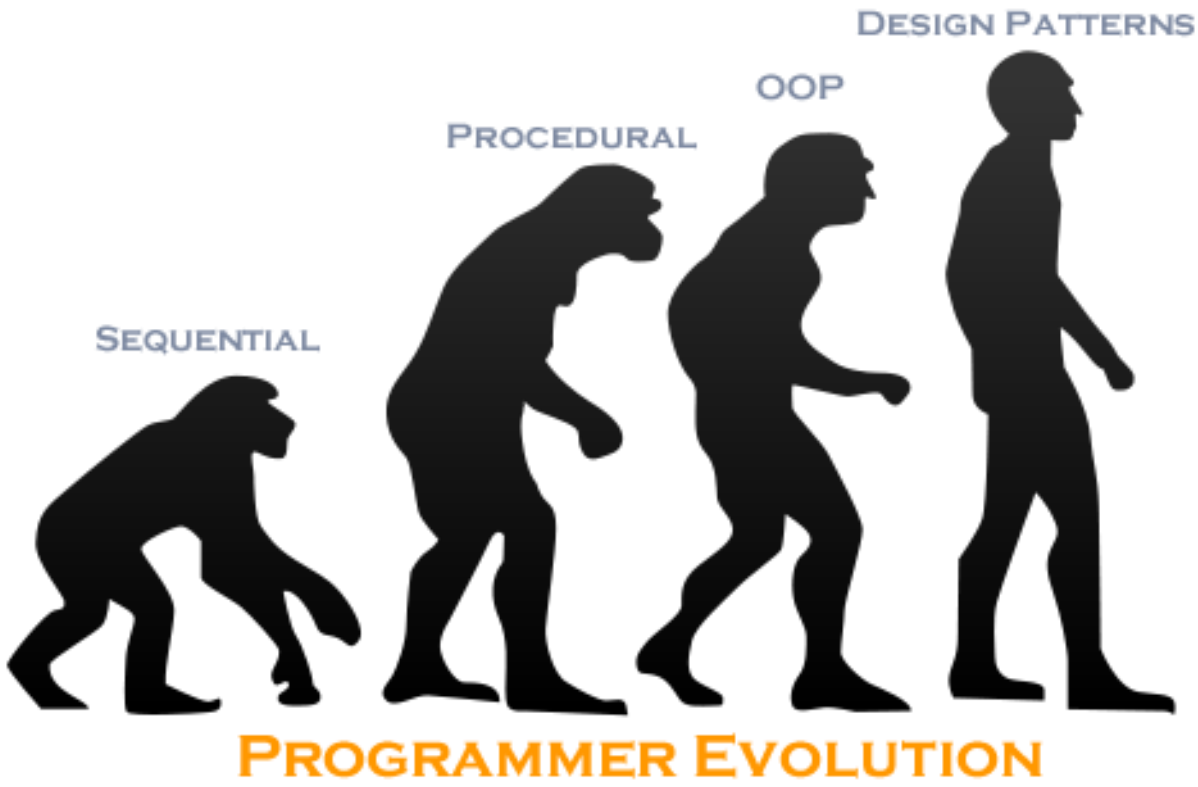


http://hrushikeshzadgaonkar.wordpress.com/

Курс: «Математические модели комплексов программ»

Модуль: **«Архитектура программных систем»**

Лекция 5 (Князьков К.В.)

Design Patterns

Шаблон проектирования или **паттерн** — повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Шаблоны могут применяться на разных уровнях:

- *На наивысшем уровне существуют **архитектурные шаблоны**, они охватывают собой архитектуру всей программной системы.*
- ***Идиомы** – «низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования. Это хорошие решения проектирования, характерные для конкретного языка или программной платформы, и потому не универсальные.*

«Банда четырех» (Gang of Four)

Книга: «Приёмы объектно-ориентированного проектирования. Паттерны проектирования»

Авторы: Эрих Гамма (Erich Gamma), Ричард Хелм (Richard Helm), Ральф Джонсон (Ralph Johnson), Джон Влиссидс (John Vlissides)

Год издания: 1994

Классификация паттернов (GoF)

Порождающие паттерны организуют создание объектов

Структурные паттерны организуют структуру классов

Паттерны поведения организуют поведение объектов

Порождающие паттерны

Abstract Factory

Builder

Factory Method

Prototype

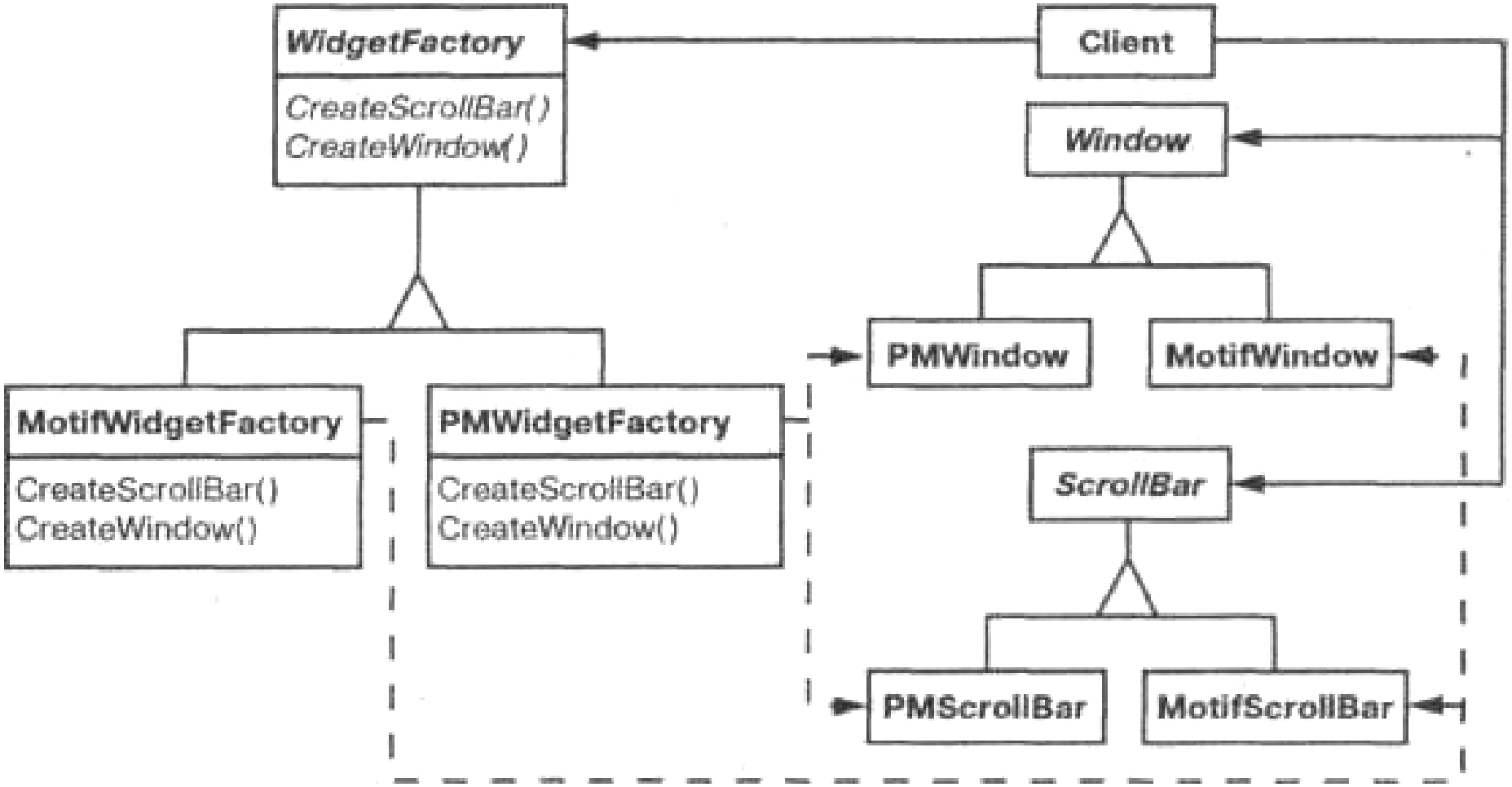
Singleton

Abstract Factory – Абстрактная фабрика

- **Назначение:** организовать интерфейс для семейства взаимосвязанных или взаимозависимых объектов

- **Применимость:**
  - Система на зависит от того, как создаются и компонуются входящие в нее объекты;
  - Объекты взаимосвязаны и должны использоваться вместе;
  - Система конфигурируется конкретным семейством объектов;
  - Библиотека объектов предоставляет только интерфейсы, но не реализацию

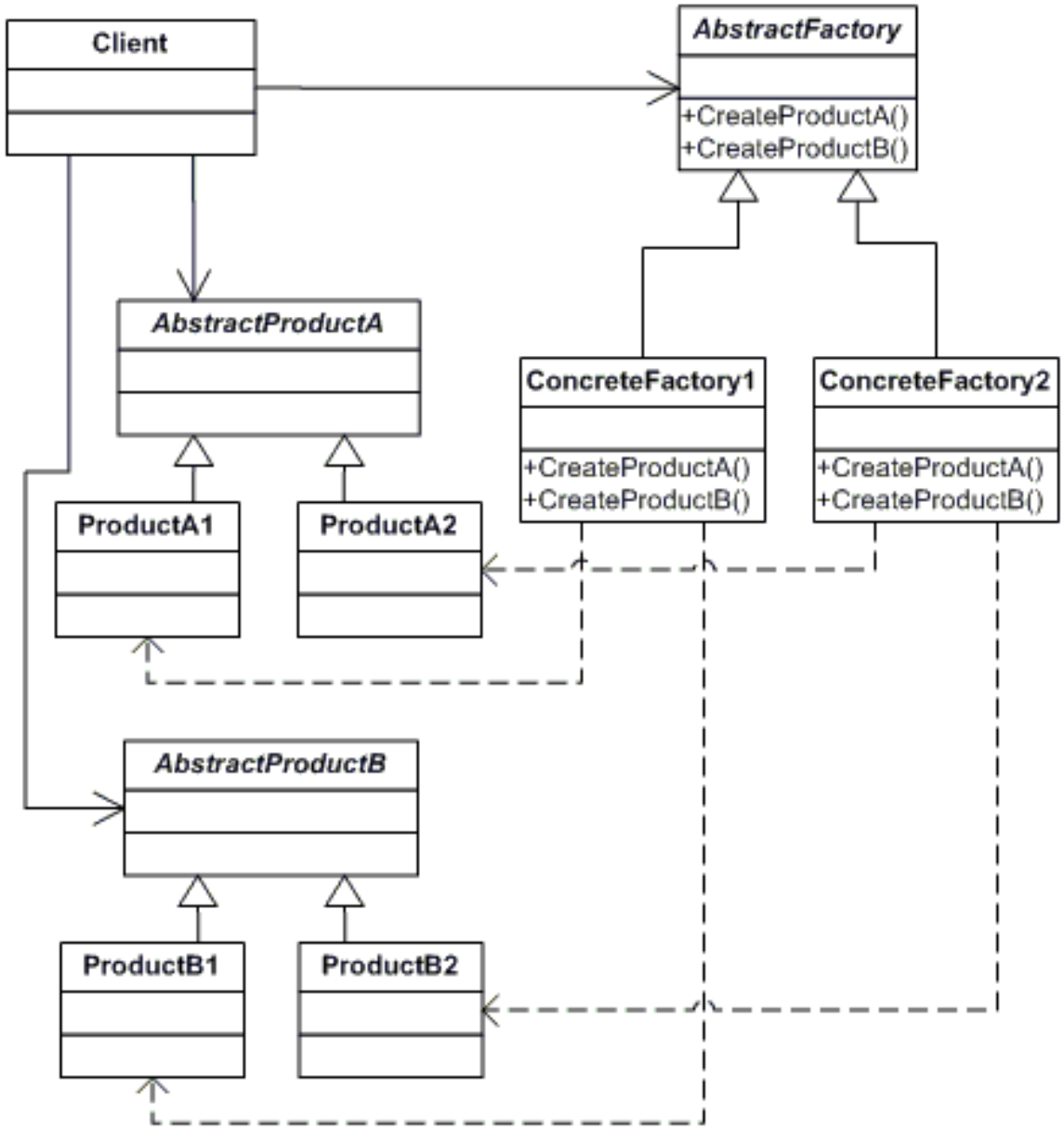
Мотивация


WidgetFactory
CreateScrollBar()
CreateWindow()
Client
Window
PMWindow
MotifWindow
MotifWidgetFactory
CreateScrollBar()
CreateWindow()
PMWidgetFactory
CreateScrollBar()
CreateWindow()
ScrollBar
PMScrollBar
MotifScrollBar

Примеры

- Портируемые библиотеки (cross platform UI)
- Обработка графов (WeightedGraphFactory,)
- Работа javax.xml.parsers.DocumentBuilderFactory
- Независимость от протокола System.Net.WebRequest

Структура


Client
AbstractFactory
+CreateProductA()
+CreateProductB()
AbstractProductA
ConcreteFactory1
+CreateProductA()
+CreateProductB()
ConcreteFactory2
+CreateProductA()
+CreateProductB()
ProductA1
ProductA2
AbstractProductB
ProductB1
ProductB2

Обсуждение

Преимущества/недостатки

+ изолирует конкретные классы
+ упрощает замену семейств продуктов
+ гарантирует сочетаемость продуктов
- сложно поддерживать новые виды продуктов

Просто «Фабрика»

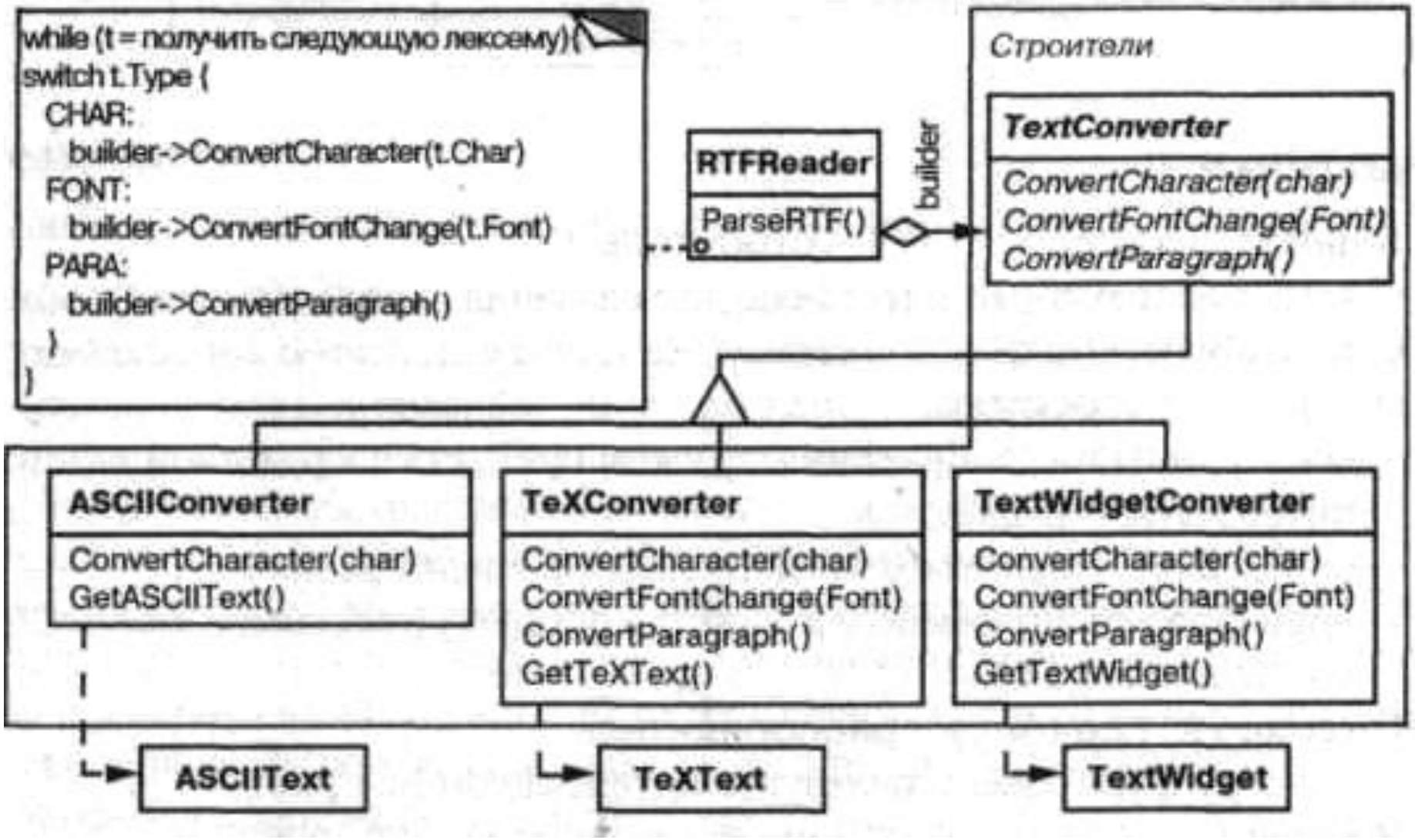
Скрытие процесса конструирования

Builder – Строитель

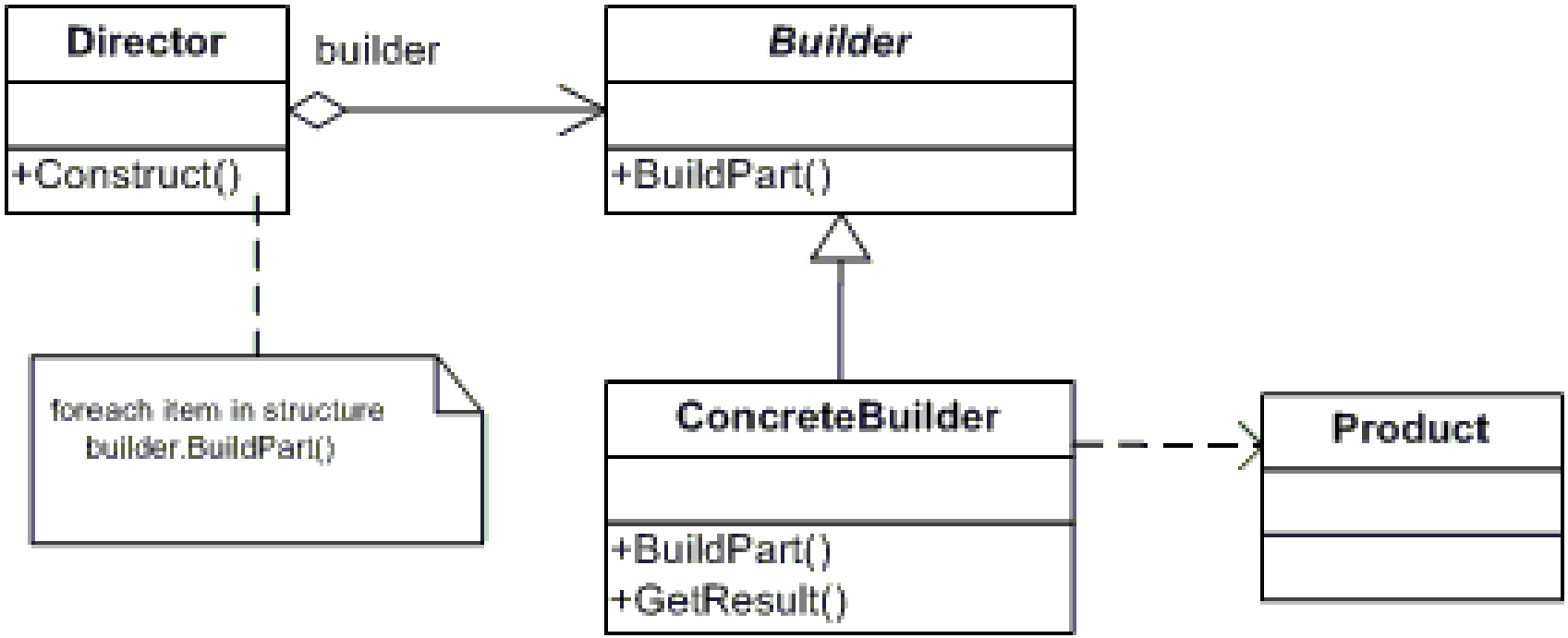
- **Назначение:** отделить конструирование сложного объекта от его представления

- **Применимость:**
  - Процесс конструирования должен обеспечивать различные представления объекта
  - Алгоритм создания сложного объекта не должен зависеть от того, из каких частей состоит объект и как они стыкуются

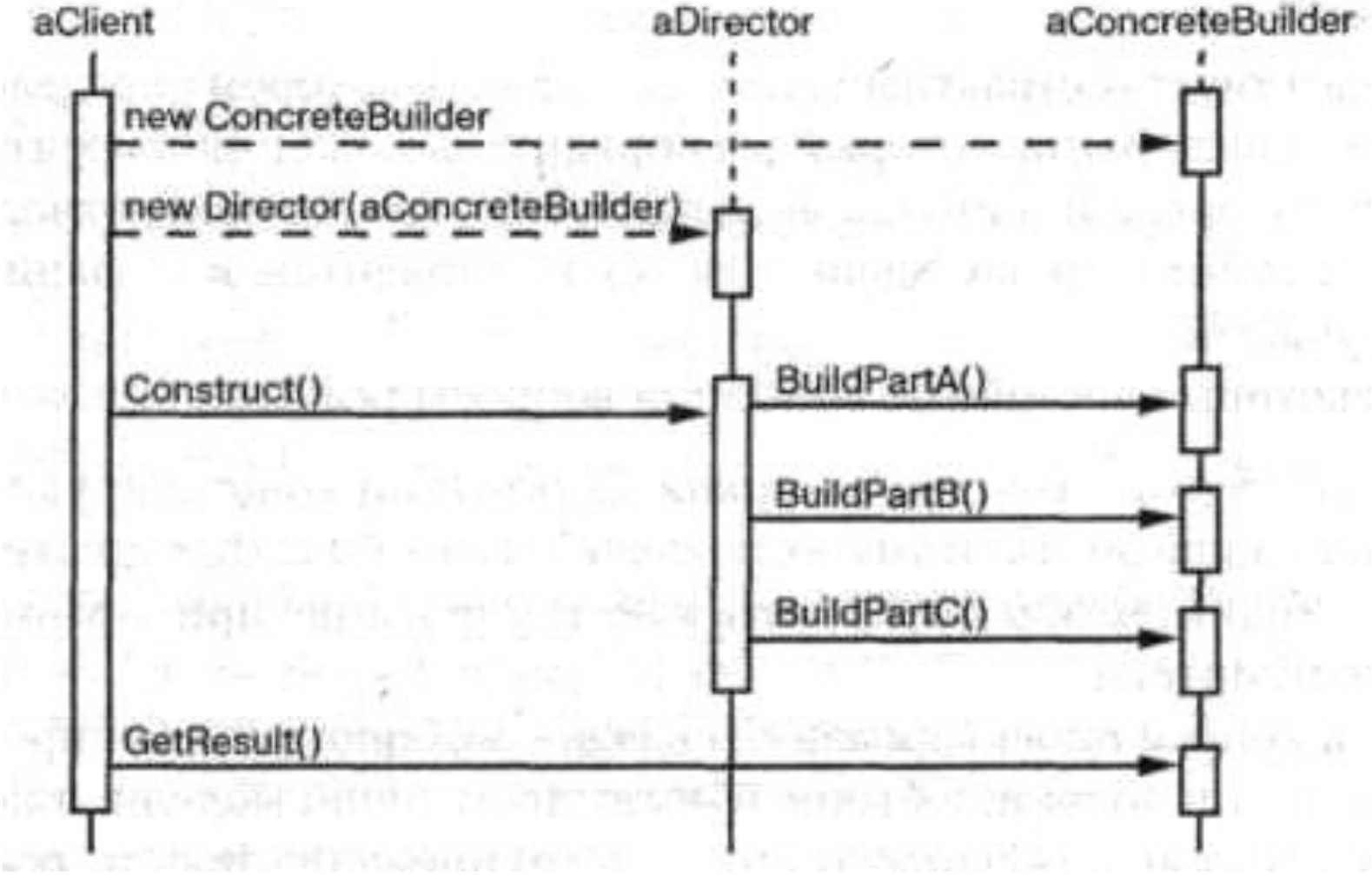
Мотивация


while (t = получить следующую лексему) {
switch t.Type {
CHAR:
builder->ConvertCharacter(t.Char)
FONT:
builder->ConvertFontChange(t.Font)
PARA:
builder->ConvertParagraph()
}
}
RTFReader
ParseRTF()
builder
Строители
TextConverter
ConvertCharacter(char)
ConvertFontChange(Font)
ConvertParagraph()
ASCIIConverter
ConvertCharacter(char)
GetASCIIText()
TeXConverter
ConvertCharacter(char)
ConvertFontChange(Font)
ConvertParagraph()
GetTeXText()
TextWidgetConverter
ConvertCharacter(char)
ConvertFontChange(Font)
ConvertParagraph()
GetTextWidget()
ASCIIText
TeXText
TextWidget

