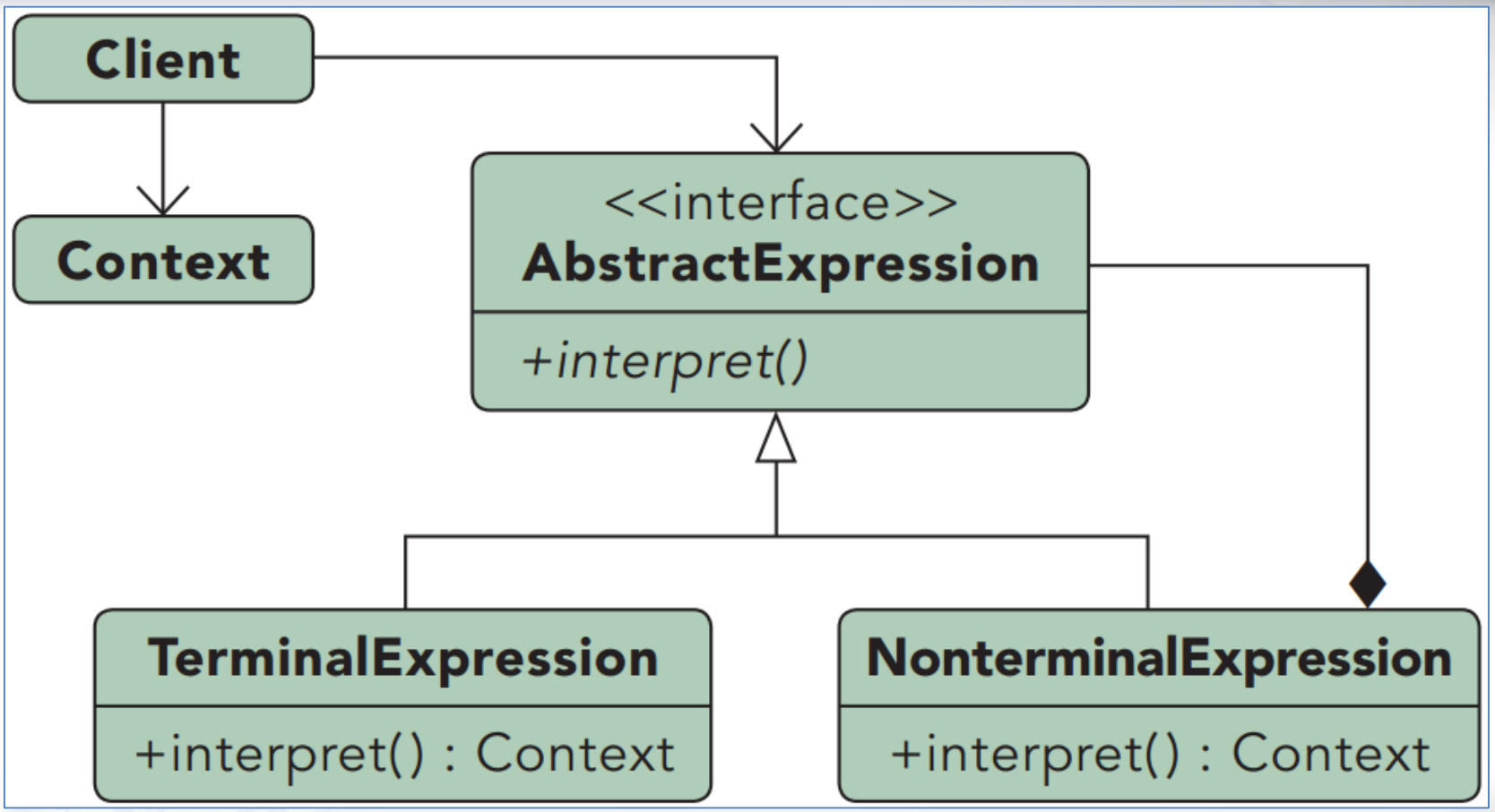
Команда / Command

Обеспечивает обработку команды в виде объекта, что позволяет сохранять её, передавать в качестве параметра методам, а также возвращать её в виде результата, как и любой другой объект.

- Создание структуры, в которой класс-отправитель и класс-получатель не зависят друг от друга напрямую.
- Организация обратного вызова к классу, который включает в себя класс-отправитель.



Интерпретатор / Interpreter

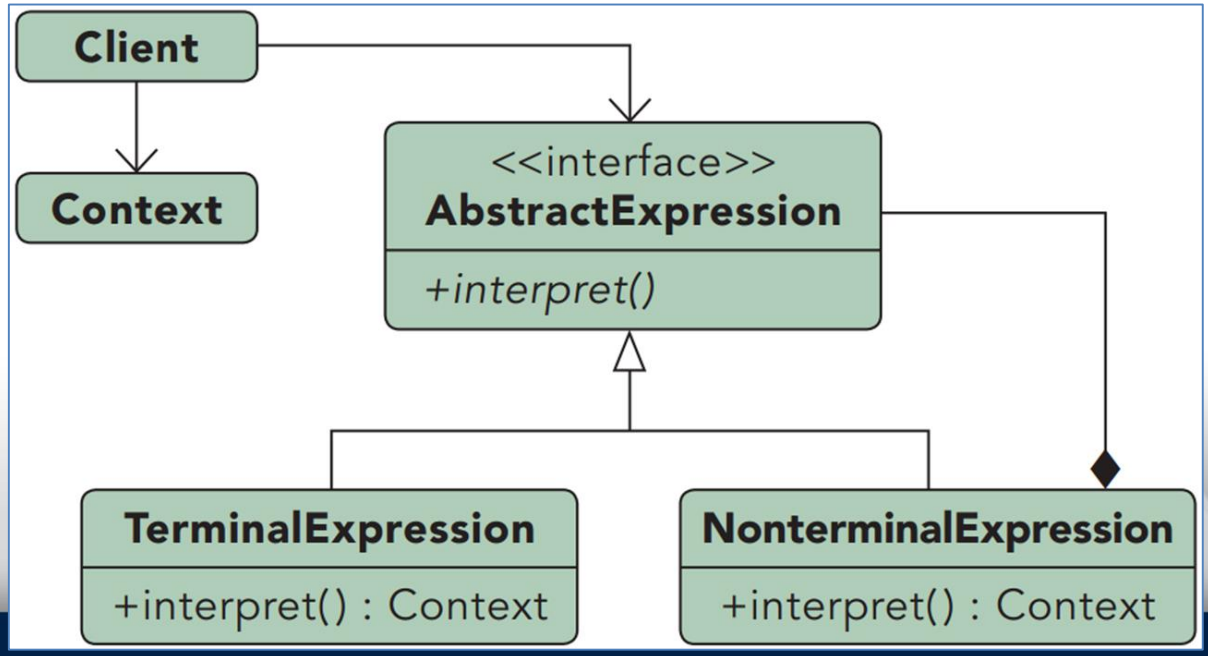


Интерпретатор / Interpreter

Нужно решать часто встречающуюся, но подверженную изменениям задачу

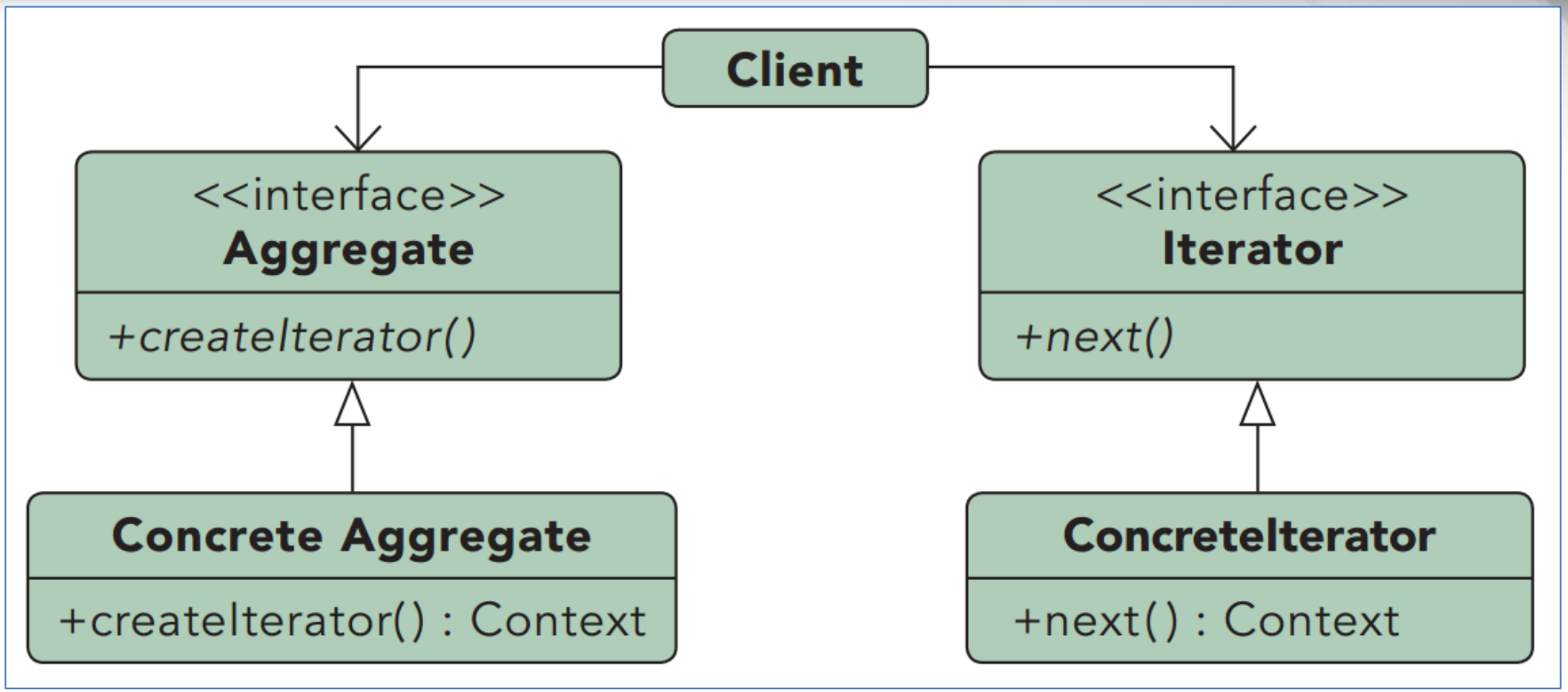
Нужно создать интерпретатор, который решает данную задачу.

