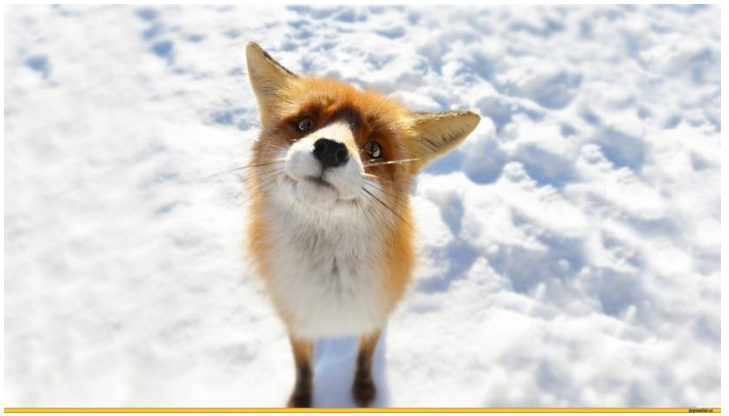
Введение в шаблоны GRASP

Сергей Немчинский, FoxmindEd,
2019

Обо мне

- Послужной список
- YouTube
- Про FoxmindEd



foxminded

Что такое

Шаблоны проектирования

Определение

Шаблон – это абстракция конкретных примеров, которые повторяются в определенных непроизвольных контекстах.

Что такое шаблоны проектирования?

"Каждый паттерн описывает некую повторяющуюся проблему и ключ к ее разгадке, причем таким образом, что этим ключом можно пользоваться при решении самых разнообразных задач".