Структура


Director
builder
Builder
+Construct()
+BuildPart()
foreach item in structure
builder.BuildPart()
ConcreteBuilder
+BuildPart()
+GetResult()
Product

Отношения


aClient
aDirector
aConcreteBuilder
new ConcreteBuilder
new Director(aConcreteBuilder)
Construct()
BuildPartA()
BuildPartB()
BuildPartC()
GetResult()

Примеры

- Разбор и представление HTML

Обсуждение

Преимущества/недостатки

+ изолирует код конструирования и представления
+ более тонкий контроль над процессом конструирования за счет поэтапной сборки

Отличие от фабрики

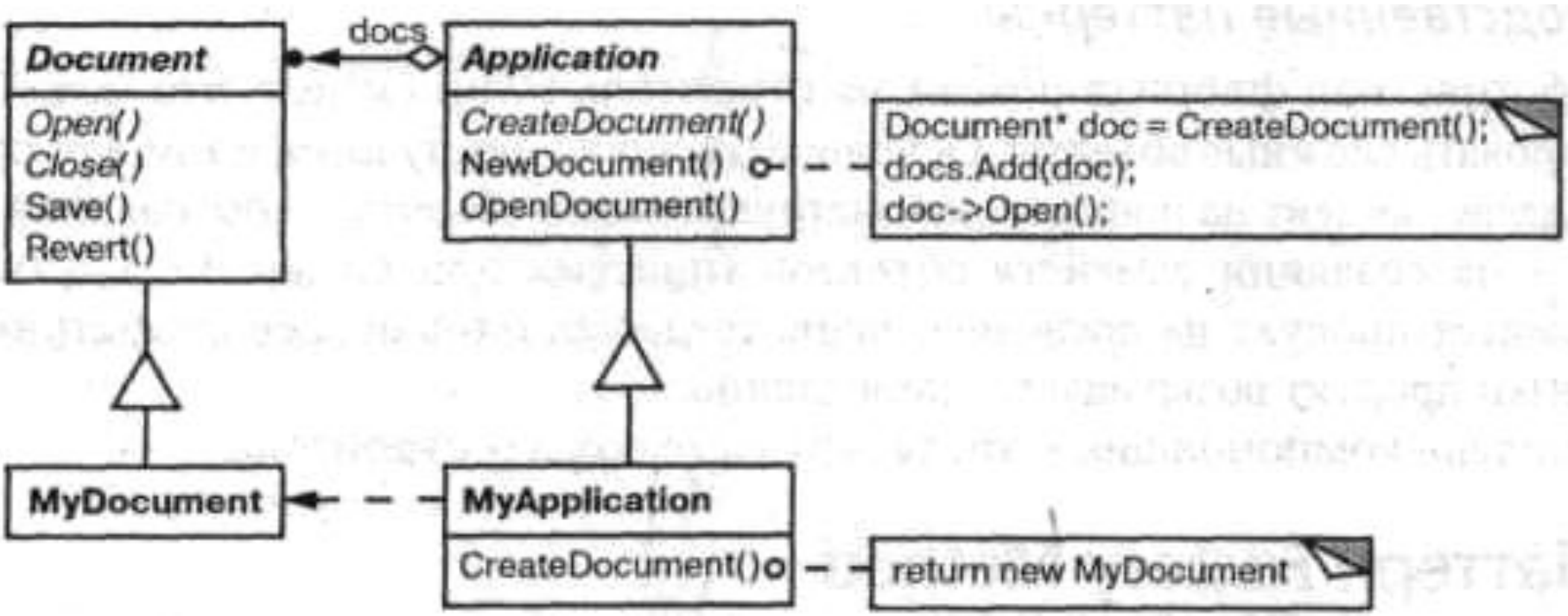
Фабрика собирает сразу весь объект, а строитель

Factory method – Фабричный метод

- **Назначение:** делегирование подклассам решения о том, какой класс инстанцировать

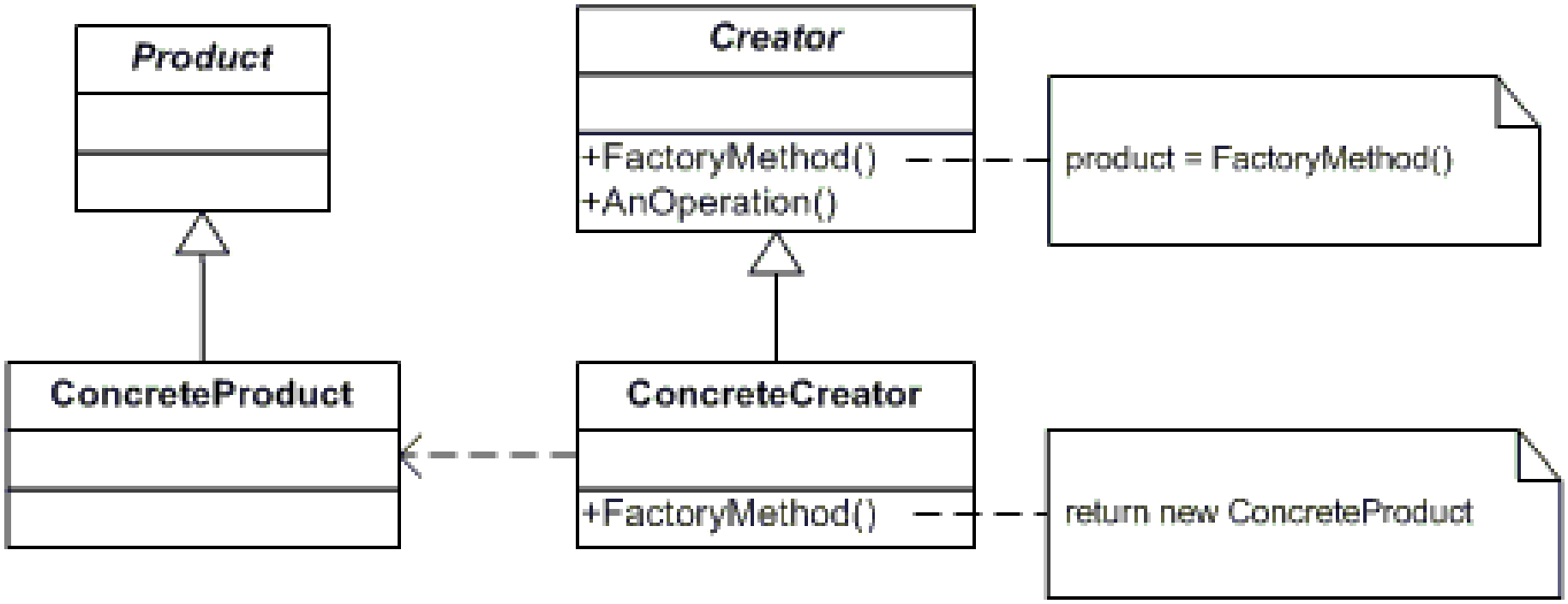
- **Применимость:**
  - Классу заранее неизвестно объекты каких классов нужно создавать
  - Делегирование обязанностей подклассу

Мотивация



- Пример использования: DrawingApplication и DrawingDocument Vs TextViewApplication и TextDocument

Структура



- Полиморфный метод для создания объектов

Обсуждение

Преимущества/недостатки

+ позволяет выносить предметно-зависимый код вне кода фреймворка, где вся логика построена на работе с интерфейсами
- Необходимость создавать подкласс класса Creator

Отличие от фабрики

Абстрактная фабрика часто реализуется с помощью фабричных методов.

Примеры

Программные каркасы и инструментальные библиотеки

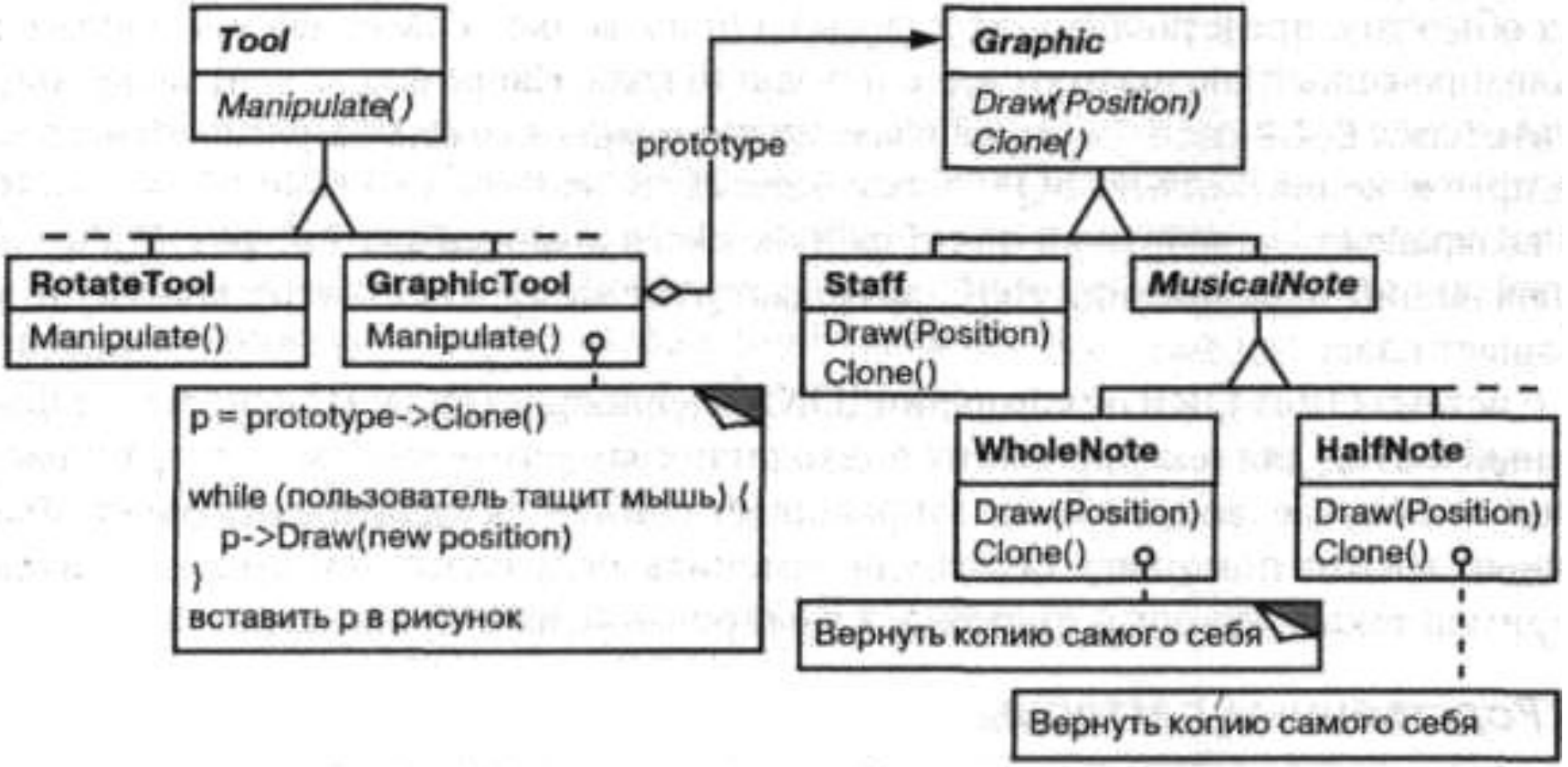
javax.xml.parsers.DocumentBuilderFactory

Prototype – Прототип

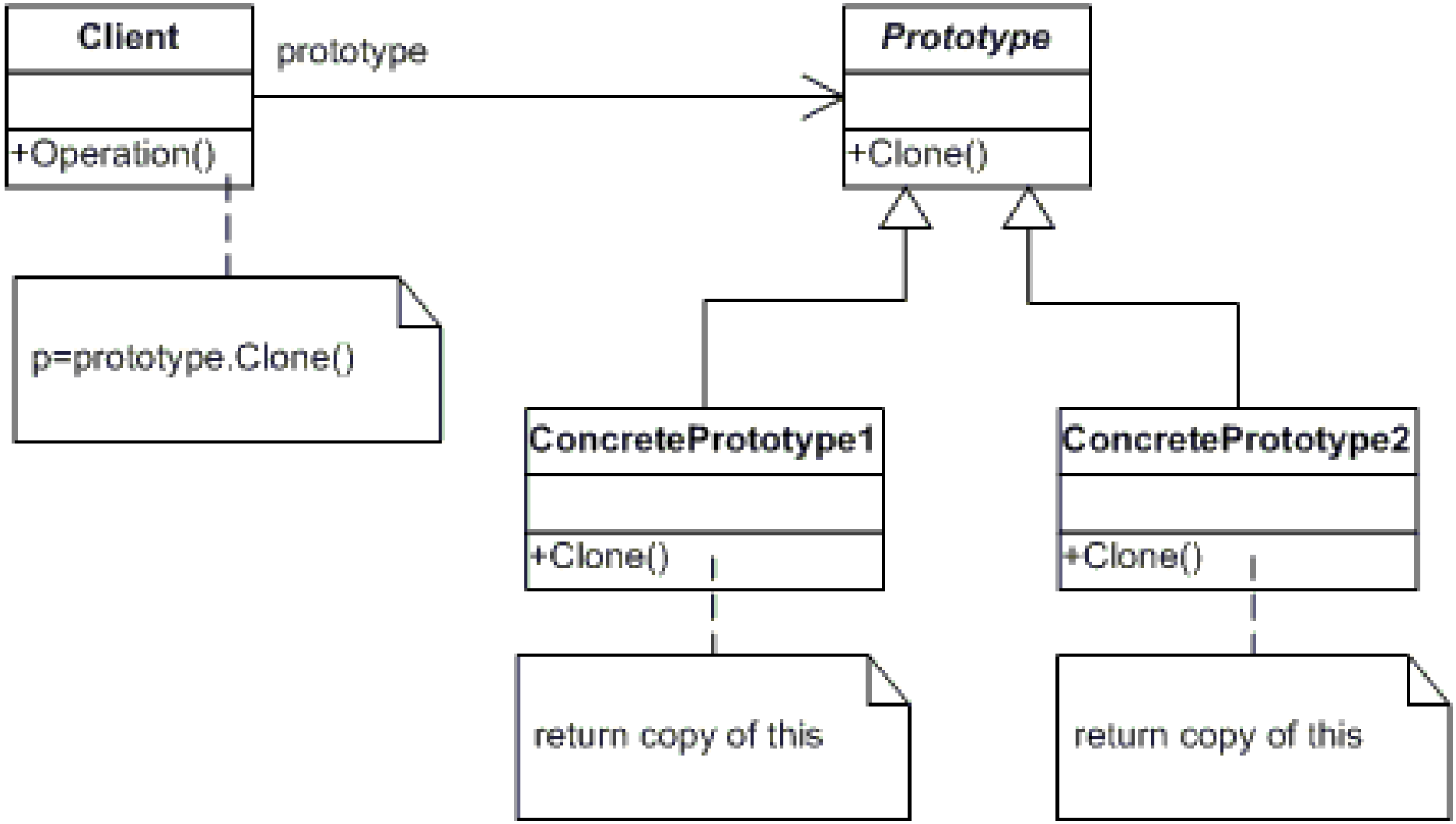
- **Назначение:** создание объектов за счет копирования экземпляра-прототипа

- **Применимость:**
  - Инстанциируемые классы определяются во время выполнения
  - Для избежания создания иерархий классов или фабрик
  - Объекты могут находиться в одном из состояний (из небольшого набора N). Может быть удобно держать N объектов - прототипов

Мотивация


Tool
Manipulate()
prototype
Graphic
Draw(Position)
Clone()
RotateTool
Manipulate()
GraphicTool
Manipulate()
Staff
Draw(Position)
Clone()
MusicalNote
p = prototype->Clone()
while (пользователь тащит мышь) {
p->Draw(new position)
}
вставить p в рисунок
WholeNote
Draw(Position)
Clone()
HalfNote
Draw(Position)
Clone()
Вернуть копию самого себя
Вернуть копию самого себя

Структура


Client
prototype
+Operation()
Prototype
+Clone()
p=prototype.Clone()
ConcretePrototype1
+Clone()
return copy of this
ConcretePrototype2
+Clone()
return copy of this

Обсуждение

Преимущества/недостатки

+ добавление классов на лету за счет добавления прототипа

  палитра инструментов в редакторе

  плагины

- Необходимость реализации Clone

Отличие от фабрики

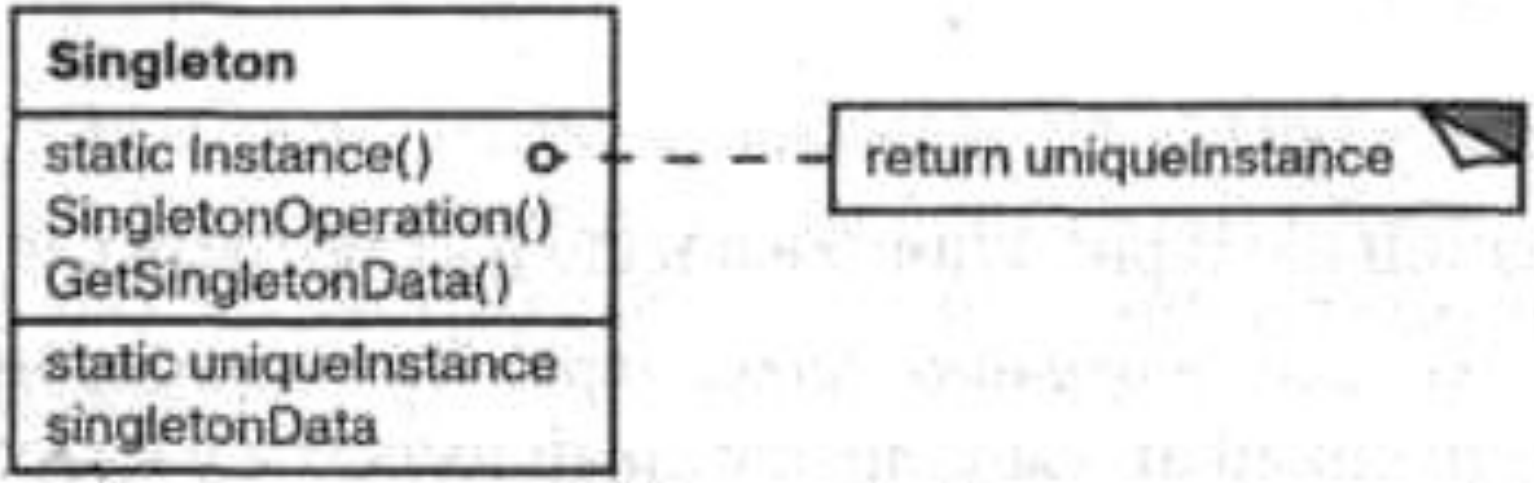
Абстрактная фабрика часто реализуется с помощью фабричных методов.

Singleton – Одиночка

- **Назначение:** обеспечить наличие только одного объекта для определенного класса

- **Применимость:**
  - Ровно один экземпляр

Структура


Singleton
static Instance()
SingletonOperation()
GetSingletonData()
static uniqueInstance
singletonData
return uniqueInstance

Обсуждение

Преимущества/недостатки

+ в отличие от глобальных переменных избавляет от засорения пространства имен
+ контролируемый доступ к экземпляру

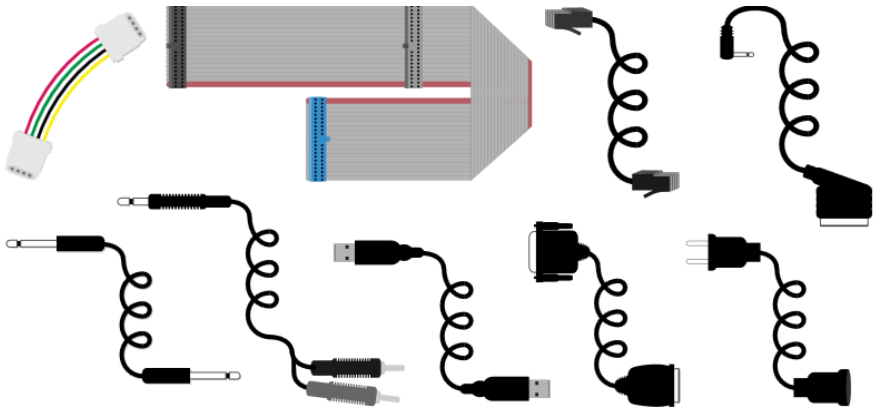
Структурные паттерны

Adapter
Bridge
Composite
Decorator
Facade
Flyweight
Proxy

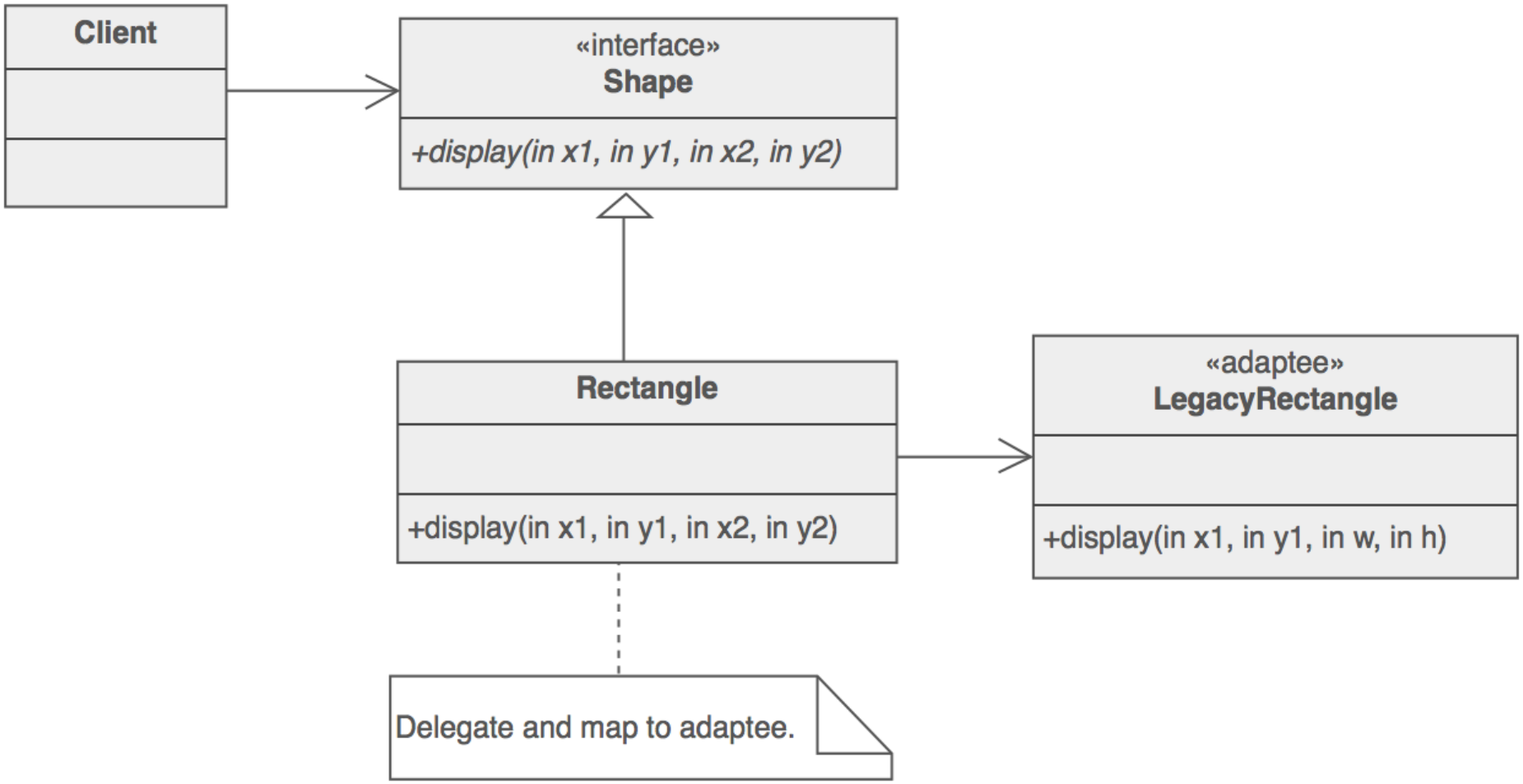
Adapter – Адаптер

- **Назначение:** преобразовать интерфейс одного класса в интерфейс другого

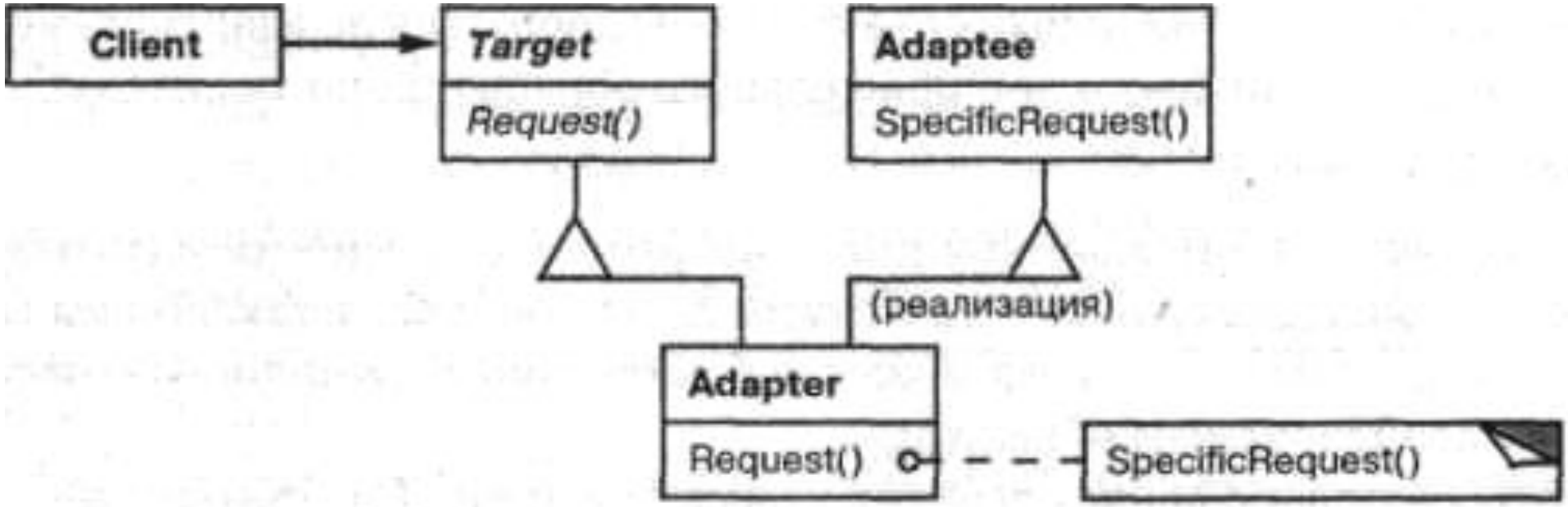
- **Применимость:**
  - Интерфейс класса не соответствует требованиям клиентов