Грамматику становится легко расширять и изменять, реализации классов, описывающих узлы абстрактного синтаксического дерева похожи (легко кодируются). Можно легко изменять способ вычисления выражений.



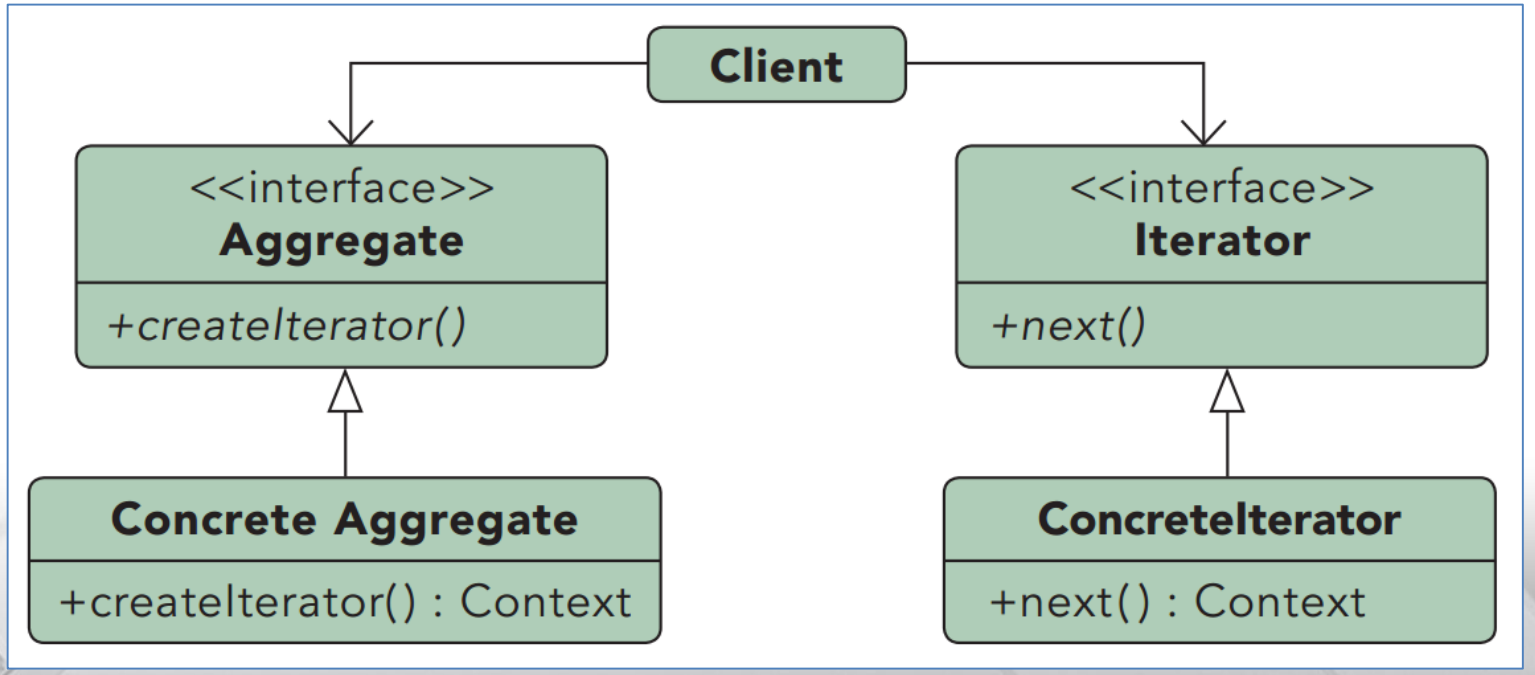
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Итератор / Iterator



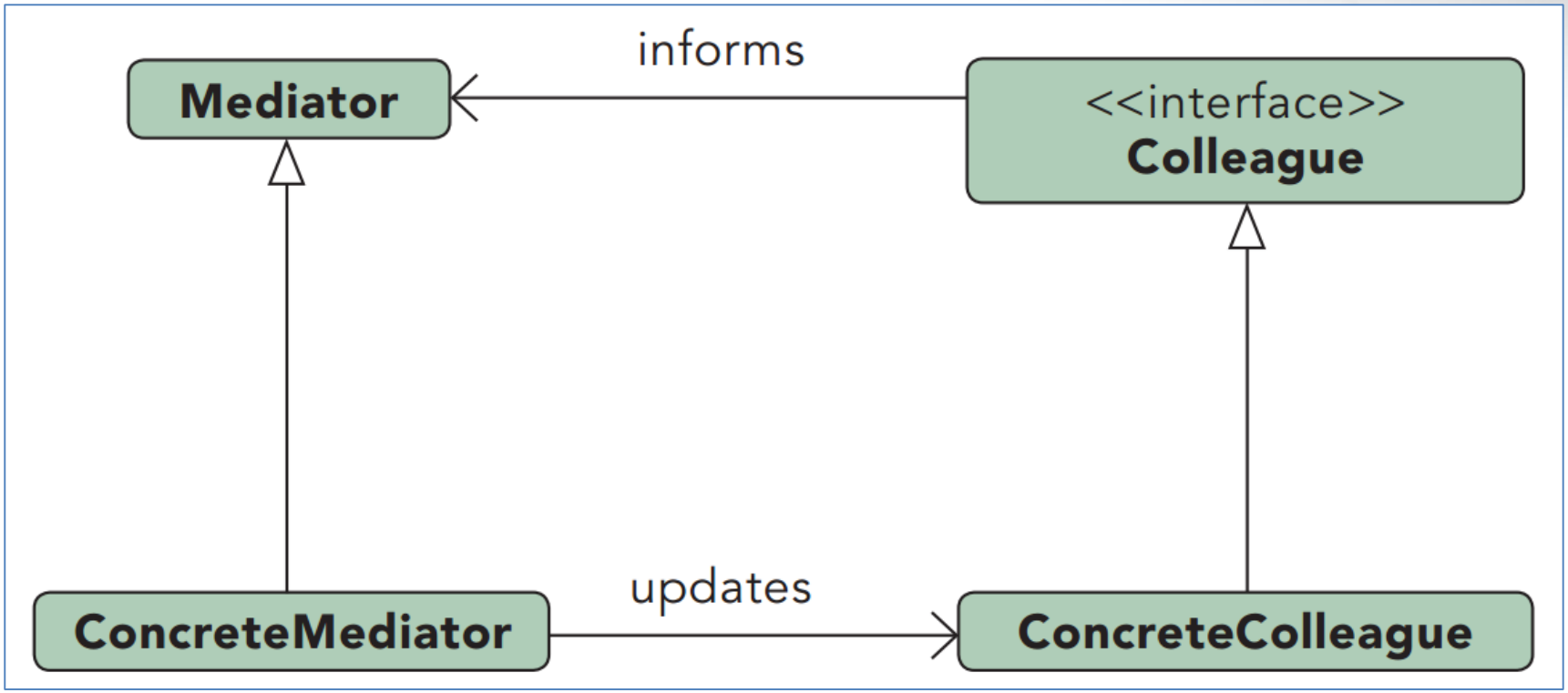
Итератор / Iterator

Представляет собой объект, позволяющий получить последовательный доступ к элементам объекта-агрегата без использования описаний каждого из объектов, входящий в состав агрегации.