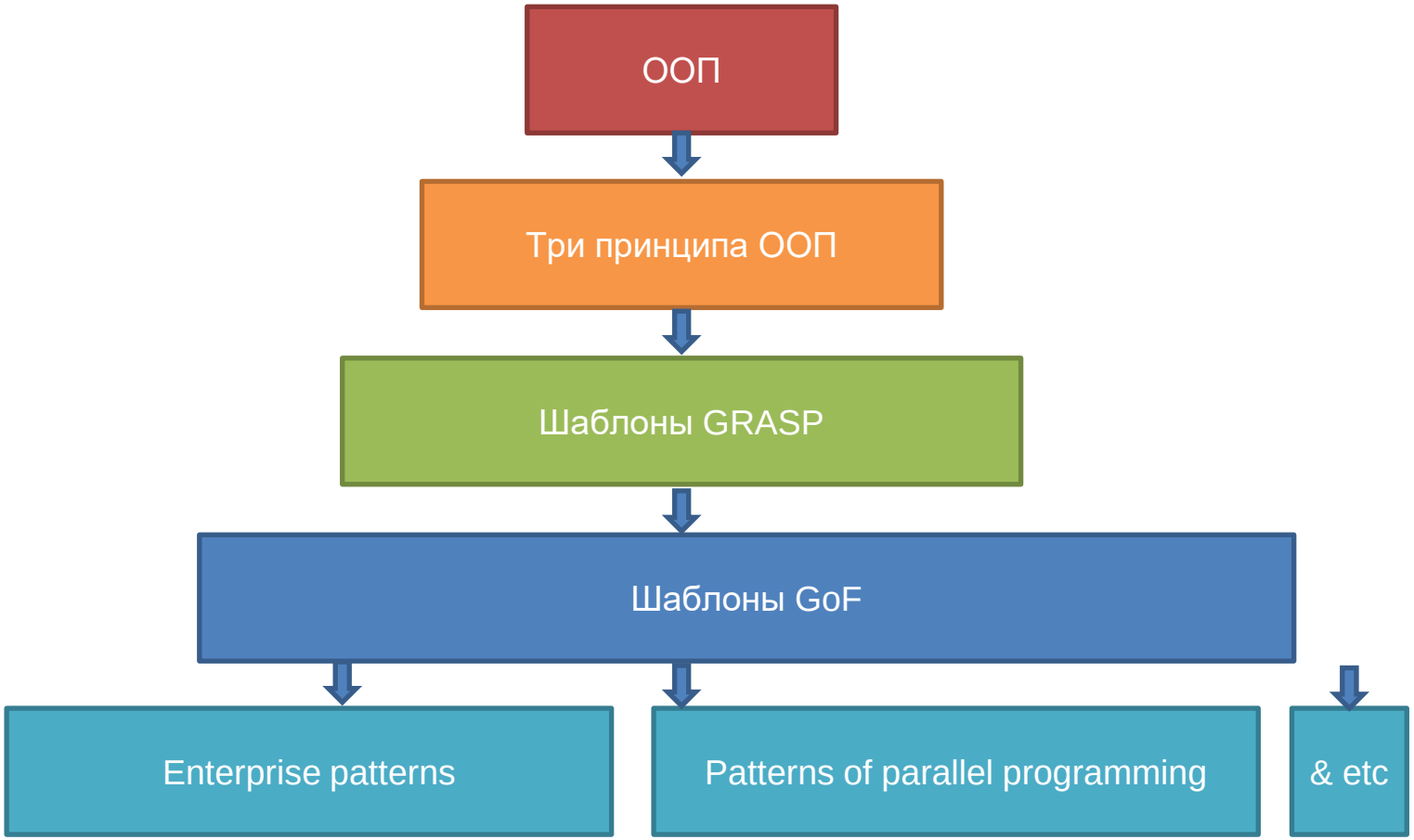
Christopher Alexander

foxminded

Что такое шаблоны проектирования?

Шаблоны проектирования (паттерн, pattern) — это эффективные способы решения характерных задач проектирования, в частности проектирования компьютерных программ. Паттерн не является законченным образцом проекта, который может быть прямо преобразован в код, скорее это описание или образец для того, как решить задачу, таким образом чтобы это можно было использовать в различных ситуациях.

Как все устроено?



foxminded

Принципы ООП

ООП (Объектно-Ориентированное Программирование)

- парадигма программирования, в которой основной концепцией является понятие объекта, отождествляя его с объектом предметной области.

Основные концепции

- Система состоит из объектов
- Объекты некоторым образом взаимодействуют между собой
- Каждый объект характеризуется своим состоянием и поведением

Принципы ООП

- Инкапсуляция
- Наследование
- Полиморфизм

Инкапсуляция

- это принцип, согласно которому любой класс и в более широком смысле – любая часть системы, должны рассматриваться как чёрный ящик — пользователь класса или подсистемы должен видеть и использовать только интерфейс (т. е. список декларируемых свойств и методов) и не вникать во внутреннюю реализацию.

- позволяет (теоретически) минимизировать число связей между классами и подсистемами и, соответственно, упростить независимую реализацию и модификацию классов и подсистем.

Наследование

- возможность порождать один класс от другого с сохранением всех свойств и методов класса-предка (иногда его называют суперклассом) добавляя, при необходимости, новые свойства и методы.
- призвано отобразить такое свойство реального мира, как иерархичность.

Полиморфизм

- классы-потомки могут изменять реализацию метода класса-предка, сохраняя его сигнатуру (таким образом, сохраняя неизменным интерфейс класса-предка).
- позволяет обрабатывать объекты классов-потомков как однотипные объекты, не смотря на то, что реализация методов у них может различаться.

Шаблоны GRASP

GRASP

- Craig Larman
- Книга Applying UML and Patterns, 1995. GRASP stands for General Responsibility Assignment Software Patterns.

Полный список шаблонов GRASP

- ❑ Information Expert
- ❑ Creator
- ❑ Controller
- ❑ Low Coupling
- ❑ High Cohesion
- ❑ Polymorphism
- ❑ Pure Fabrication
- ❑ Indirection
- ❑ Protected Variations