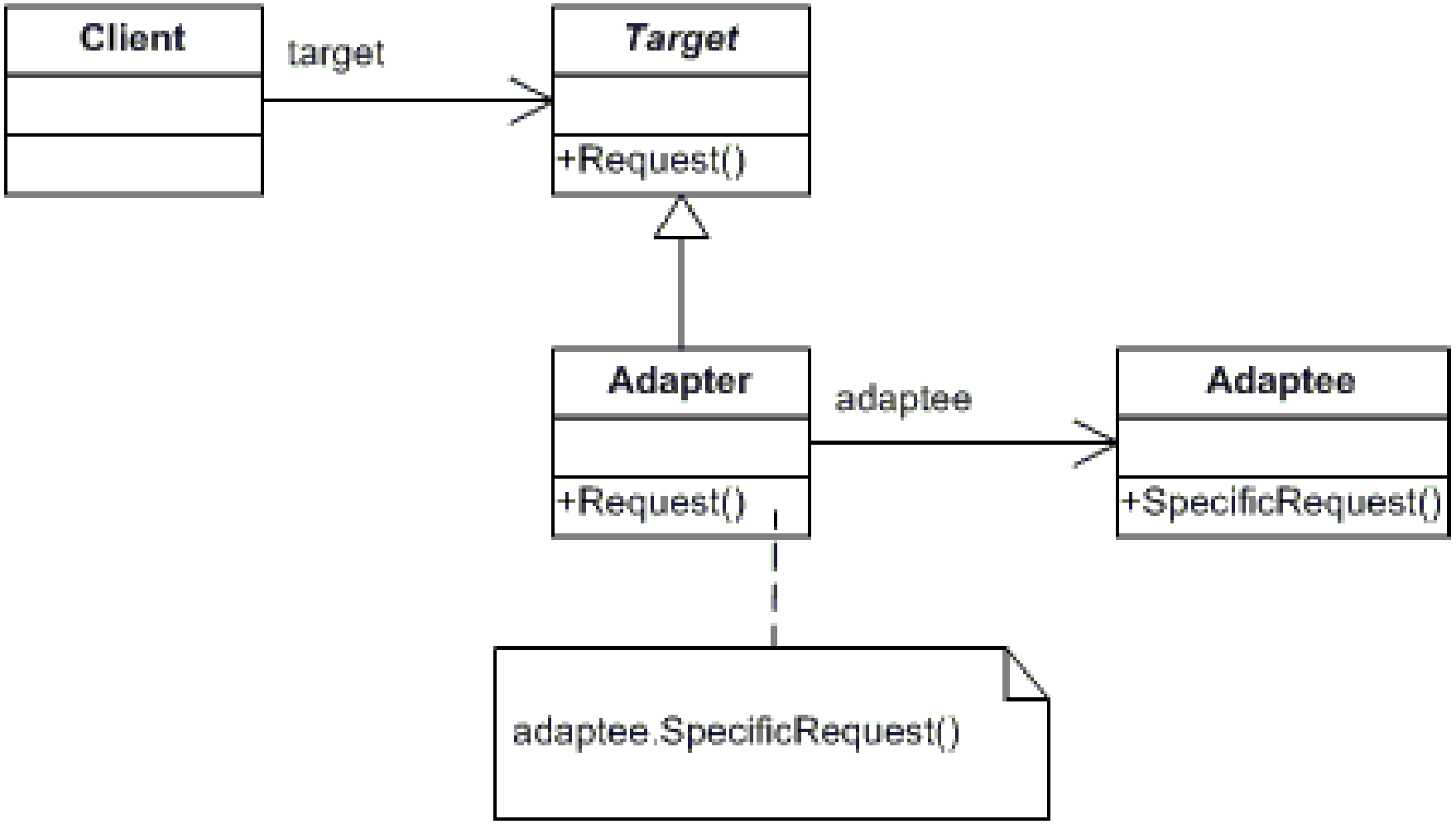
Мотивация


Client
«interface»
Shape
+display(in x1, in y1, in x2, in y2)
Rectangle
+display(in x1, in y1, in x2, in y2)
«adaptee»
LegacyRectangle
+display(in x1, in y1, in w, in h)
Delegate and map to adaptee.

Структура


Client
Target
Request()
Adaptee
SpecificRequest()
(реализация)
Adapter
Request()
SpecificRequest()

Структура


Client
target
Target
+Request()
Adapter
+Request()
adaptee
Adaptee
+SpecificRequest()
adaptee.SpecificRequest()

Примеры

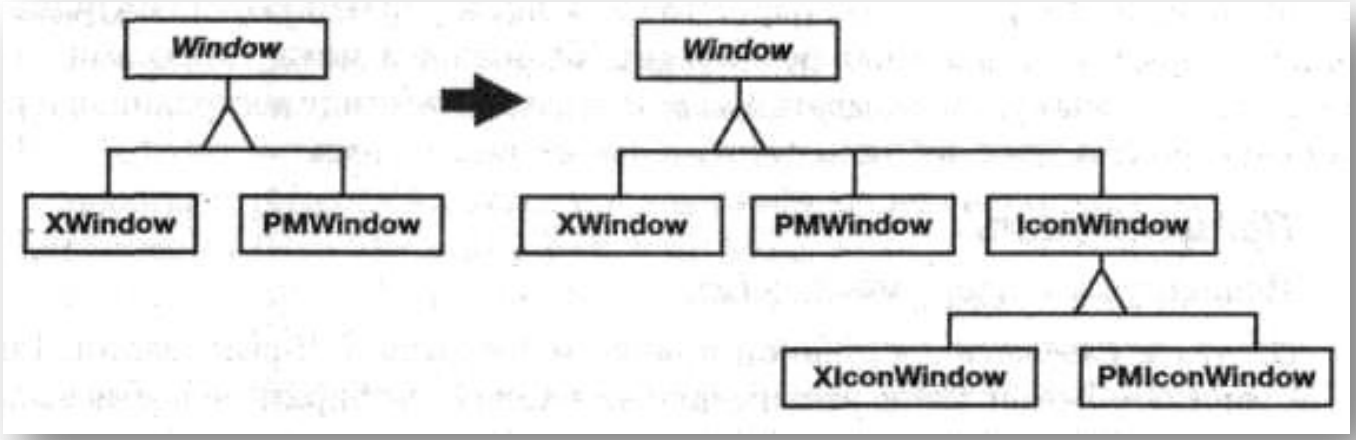
java.util.Arrays#asList()

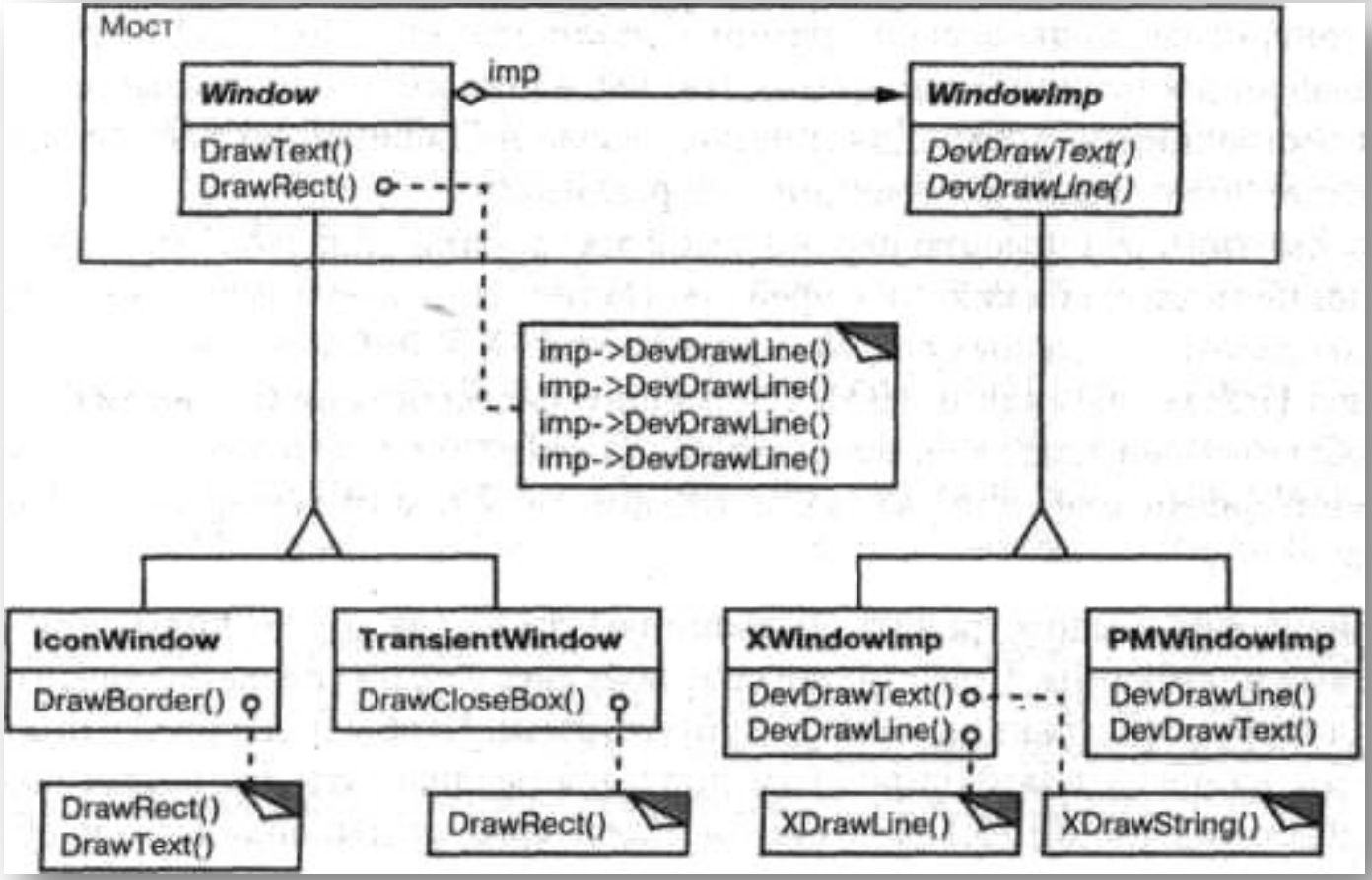
Bridge – Мост

- **Назначение:** отделить абстракцию от реализации так, чтобы и то и другое можно было изменять независимо

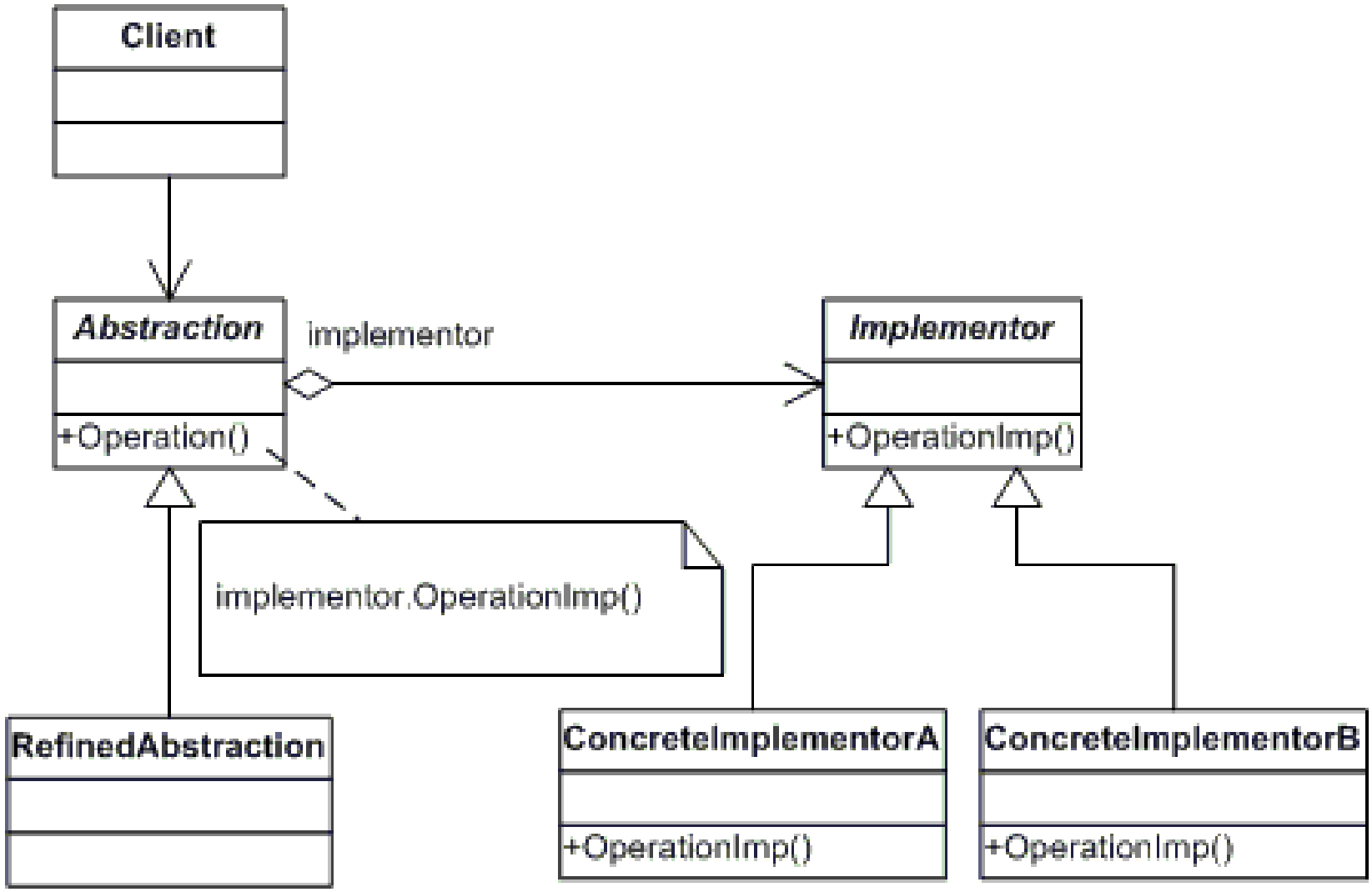
- **Применимость:**
  - Изменение реализации в динамике
  - Возможность независимо расширять иерархии классов
  - Клиентский код завязан только на абстракцию и при изменении реализации перекомпиляции не происходит

Мотивация


Window
XWindow
PMWindow
Window
XWindow
PMWindow
IconWindow
XIconWindow
PMIconWindow


Мост
Window
DrawText()
DrawRect()
imp
WindowImp
DevDrawText()
DevDrawLine()
imp->DevDrawLine()
imp->DevDrawLine()
imp->DevDrawLine()
imp->DevDrawLine()
IconWindow
DrawBorder()
TransientWindow
DrawCloseBox()
XWindowImp
DevDrawText()
DevDrawLine()
PMWindowImp
DevDrawLine()
DevDrawText()
DrawRect()
DrawText()
DrawRect()
XDrawLine()
XDrawString()

Структура

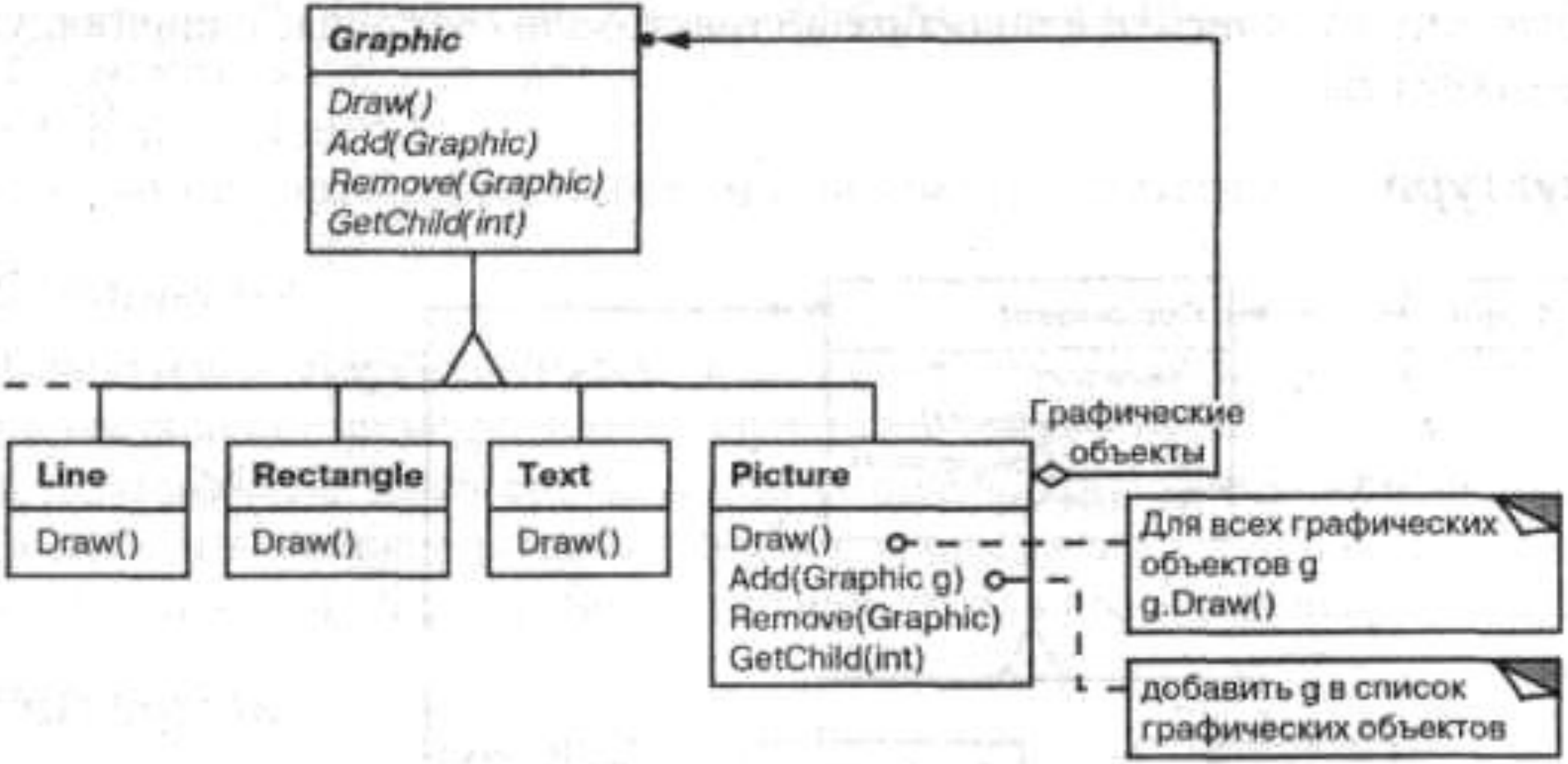

Client
Abstraction
implementor
Implementor
+Operation()
+OperationImp()
implementor.OperationImp()
RefinedAbstraction
ConcreteImplementorA
+OperationImp()
ConcreteImplementorB
+OperationImp()

Composite – Компоновщик

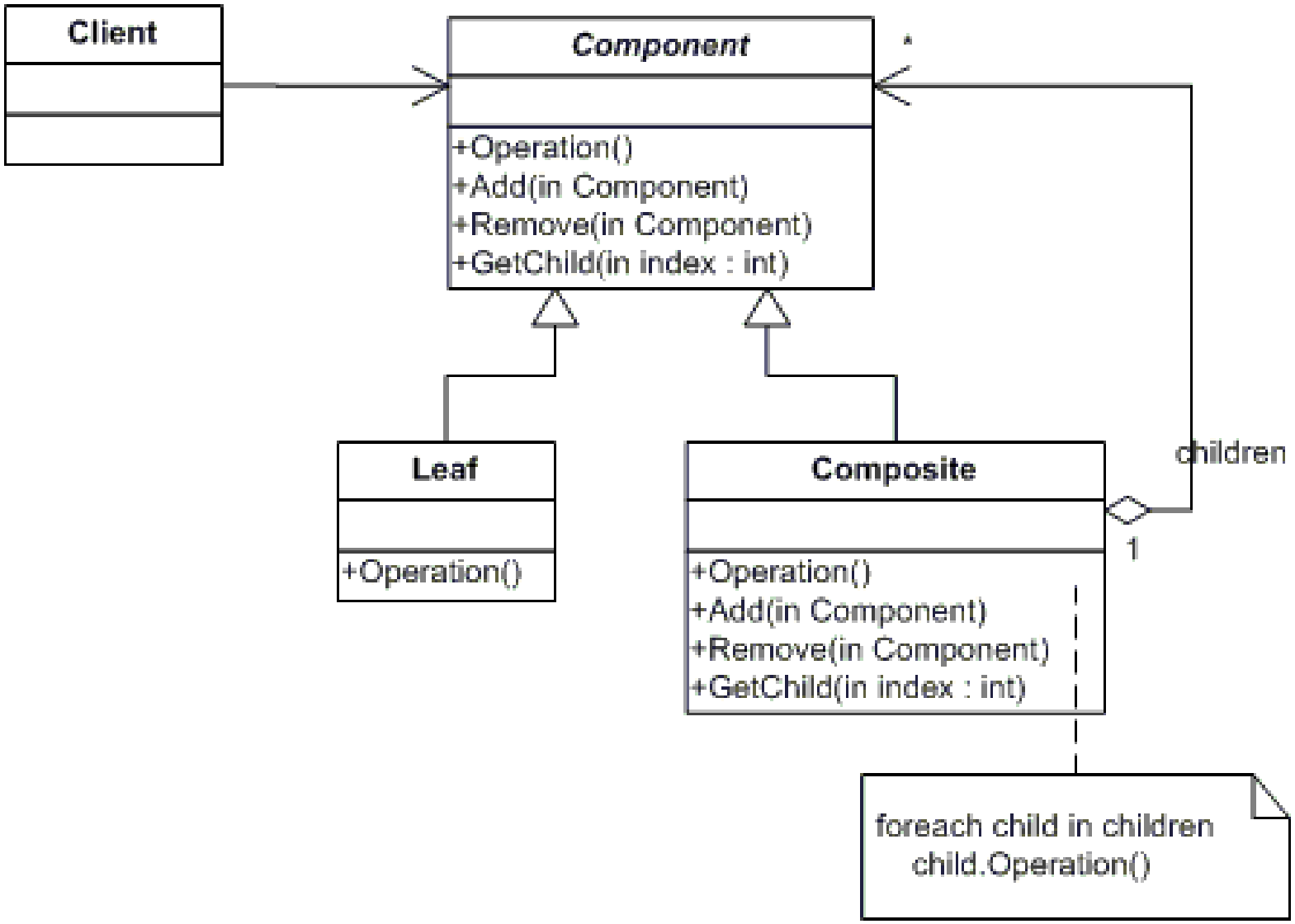
- **Назначение:** скомпоновать объекты в виде древовидной структуры для представления иерархий типа часть-целое