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Посредник / Mediator

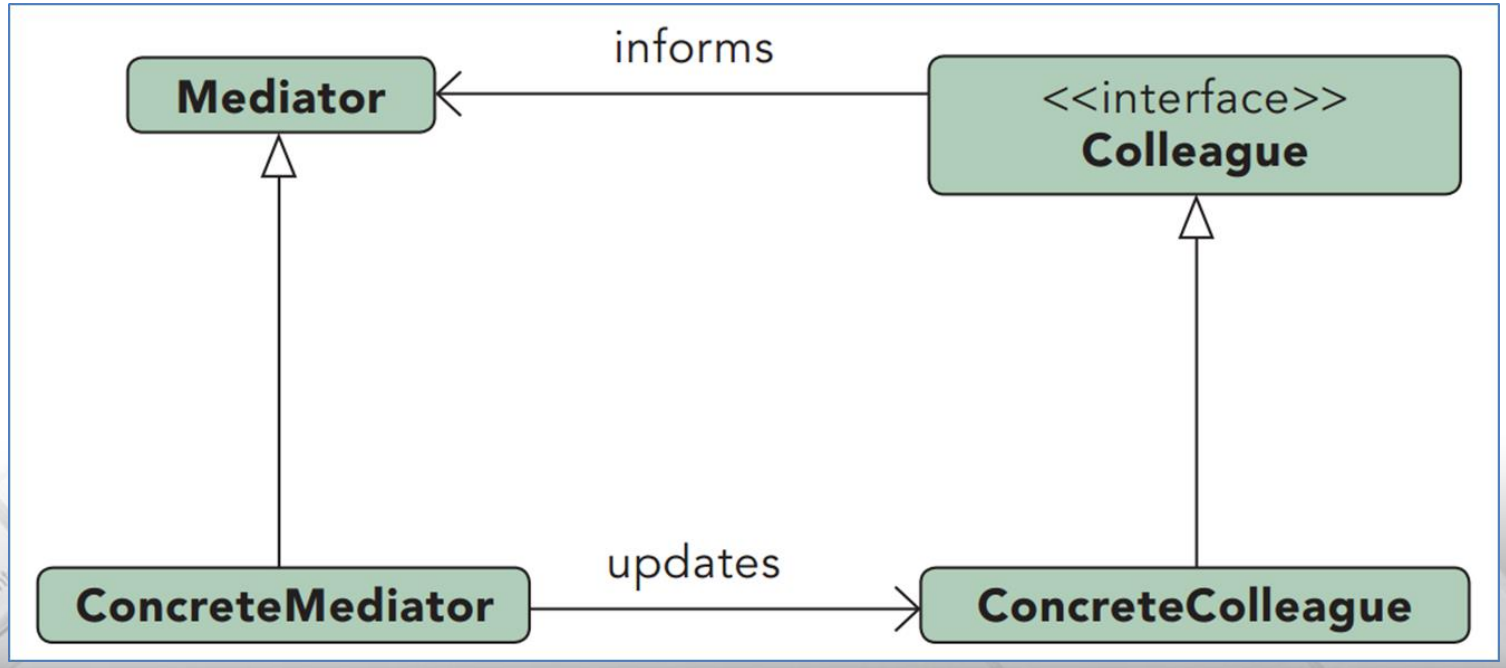


НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

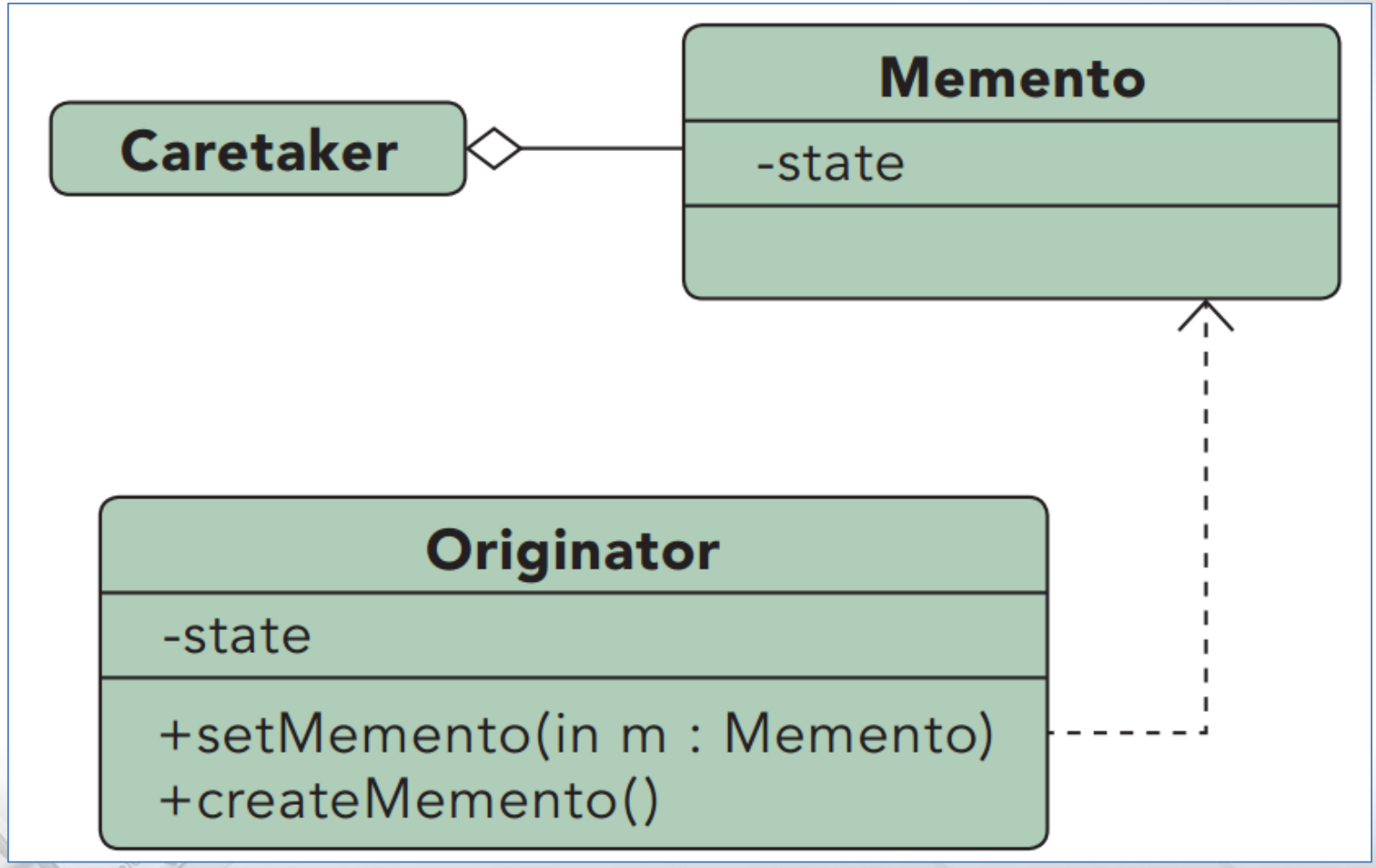
Посредник / Mediator

Решает задачу обеспечения взаимодействия множества объектов, сформировав при этом слабую связанность и избавив объекты от необходимости явно ссылаться друг на друга.

Посредник похож на Фасад в том что он абстрагирует функциональность существующих классов. Посредник абстрагирует/централизует произвольные коммуникации между объектами-сотрудниками. Он просто "добавляет значение" и оно известно (или на него ссылается) объекту-сотруднику.