foxminded

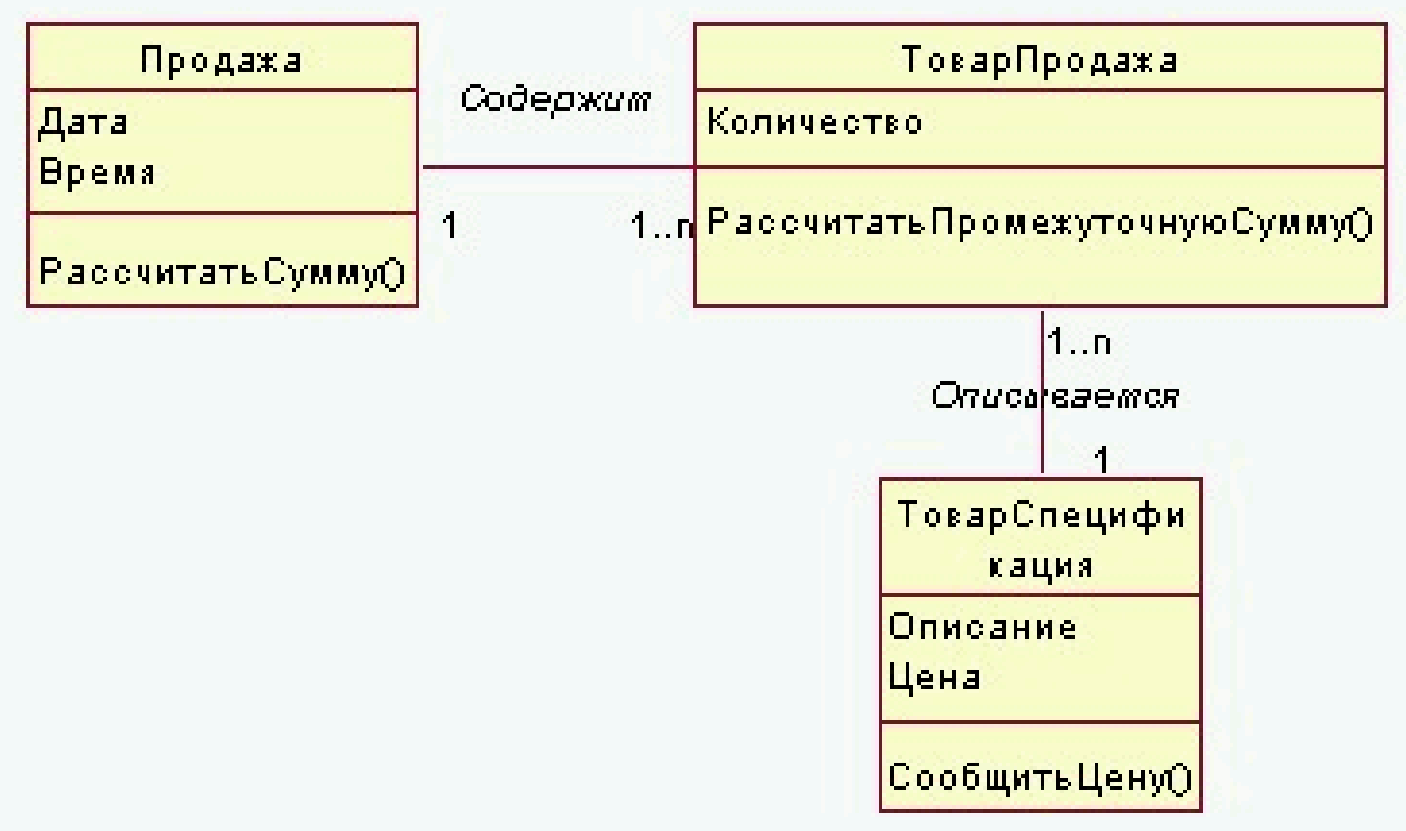
Шаблон информационный эксперт (Information Expert)- GRASP

- Проблема
  - В системе должна аккумулироваться, рассчитываться и т. п. необходимая информация.
- Решение
  - Назначить обязанность аккумуляции информации, расчета и т. п. некоему классу (информационному эксперту), обладающему необходимой информацией.
- Рекомендации
  - Информационным экспертом может быть не один класс, а несколько.

Эксперт. Пример

- Необходимо рассчитать общую сумму продажи. Имеются классы проектирования "Продажа", "ТоварПродажа" (продажа отдельного вида товара в рамках продажи в целом), "ТоварСпецификация" (описание конкретного вида товара).
- Необходимо распределить обязанности по предоставлению информации и расчету между этими классами.
- Таким образом, все три объекта являются информационными экспертами.

