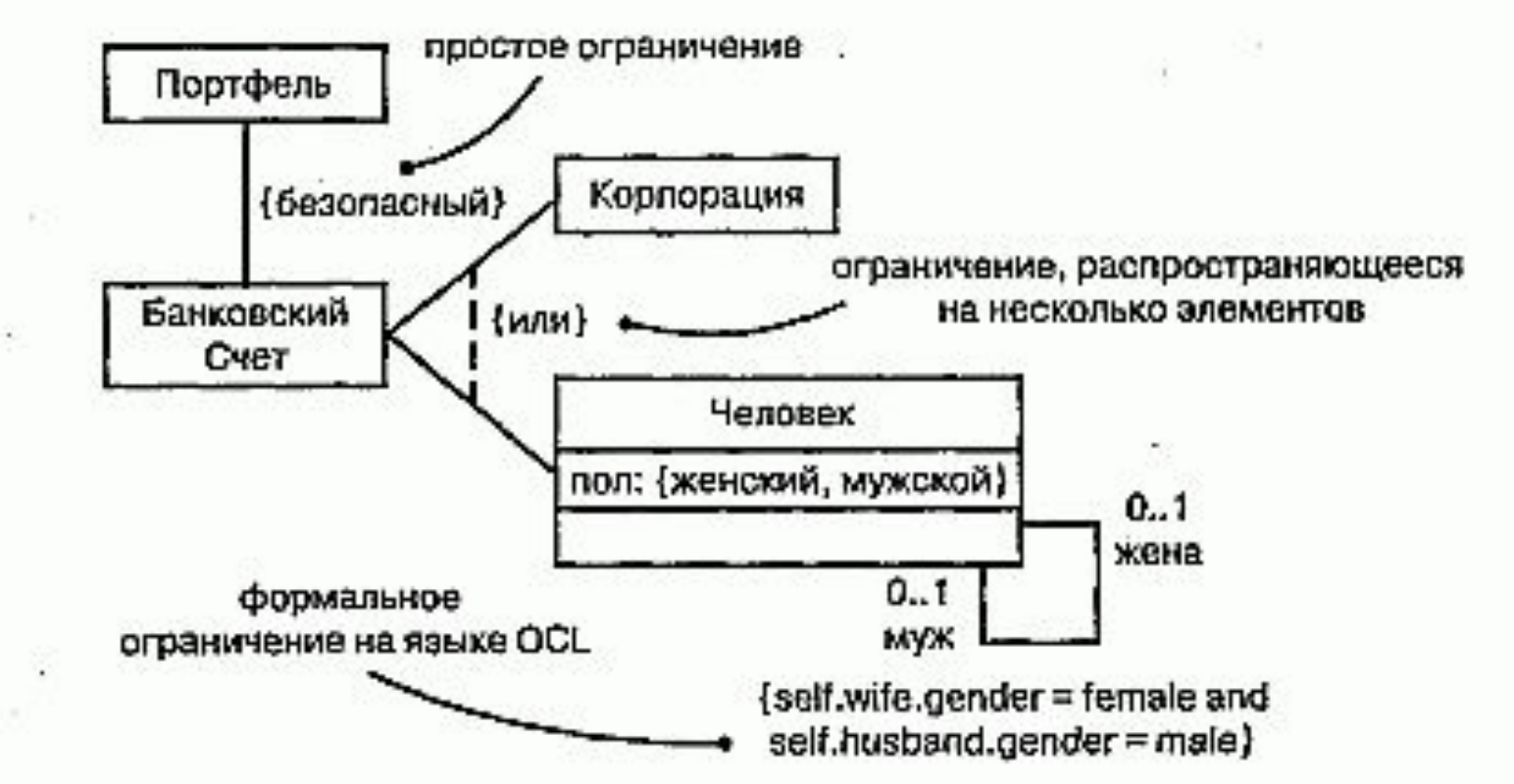
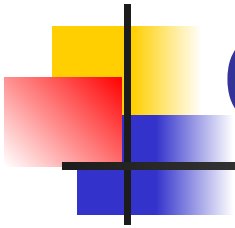
Информационные технологии

- Диаграммы классов (продолжение)

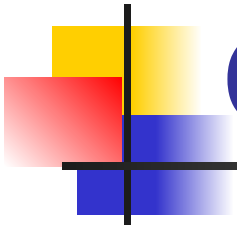
Основы структурного моделирования





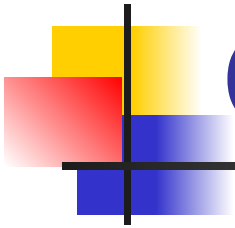
Ограничения

- Используются в языке Eiffel (Design by contract, Бертран Мейер)
- В основе лежит понятие утверждения: булевское высказывание, которое всегда истинно
- Предусловия, условия и инвариант



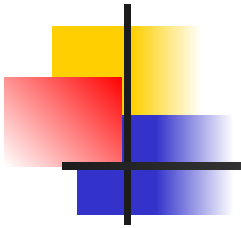
Ограничения

- Пусть A – это некоторая операция, тогда формула корректности (correctness formula)
- {P} A {Q} (Триада Хоара)
- {x = 5} x = x ^ 2 {x > 0}

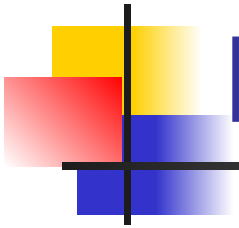


Ограничения

- Кто ответственен за выполнение проверки?
- Для предусловия ответственен вызывающий класс



- Самая трудная задача в объектно-ориентированном проектировании – разложить систему на объекты
- Можно сформулировать задачу письменно, выделить из получившейся фразы существительные и глаголы, после чего создать соответствующие классы и операции.



Моделирование

- Другой путь – сосредоточиться на отношениях и разделении обязанностей в системе.
- Согласие по поводу того, какой подход самый лучший, никогда не будет достигнуто. (GoF)

CRC – карточки

- Уорд Каннингхем и Кент Бек (разработчики Smalltalk) в конце 80-х, Удобны при построении диаграмм взаимодействия
- CRC: Class-Responsibility-Collaboration (Класс- Ответсвенность- Кооперация)
- Технология использовалась для проектирования модели классов

CRC карточки

ИМЯ КЛАССА	
ОТВЕТСТВЕННОСТЬ	КООПЕРАЦИЯ

CRC – карточки

- Небольшие карточки, размером 4 x 6см

Заказ	
Проверить наличие товара	Строка заказа
Определить цену	
Проверить факт оплаты	Клиент
Отправить по адресу доставки	

CRC карточки

Диаграммы классов используются:

диаграммы классов используются в следующих целях:

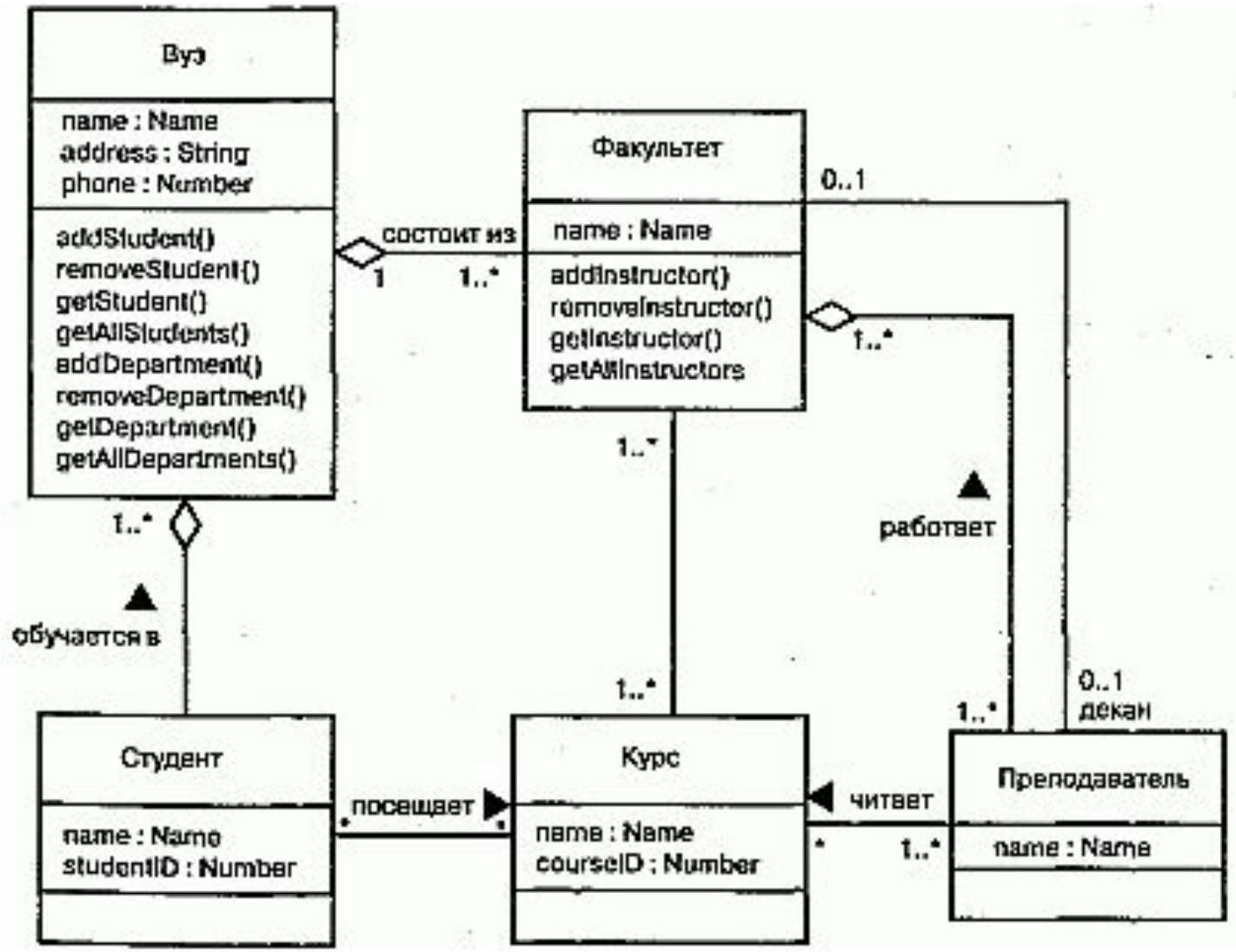
- для *моделирования словаря* системы
- для *моделирования простых кооперацій*
- для *моделирования логической схемы базы данных*.

Диаграммы классов используются:

Моделирование словаря системы:

- Определите, какие элементы пользователи и разработчики применяют для описания задачи или ее решения. Используйте CRC-карточки.
- Выявите для каждой абстракции соответствующее ей множество обязанностей.
- Разработайте атрибуты и операции, необходимые для выполнения класса ими своих обязанностей.