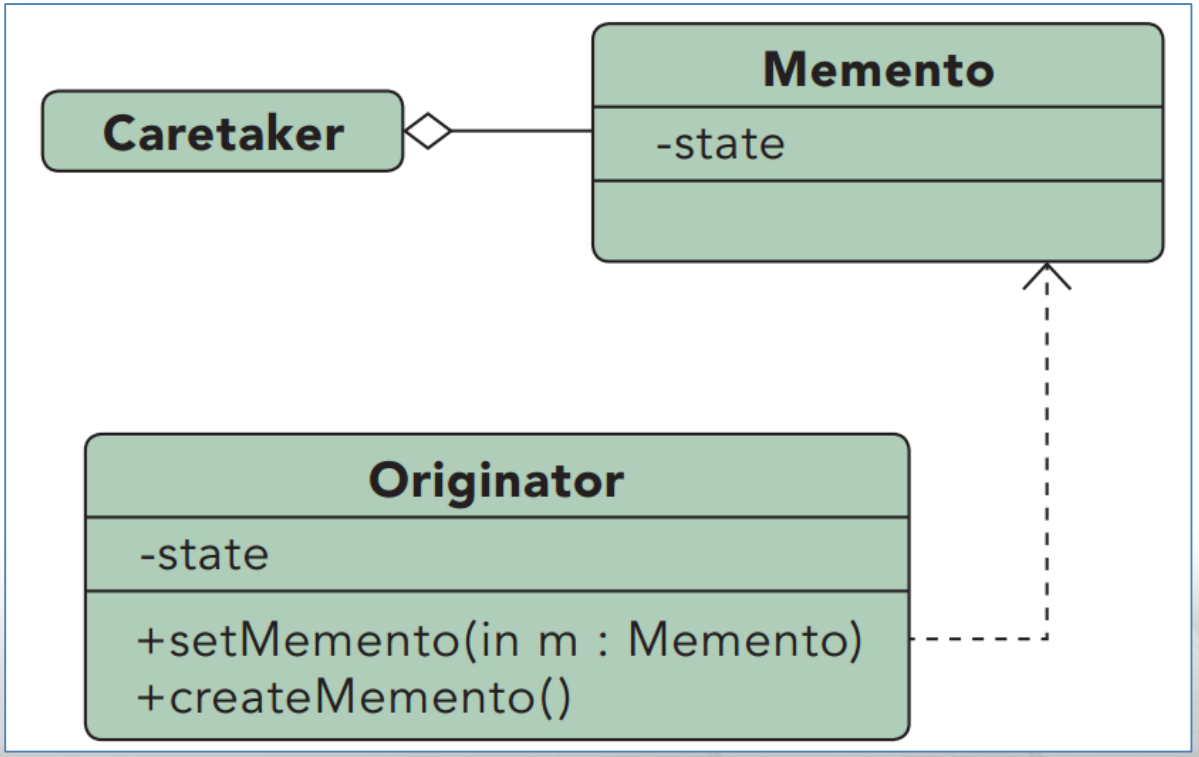
Напоминание / Memento



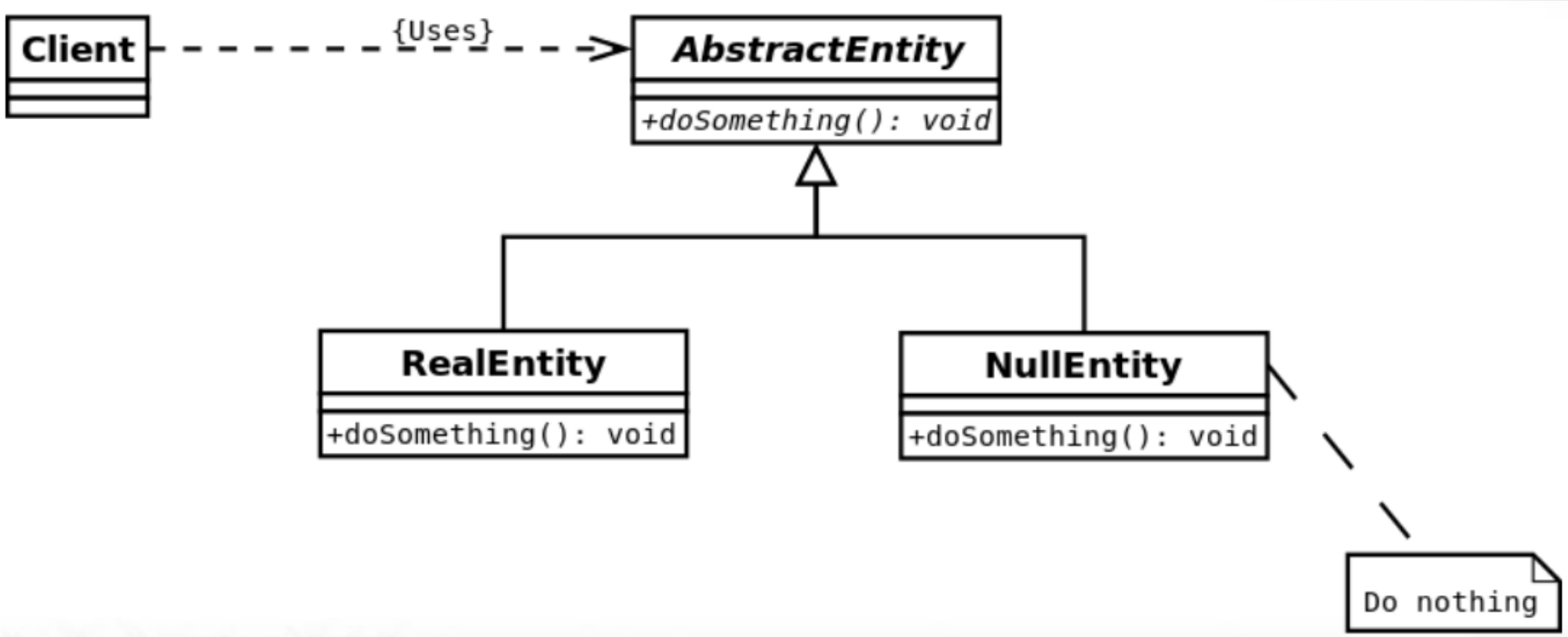
Напоминание / Memento

Используется, когда:

- необходимо сохранить снимок состояния объекта (или его части) для последующего восстановления
- прямой интерфейс получения состояния объекта раскрывает детали реализации и нарушает инкапсуляцию объекта



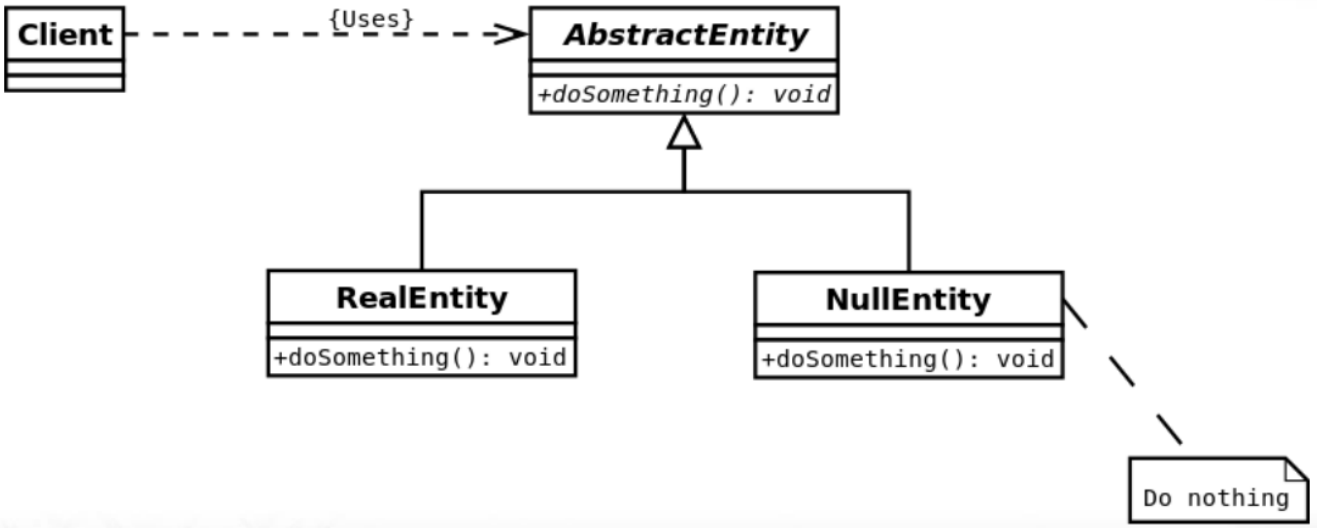
Null Object


Client
{Uses}
AbstractEntity
+doSomething(): void
RealEntity
+doSomething(): void
NullEntity
+doSomething(): void
Do nothing