- **Применимость:**
  - единообразная трактовка со стороны клиента как составного объекта, так и индивидуального

Мотивация


Graphic
Draw()
Add(Graphic)
Remove(Graphic)
GetChild(int)
Line
Draw()
Rectangle
Draw()
Text
Draw()
Picture
Draw()
Add(Graphic g)
Remove(Graphic)
GetChild(int)
Графические объекты
Для всех графических объектов g
g.Draw()
добавить g в список графических объектов

Структура


Client
Component
*
+Operation()
+Add(in Component)
+Remove(in Component)
+GetChild(in index : int)
Leaf
+Operation()
Composite
+Operation()
+Add(in Component)
+Remove(in Component)
+GetChild(in index : int)
children
1
foreach child in children
child.Operation()

Обсуждение

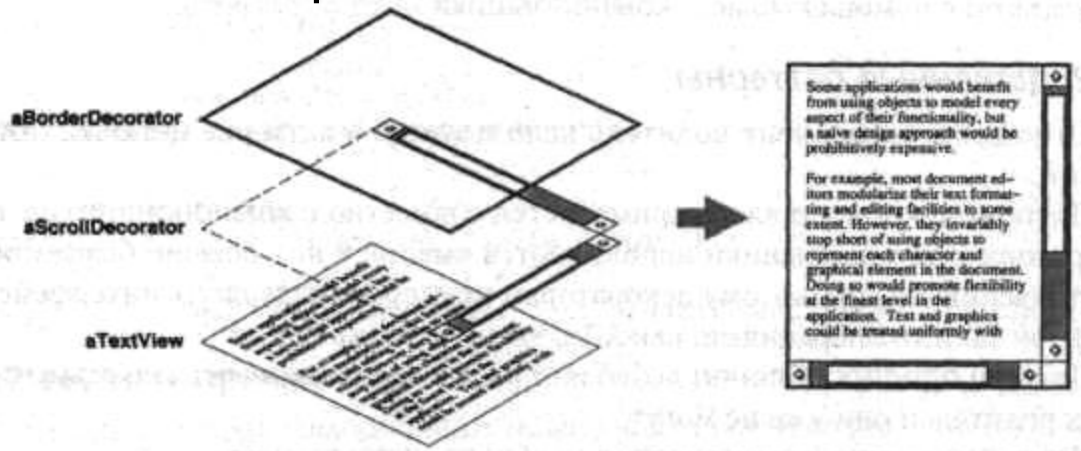
Преимущества/недостатки

+ упрощение архитектуры клиента
+ облегчает добавление новых типов компонентов
- трудно наложить ограничения на то, какие объекты могут быть скомпонованы

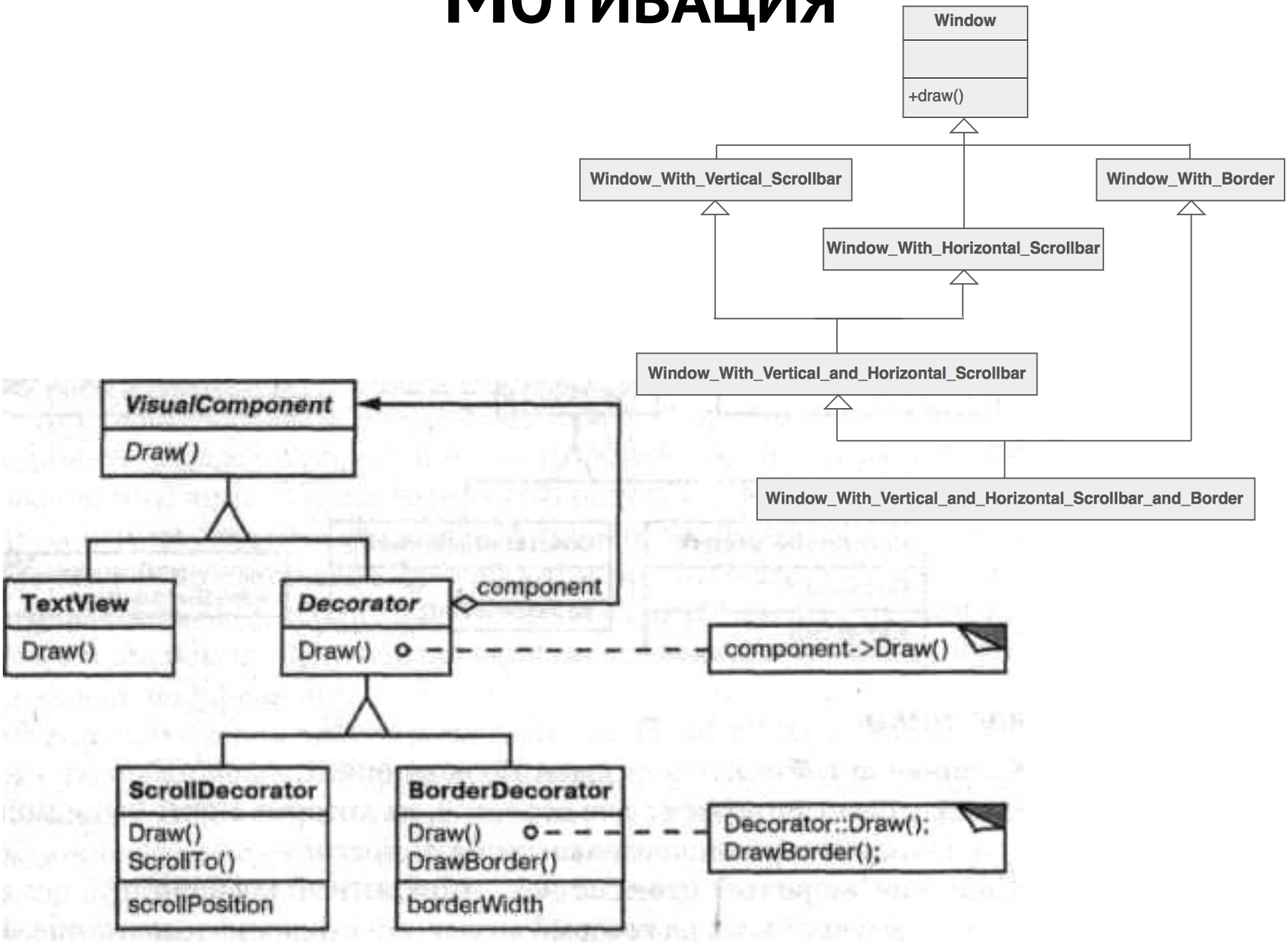
Decorator – Декоратор

- **Назначение:** динамически добавить объекту новые функции

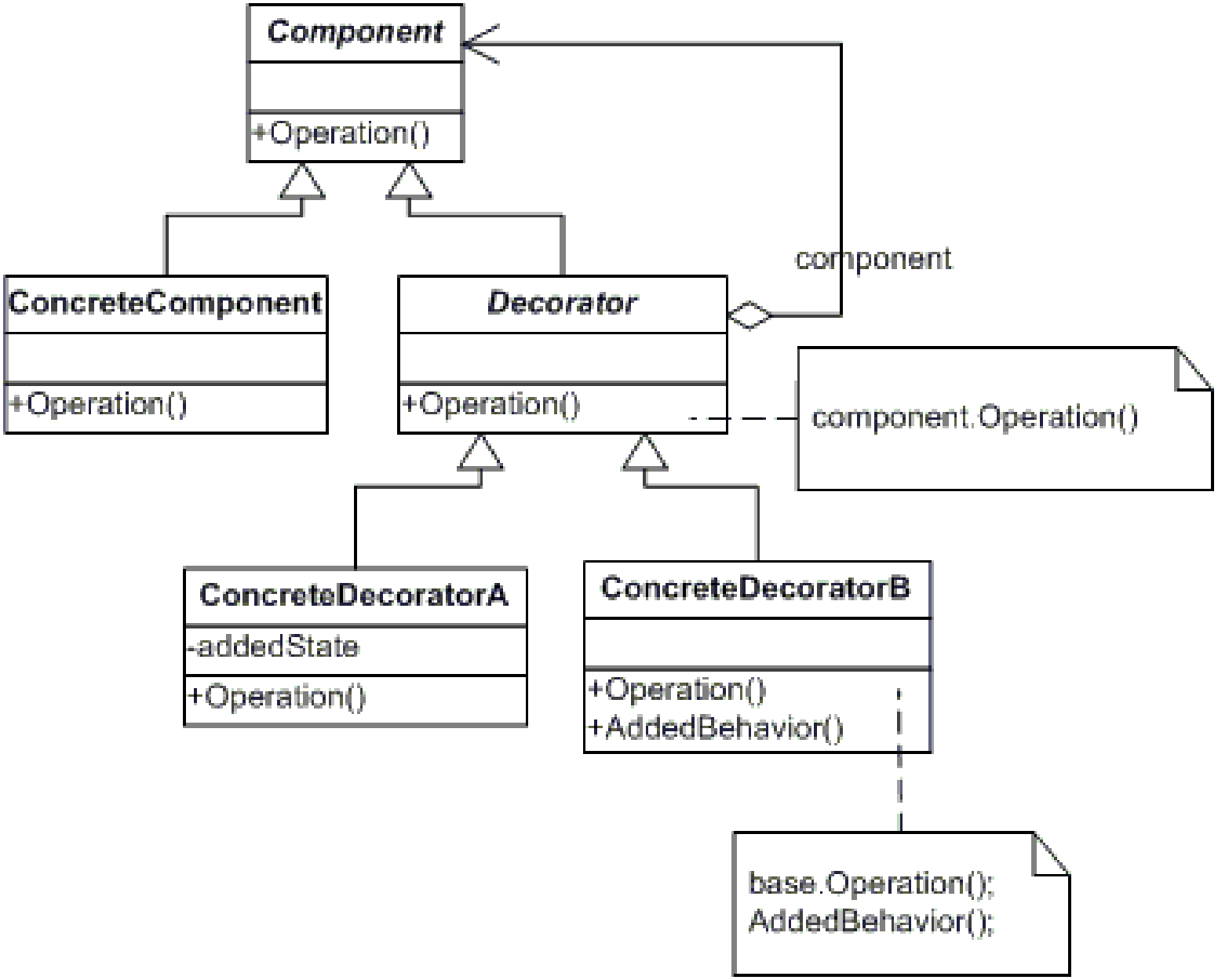
- **Применимость:**
  - для динамического, прозрачного для клиентов добавления обязанностей объектам
  - для реализации обязанностей, которые могут быть сняты с объекта
  - в случаях, когда расширение функциональности посредством наследования не работает



Мотивация

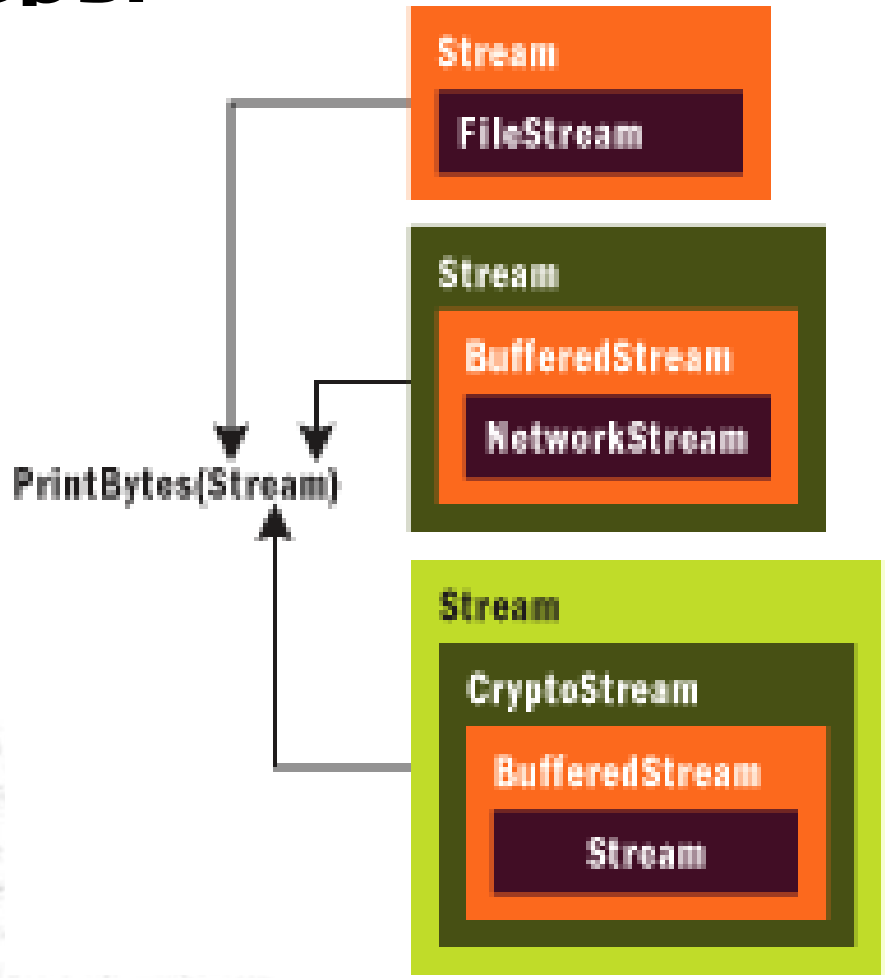

Window
+draw()
Window_With_Vertical_Scrollbar
Window_With_Horizontal_Scrollbar
Window_With_Border
Window_With_Vertical_and_Horizontal_Scrollbar
Window_With_Vertical_and_Horizontal_Scrollbar_and_Border
VisualComponent
Draw()
TextView
Draw()
Decorator
Draw()
component
component->Draw()
ScrollDecorator
Draw()
ScrollTo()
scrollPosition
BorderDecorator
Draw()
DrawBorder()
borderWidth
Decorator::Draw();
DrawBorder();

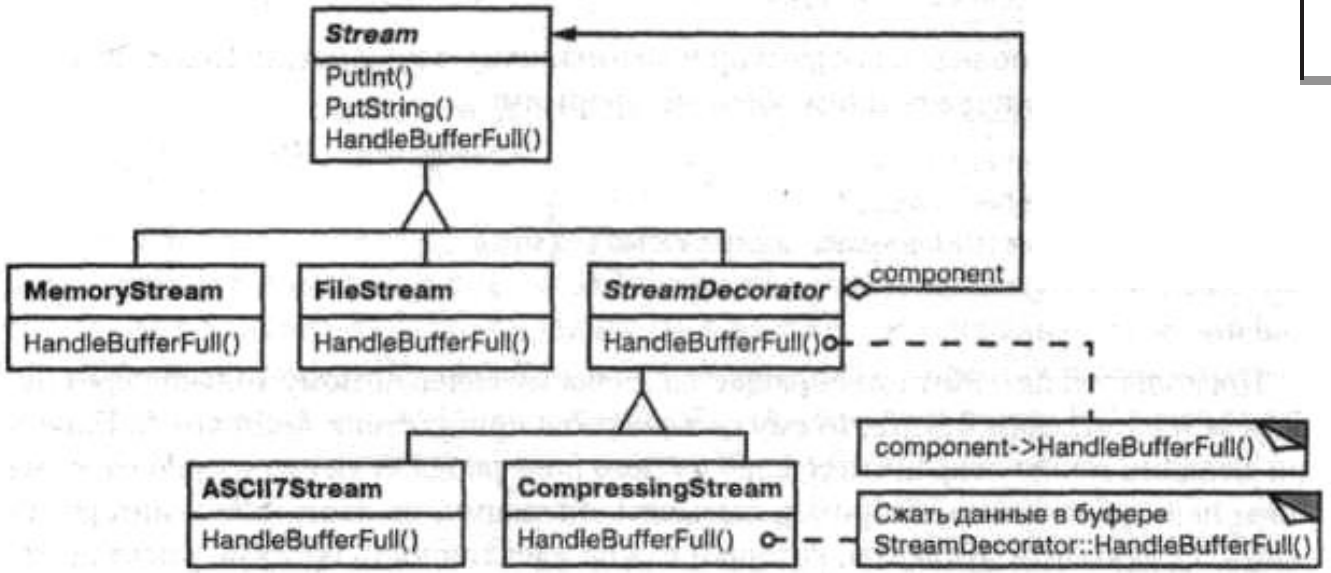
Структура


Component
+Operation()
component
ConcreteComponent
+Operation()
Decorator
+Operation()
component.Operation()
ConcreteDecoratorA
-addedState
+Operation()
ConcreteDecoratorB
+Operation()
+AddedBehavior()
base.Operation();
AddedBehavior();

Примеры

- Java synchronized*Set*
- Java Streams
- .NET Streams →



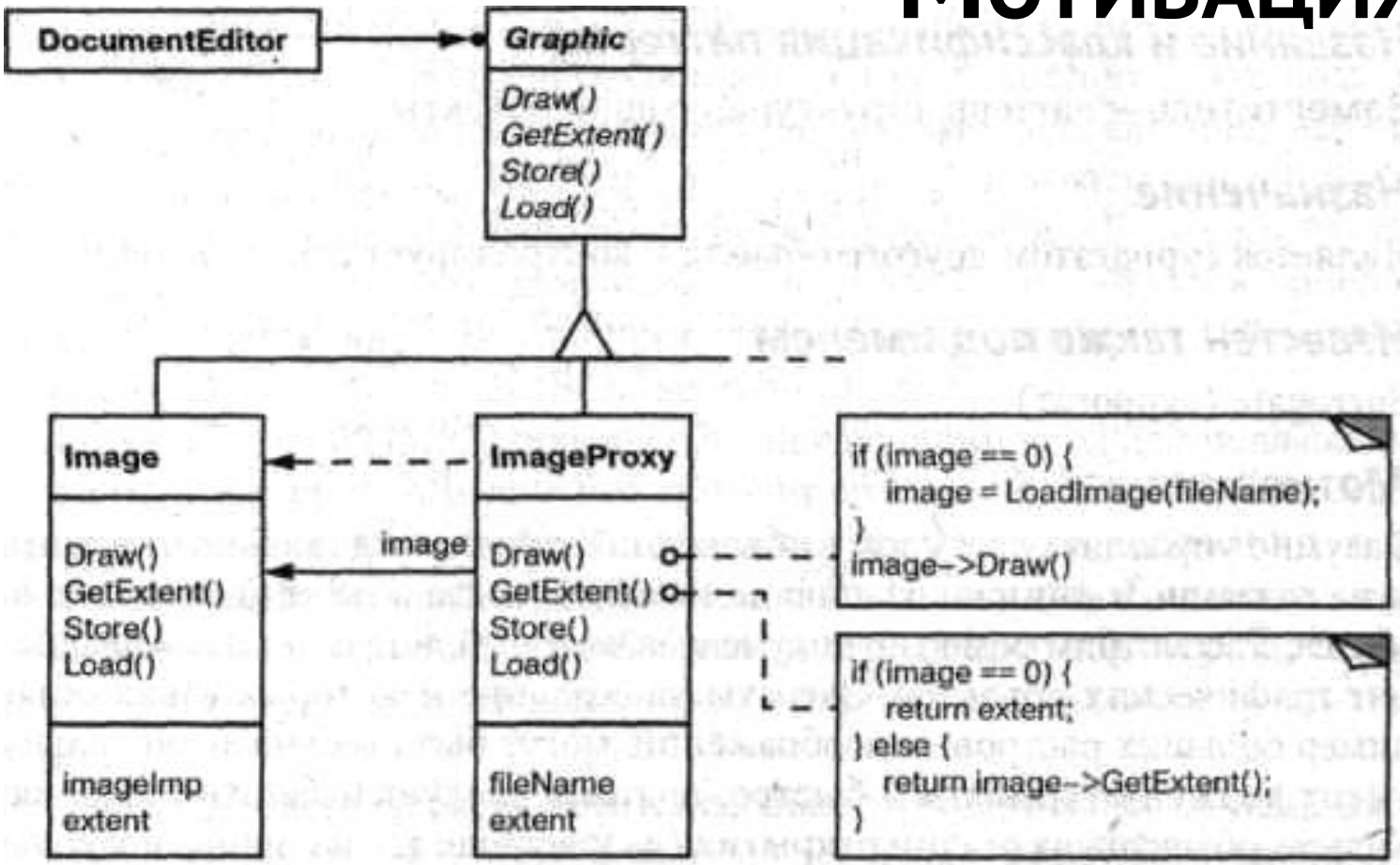


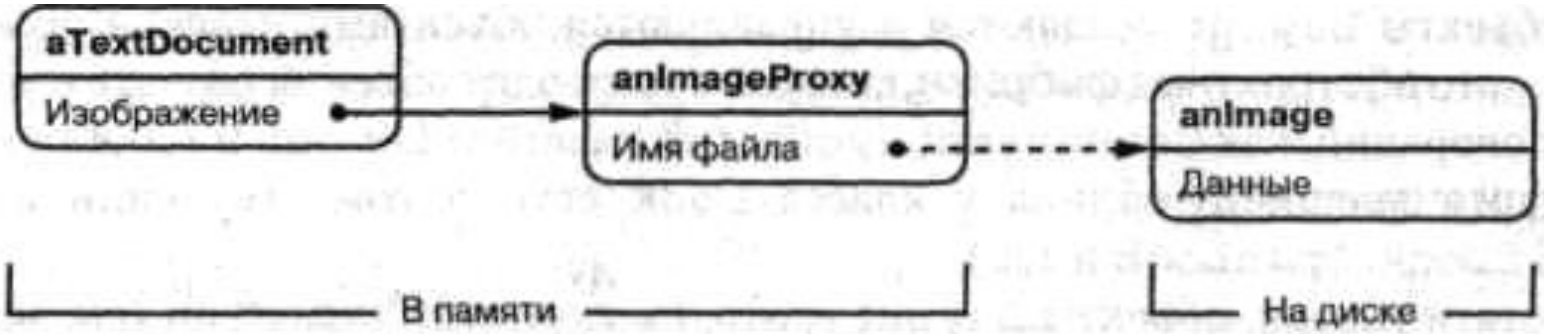
Proxy – Заместитель

- **Назначение:** оперировать объектом-заместителем для предоставления доступа к целевому объекту согласно определенной дополнительной логике