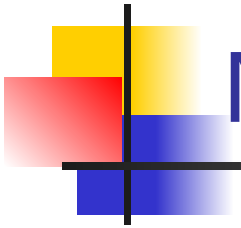
Моделирование схемы БД



Моделирование схемы БД

- используйте зависимость, только если моделируемое отношение не является структурным;
- используйте обобщение, только если имеет место отношение типа "является";
- множественное наследование часто можно заменить агрегированием;
- остерегайтесь циклических отношений обобщения;

Моделирование схемы БД

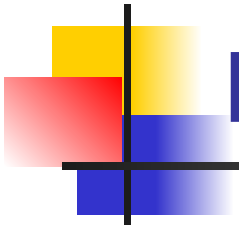


- поддерживайте баланс в отношениях обобщения: иерархия наследования не должна быть ни слишком глубокой (желательно не более пяти уровней), ни слишком широкой (лучше прибегнуть к промежуточным абстрактным классам);
- применяйте ассоциации прежде всего там, где между объектами существуют структурные отношения.

Моделирование схемы БД

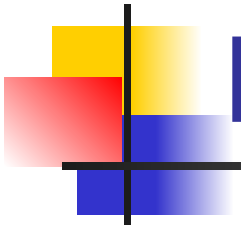
- выбрав один из стилей оформления линий (прямые или наклонные), в дальнейшем старайтесь его придерживаться. Прямые линии подчеркивают, что соединения идут от родственных сущностей к одному общему родителю. Наклонные линии позволяют существенно сэкономить пространство в сложных диаграммах. Если вы хотите привлечь внимание к разным группам отношений, применяйте одновременно оба типа линий;

Моделирование схемы БД



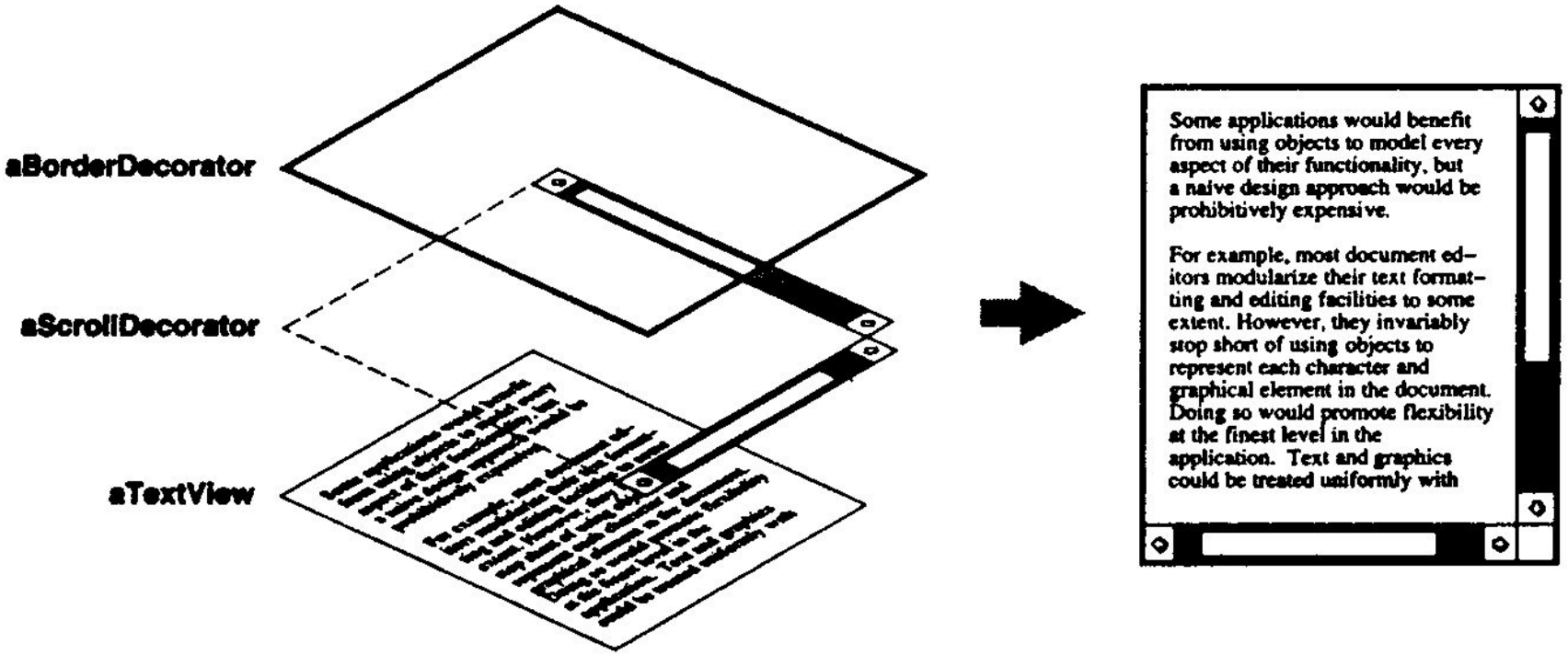
- избегайте пересечения линий;
- показывайте только такие отношения, которые необходимы для понимания особенностей группирования элементов модели; скрывайте несущественные (особенно избыточные) ассоциации.

Шаблоны: Decorator

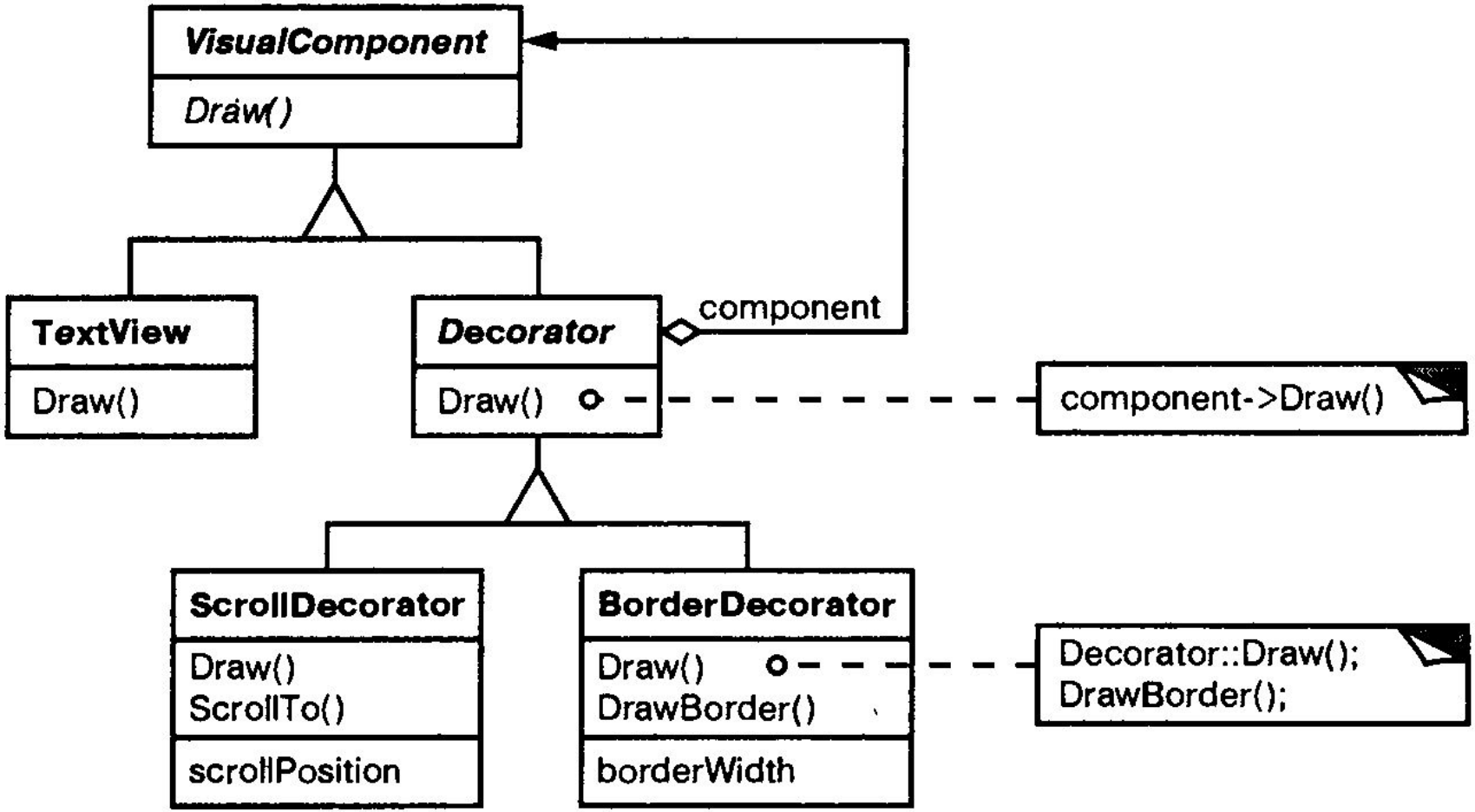


- ***Назначение***: Динамически добавляет объекту новые обязанности. Является гибкой альтернативой порождению подклассов с целью расширения функциональности