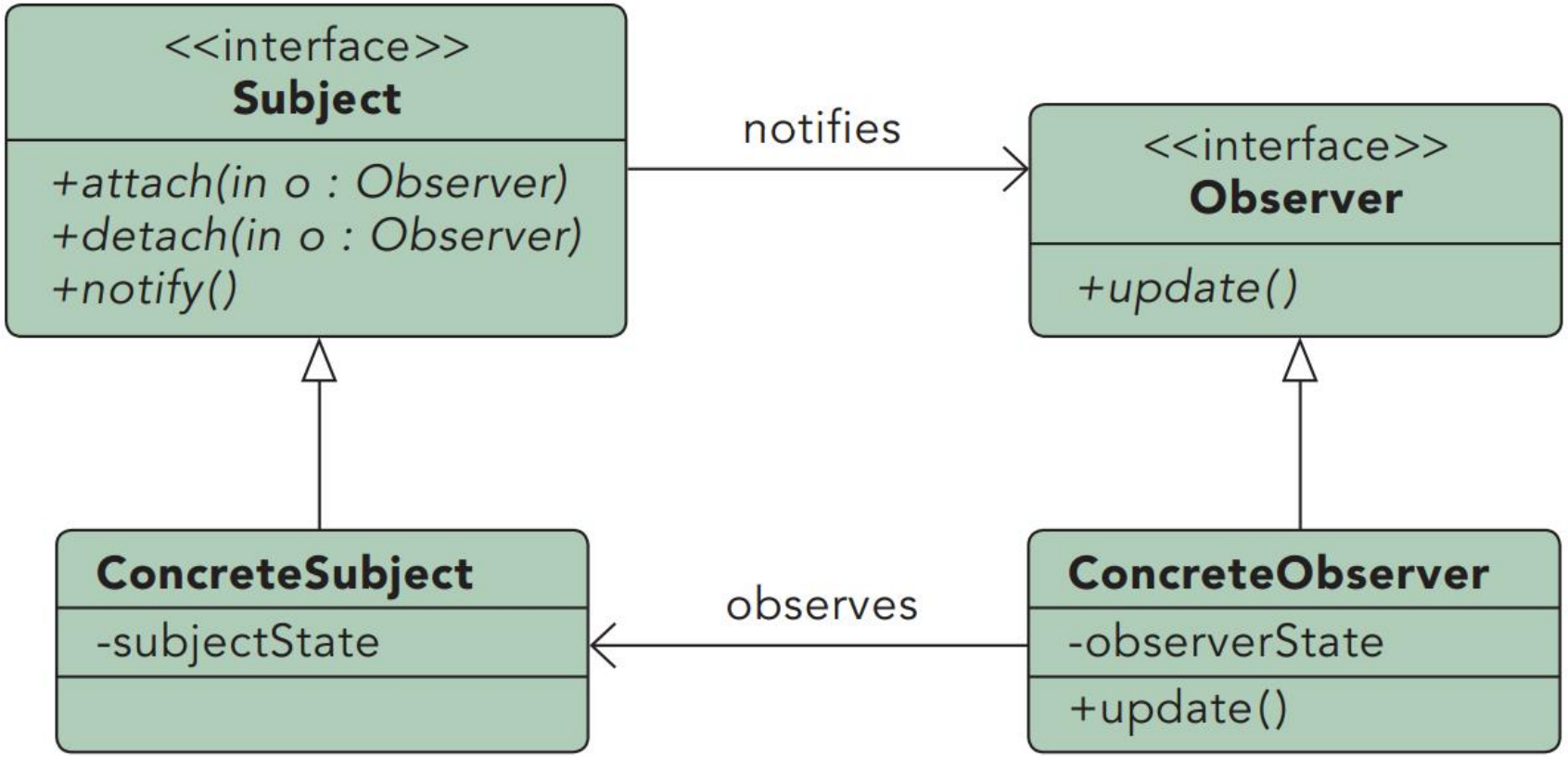
Null Object

Целью Null-object является инкапсулирование отсутствия объекта путем замещения его другим объектом, который ничего не делает.
Рекомендуется использовать, когда:

- Объект требует взаимодействия с другими объектами.
- Null Object не устанавливает нового взаимодействия — он использует уже установленное взаимодействие.
- Какие-то из взаимодействующих объектов должны бездействовать
- Требуется абстрагирование «общения» с объектами, имеющими NULL-значение.



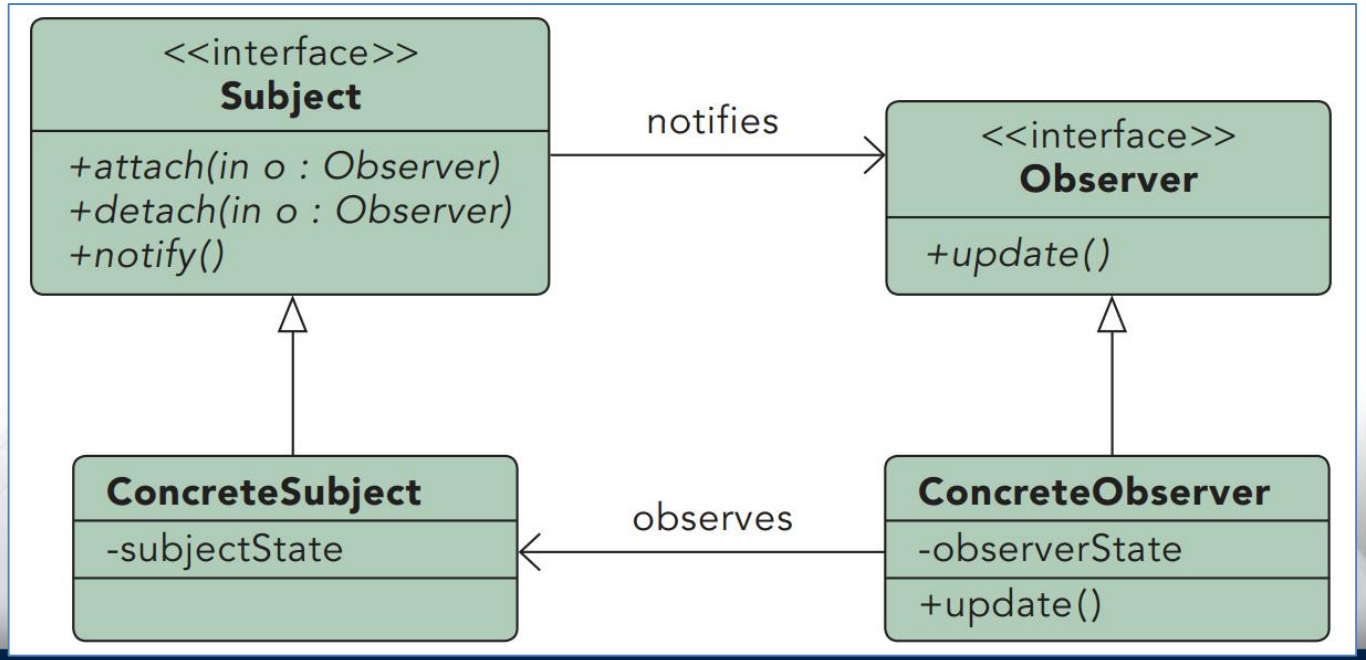
Наблюдатель / Observer



Наблюдатель / Observer

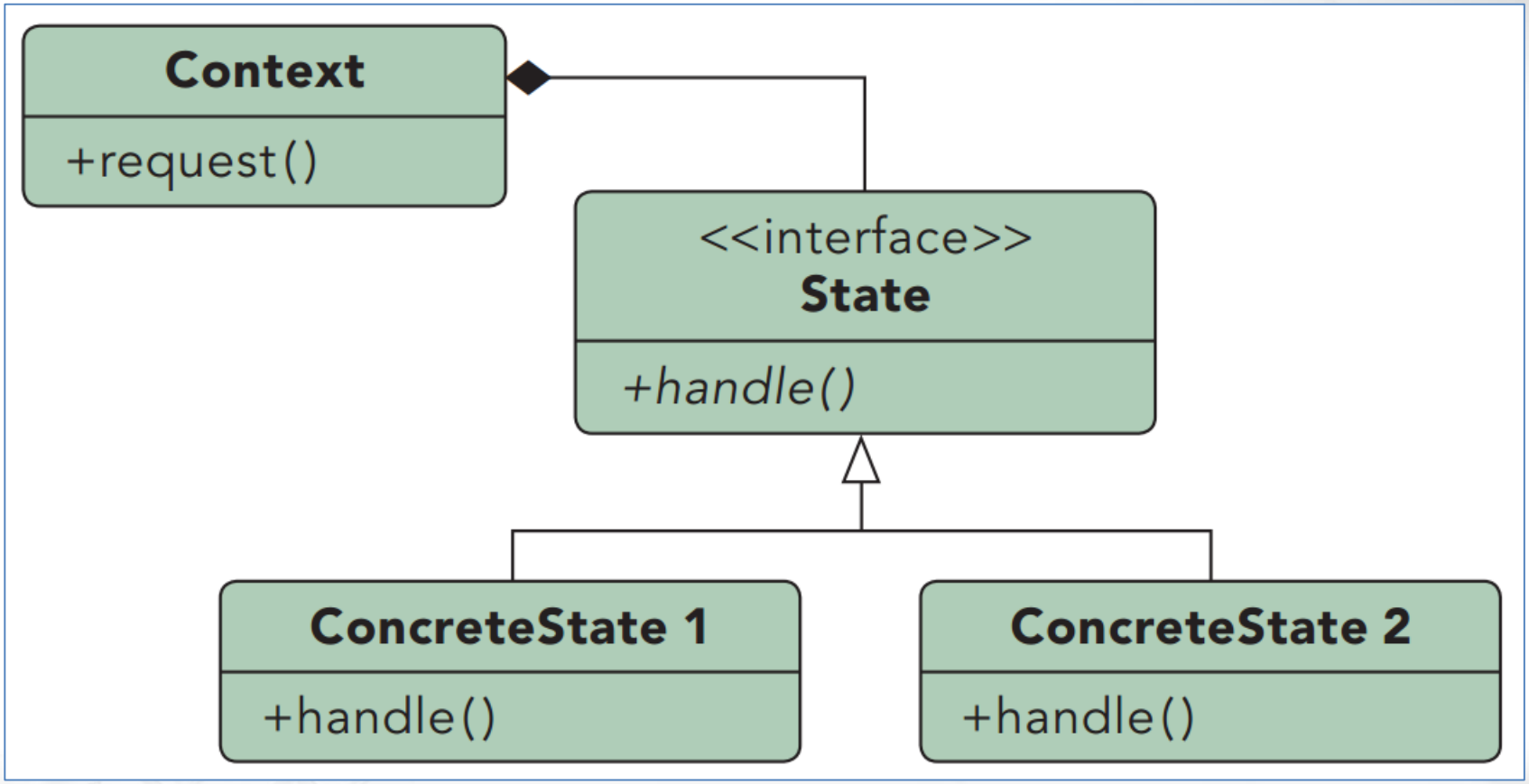
Применяется, когда система обладает следующими свойствами:

- существует, как минимум, один объект, рассылающий сообщения;
- имеется не менее одного получателя сообщений, причём их количество и состав могут изменяться во время работы приложения;
- нет надобности очень сильно связывать взаимодействующие объекты, что полезно для повторного использования.



