Шаблоны проектирования

Accessors, Exception, Singleton, Factory, Strategy

LoftSchool
от мыслителя к создателю

Исключения (Exception)

Исключение - это специальный объект, который является экземпляром встроенного класса Exception (или его произвольного класса). Объекты данного типа предназначены для хранения информации об ошибках и выдачи сообщений о них.

getMessage()	Получить строку сообщения, переданную в конструктор
getCode()	Получить код ошибки
getFile	Получить имя файла, в котором возникло исключение
getLine	Получить номер строки, в котором возникло исключение
getTrace()	Получить многомерный массив, отслеживающий вызовы метода, которые привели к исключению, включая имя метода, класса, файла и значения аргумента
getTraceAsString()	Получить строковую версию данных, возвращенных методом getTrace()

Обработка исключений

```
try{
    /// ваш код в котором может возникнуть исключение

} catch(Exception $e){

    // обработка исключения

} finally{
    // ваш код который выполнится в любом случае
}
```

Создание пользовательских исключений

Причины:

1. Можно расширить функциональность базового класса
2. Новый класс определяет новый тип ошибок, который можно будет перехватывать в своем коде

```
class XmlException extends Exception{
    ...
    function __construct(LibXMLError $error){
        $shortfile = basename($error->file);
        $msg = "[{$shortfile}], строка {$error->message}";
        $this->error = $error;
        parent::__construct($msg, $error->code);
    }
}
```

Демонстрация

Методы аксессоры. __set() и __get()

Это специализированные методы позволяющие обращаться и манипулировать закрытыми полями класса, если бы они были открытыми

```
class Employee{
    ///
       public function __get($index){
        return $this->$index;
    }
    ///
}
```

* необходимо не забывать про функцию property_exists

Интерполяция объекта. __toString()

Специальный метод __toString() позволяет интерполировать (подставлять) объект в строку. Для подстановки значений переменных необходимо заключить строку в двойные кавычки

```php
<?php

class Employee{
        ...
    public function __toString(){
        return "ФИО: ".$this->lastname.' '.$this->name.' '
                .$this->patronomic.'<br />Мой возвраст '.$this->age;
    }
}
```

Демонстрация

Что такое шаблоны проектирования?

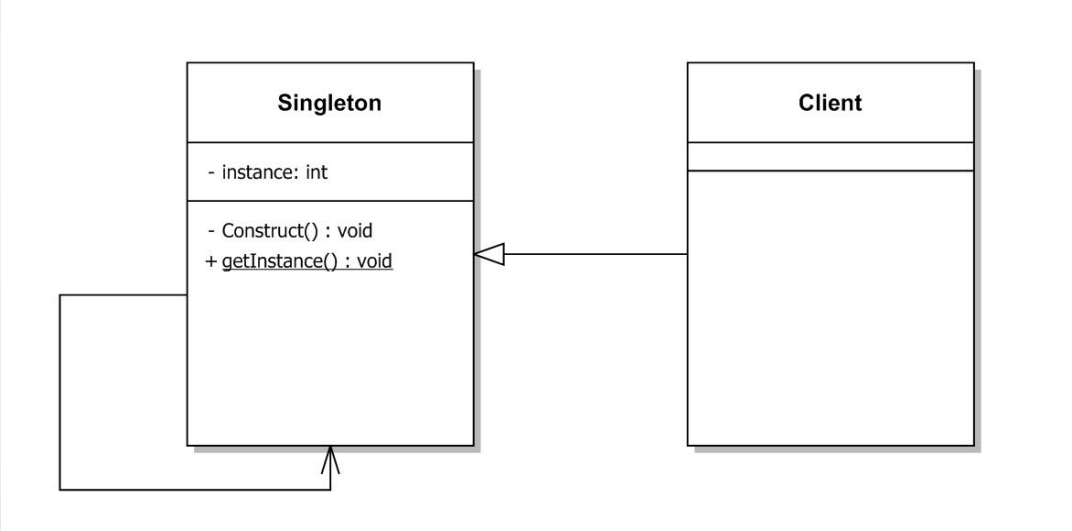
Шаблон проектирования или паттерн в разработке программного обеспечения — повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях. Объектно-ориентированные шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться.