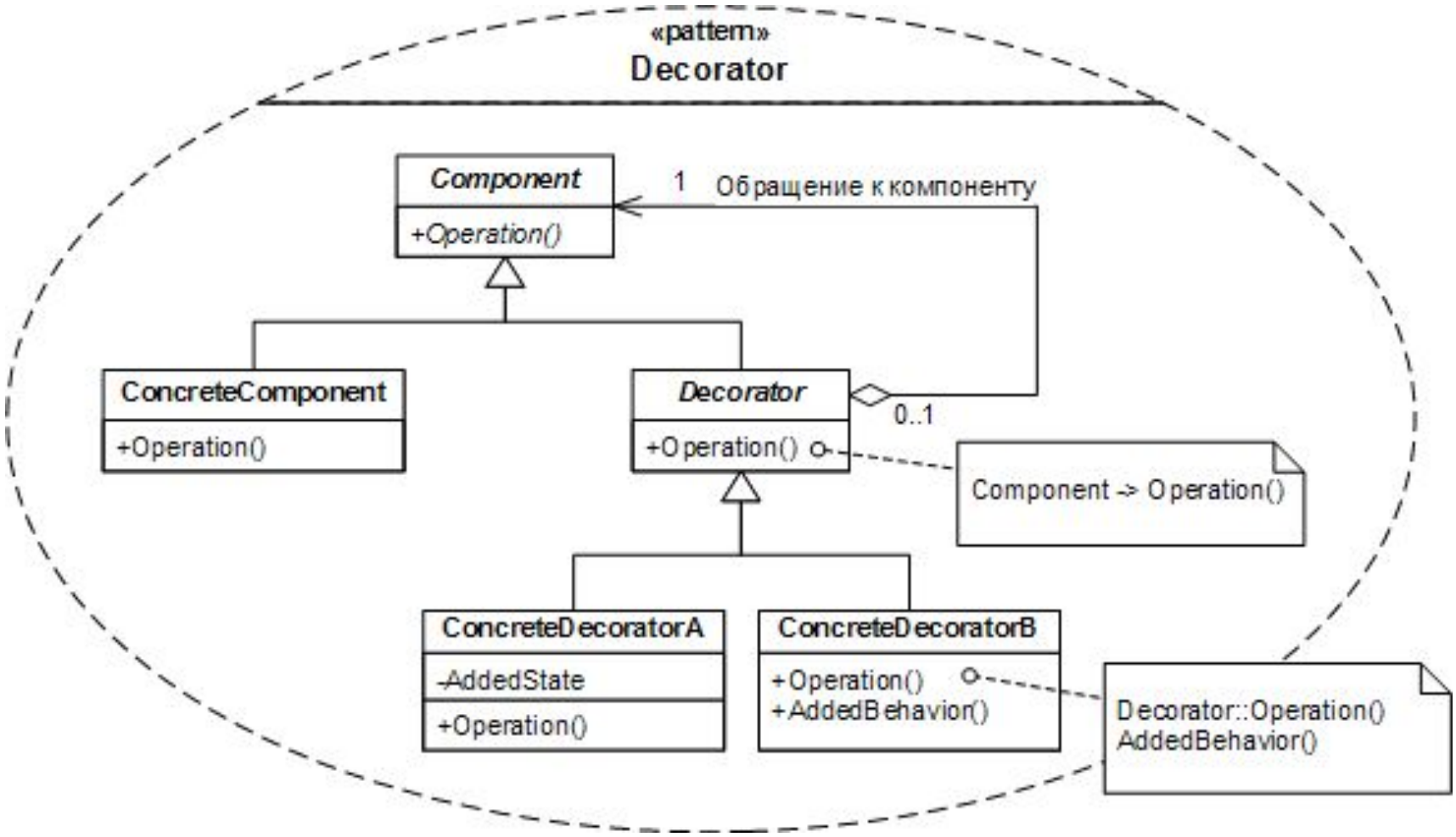
Шаблоны: Decorator

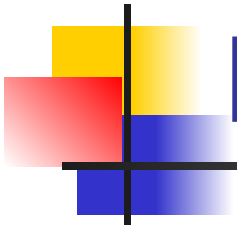


Шаблоны: Decorator



Шаблоны: Decorator



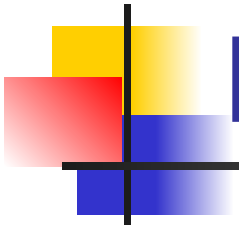


Шаблоны: Decorator

Используйте паттерн:

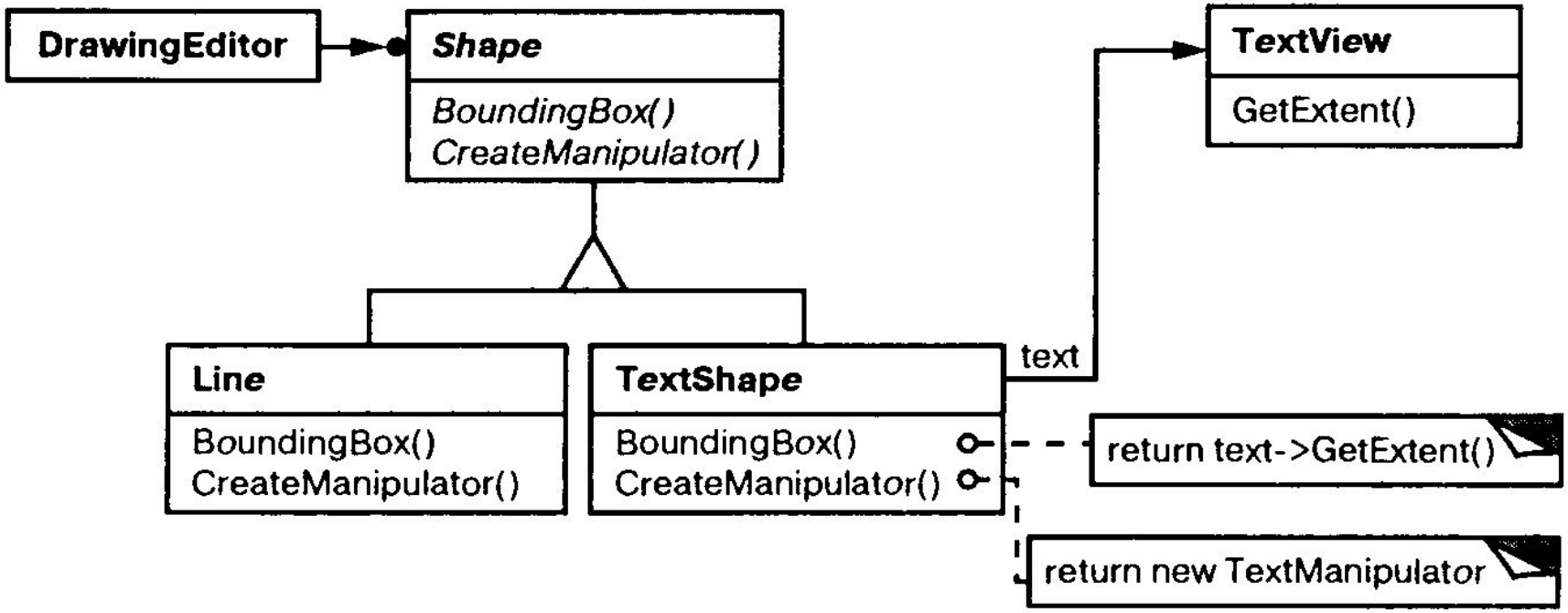
- для динамического, прозрачного для клиентов добавления обязанностей объектам;
- для реализации обязанностей, которые могут быть сняты с объекта;
- когда расширение путем порождения подклассов по каким-то причинам неудобно

Шаблоны: Adapter

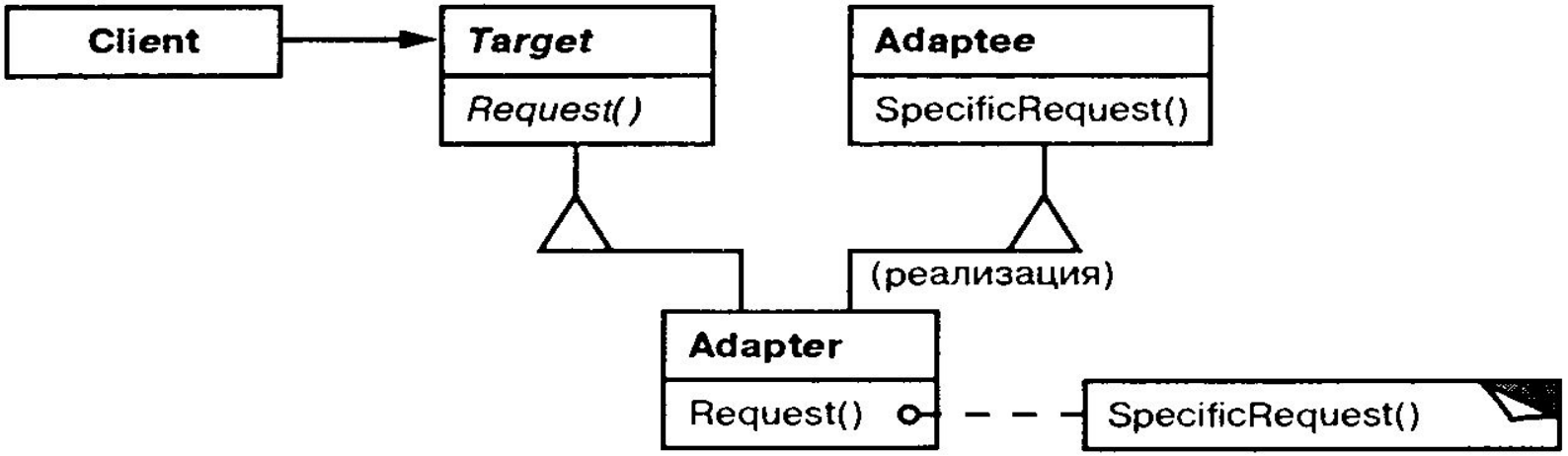


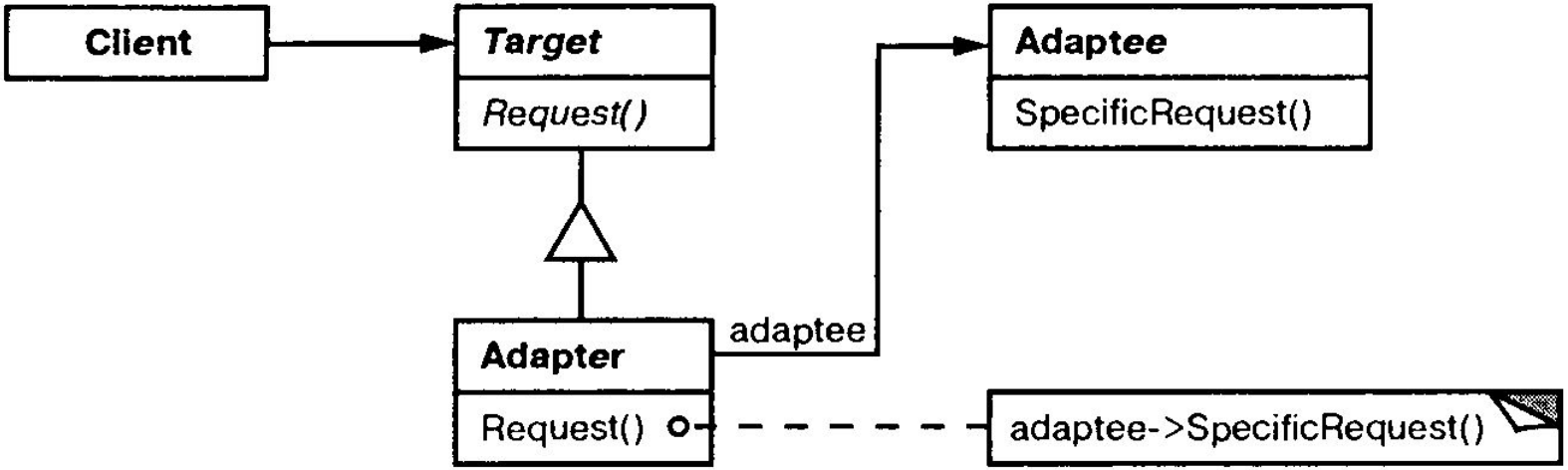
- ***Назначение***: Преобразует интерфейс одного класса в интерфейс другого, который ожидают клиенты