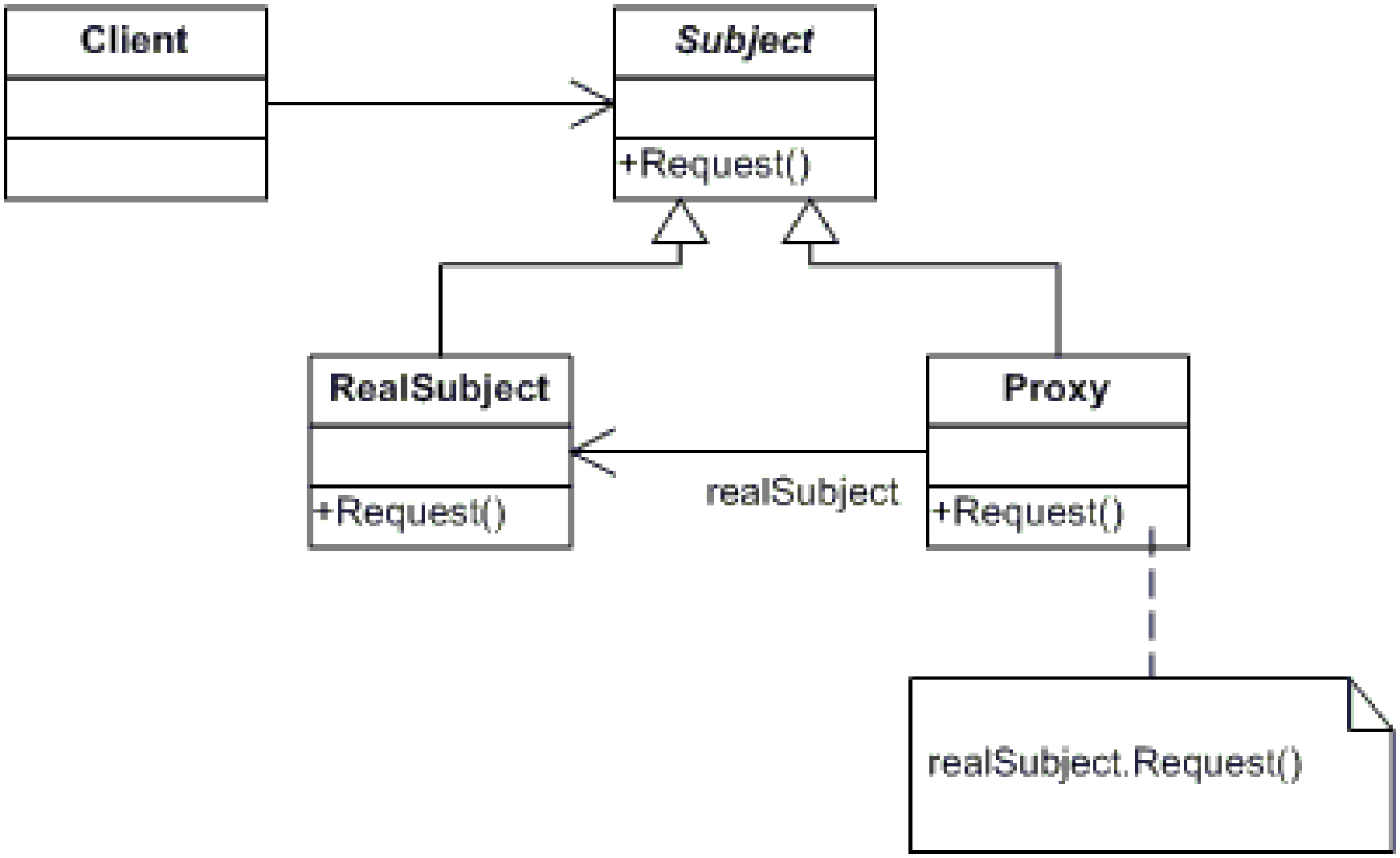
- **Условия применения:**
  - удаленный заместитель
  - виртуальный заместитель для тяжелых объектов
  - защищающий заместитель
  - умный указатель

Мотивация


DocumentEditor
Graphic
Draw()
GetExtent()
Store()
Load()
Image
Draw()
GetExtent()
Store()
Load()
imageImp
extent
image
ImageProxy
Draw()
GetExtent()
Store()
Load()
fileName
extent
if (image == 0) {
image = LoadImage(fileName);
}
image->Draw()
if (image == 0) {
return extent;
} else {
return image->GetExtent();
}


aTextDocument
Изображение
anImageProxy
Имя файла
anImage
Данные
В памяти
На диске

Структура


Client
Subject
+Request()
RealSubject
+Request()
realSubject
Proxy
+Request()
realSubject.Request()

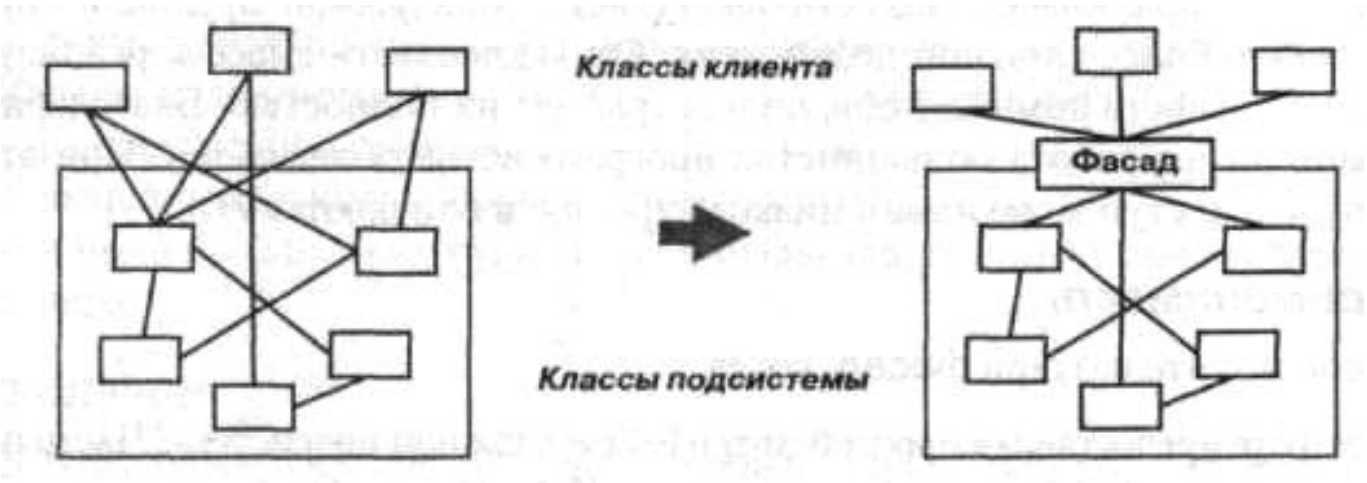
Примеры

- Lazy images loading
- RPC (Java RMI, .NET Remoting)
- SOAP WebServices

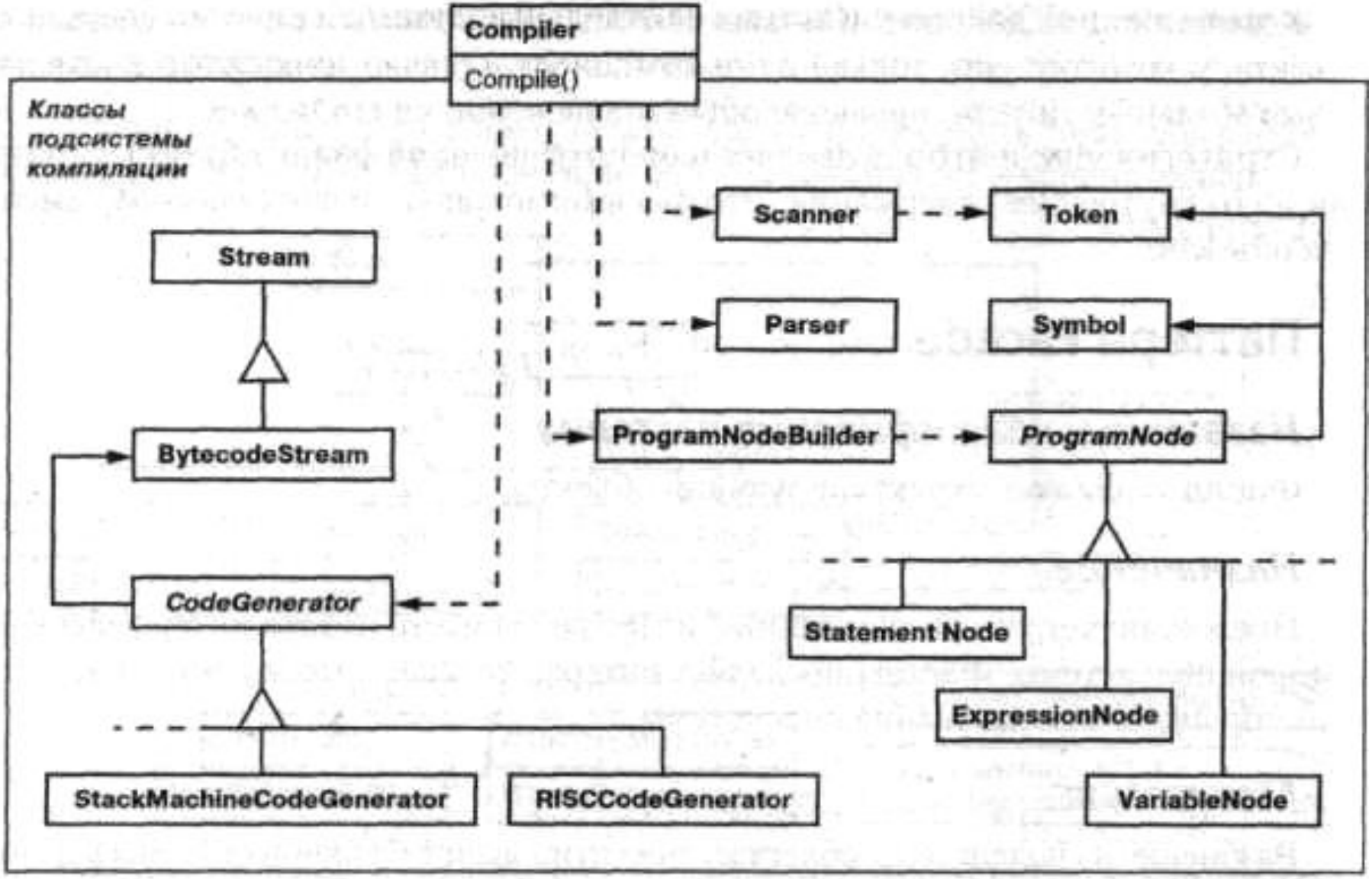
Facade

- **Назначение:** предоставить унифицированный высокоуровневый интерфейс вместо набора интерфейсов подсистемы

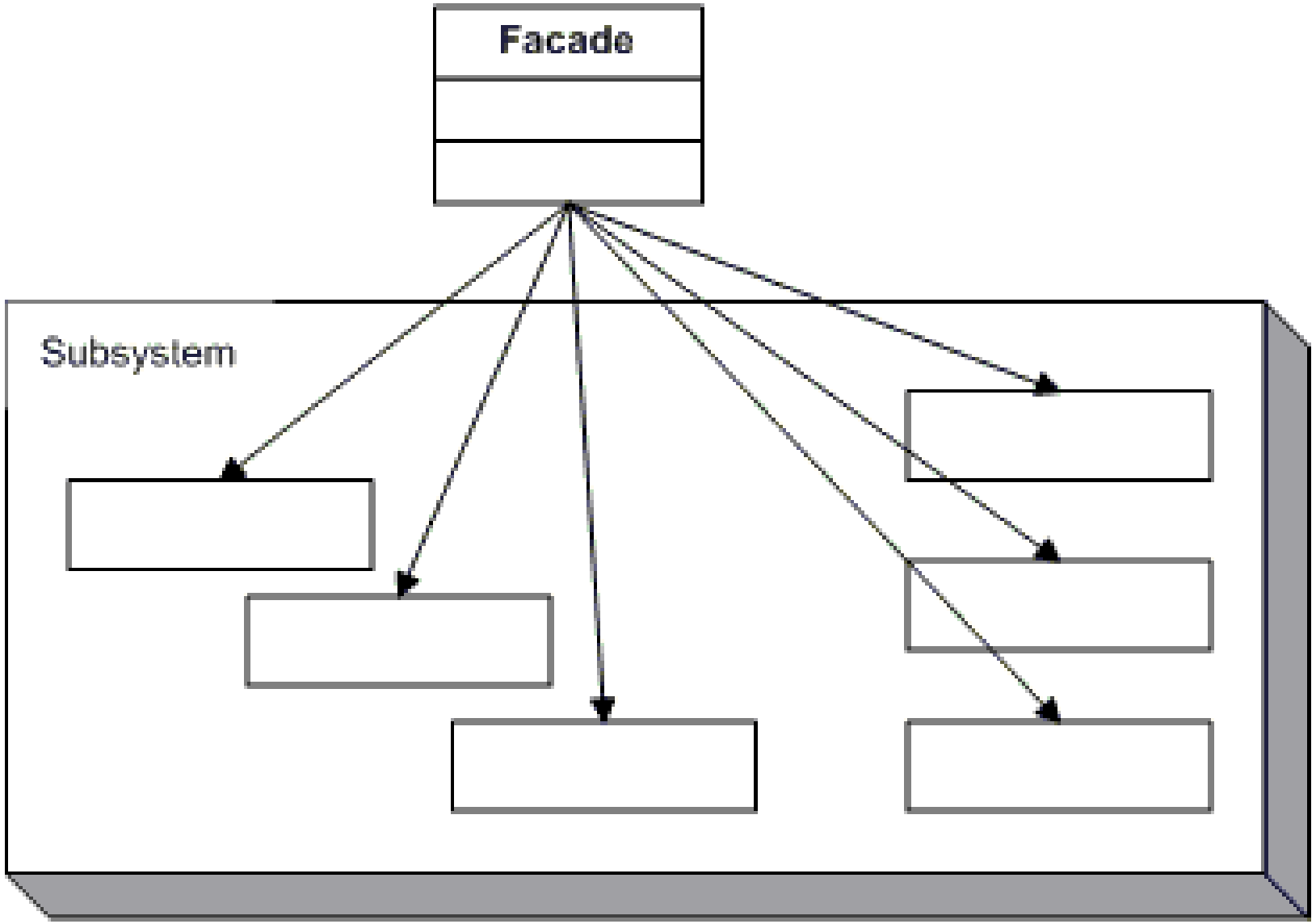
- **Применимость:**
  - упрощение использования подсистем
  - для реализации обязанностей, которые могут быть сняты с объекта
  - в случаях, когда расширение функциональности посредством наследования не работает



Мотивация


Compiler
Compile()
Классы подсистемы компиляции
Scanner
Token
Stream
Parser
Symbol
ProgramNodeBuilder
ProgramNode
BytecodeStream
CodeGenerator
Statement Node
ExpressionNode
StackMachineCodeGenerator
RISCCodeGenerator
VariableNode

Структура

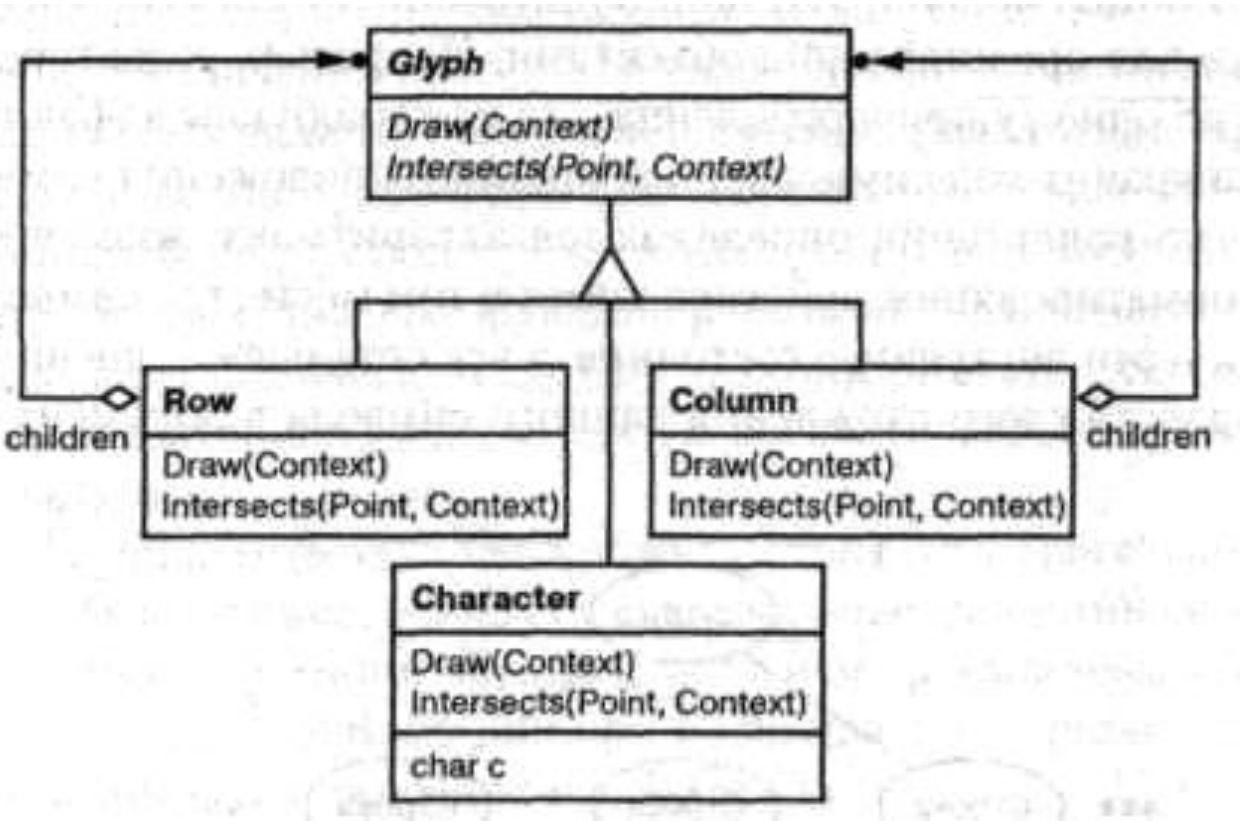


Flyweight - Приспособленец

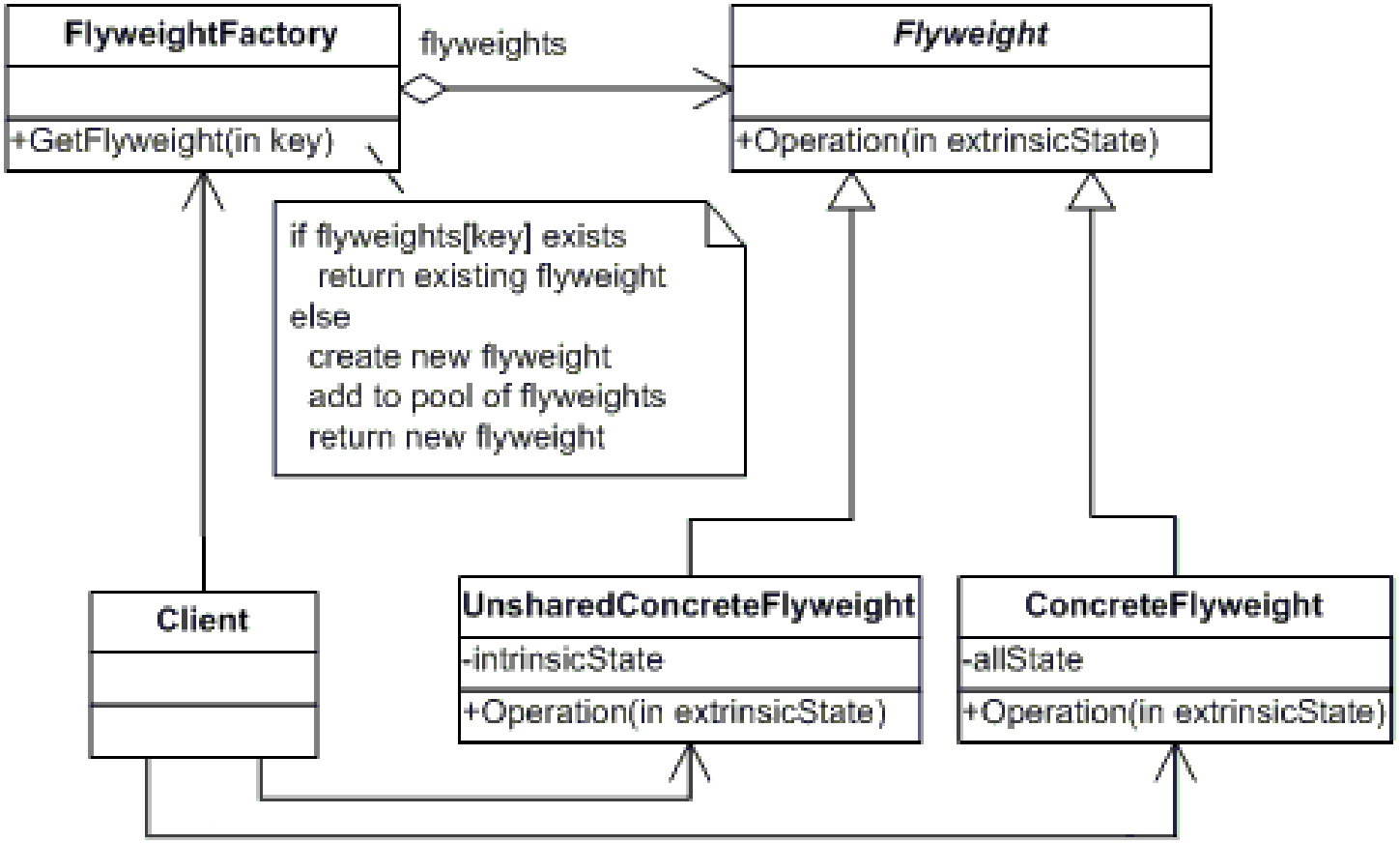
- **Назначение:** оптимизировать представление большого количества мало различающихся однородных объектов

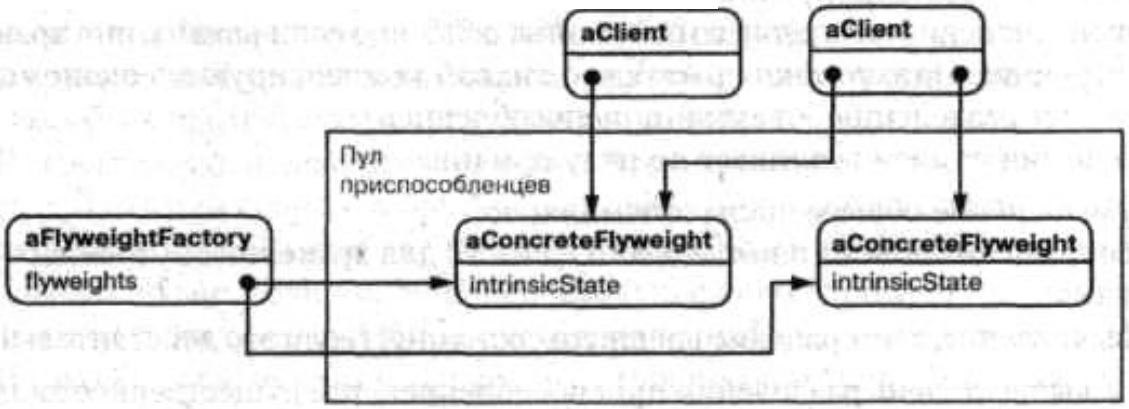
- **Условия применения:**
  - большое число объектов
  - накладные расходы на их хранение высоки
  - большую часть состояния объектов можно вынести вовне

Мотивация


Glyph
Draw(Context)
Intersects(Point, Context)
Row
Draw(Context)
Intersects(Point, Context)
Column
Draw(Context)
Intersects(Point, Context)
Character
Draw(Context)
Intersects(Point, Context)
char c
children
children

Структура


FlyweightFactory
flyweights
Flyweight
+GetFlyweight(in key)
+Operation(in extrinsicState)
if flyweights[key] exists
return existing flyweight
else
create new flyweight
add to pool of flyweights
return new flyweight
Client
UnsharedConcreteFlyweight
-intrinsicState
+Operation(in extrinsicState)
ConcreteFlyweight
-allState
+Operation(in extrinsicState)


aClient
aClient
Пул
приспособленцев
aFlyweightFactory
flyweights
aConcreteFlyweight
intrinsicState
aConcreteFlyweight
intrinsicState

Примеры

- Текстовый редактор

Browser loads images just once and then reuses them from pool:

Паттерны поведения

Chain of Responsibility

Command

Interpreter

Iterator

Mediator

Memento

Observer

State

Strategy

Template Method

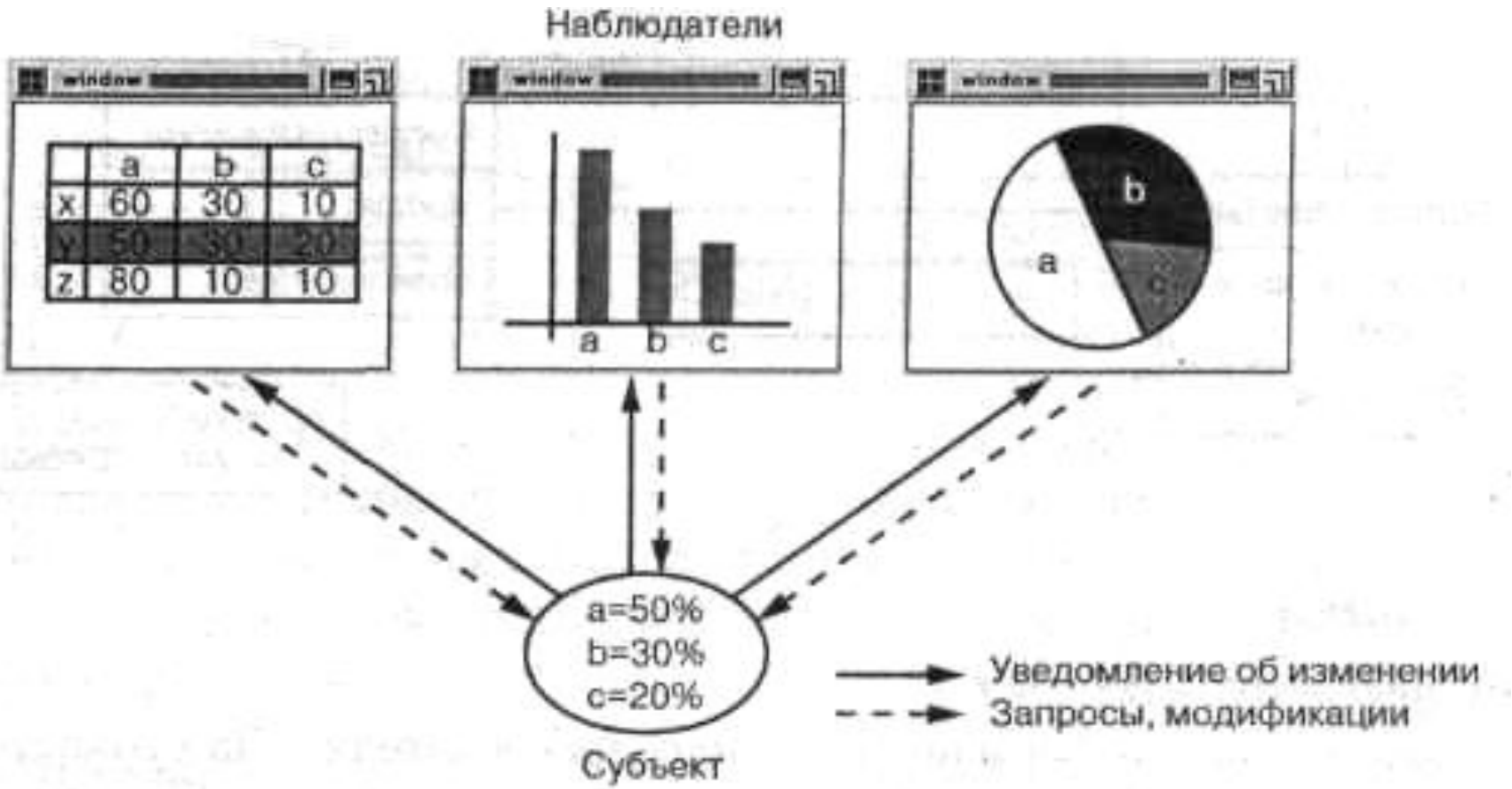
Visitor

Observer

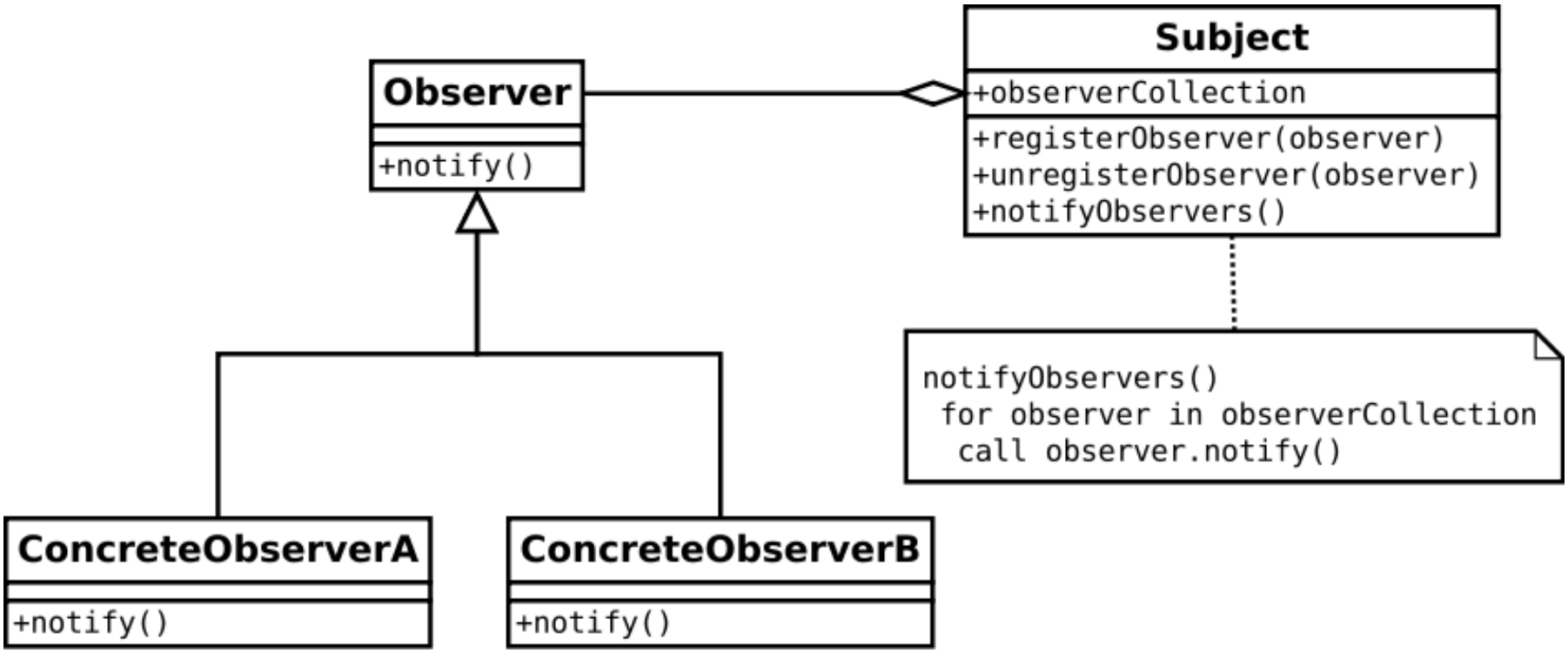
- **Назначение:** представить зависимость типа один-ко-многим так, что при изменении состояния одного объекта все зависящие оповещаются и обновляются

- **Применимость:**
  - объект должен оповещать другие, не зная о них заранее
  - соблюдение согласованности между множеством взаимосвязанных объектов

Мотивация


Наблюдатели
window
window
window
a b c
x 60 30 10
y 50 30 20
z 80 10 10
a
b
c
a b c
a=50%
b=30%
c=20%
Субъект
Уведомление об изменении
Запросы, модификации

Структура


Subject
+observerCollection
+registerObserver(observer)
+unregisterObserver(observer)
+notifyObservers()
Observer
+notify()
notifyObservers()
for observer in observerCollection
call observer.notify()
ConcreteObserverA
+notify()
ConcreteObserverB
+notify()

Примеры

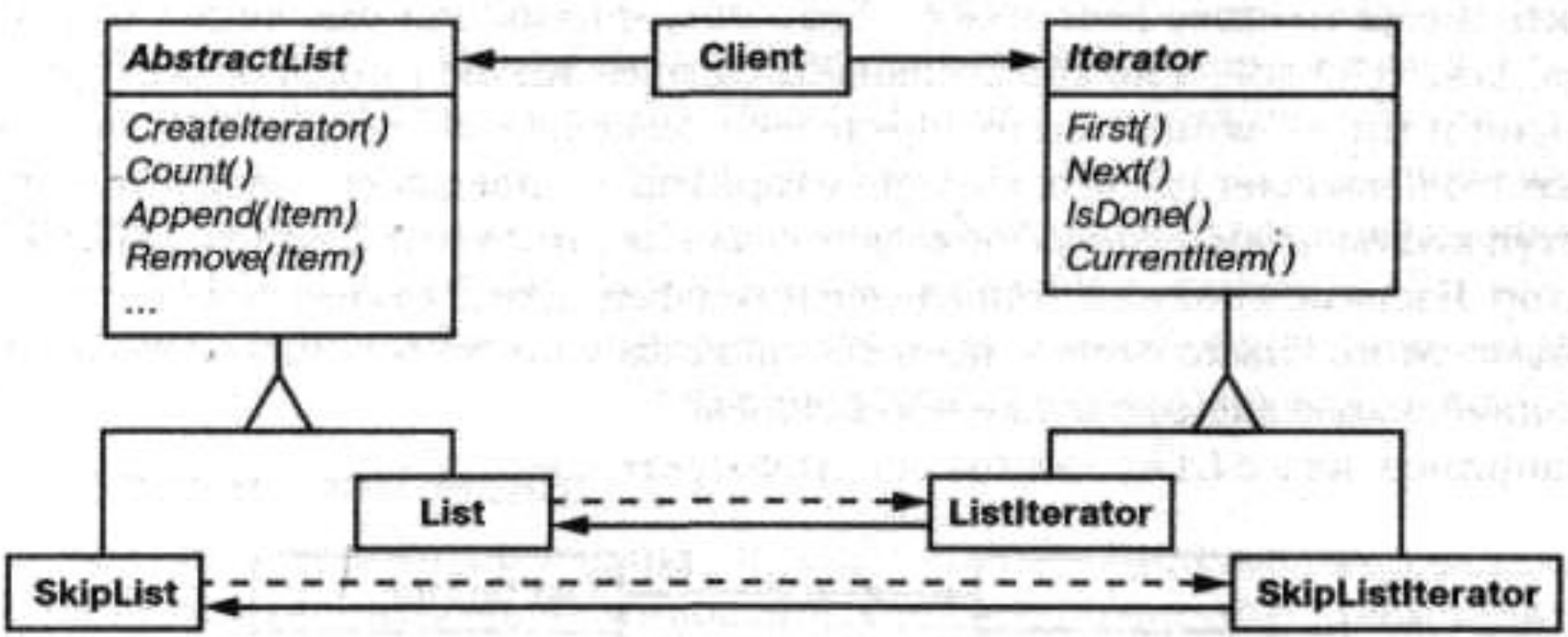
- .NET System.IObservable<T>
- java.util.Observable
- M**V**C

Iterator

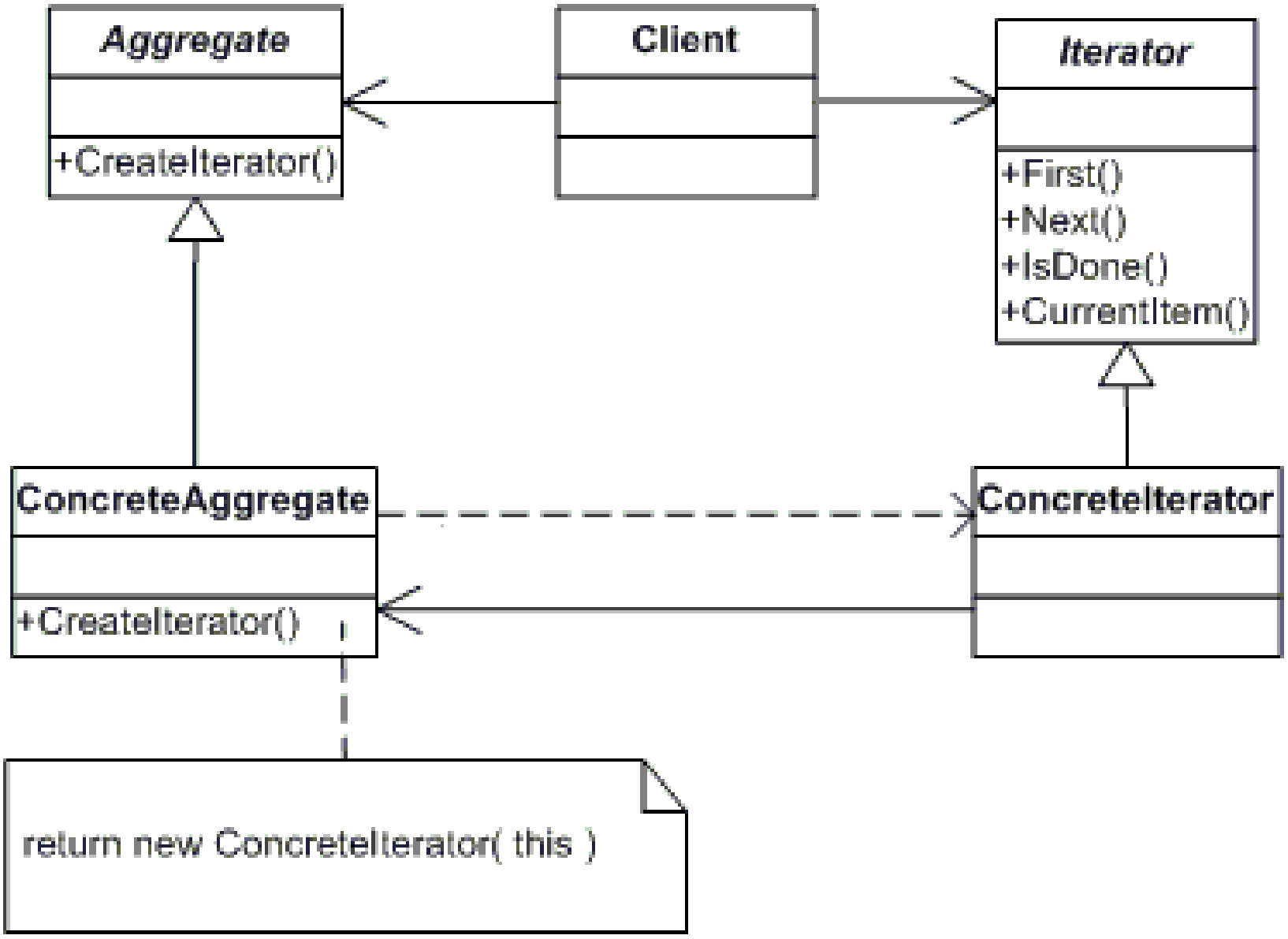
- **Назначение:** обеспечить абстракцию последовательного доступа к элементам множества без раскрытия реализации

- **Применимость:**
  - Доступ к содержимому агрегированных объектов без раскрытия их устройства
  - Разные обходы одного множества
  - Единообразный интерфейс для обхода различных агрегированных структур

Мотивация


AbstractList
CreateIterator()
Count()
Append(Item)
Remove(Item)
...
Client
Iterator
First()
Next()
IsDone()
CurrentItem()
List
ListIterator
SkipList
SkipListIterator

Структура


Aggregate
+CreateIterator()
Client
Iterator
+First()
+Next()
+IsDone()
+CurrentItem()
ConcreteAggregate
+CreateIterator()
ConcreteIterator
return new ConcreteIterator(this)

Примеры

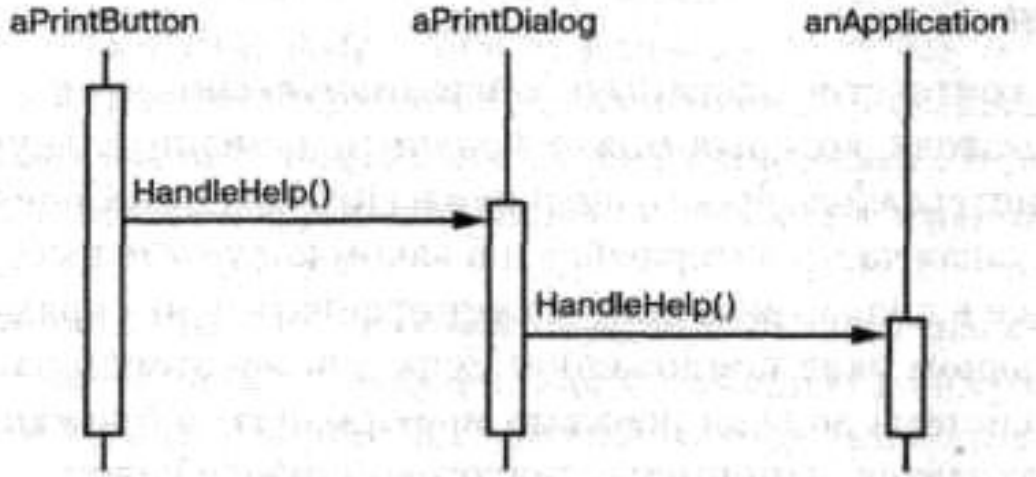
- .NET System.Collections.Generic.IEnumerable<T>
- java.util.Observable

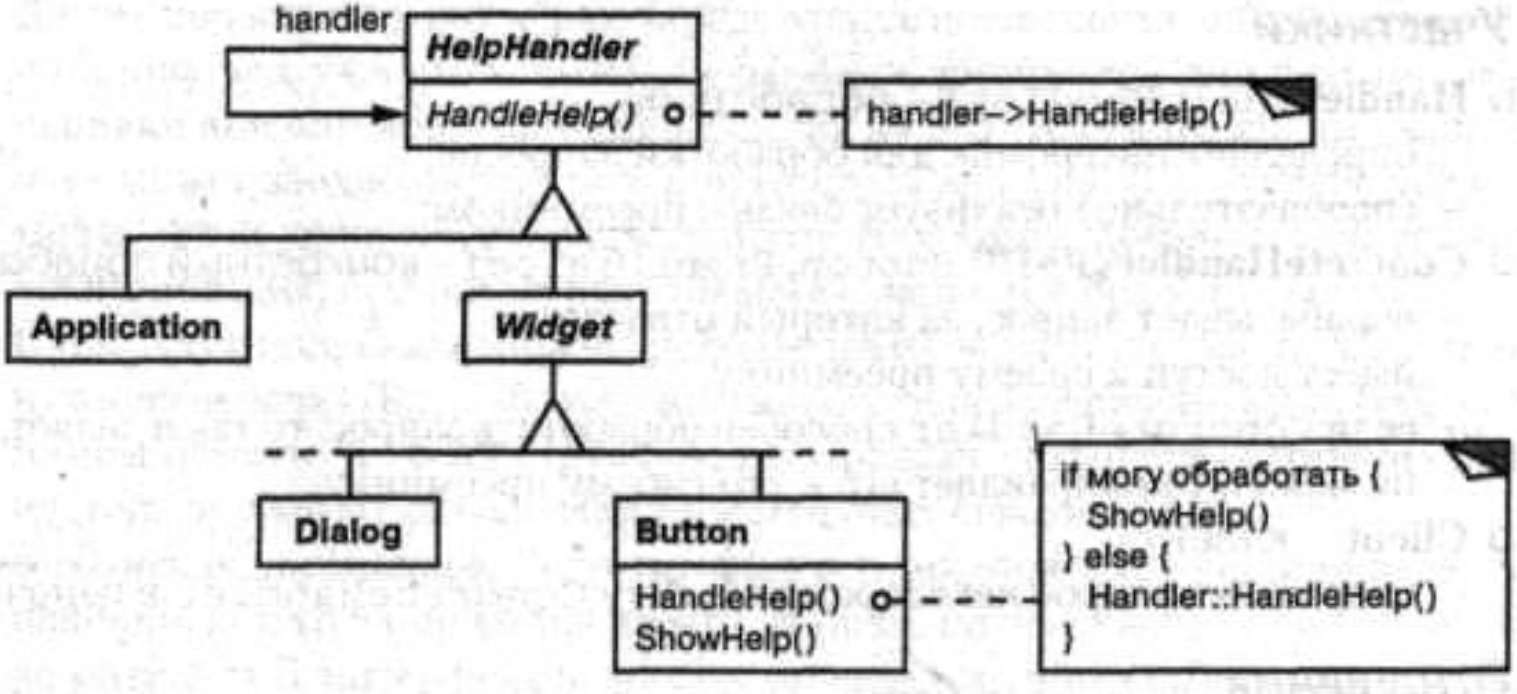
Chain of responsibility

- **Назначение:** избежать жесткого связывания отправителя запроса с получателем, запрос может быть обработан несколькими объектами

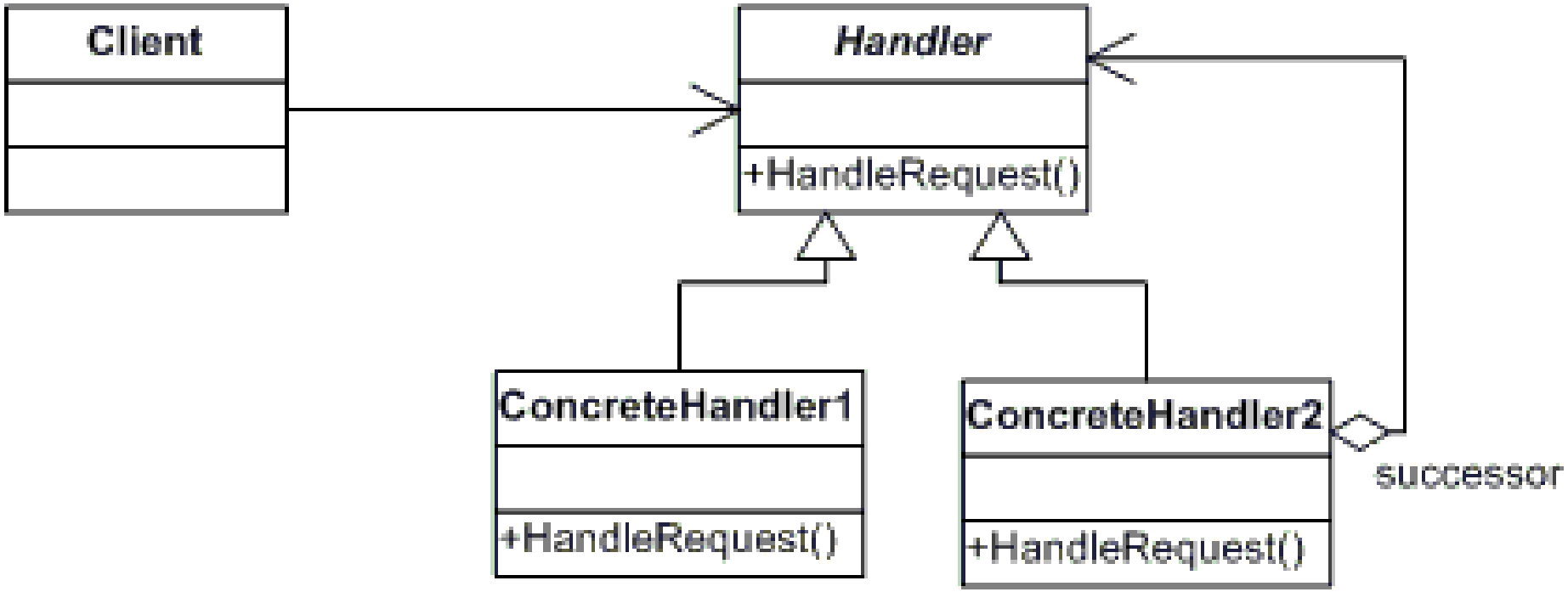
- **Применимость:**
  - Более одного объекта могут обработать запрос
  - Список обработчиков может формироваться в динамике
  - Запрос посылается без явного указания того, кем он должен быть обработан

Мотивация


aPrintButton
aPrintDialog
anApplication
HandleHelp()
HandleHelp()


handler
HelpHandler
HandleHelp()
handler->HandleHelp()
Application
Widget
Dialog
Button
HandleHelp()
ShowHelp()
if могу обработать {
ShowHelp()
} else {
Handler::HandleHelp()
}