(с) Википедия

Singleton

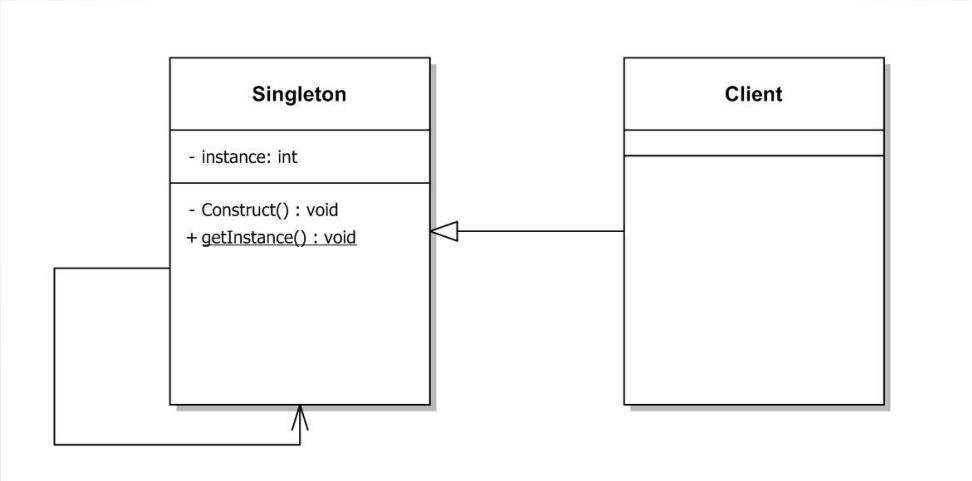
Одиночка (англ. Singleton) — порождающий шаблон проектирования, гарантирующий, что в однопоточном приложении будет единственный экземпляр класса с глобальной точкой доступа.



Singleton

Какие проблемы решает данный шаблон:

- Создается объект доступный их любого места системы, но сохраняется не в глобальной области видимости, где он может быть случайно изменен
- Уменьшить количество создаваемых объектов в системе, что повышает ее производительность

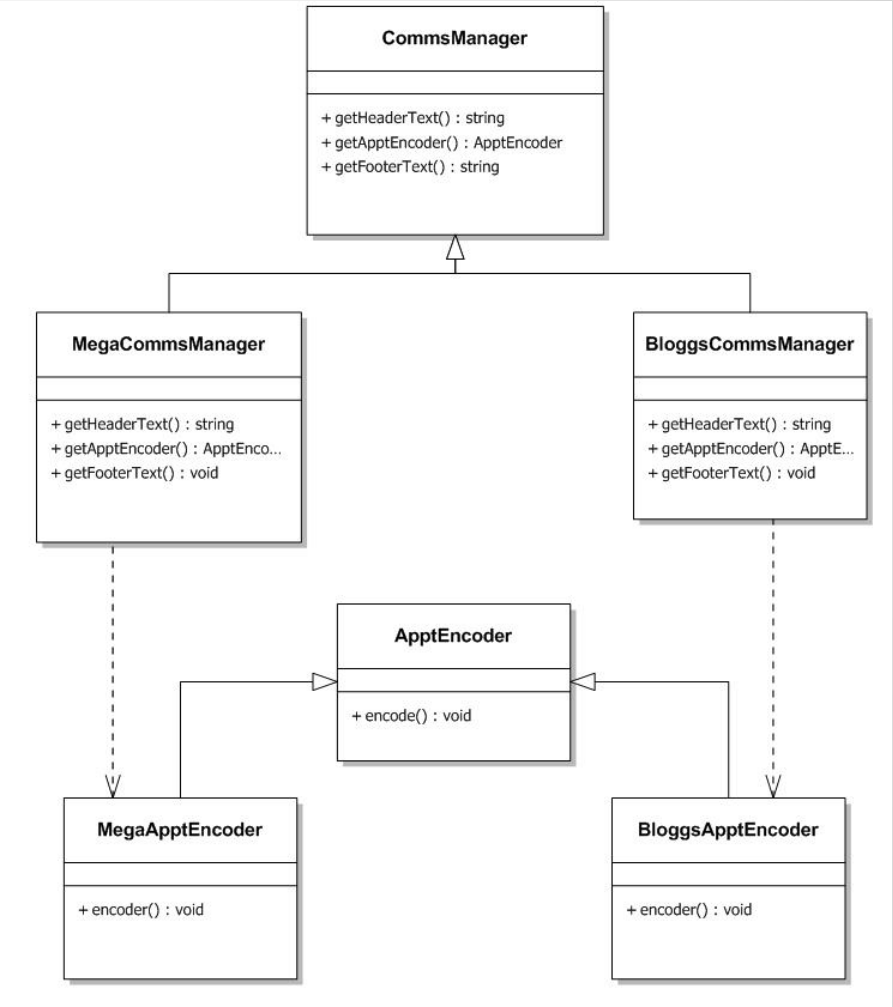


Демонстрация

(пример реализации)

Factory method

Фабричный метод - порождающий шаблон проектирования, предоставляющий подклассам интерфейс для создания экземпляров некоторого класса. В момент создания наследники могут определить, какой класс создавать. Иными словами, Фабрика делегирует создание объектов наследникам родительского класса. Это позволяет использовать в коде программы не специфические классы, а манипулировать абстрактными объектами на более высоком уровне.



Factory method

Какие проблемы решает данный шаблон:

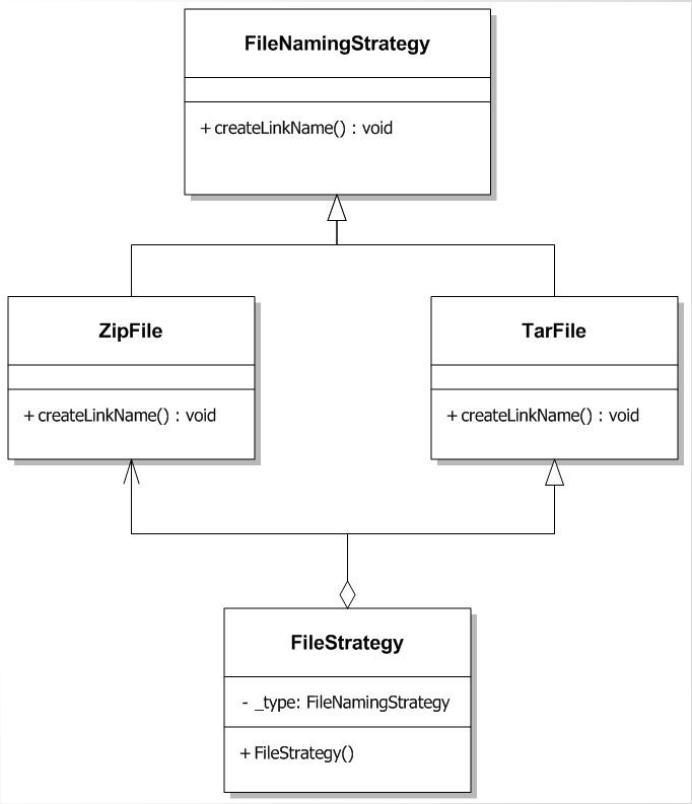
- Позволяет достаточно просто добавлять новые типы для обработки объектов
- Позволяет сделать код создания объектов более универсальным, не привязываясь к конкретным классам, а оперируя лишь общим интерфейсом

Демонстрация

(пример реализации)

Strategy method

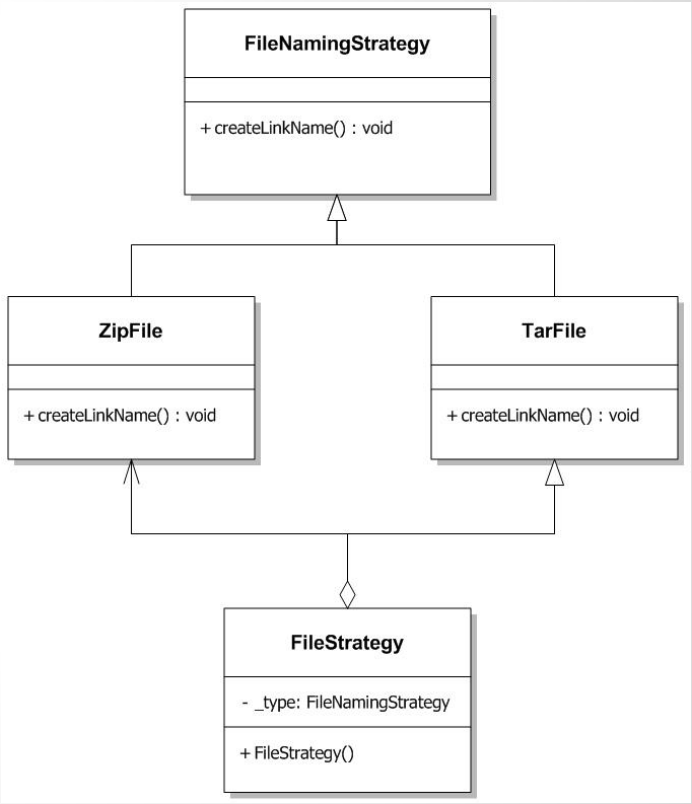
Strategy - поведенческий шаблон проектирования, предназначенный для определения семейства алгоритмов, инкапсуляции каждого из них и обеспечения их взаимозаменяемости. Это позволяет выбирать алгоритм путем определения соответствующего класса. Шаблон Strategy позволяет менять выбранный алгоритм независимо от объектов-клиентов, которые его используют.



Strategy method

Какие проблемы решает данный шаблон:

- Обеспечивает различные варианты алгоритмов поведения на основании типа клиента

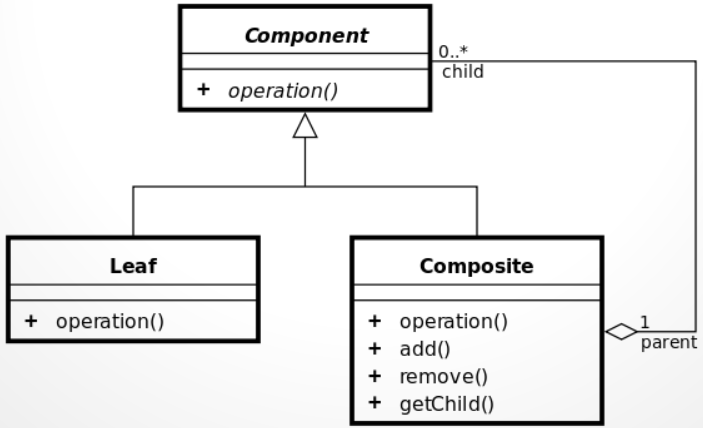


Демонстрация

(пример реализации)

Компоновщик (composite)

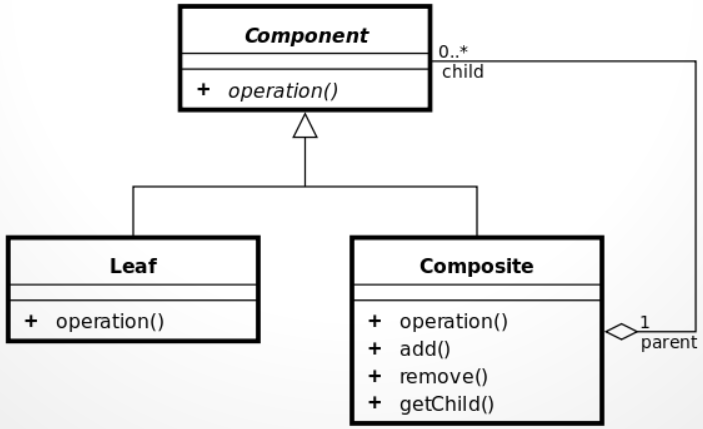
Структурный шаблон проектирования, объединяющий объекты в древовидную структуру для представления иерархии от частного к целому. Компоновщик позволяет клиентам обращаться к отдельным объектам и к группам объектов одинаково.



Компоновщик (composite)

Какие проблемы решает данный шаблон:

- Паттерн определяет иерархию классов, которые одновременно могут состоять из примитивных и сложных объектов, упрощает архитектуру клиента, делает процесс добавления новых видов объекта более простым.



Демонстрация

(пример реализации)

Что нужно сделать после вебинара?

1. Пересмотреть запись вебинара
2. Прочитать методичку
3. Выполнить домашнее задание
4. Продолжить чтение книг из списка “рекомендованной литературы”
5. Задавать свои вопросы в общем чате
6. Ожидать следующий вебинар