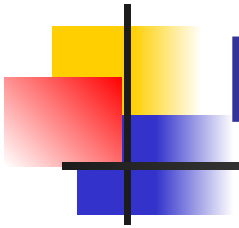
Шаблоны: Adapter



Шаблоны: Adapter





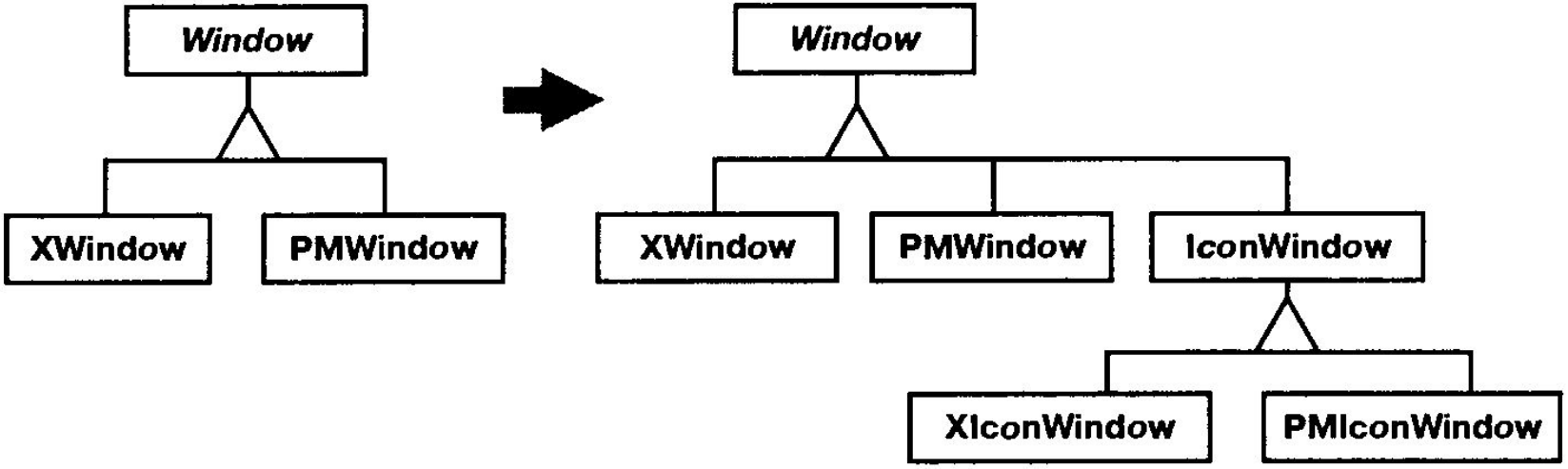


Шаблоны: Bridge

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

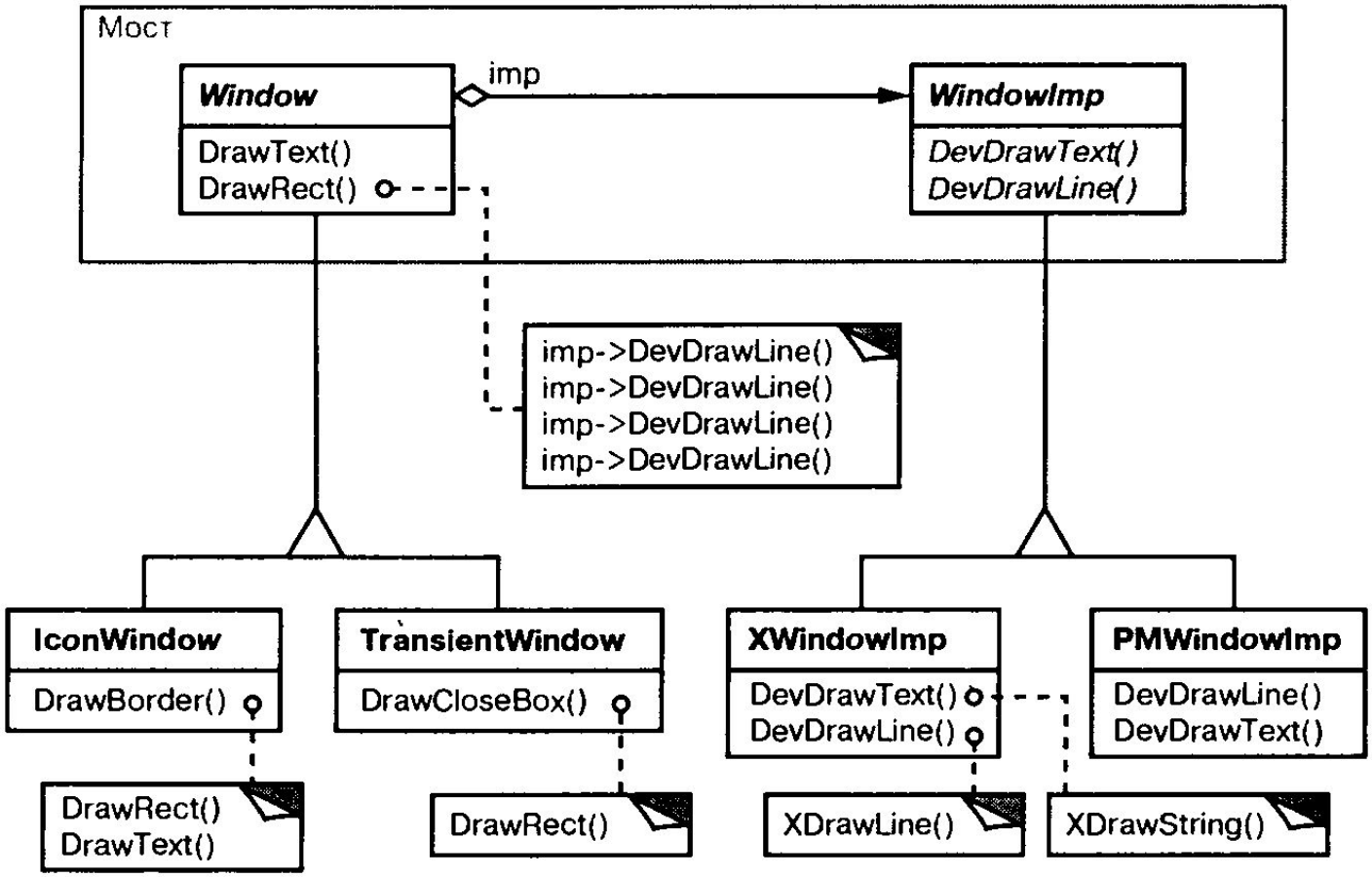
Шаблоны: Bridge

- ***Задача***: 2 среды Xwindows и Presentation Manager (PM) от компании IBM



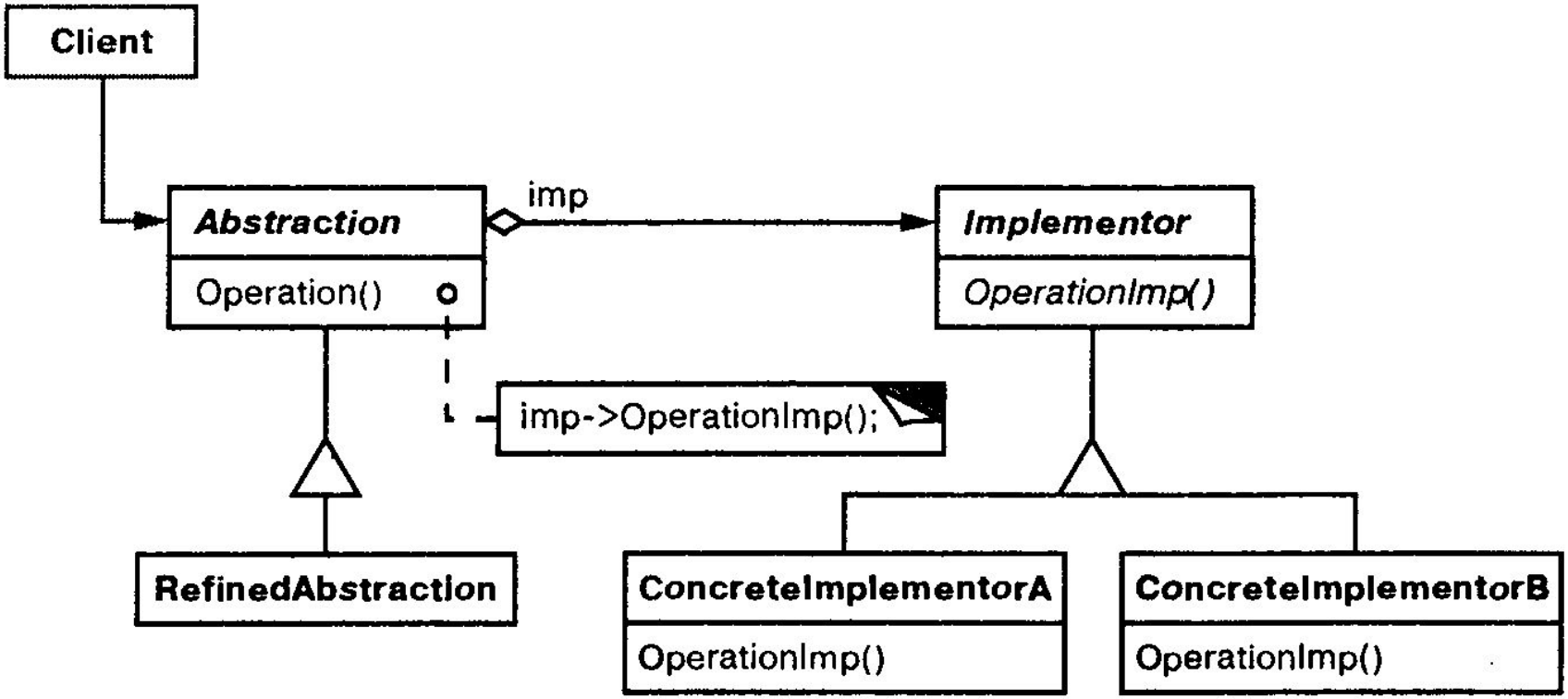
Шаблоны: Bridge

- ***Решение***:



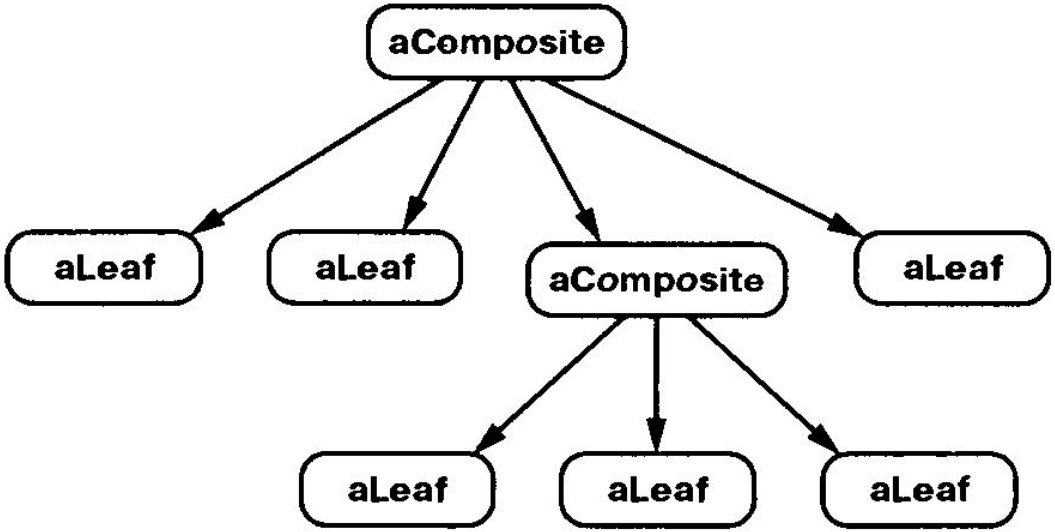
Шаблоны: Bridge

- ***Решение***:

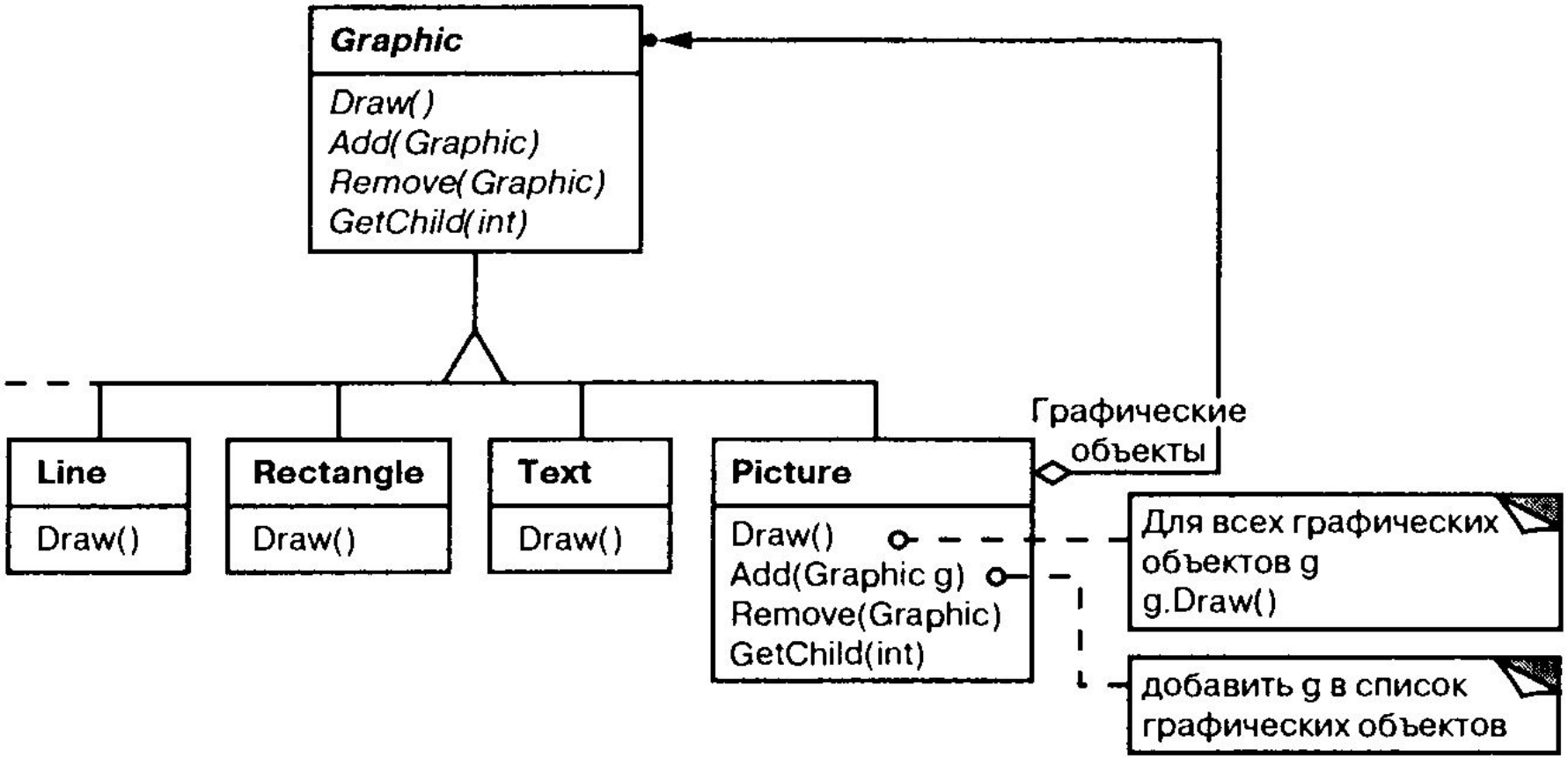


Composite (Компоновщик)

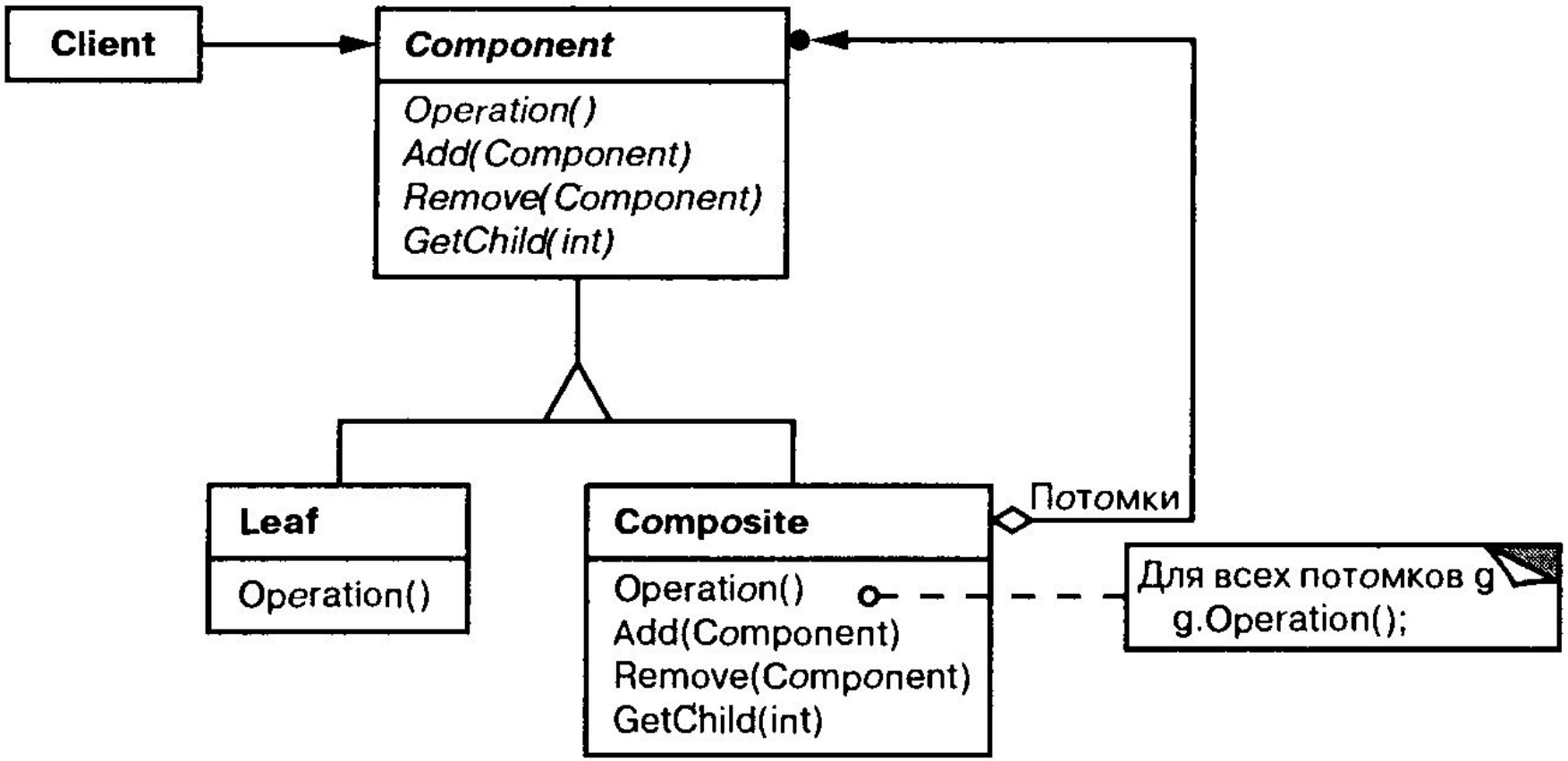
- ***Назначение***: Компонует объекты в древовидные структуры для представления иерархий часть-целое.



Composite (Компоновщик)

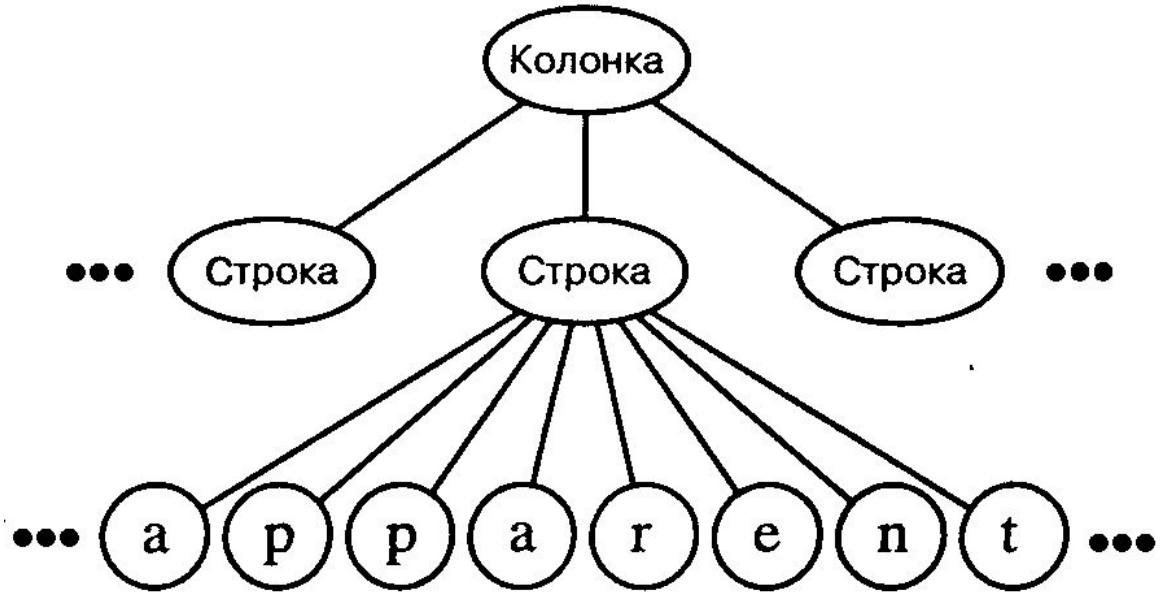


Composite (Компоновщик)