Эксперт. Диаграмма классов



foxminded

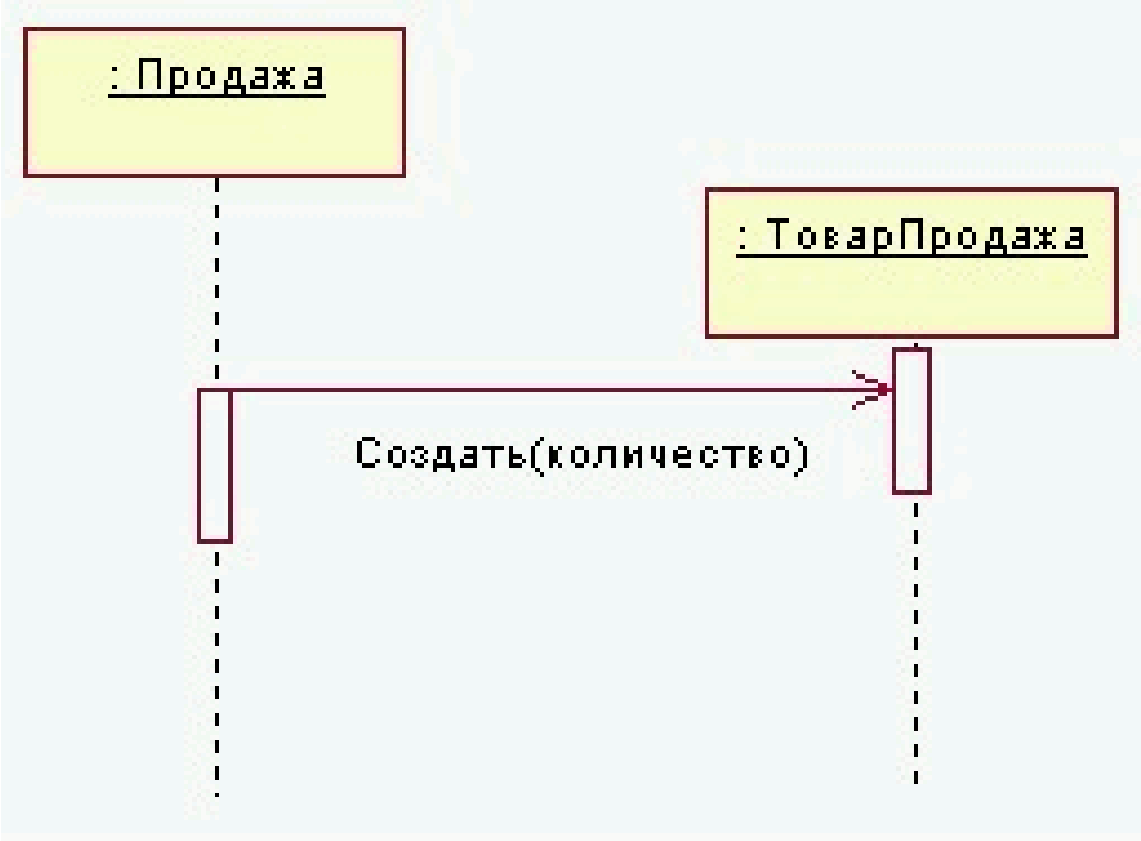
Создатель экземпляров класса (Creator) - GRASP

- Проблема
  - "Кто" должен отвечать за создание экземпляров класса.
- Решение
  - Назначить классу B обязанность создавать объекты другого класса A
- Рекомендации
  - Логично использовать паттерн если класс B содержит, агрегирует, активно использует и т.п. объекты класса A.

Creator. Пример.

- необходимо определить, какой объект должен отвечать за создание экземпляра "ТоварПродажа".
- Логично, чтобы это был объект "Продажа", поскольку он содержит (агрегирует) несколько обьектов "ТоварПродажа".

Creator. Диаграмма последовательности



Creator. Критика

- Преимущества
  - Использование этого паттерна не повышает связанности, поскольку созданный класс, как правило, виден только для класса - создателя.
- Недостатки
  - Если процедура создания объекта достаточно сложная (например выполняется на основе некоего внешнего условия), логично использовать паттерн "Абстрактная Фабрика",, то есть, делегировать обязанность создания объектов специальному классу.