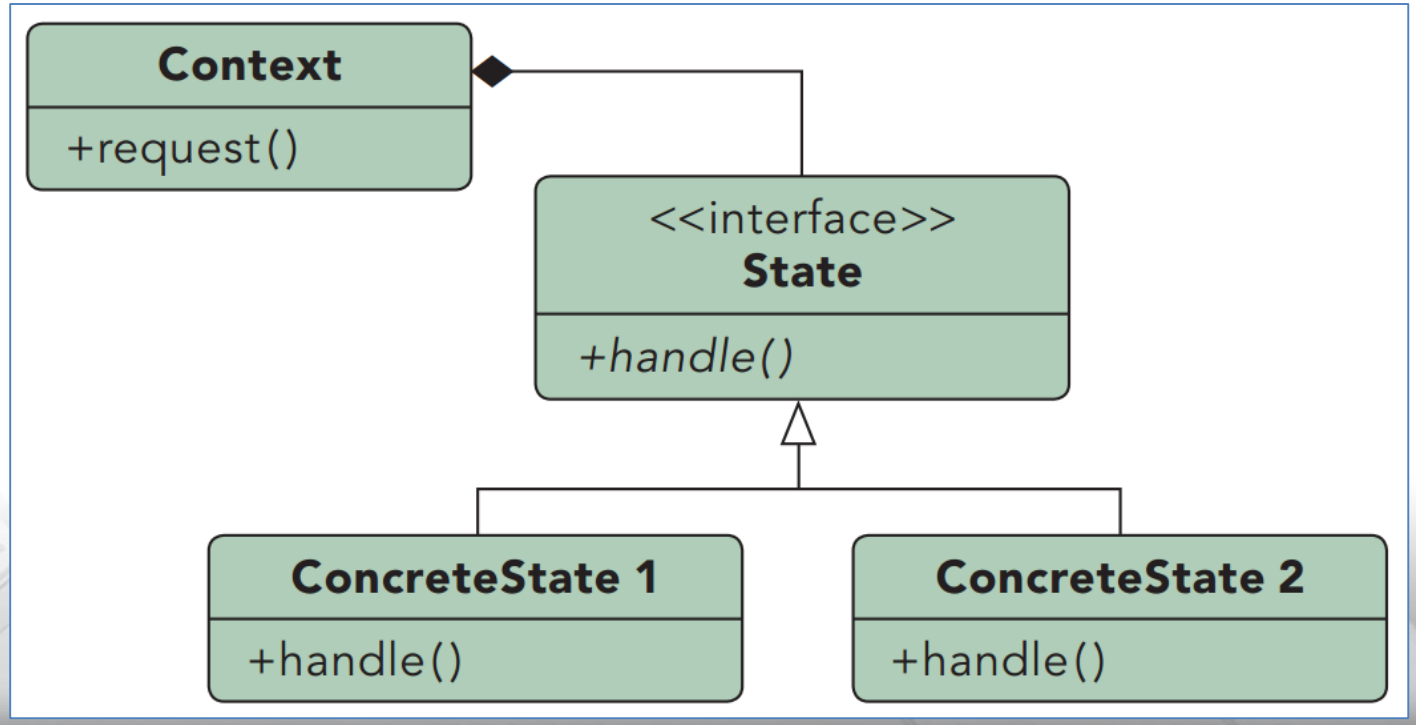
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Состояние / State

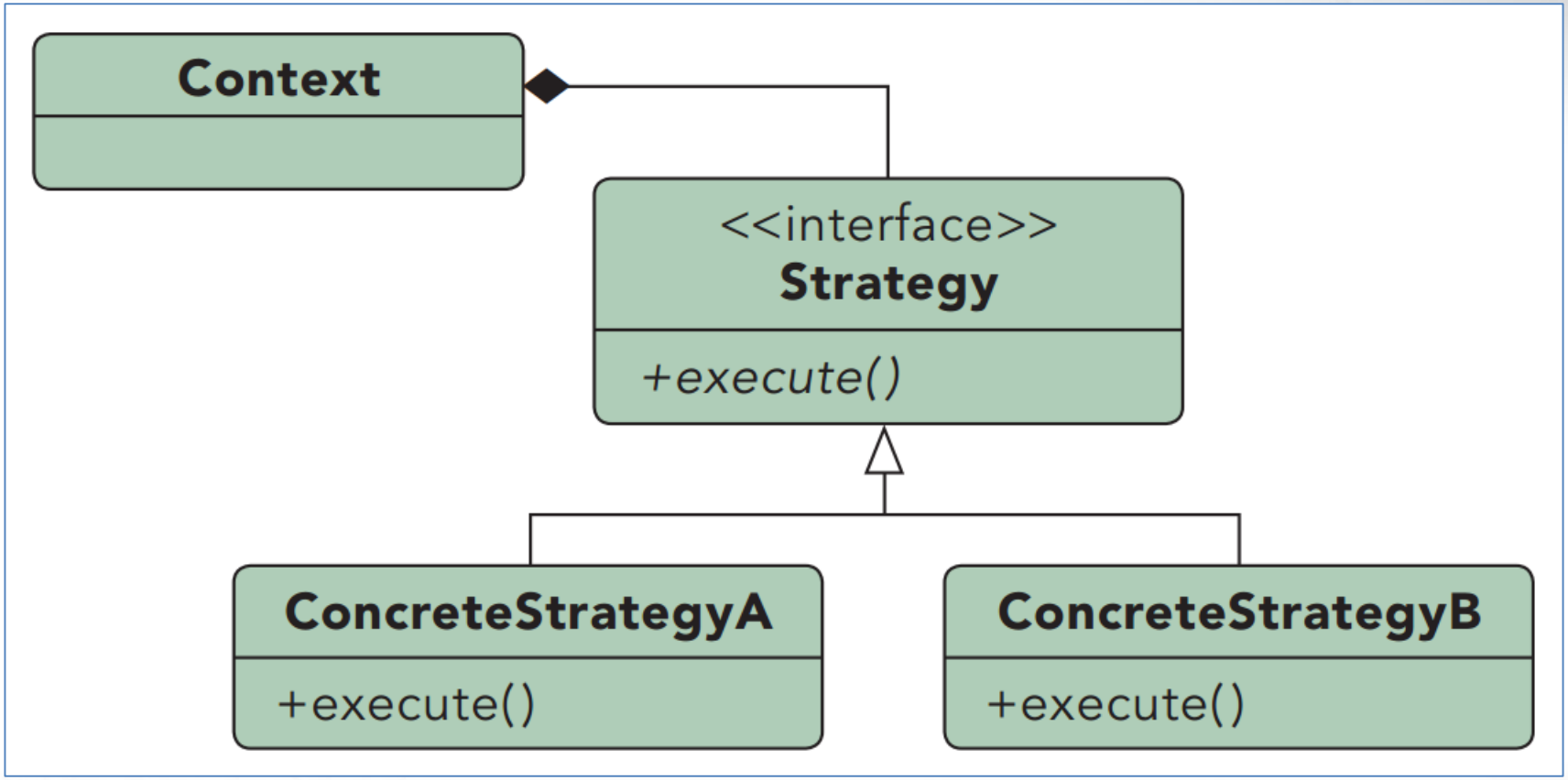


Состояние / State

Используется в тех случаях, когда во время выполнения программы объект должен менять свое поведение в зависимости от своего состояния.