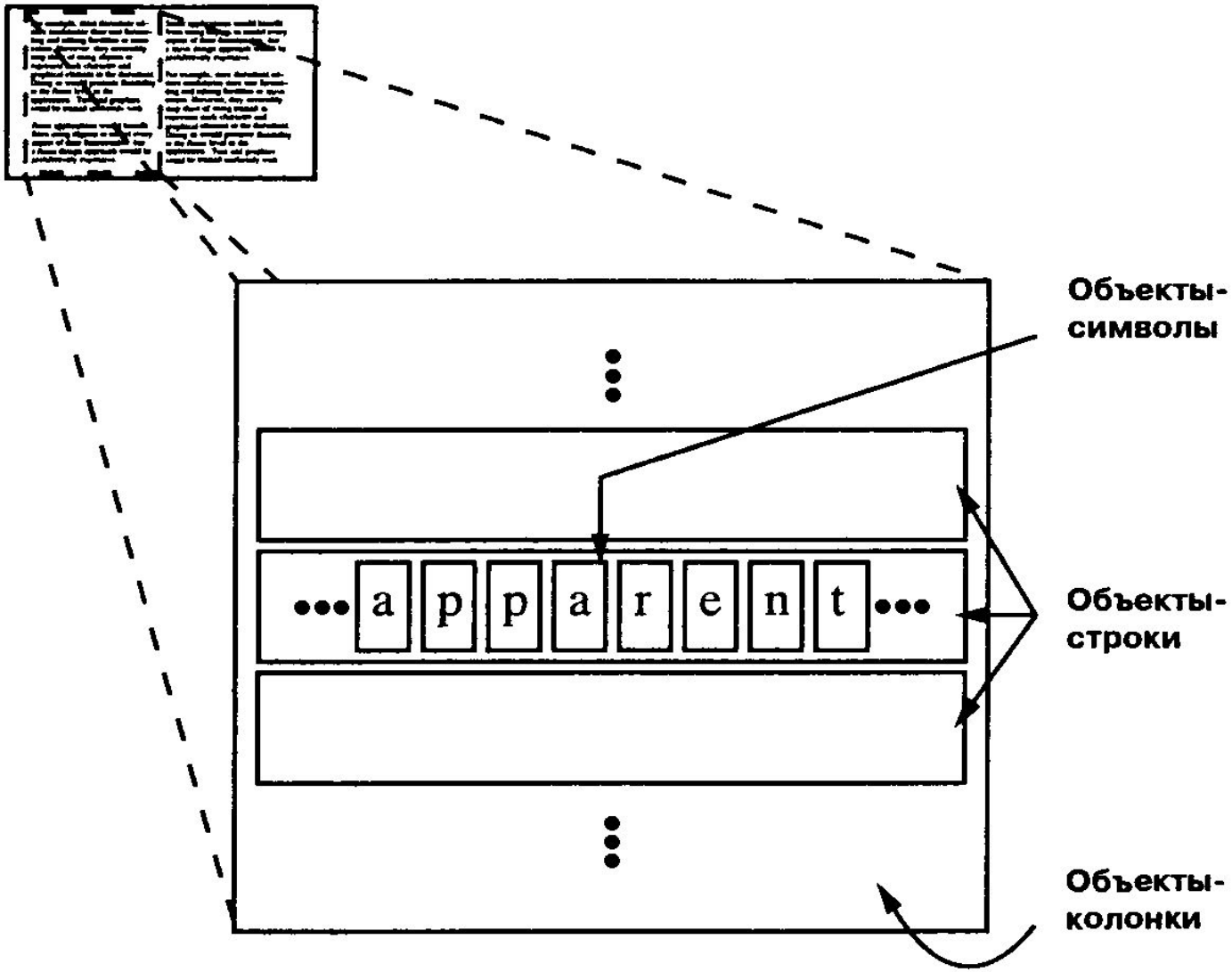


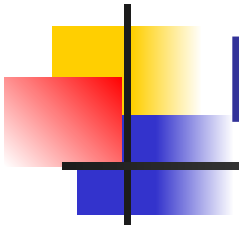
Flyweight (Приспособленец)

- ***Назначение***: Использует разделение для эффективной поддержки множества мелких объектов.



Flyweight (Приспособленец)





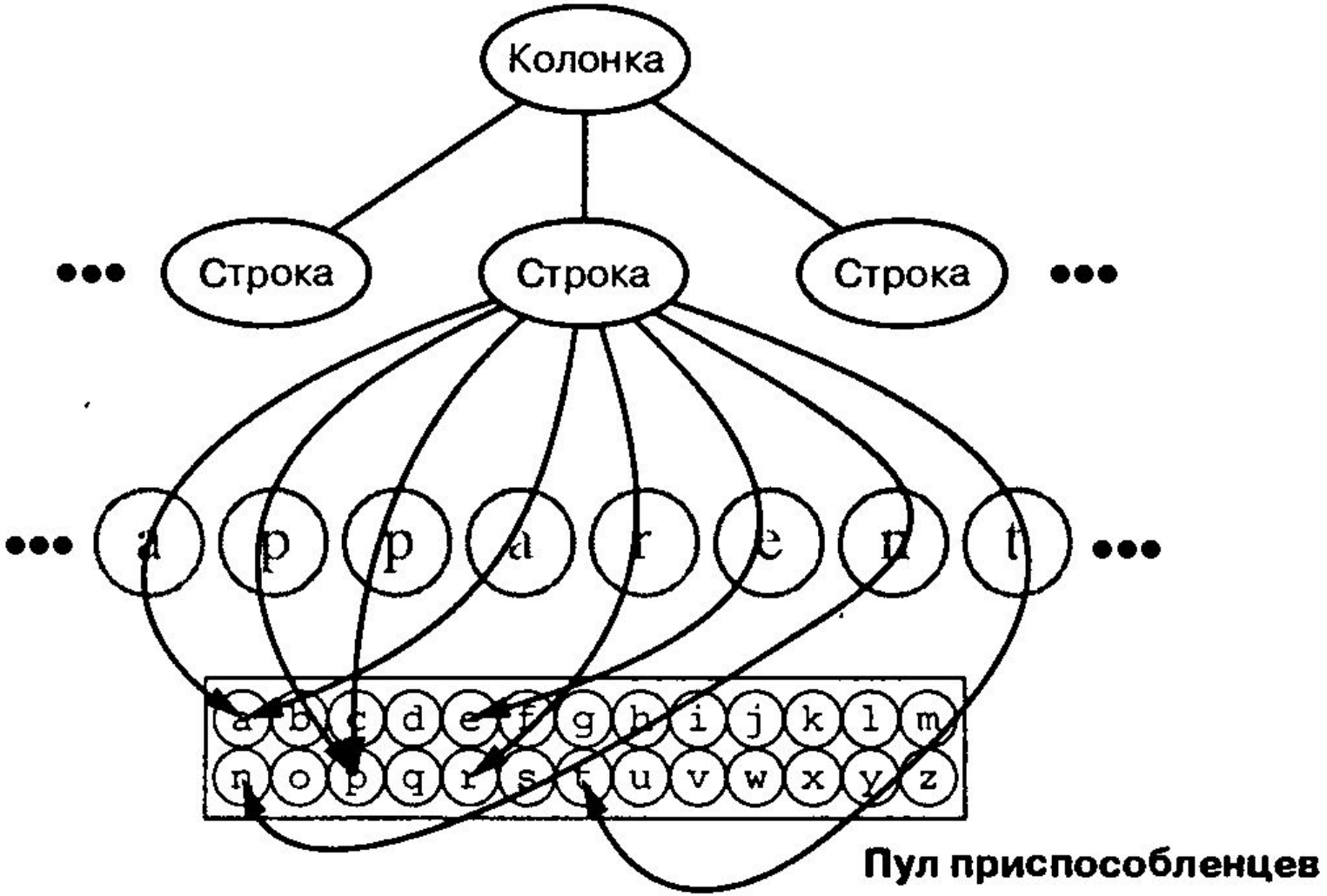
Flyweight (Приспособленец)

Ключевая идея здесь – различие между *внутренним* и *внешним* состояниями

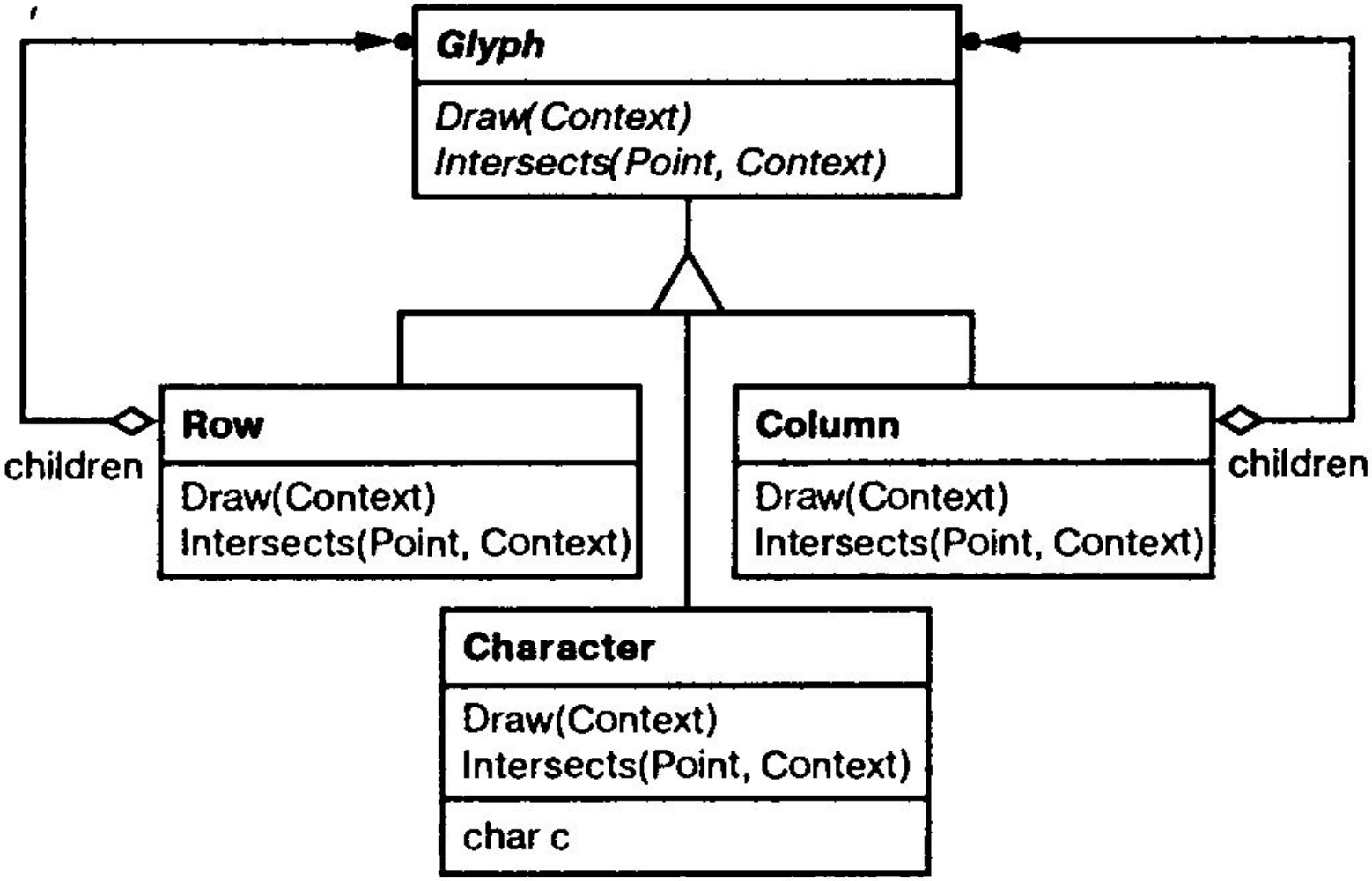
Приспособленец – это разделяемый объект.

Объекты-клиенты отвечают за передачу внешнего состояния приспособленцу, когда в этом возникает необходимость

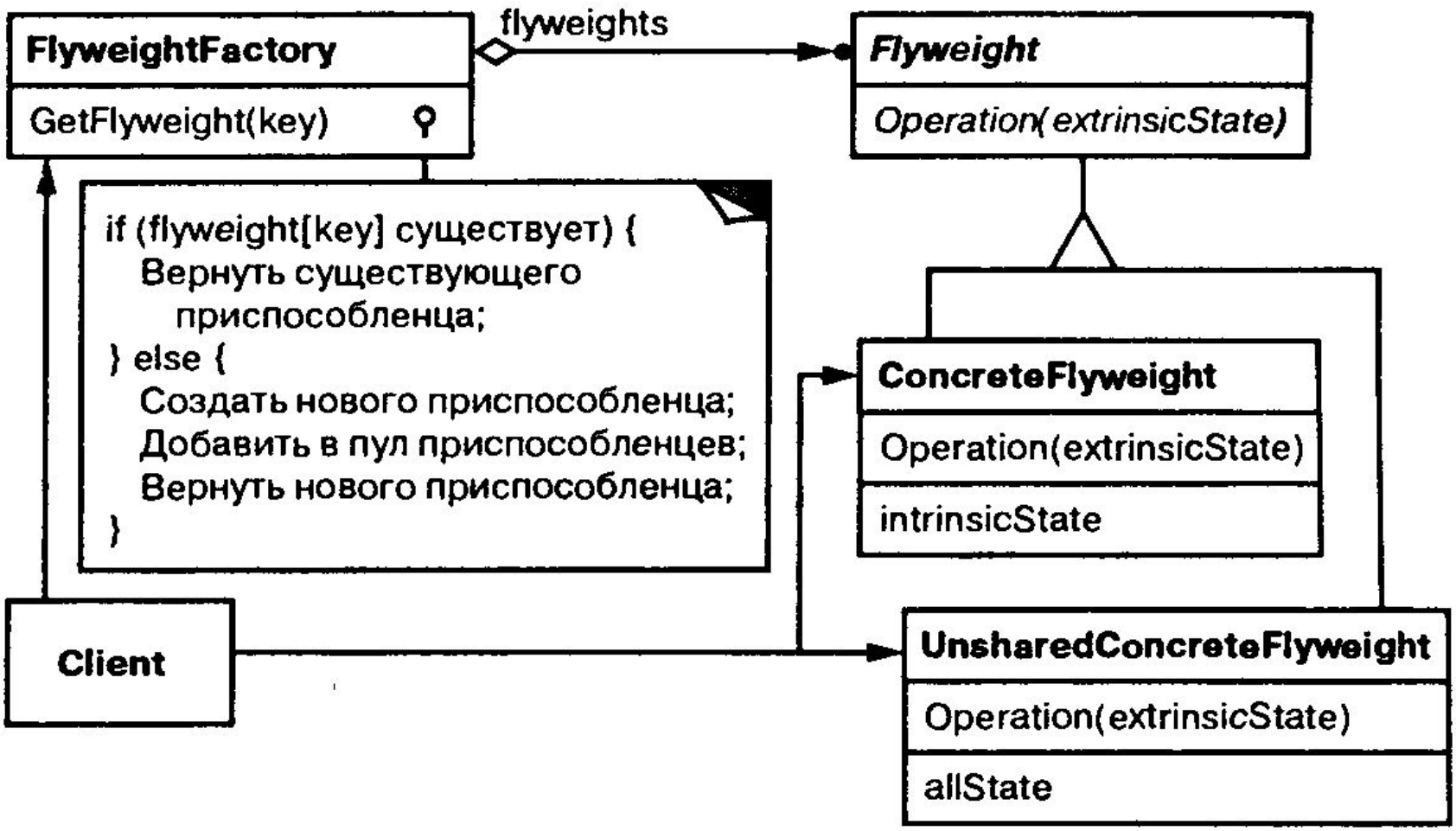
Flyweight (Приспособленец)

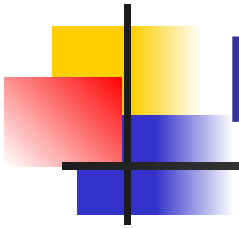


Flyweight (Приспособленец)



Flyweight (Приспособленец)

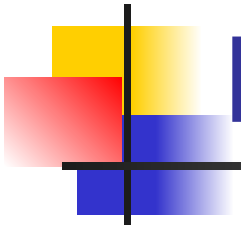




Flyweight (Приспособленец)

состояние, необходимое приспособленцу для нормальной работы, можно охарактеризовать как внутреннее или внешнее. Первое хранится в самом объекте ConcreteFlyweight. Внешнее состояние хранится или вычисляется клиентами

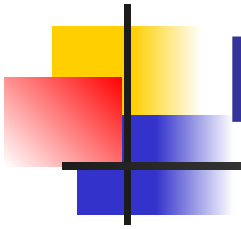
Flyweight (Приспособленец)



При реализации приспособленца следует обратить внимание на следующие вопросы:

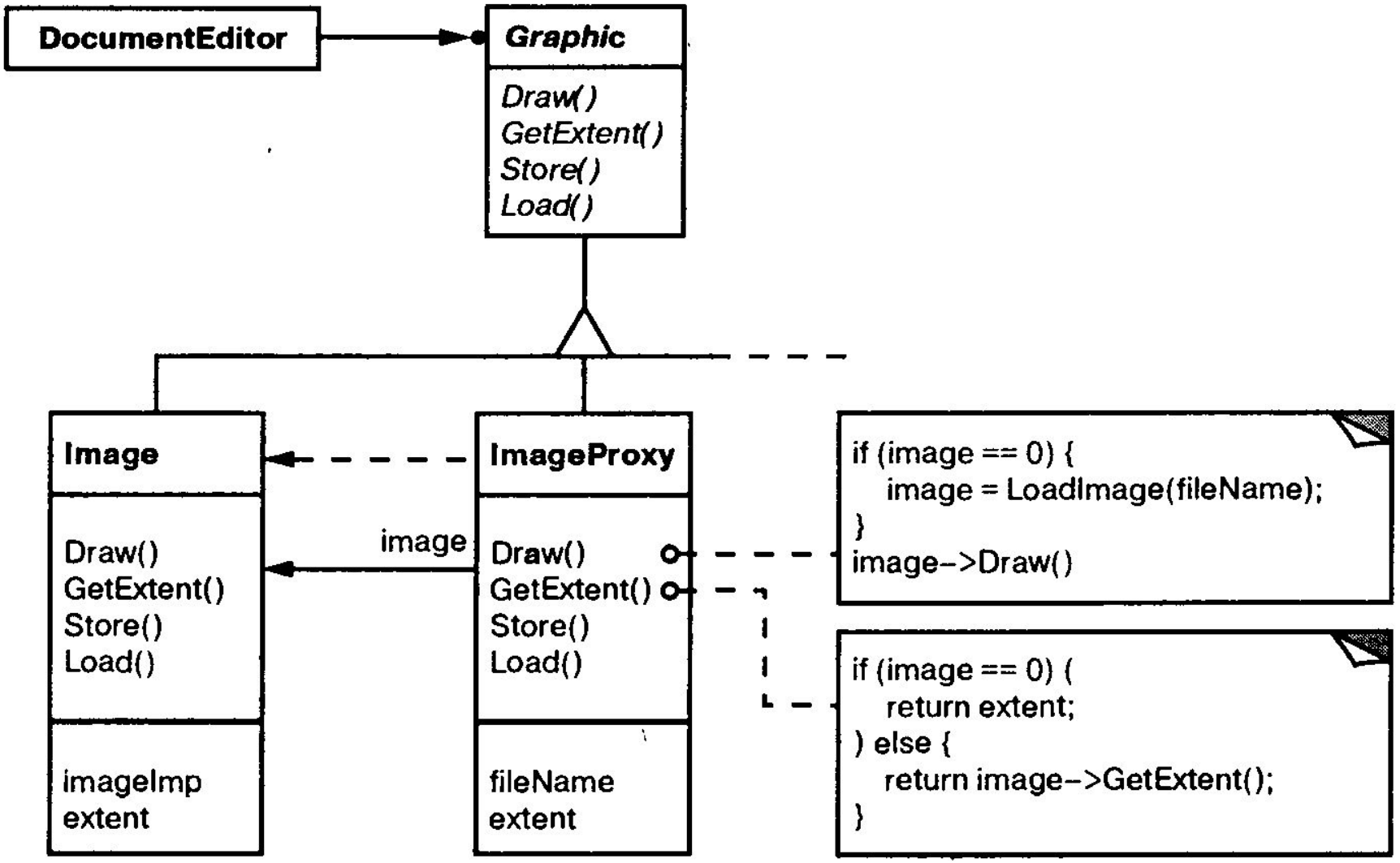
- *вынесение внешнего состояния*.
- *управление разделяемыми объектами*

Паттерн Proxy (Заместитель)

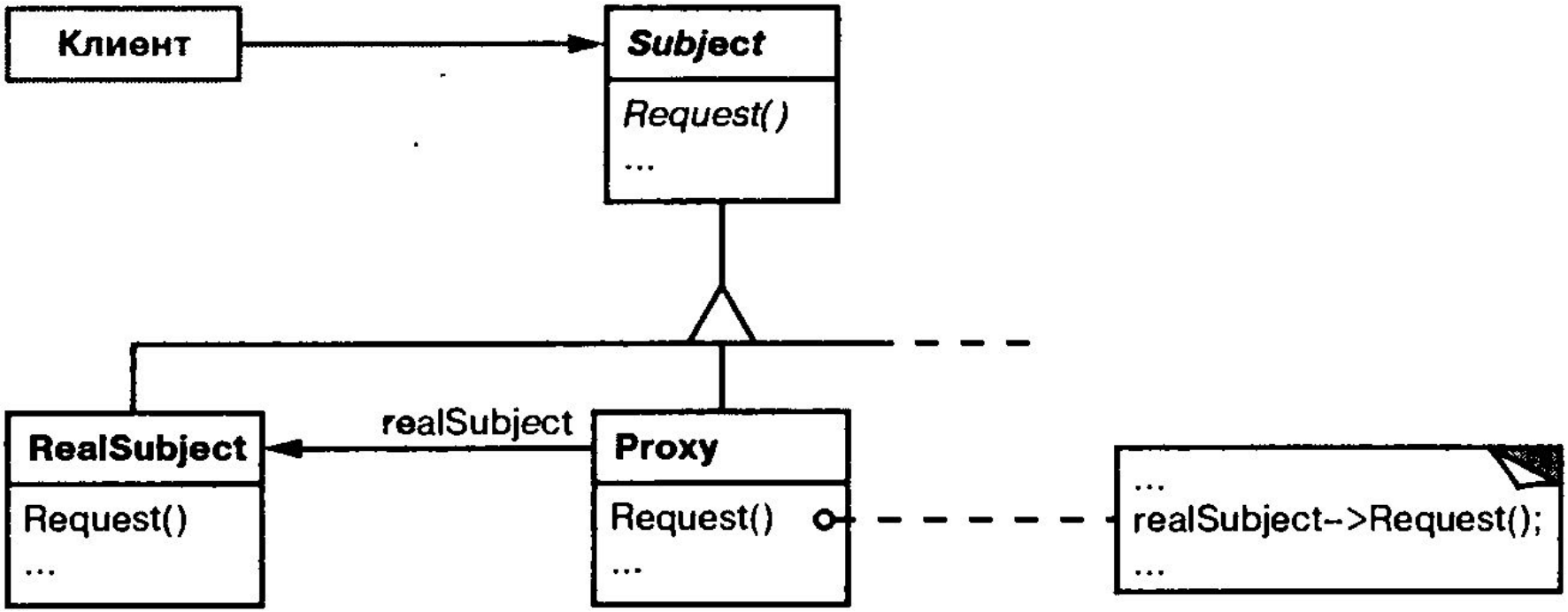


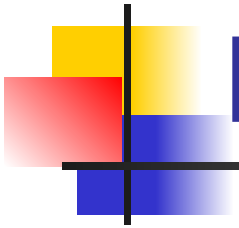
- ***Назначение***: Является суррогатом другого объекта и контролирует доступ к нему.

Паттерн Proxy (Заместитель)



Паттерн Proxy (Заместитель)

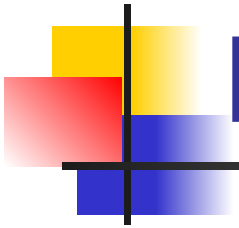




Паттерн Proxy (Заместитель)

У паттерна заместитель при доступе к объекту вводится дополнительный уровень косвенности:

- удаленный заместитель может скрыть тот факт, что объект находится в другом адресном пространстве;
- виртуальный заместитель может выполнять оптимизацию, например создание объекта по требованию;
- защищающий заместитель и «умная» ссылка позволяют решать дополнительные задачи при доступе к объекту.

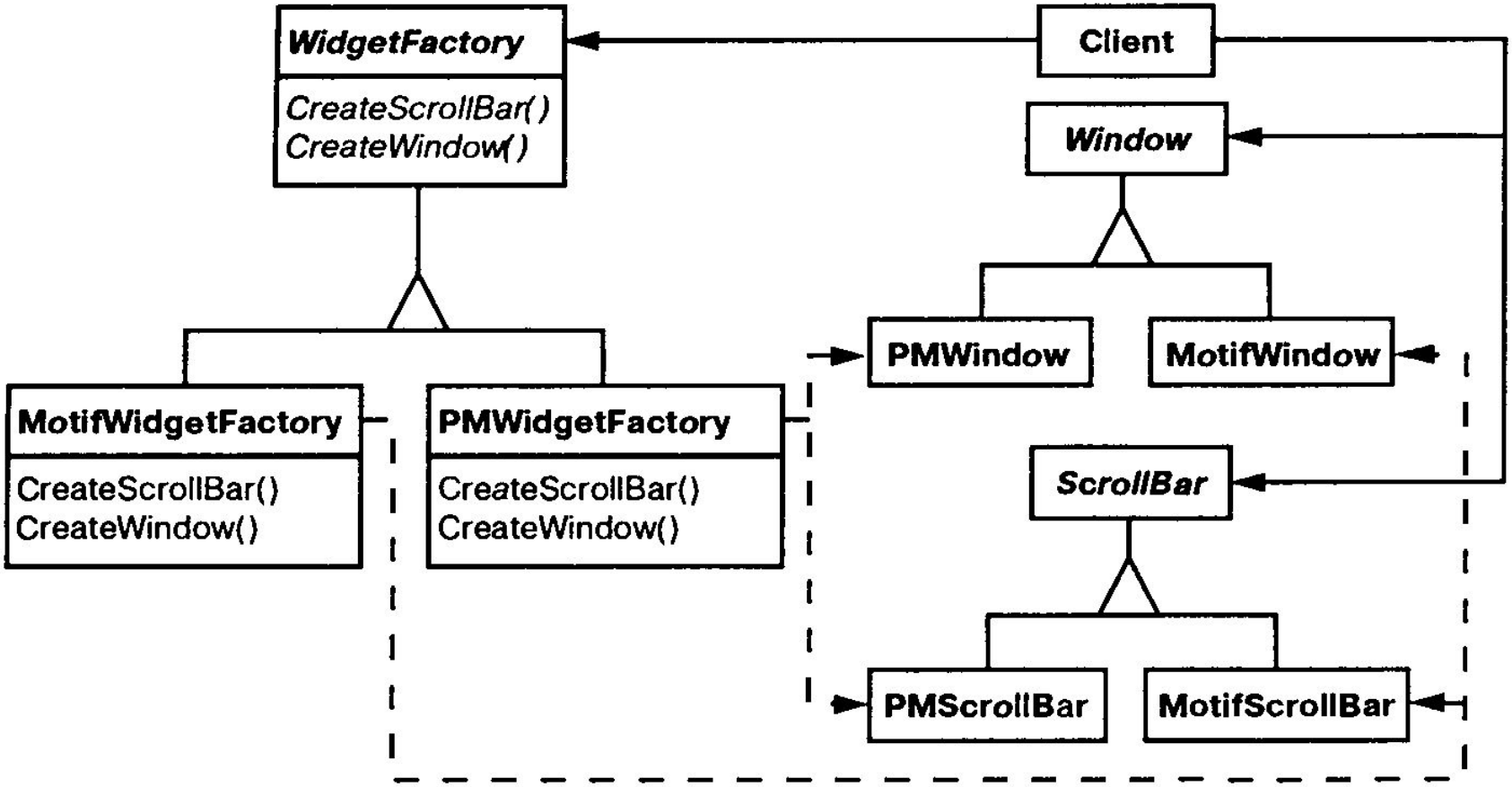


Паттерн Abstract Factory

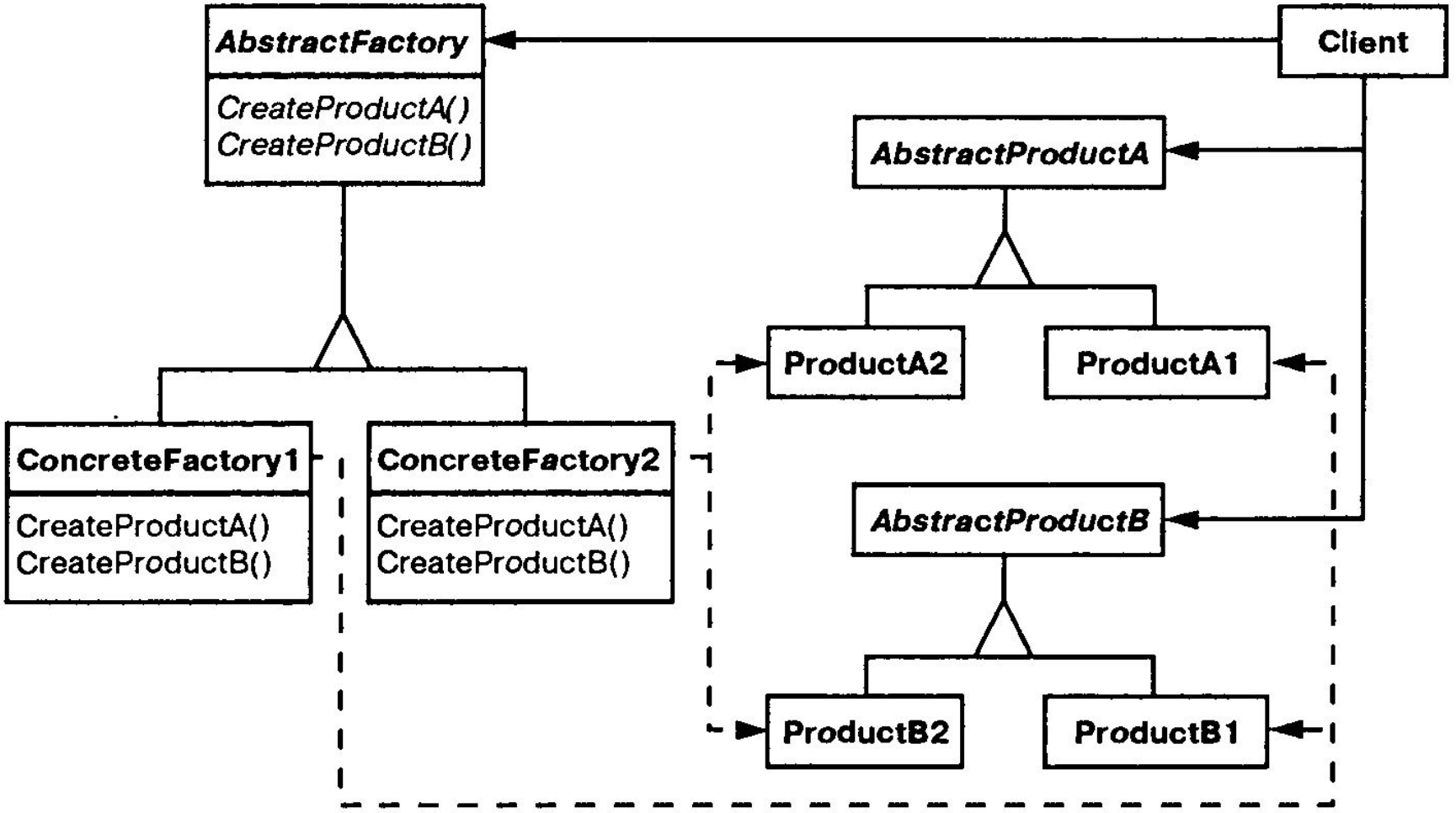
Назначение:

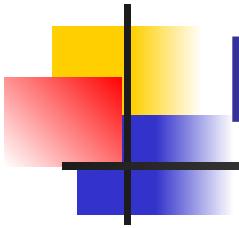
- Предоставляет интерфейс для создания семейств взаимосвязанных или взаимозависимых объектов, не специфицируя их конкретных классов.

Паттерн Abstract Factory



Паттерн Abstract Factory





Паттерн Prototype

Назначение

- Задает виды создаваемых объектов с помощью экземпляра-прототипа и создает новые объекты путем копирования этого прототипа.

Паттерн Prototype