Структура

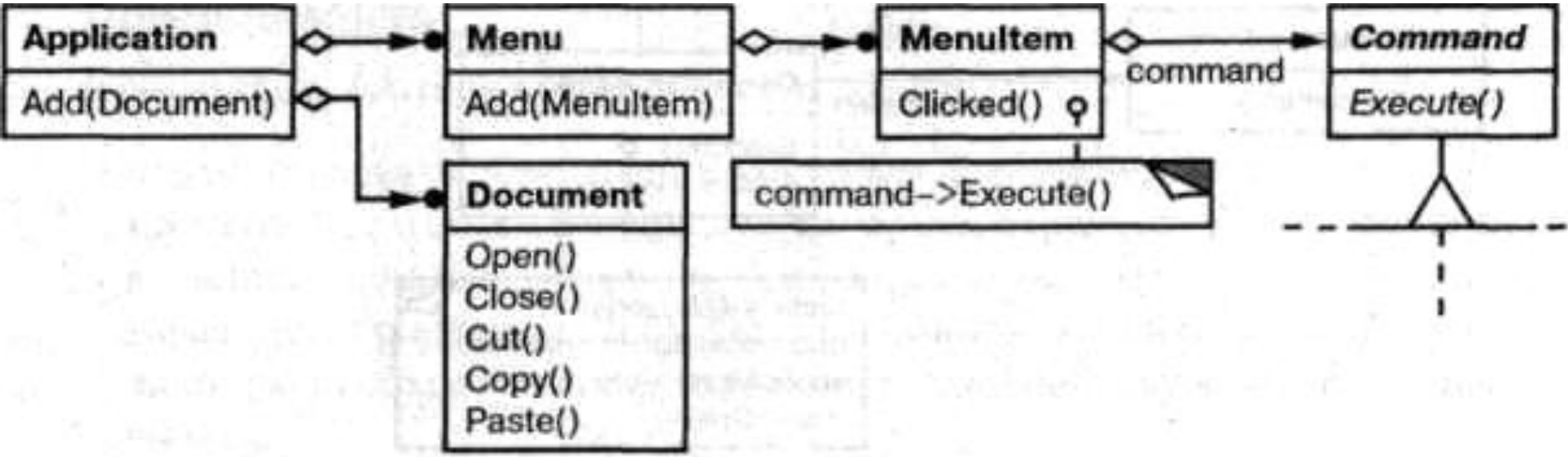

Client
Handler
+HandleRequest()
ConcreteHandler1
+HandleRequest()
ConcreteHandler2
+HandleRequest()
successor

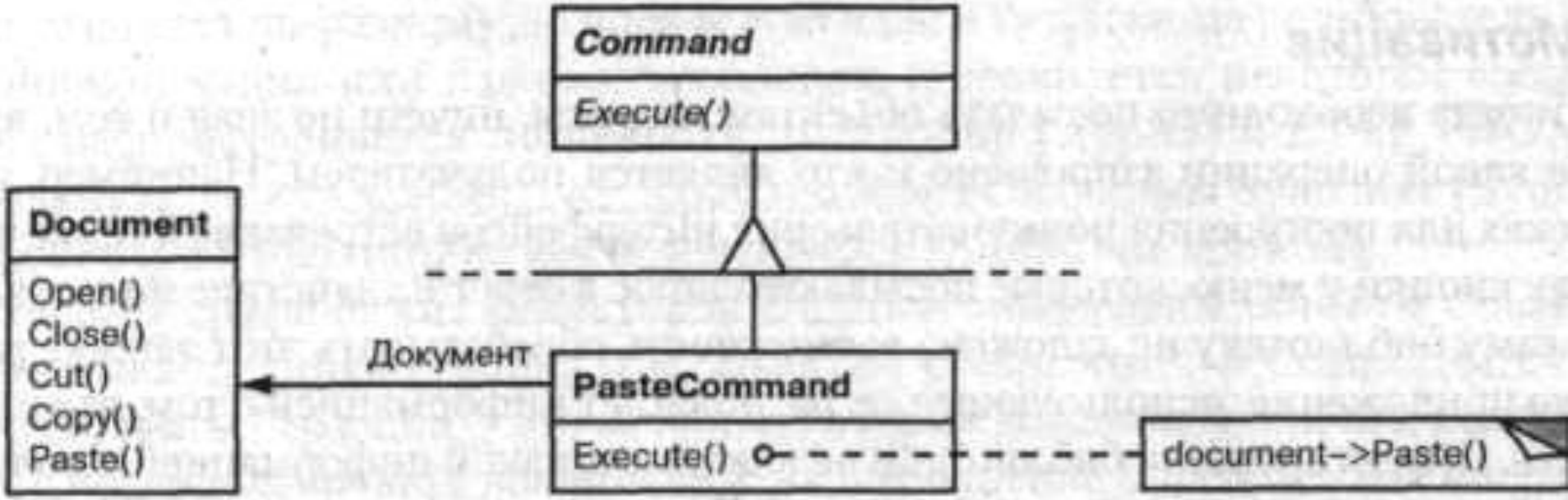
Command

- **Назначение:** обрабатывать запрос в виде объекта (ставить в очередь, протоколировать, отменять)

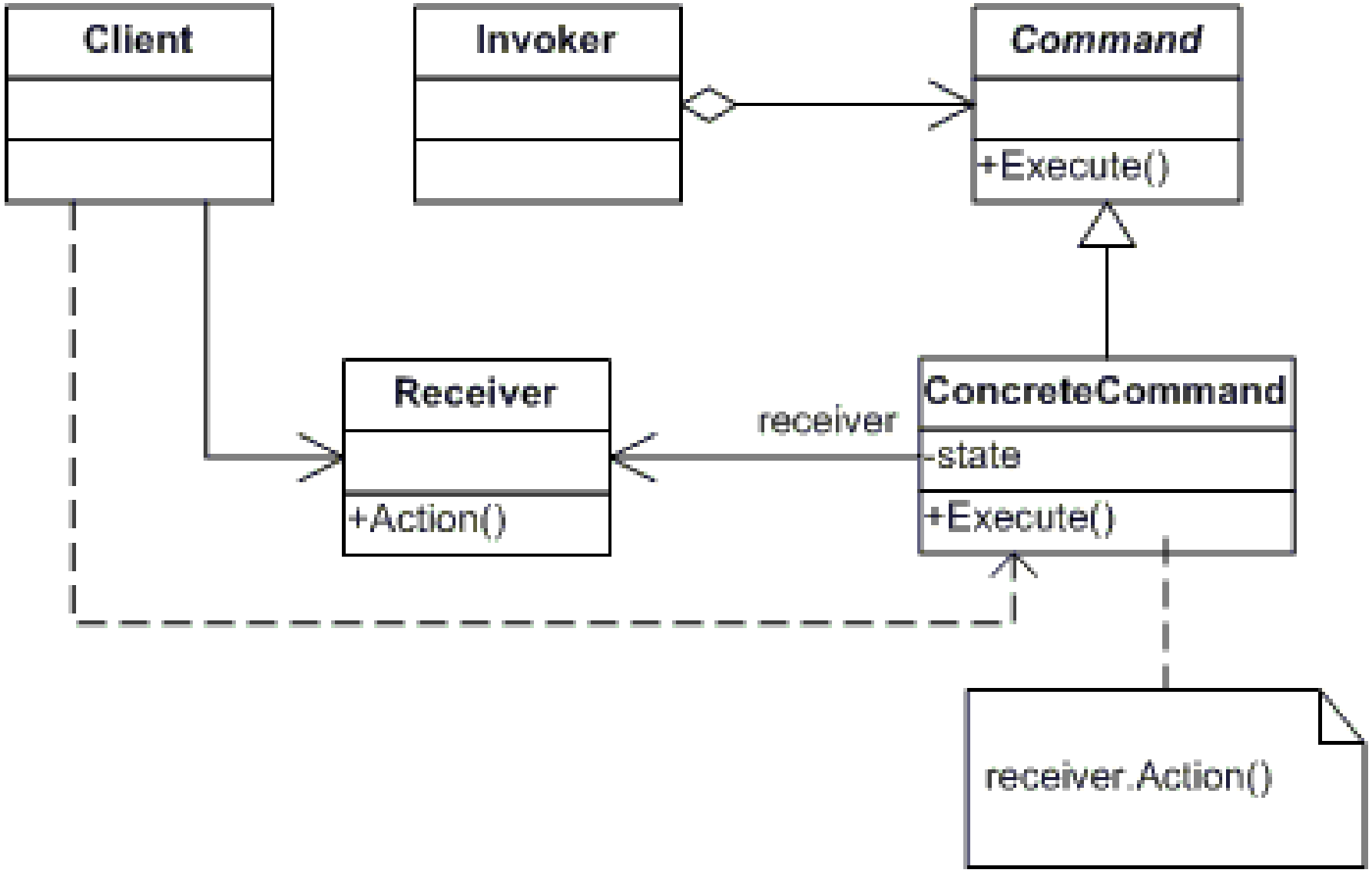
- **Применимость:**
  - Параметризовать объекты действием
  - Регулировать время (момент, когда) или место (где) исполнения запроса, ставить в очередь, маршрутизировать
  - Отмена операций
  - Протоколирование изменений
  - Поддержка сложных действий, составленных из более простых – транзакции

Мотивация


Application
Add(Document)
Menu
Add(MenuItem)
MenuItem
Clicked()
command
Command
Execute()
Document
Open()
Close()
Cut()
Copy()
Paste()
command->Execute()


Command
Execute()
Document
Open()
Close()
Cut()
Copy()
Paste()
Документ
PasteCommand
Execute()
document->Paste()

Структура

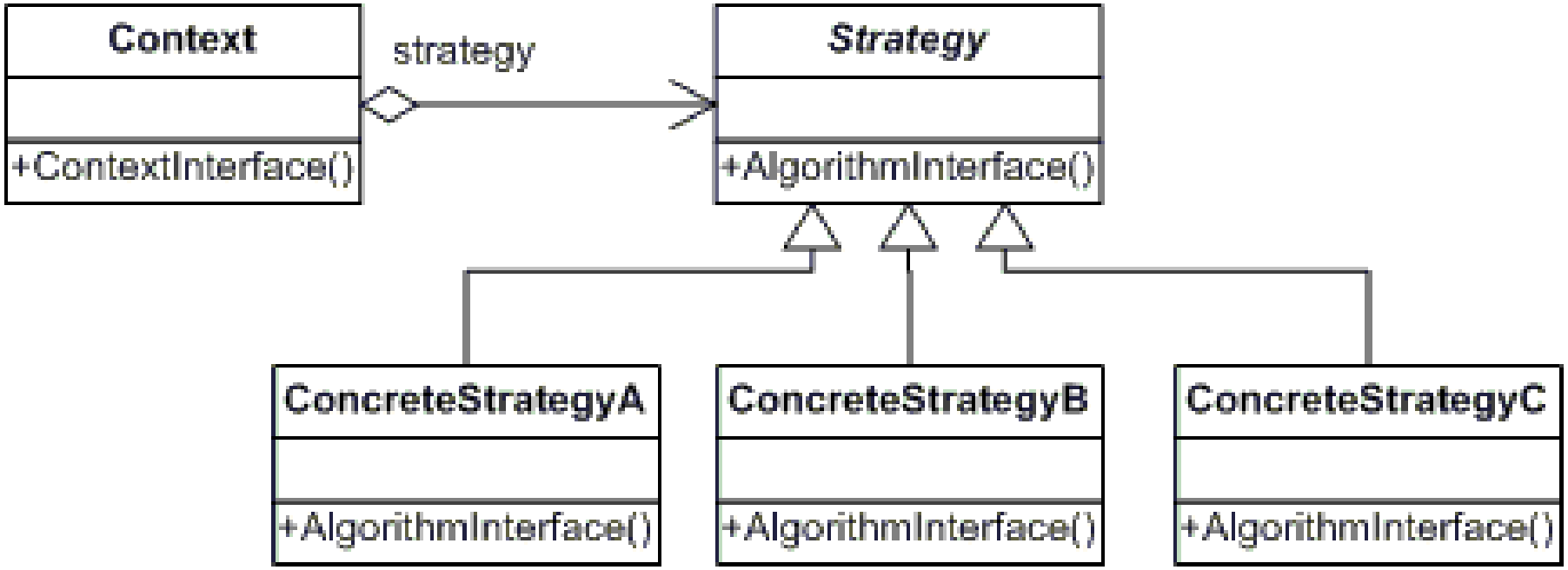

Client
Invoker
Command
+Execute()
Receiver
+Action()
receiver
ConcreteCommand
-state
+Execute()
receiver.Action()

Strategy

- **Назначение:** определяет семейство взаимозаменяемых алгоритмов

- **Применимость:**
  - Возможность динамически менять алгоритм без влияния на клиента
  - Параметризация классов алгоритмами
  - Сокрытие реализации алгоритмов

Структура

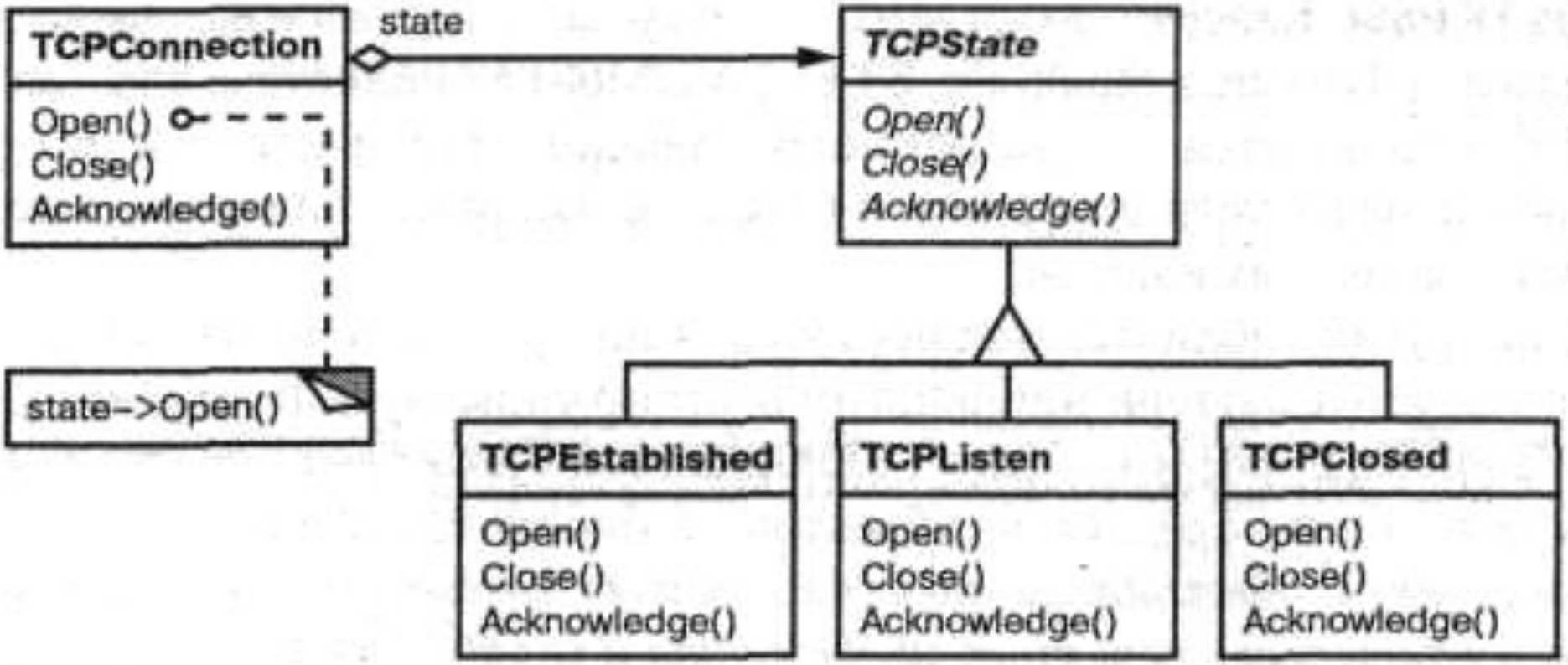

Context
+ContextInterface()
strategy
Strategy
+AlgorithmInterface()
ConcreteStrategyA
+AlgorithmInterface()
ConcreteStrategyB
+AlgorithmInterface()
ConcreteStrategyC
+AlgorithmInterface()

State

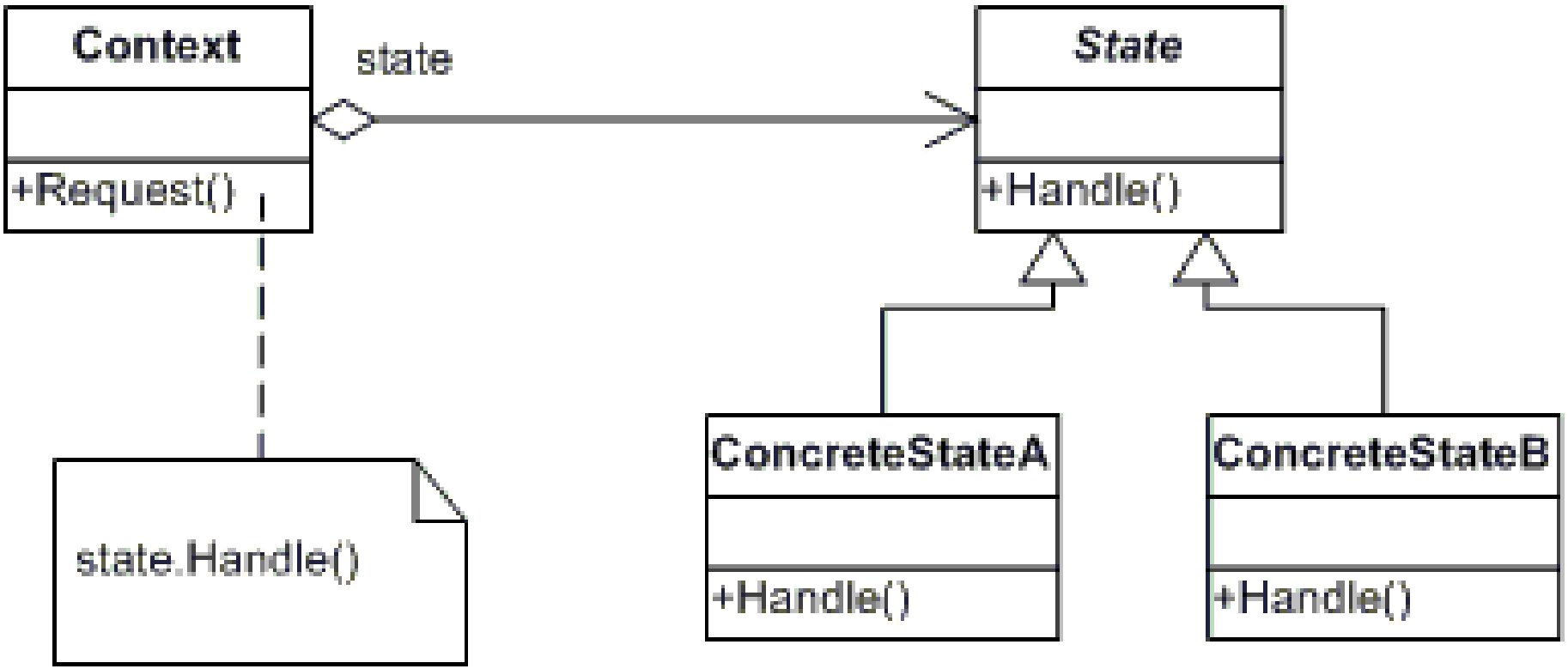
- **Назначение:** позволяет объекту варьировать свое поведение в зависимости от внутреннего состояния

- **Применимость:**
  - Поведение объекта меняется при смене его состояния
  - Замена множества условных операторов, связанных с состоянием объекта

Мотивация


TCPConnection
Open()
Close()
Acknowledge()
state
TCPState
Open()
Close()
Acknowledge()
state->Open()
TCPEstablished
Open()
Close()
Acknowledge()
TCPListen
Open()
Close()
Acknowledge()
TCPClosed
Open()
Close()
Acknowledge()

Структура

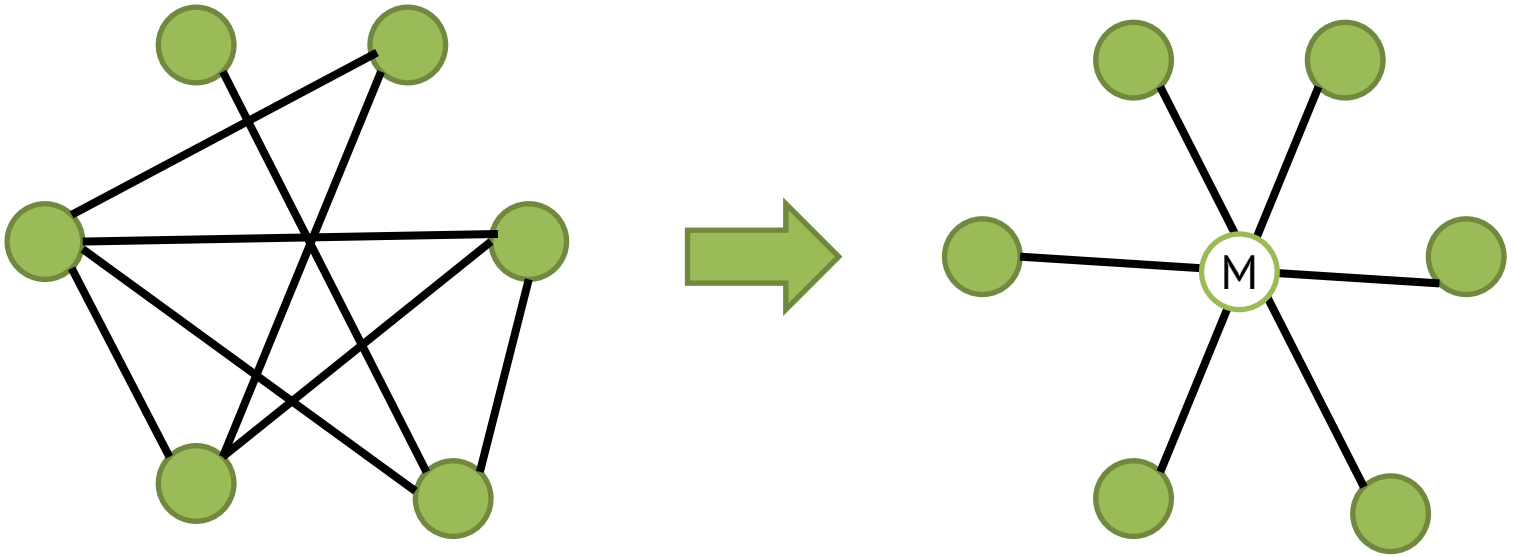

Context
state
State
+Request()
+Handle()
state.Handle()
ConcreteStateA
+Handle()
ConcreteStateB
+Handle()

Mediator – Посредник

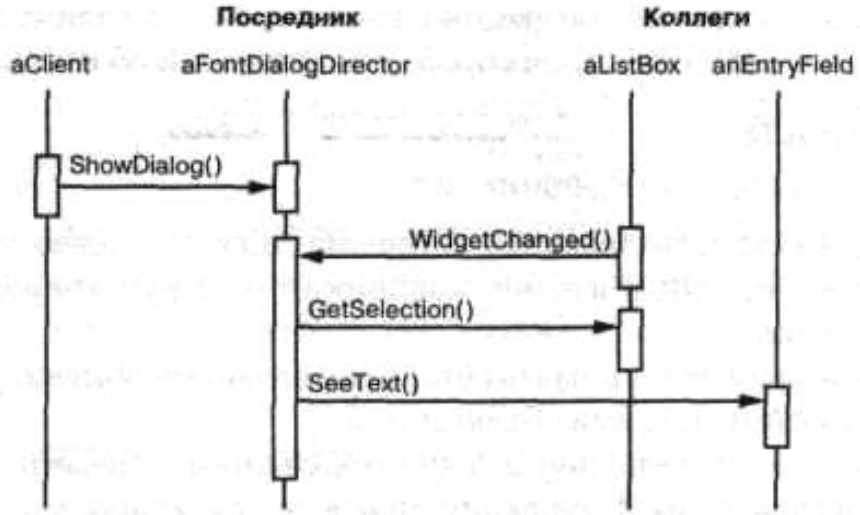
- **Назначение:** инкапсулировать способ взаимодействия между объектами в рамках специального объекта-посредника

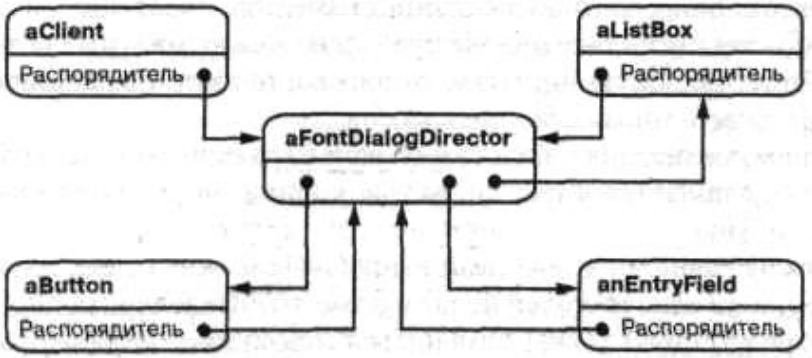
- **Применимость:**
  - Для уменьшения связности объектов в системе
  - Маршрутизация вызовов конкретных объектов

Мотивация

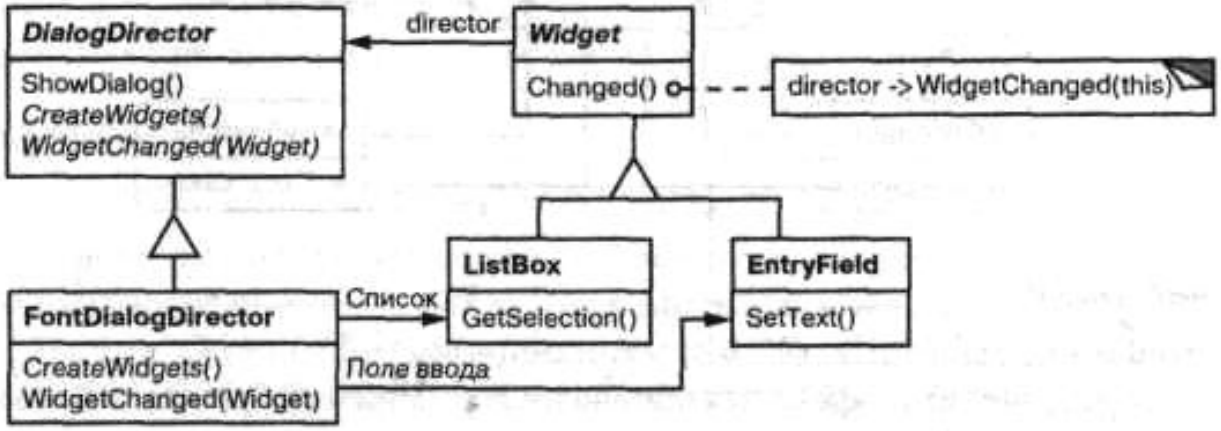


Мотивация

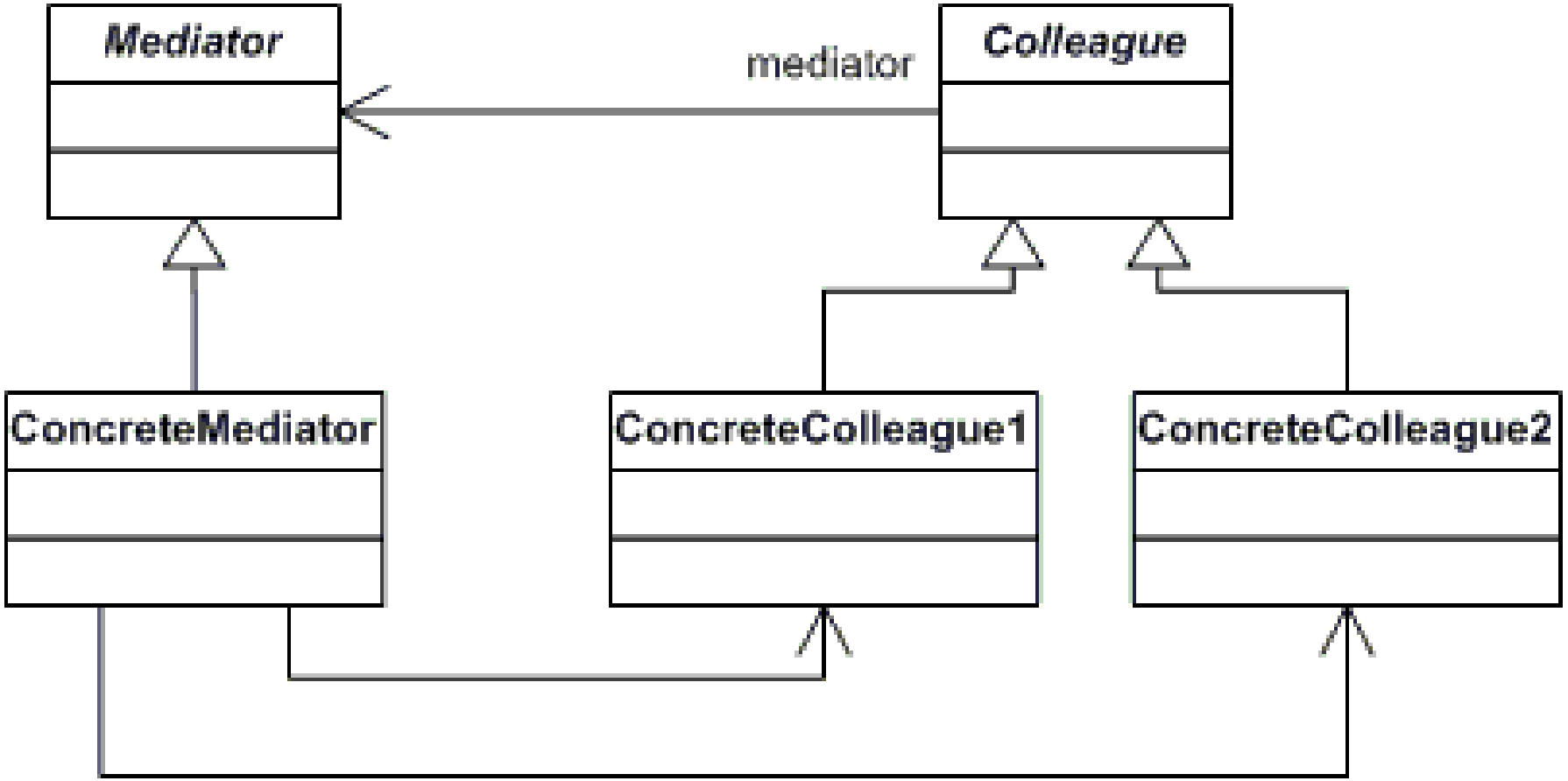

Посредник
Коллеги
aClient
aFontDialogDirector
aListBox
anEntryField
ShowDialog()
WidgetChanged()
GetSelection()
SeeText()


aClient
Распорядитель
aListBox
Распорядитель
aFontDialogDirector
aButton
Распорядитель
anEntryField
Распорядитель


The quick brown fox...
Family
new century schoolbook
avant garde
chicago
courier
helvetica
new century schoolbook
palatino
times roman
zapf dingbats
Weight
medium
bold
demibold
Slant
roman
italic
oblique
Size
34 pt
condensed
Cancel
OK


DialogDirector
ShowDialog()
CreateWidgets()
WidgetChanged(Widget)
director
Widget
Changed()
director -> WidgetChanged(this)
FontDialogDirector
CreateWidgets()
WidgetChanged(Widget)
Список
Поле ввода
ListBox
GetSelection()
EntryField
SetText()

Структура

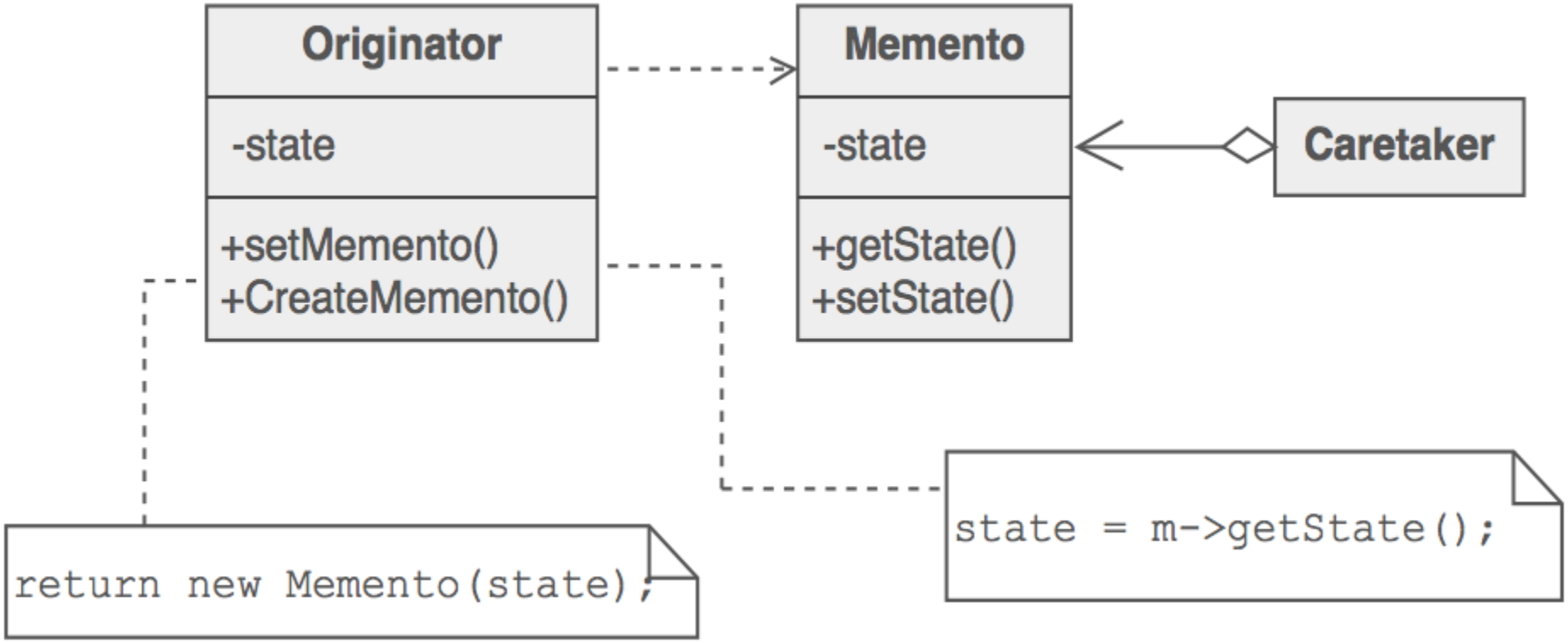

Mediator
mediator
Colleague
ConcreteMediator
ConcreteColleague1
ConcreteColleague2

Memento – Хранитель

- **Назначение:** фиксирует и выносит за пределы объекта его внутреннее состояние так, чтобы позже его можно было восстановить

- **Применимость:**
  - Скрытие информации о реализации
  - Сохранение мгновенного снимка состояния объекта (snapshot)

Структура

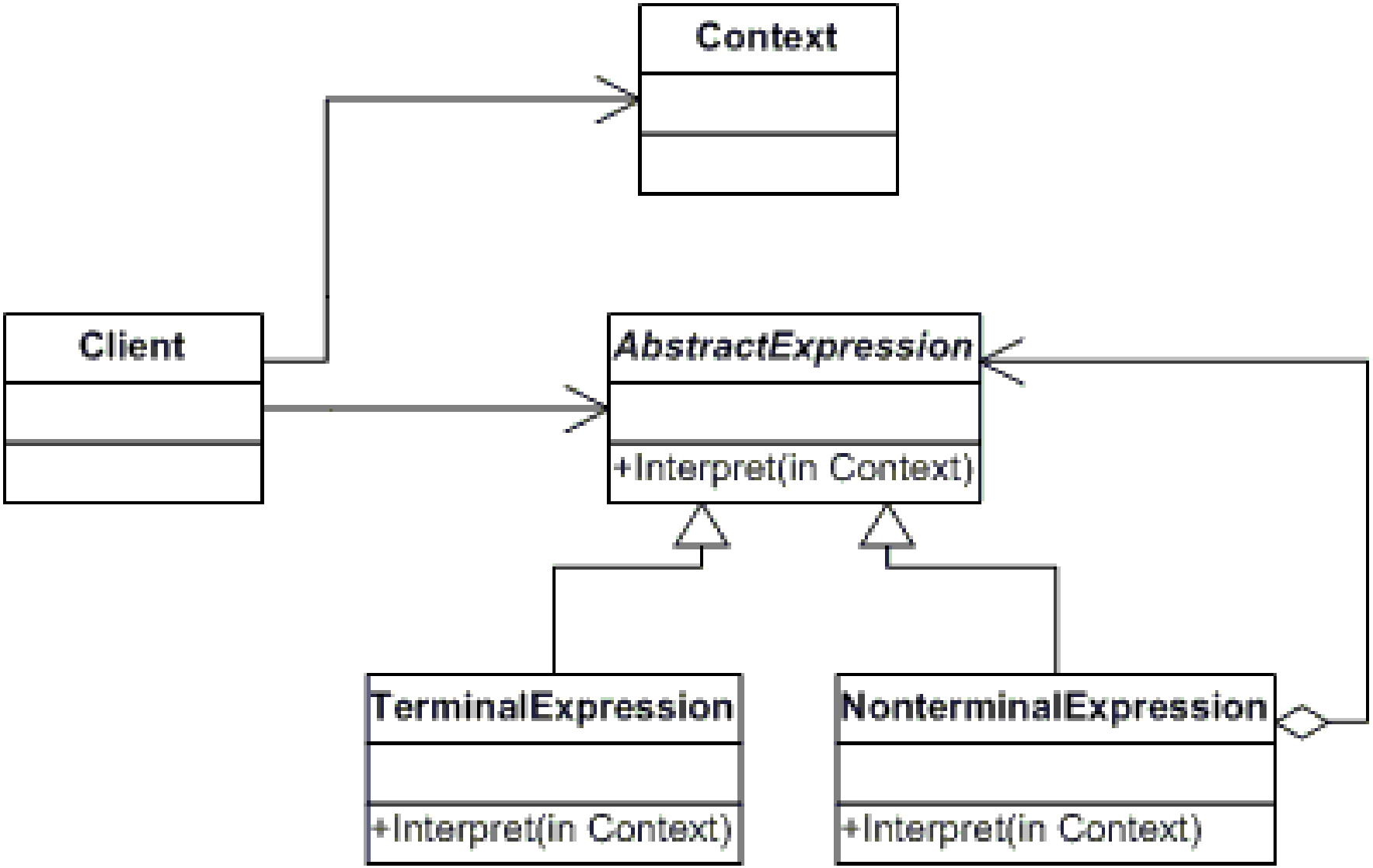

Originator
-state
+setMemento()
+CreateMemento()
Memento
-state
+getState()
+setState()
Caretaker
return new Memento(state);
state = m->getState();

Interpreter

- **Назначение:** для языка определяет представление его грамматики, а также интерпретатор предложений языка

- **Применимость:**
  - Язык
  - Грамматика проста
  - Эффективность не является главным критерием

Структура

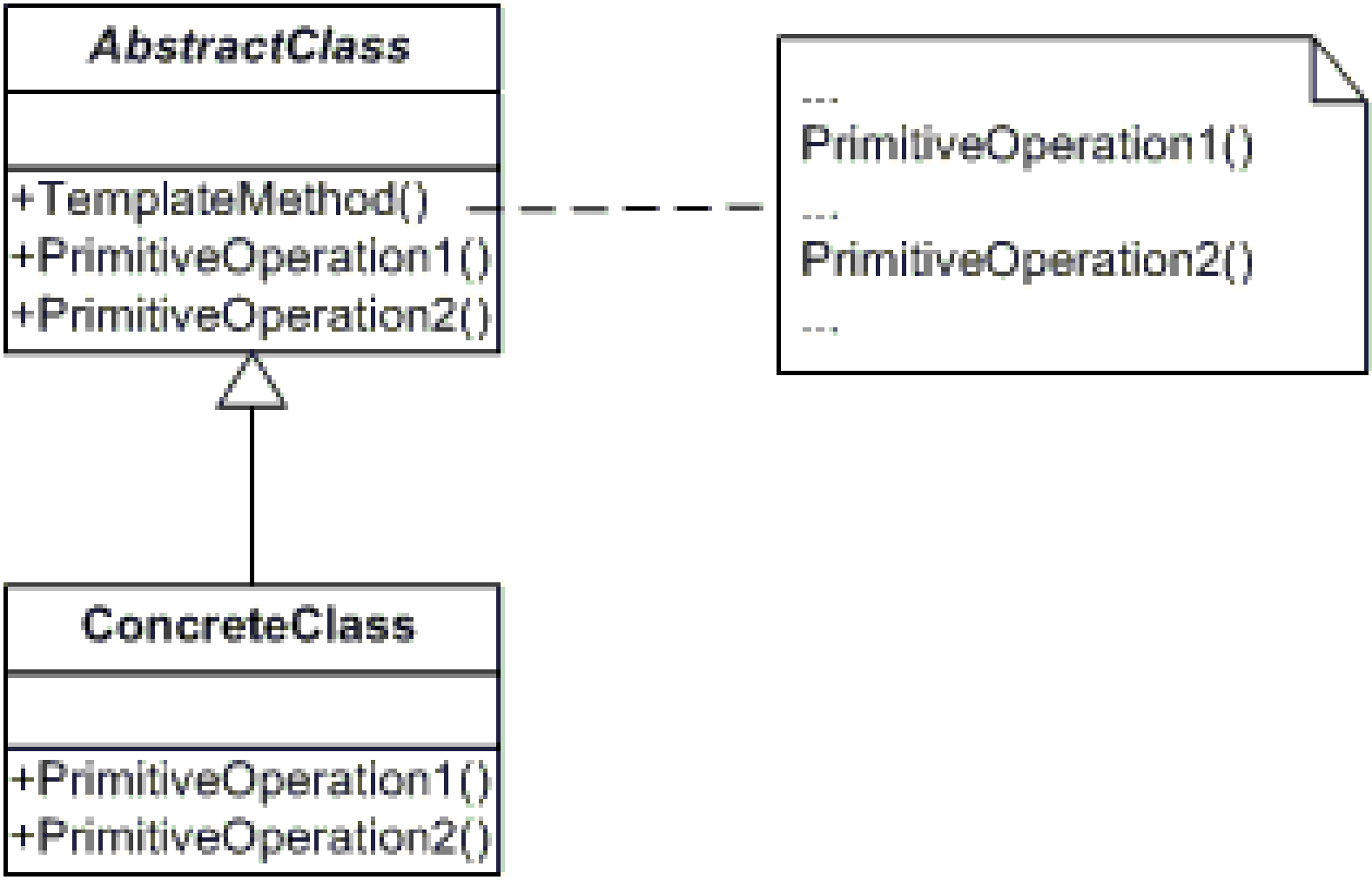

Context
Client
AbstractExpression
+Interpret(in Context)
TerminalExpression
+Interpret(in Context)
NonterminalExpression
+Interpret(in Context)

Template method – Шаблонный метод

- **Назначение:** определяет основу алгоритма и позволяет подклассам переопределять отдельные шаги, не изменяя структуры в целом

- **Применимость:**
  - Вынесение поведения, общего для всех подклассов
  - Управление расширением подклассов

Структура

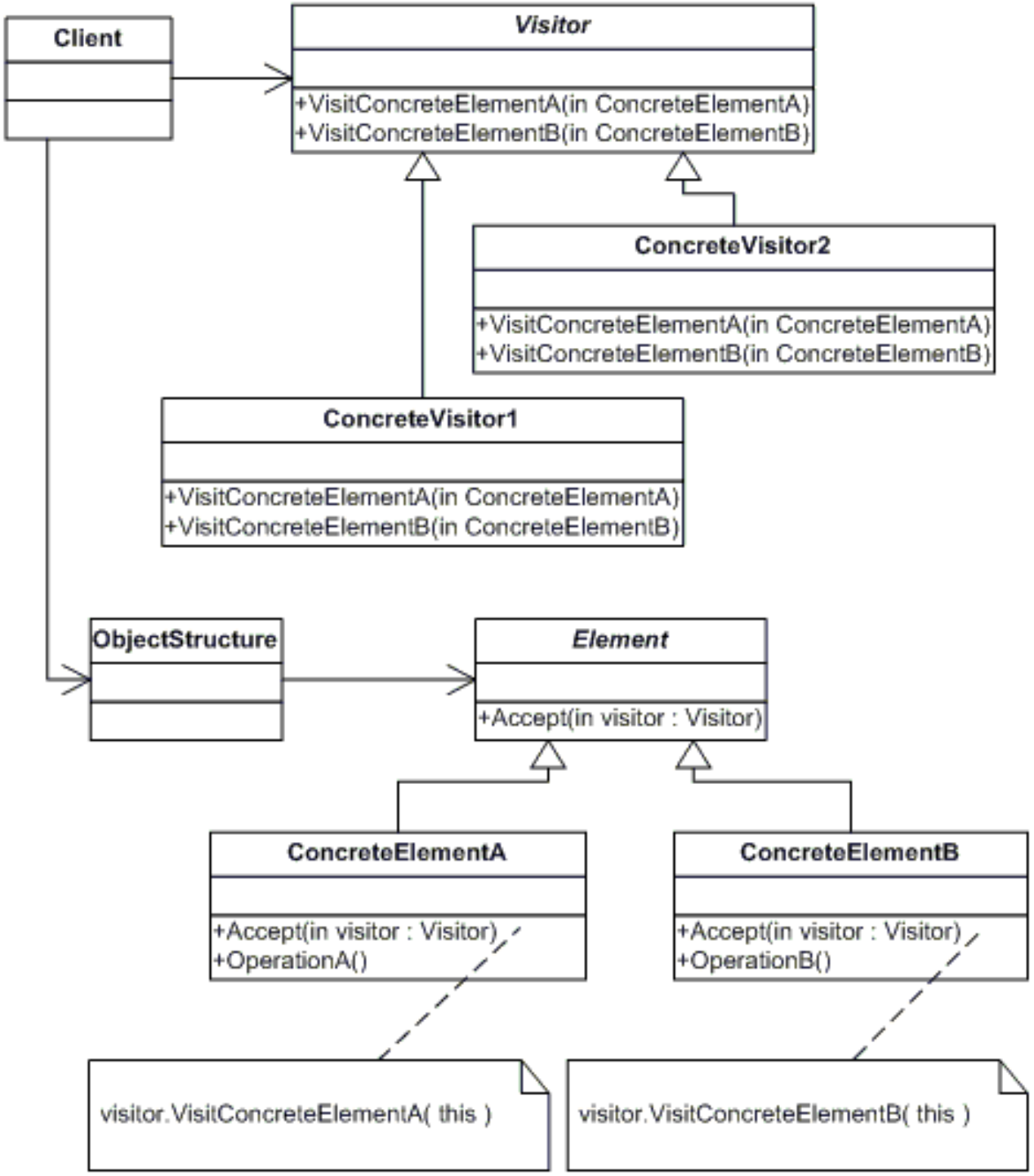

AbstractClass
+TemplateMethod()
+PrimitiveOperation1()
+PrimitiveOperation2()
...
PrimitiveOperation1()
...
PrimitiveOperation2()
...
ConcreteClass
+PrimitiveOperation1()
+PrimitiveOperation2()

Visitor

- **Назначение:** описывает операцию, выполняемую с каждым объектом из некоторой структуры

- **Применимость:**
  - Выполнение операций надо объектами различных типов
  - Вынесение логики операции из всех объектов структуры в отдельный объект-посетитель
  - Грамматика проста
  - Эффективность не является главным критерием

Структура


Client
Visitor
+VisitConcreteElementA(in ConcreteElementA)
+VisitConcreteElementB(in ConcreteElementB)
ConcreteVisitor2
+VisitConcreteElementA(in ConcreteElementA)
+VisitConcreteElementB(in ConcreteElementB)
ConcreteVisitor1
+VisitConcreteElementA(in ConcreteElementA)
+VisitConcreteElementB(in ConcreteElementB)
ObjectStructure
Element
+Accept(in visitor : Visitor)
ConcreteElementA
+Accept(in visitor : Visitor)
+OperationA()
ConcreteElementB
+Accept(in visitor : Visitor)
+OperationB()
visitor.VisitConcreteElementA(this)
visitor.VisitConcreteElementB(this)

Источники

- **Книги**
  - Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. «Приемы объектно-ориентированного проектирования. Паттерны проектирования» Питер-ДМК, 2001.
  - Эрик Фримен, Элизабет Фримен, Кэтти Сьерра, Берт Бейтс. «Паттерны проектирования», Питер, 2011.
- **Курсы**
  - http://sourcemaking.com/design_patterns
  - Курс Андрея Бреслава «Software Design» https://sites.google.com/site/abreslav2/softwaredesign2
- **Полезные статьи**
  - http://wwwswt.informatik.uni-rostock.de/deutsch/Lehre/Uebung/Beispiele/PatternExamples/patexamples.htm
- **Справочники**
  - http://en.wikipedia.org/wiki/Software_design_pattern
  - http://www.dofactory.com/net/design-patterns
  - http://sourcemaking.com/design_patterns
- **Примеры**
  - Examples of GoF Design Patterns in Java SE http://stackoverflow.com/questions/1673841/examples-of-gof-design-patterns
  - Discover the Design Patterns You're Already Using in the .NET Framework http://msdn.microsoft.com/en-us/magazine/cc188707.aspx
- **Другое**
  - Cheat Sheet http://www.mcdonaldland.info/2007/11/28/40/