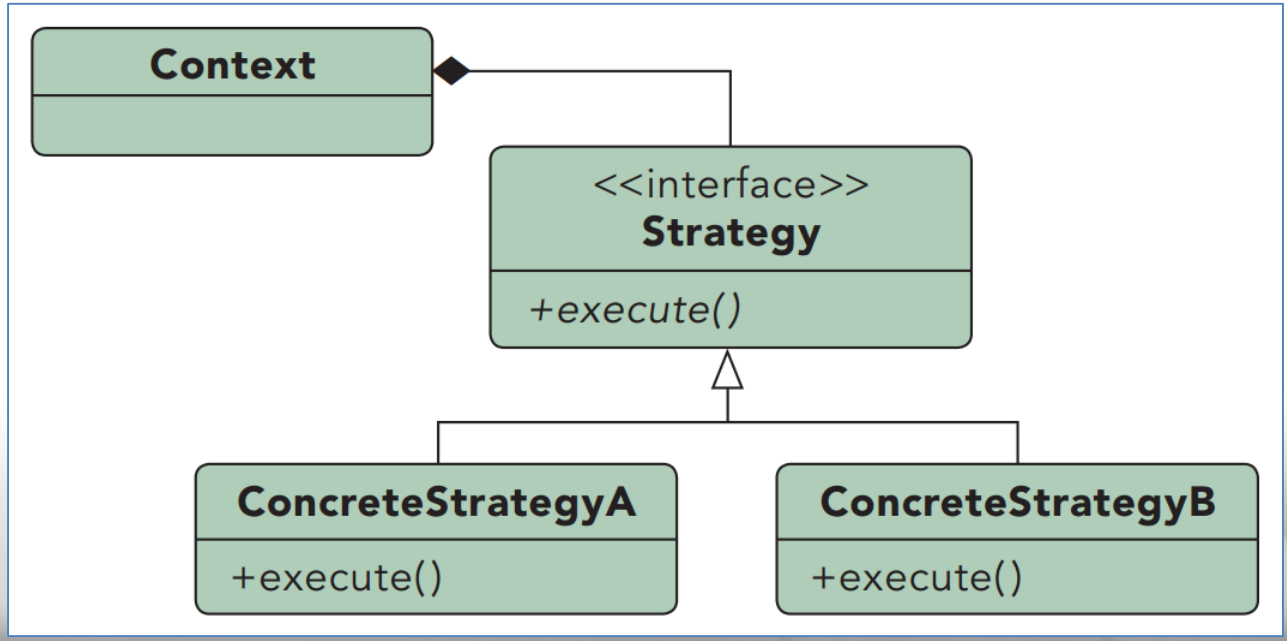


Стратегия / Strategy

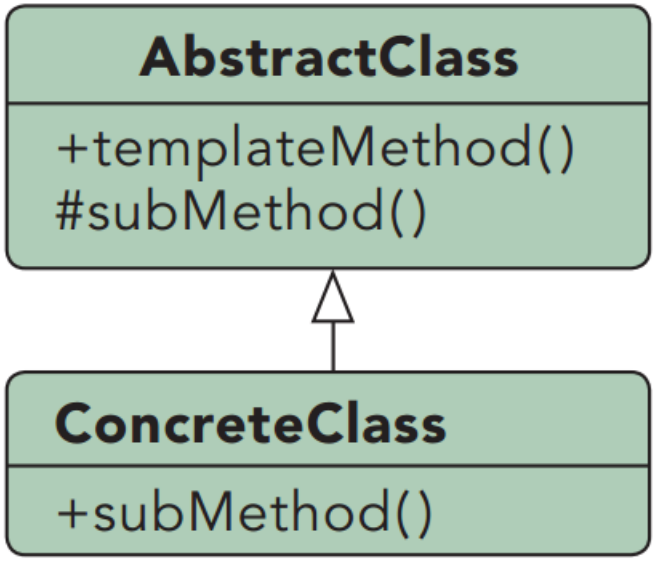


Стратегия / Strategy

- Программа должна обеспечивать различные варианты алгоритма или поведения
- Нужно изменять поведение каждого экземпляра класса
- Необходимо изменять поведение объектов на стадии выполнения
- Введение интерфейса позволяет классам-клиентам ничего не знать о классах, реализующих этот интерфейс и инкапсулирующих в себе конкретные алгоритмы

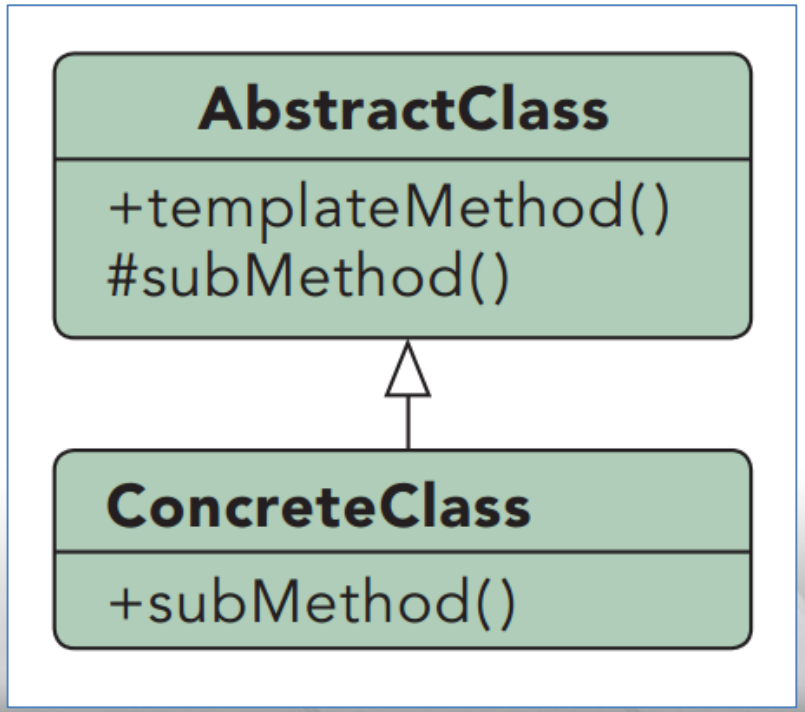


Шаблонный метод Template Method

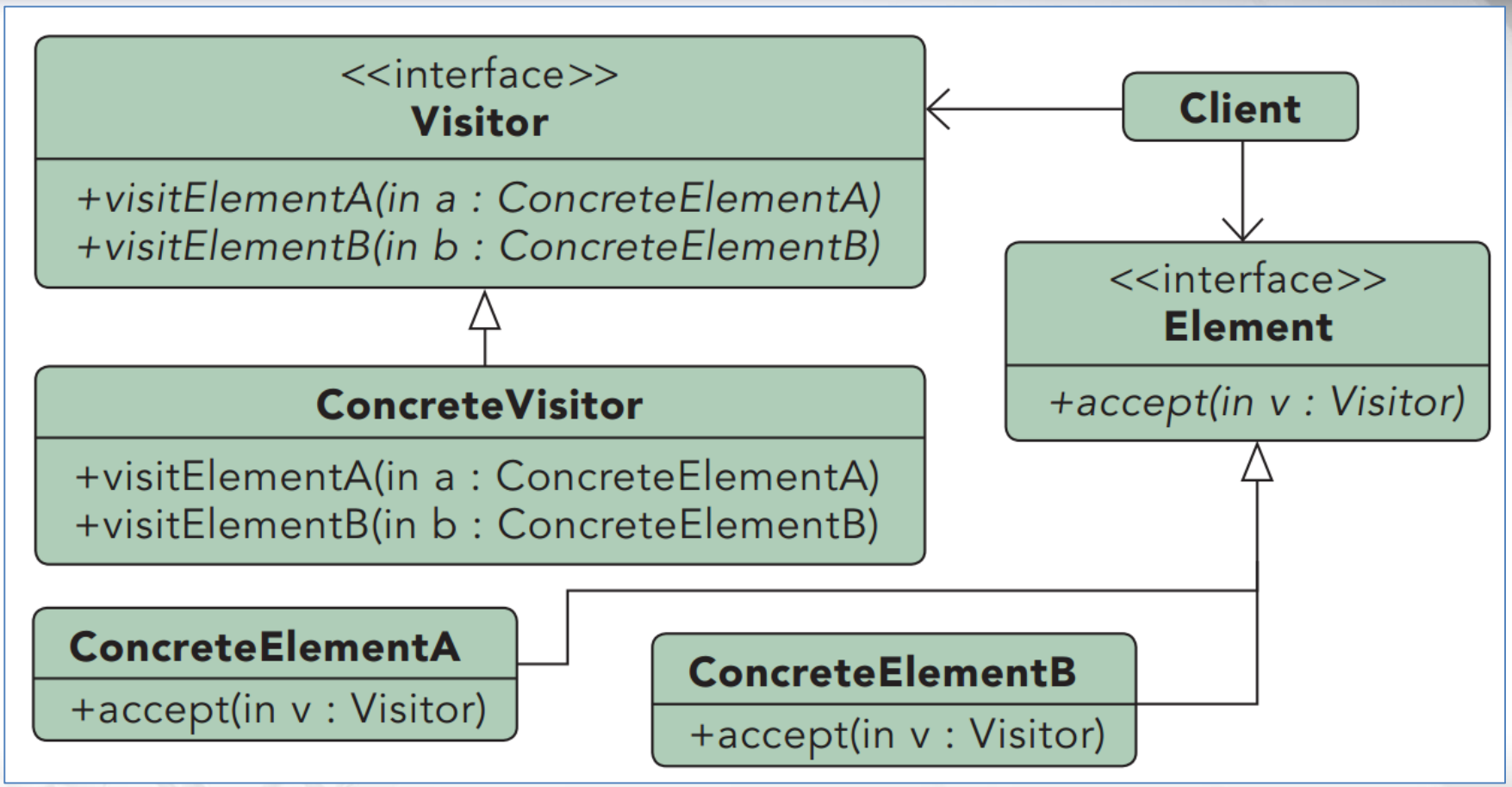


Шаблонный метод Template Method

- Однократное использование инвариантной части алгоритма, с оставлением изменяющейся части на усмотрение наследникам.
- Локализация и вычленение общего для нескольких классов кода для избегания дублирования.
- Разрешение расширения кода наследниками только в определенных местах.