Контроллер (Controller) - GRASP

- Проблема
  - "Кто" должен отвечать за обработку входных системных событий?
- Решение
  - Обязанности по обработке системных сообщений делегируются специальному классу. Контроллер - это объект, который отвечает за обработку системных событий и не относится к интерфейсу пользователя. Контроллер определяет методы для выполнения системных операций.
- Рекомендации
  - Для различных прецедентов логично использовать разные контроллеры (контроллеры прецедентов) - контроллеры не должны быть перегружены. Внешний контроллер представляет всю систему целиком, его можно использовать, если он будет не слишком перегруженным (то есть, если существует лишь несколько системных событий).

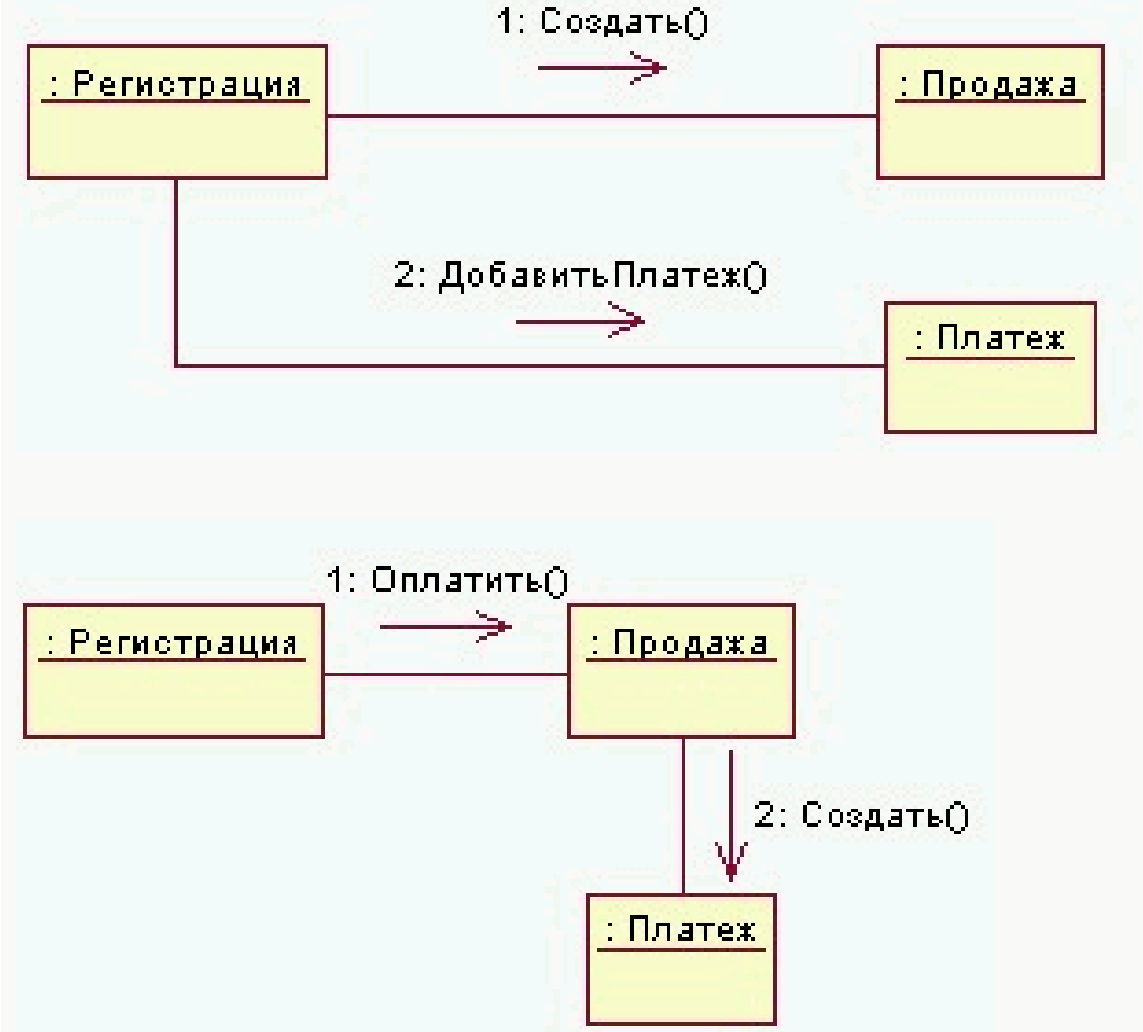
Controller. Критика

- Преимущества
  - Удобно накапливать информацию о системных событиях (в случае, если системные операции выполняются в некоторой определенной последовательности).
  - Улучшаются условия для повторного использования компонентов (системные события обрабатываются Контроллером а не элементами интерфейса пользователя).
- Недостатки
  - Контроллер может оказаться перегружен.

Низкая связанность (Low Coupling)

- Проблема
  - Обеспечить низкую связанность при создании экземпляра класса и связывании его с другим классом.
- Решение
  - Распределить обязанности между объектами так, чтобы степень связанности оставалась низкой.

Низкая связанность. Пример



Высокое зацепление (High Cohesion) - GRASP

- Проблема
  - Необходимо обеспечить выполнение объектами разнородных функций.
- Решение
  - Обеспечить распределение обязанностей с высоким зацеплением.

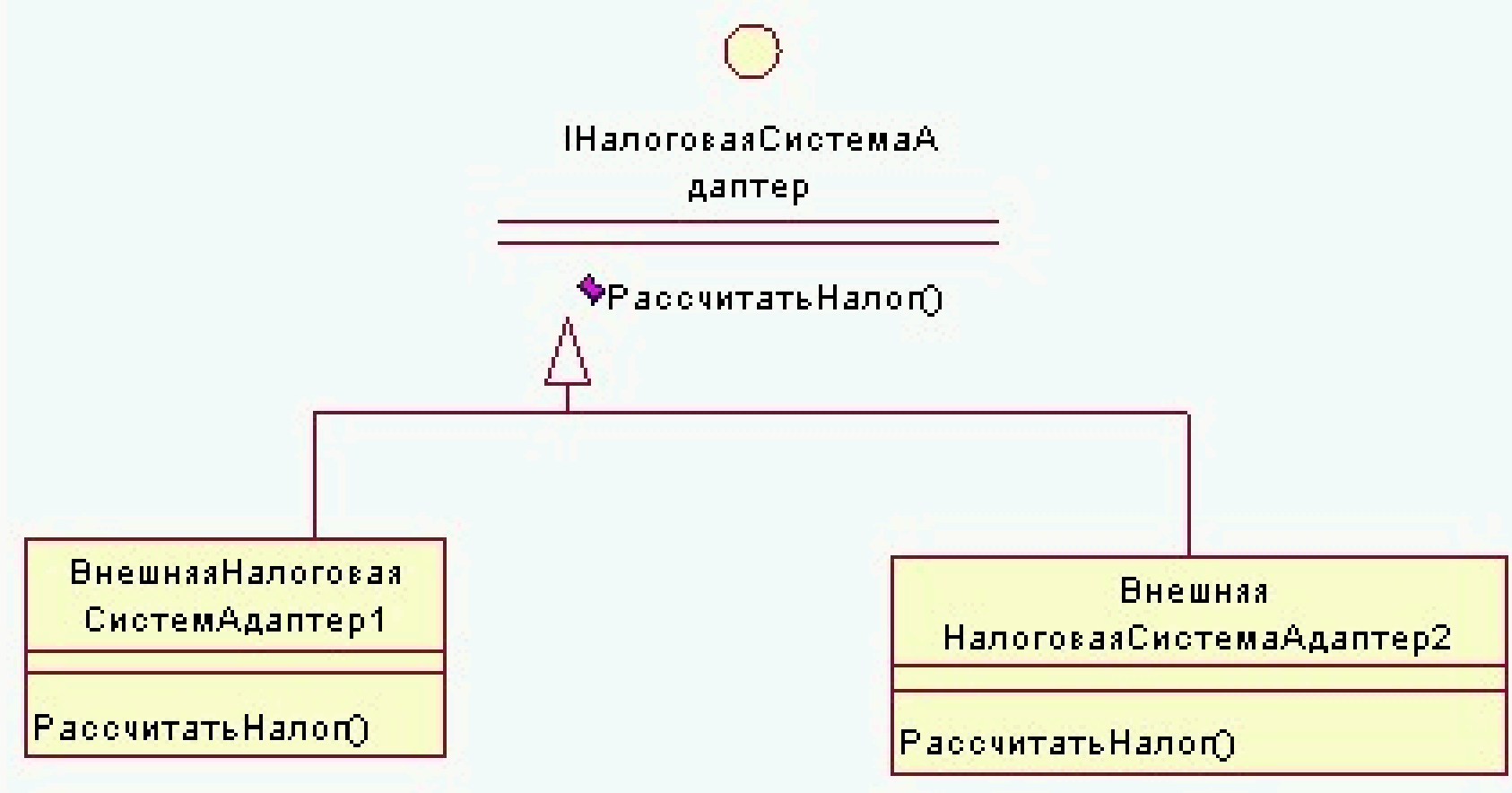
Высокое зацепление. Критика

- Преимущества
  - Классы с высокой степенью зацепления просты в поддержке и повторном использовании.
- Недостатки
  - Иногда бывает неоправданно использовать высокое зацепление для распределенных серверных объектов. В этом случае, для обеспечения быстродействия, необходимо создать несколько более крупных серверных объектов со слабым зацеплением

Полиморфизм (Polymorphism) - GRASP

- Проблема
  - Как обрабатывать альтернативные варианты поведения на основе типа?
  - Как заменять подключаемые компоненты системы?
- Решение
  - Обязанности распределяются для различных вариантов поведения с помощью полиморфных операций для этого класса.
  - Каждая внешняя система имеет свой интерфейс.

Полиморфизм. Пример



foxminded

Полиморфизм. Критика

- Преимущества
  - Впоследствии легко расширять и модернизировать систему.
- Недостатки
  - Не следует злоупотреблять добавлением интерфейсов с применением принципа полиморфизма с целью обеспечения дееспособности системы в неизвестных заранее новых ситуациях.

Искусственный (Pure Fabrication) - GRASP

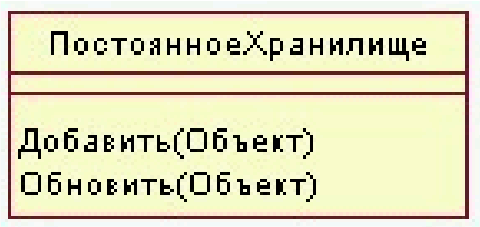
- Проблема
  - Какой класс должен обеспечивать реализацию паттернов "Высокое зацепление", и "Низкая связанность"?
- Решение
  - Присвоить группу обязанностей с высокой степенью зацепления классу, который не представляет конкретного понятия из предметной области (синтезировать искусственную сущность для обеспечения высокого зацепления и слабого связывания).

Искусственный. Пример

- Какой класс должен сохранять экземпляры класса "Продажа" в реляционной базе данных?
- Если возложить эту обязанность на класс "Продажа", то будем иметь низкую степень зацепления и высокую степень связывания (поскольку класс "Продажа" должен быть связан с интерфейсом реляционной базы данных.
- Хранение объектов в реляционной базе данных - это общая задача, которую придется решать для многих классов.

Искусственный. Пример

- Решением данной проблемы будет создание нового класса "ПостоянноеХранилище", ответственного за сохранение обьектов некоторого вида в базе данных.



foxminded

Искуственный. Критика

- Преимущества
  - Класс "ПостоянноеХранилище" будет обладать низкой степенью связывания и высокой степенью зацепления.
- Недостатки
  - Данным паттерном не следует злоупотреблять иначе все функции системы превратятся в объекты.

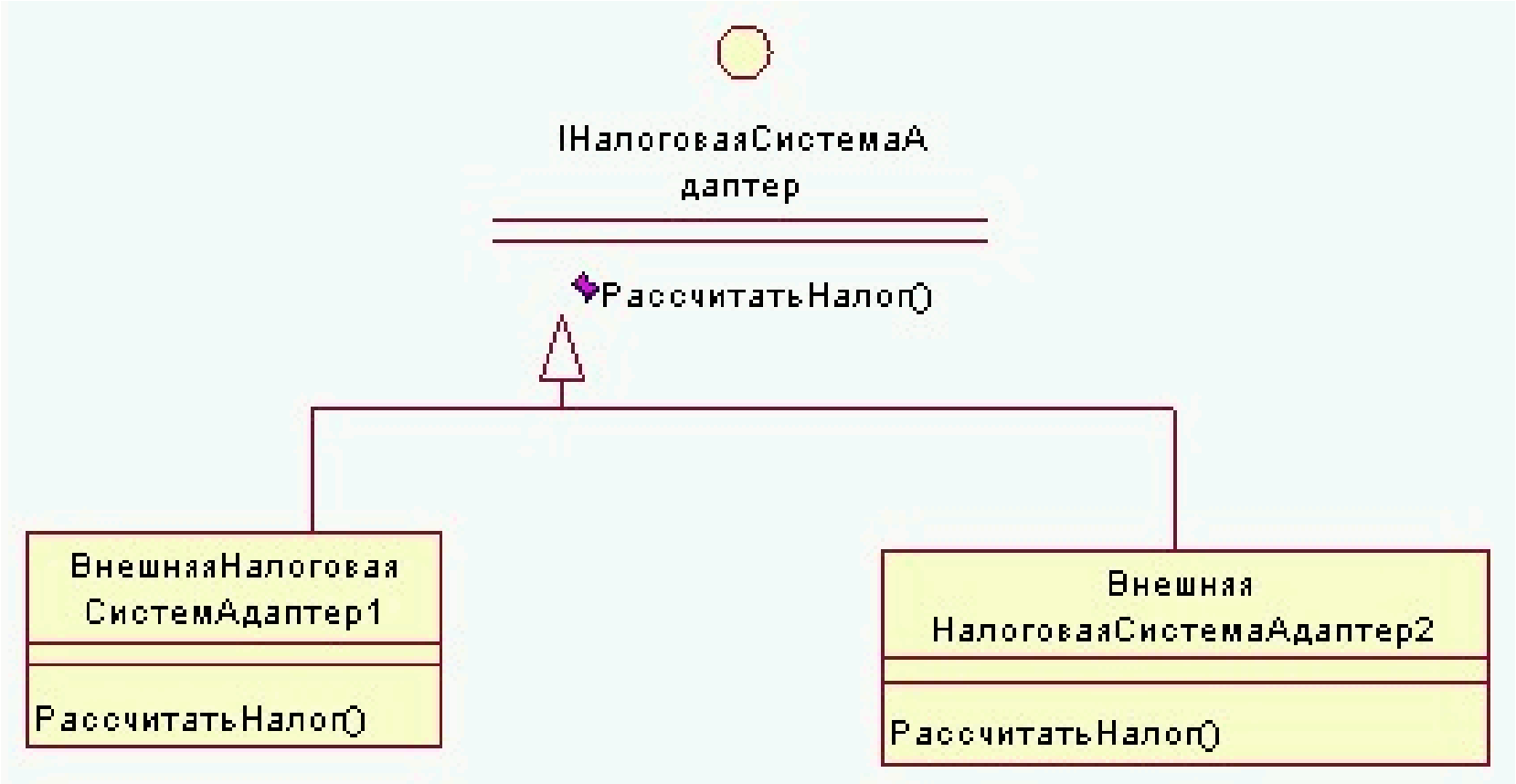
Перенаправление (Indirection) - GRASP

- Проблема
  - Как перераспределить обязанности обьектов, чтобы обеспечить отсутствие прямого связывания?
- Решение
  - Присвоить обязанности по обеспечению связи между службами или компонентами промежуточному объекту.
- Пример
  - См. пример к паттерну "Искусственный". Класс "Хранилище" выступает в роли промежуточного звена между классом "Продажа" и базой данных.

Устойчивый к изменениям (Protected Variations) - GRASP

- Проблема
  - Как спроектировать систему так, чтобы изменение одних ее элементов не влияло на другие?
- Решение
  - Идентифицировать точки возможных изменений или неустойчивости и распределить обязанности таким образом, чтобы обеспечить устойчивую работу системы.

Устойчивый к изменениям. Пример



foxminded

Итоги

- Паттерны GRASP:
- Information Expert
- Creator
- Controller
- Low Coupling
- High Cohesion
- Polymorphism
- Pure Fabrication
- Indirection
- Protected Variations

Ваши вопросы

https://foxmindEd.com.ua

facebook.com/foxmindedco/
pro100fox@gmail.com
Telegram: @nemchinskiy

foxminded