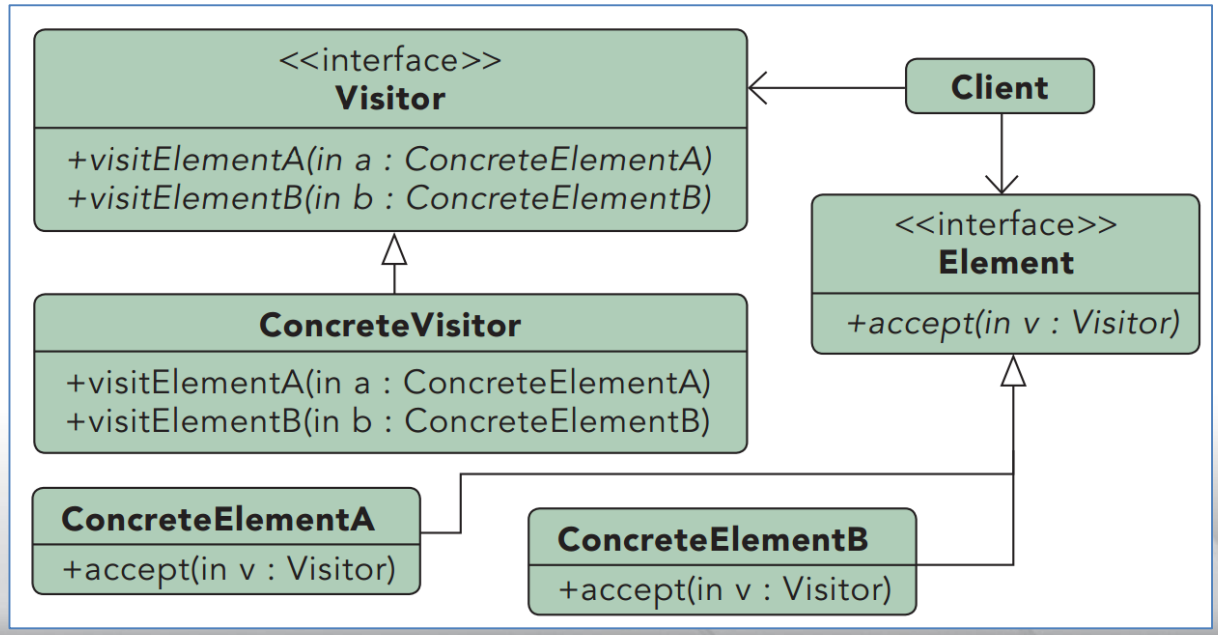
Посетитель / Visitor



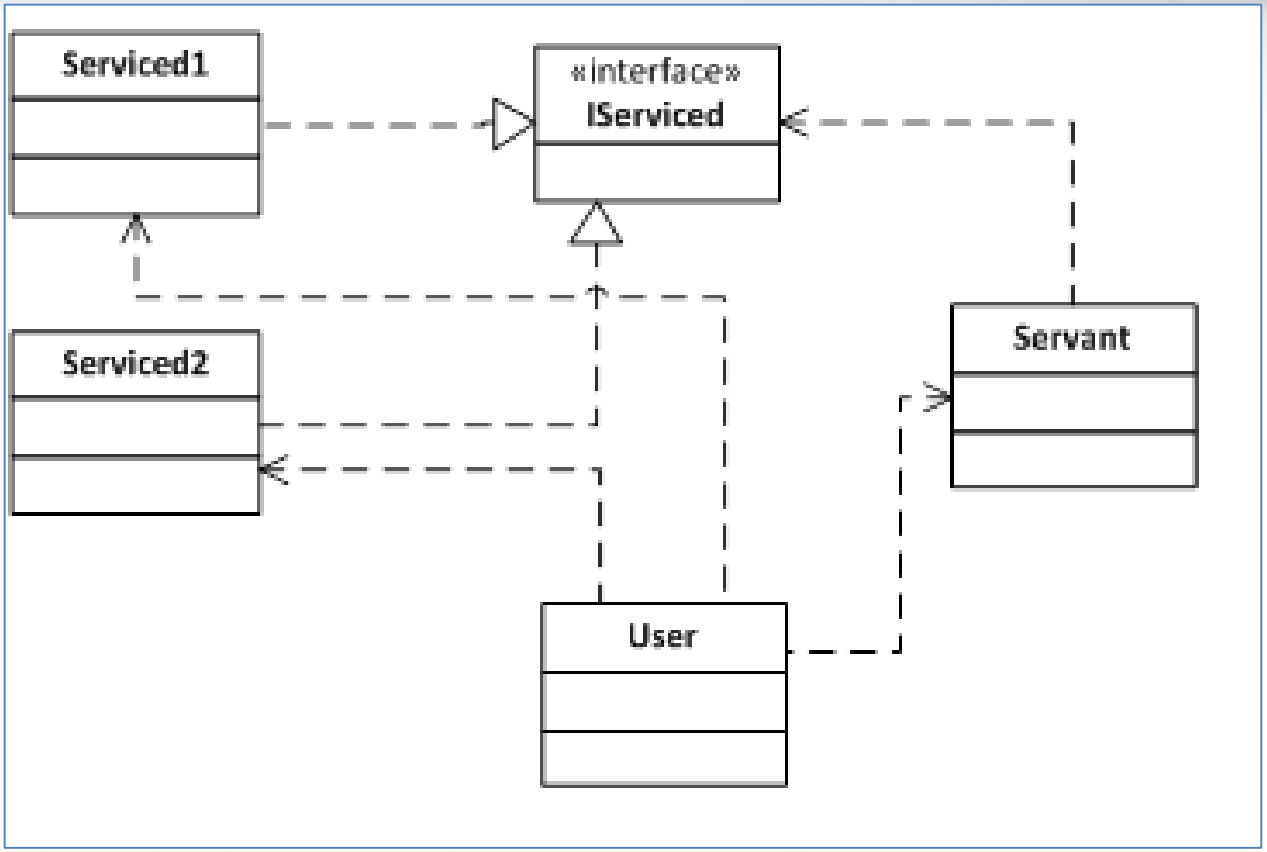
Посетитель / Visitor

Следует использовать, если:

- Имеются различные объекты разных классов с разными интерфейсами, но над ними нужно совершать операции, зависящие от конкретных классов
- Необходимо над структурой выполнить различные, усложняющие структуру операции
- Часто добавляются новые операции над структурой

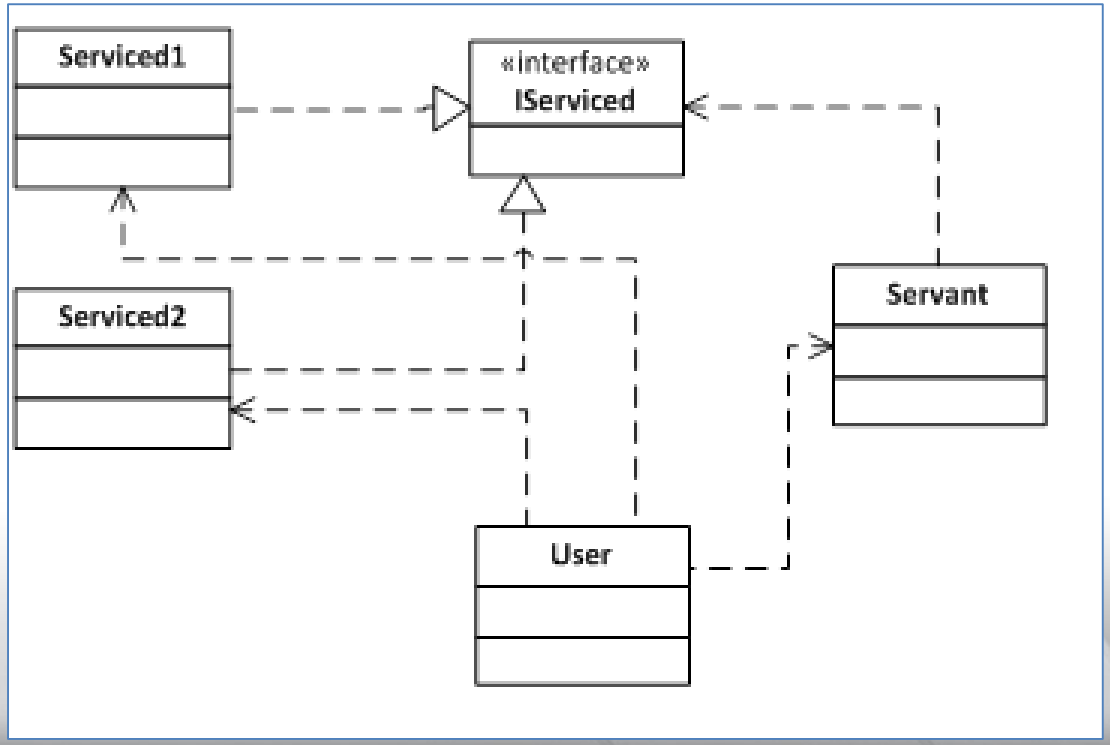


Слуга / Servant



Слуга / Servant

- Слуга используется для предоставления некоторого поведения группе классов.
- Вместо определения поведение в каждом классе, или в случае, когда мы не можем вынести это поведение в общий родительский класс - оно определяется один раз в классе Слуги.



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Вопросы?

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Паттерны (шаблоны) проектирования

Поведенческие паттерны

Eugeny Berkunsky, Computer Science dept.,
National University of Shipbuilding
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua