

Практическое изучение объектно-ориентированного программирования с применением электронного задачника Programming Taskbook for OOP

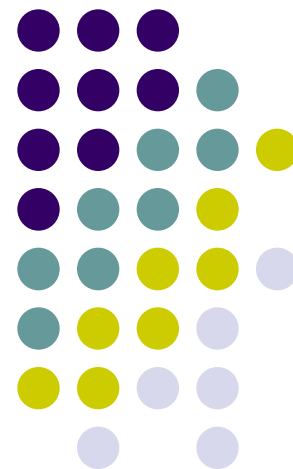


Абрамян М.Э.^{1,2}

¹*Южный федеральный университет,
Институт математики, механики
и компьютерных наук им. И. И. Воровича*

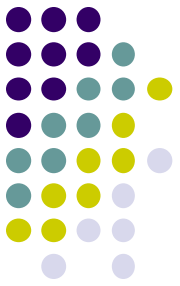
²*Университет МГУ-ППИ
в Шэньчжэне (КНР)*

mabr@sfedu.ru

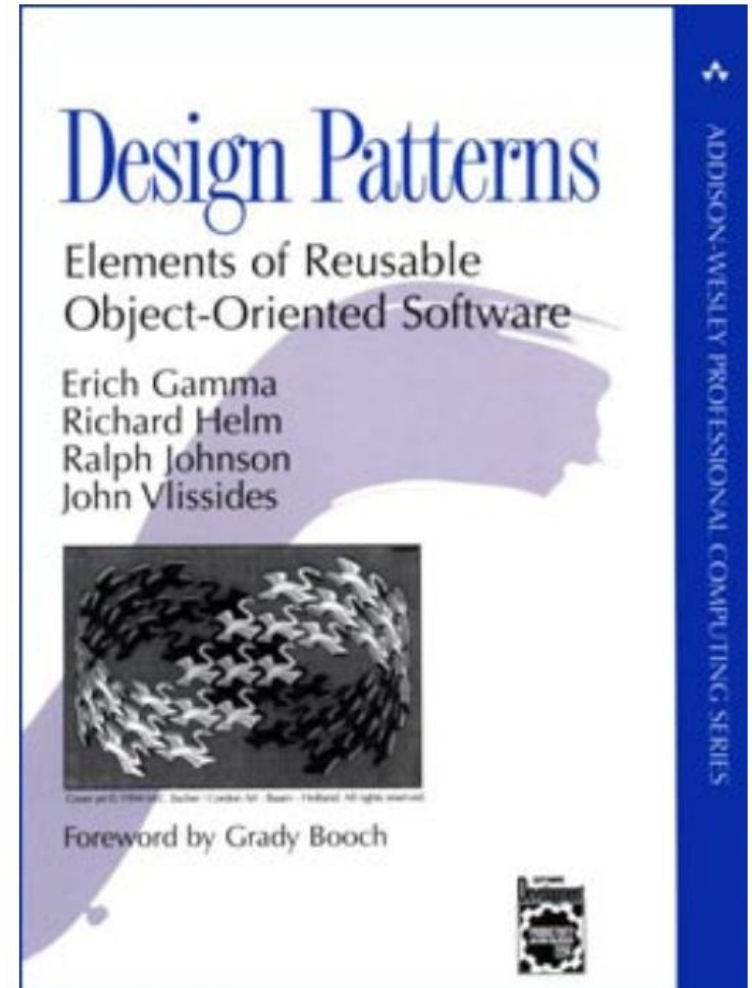


Доклад на пятой Всероссийской научно-методической конференции
**«Использование системы программирования PascalABC.NET
в обучении программированию»**
Ростов-на-Дону, 28-29 марта 2025 г.

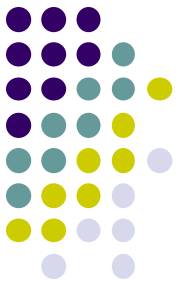
Паттерны объектно-ориентированного проектирования



- **Паттерн** (англ. design pattern) в разработке программного обеспечения – повторяемая конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.
- Книга **Design Patterns** четырех авторов (Gamma, Helm, Johnson, Vlissides) представляет собой классическое описание 23 распространенных паттернов проектирования.

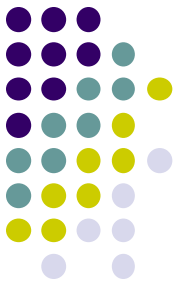


Обучение паттернам проектирования на примерах



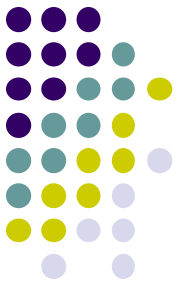
- Рассматривается модельная задача, для которой разрабатывается система классов, соответствующая изучаемому паттерну.
- Правильность системы проверяется путем обработки наборов тестовых данных.
- Для повышения надежности тестирования желательно **автоматически** генерировать исходные данные и проверять правильность полученных результатов.

Электронный задачник по паттернам проектирования



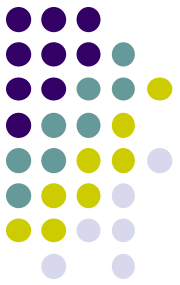
- Является расширением универсального задачника Programming Taskbook
- Предназначен для поддержки практической части курсов по ООП
- Позволяет изучить и реализовать на практике 21 паттерн проектирования
- Предусматривает дополнительную поддержку для языков PascalABC.NET, C++, C#, Java, Python, Ruby, Julia
- Интегрирован в среду программирования PascalABC.NET

Состав электронного задачника по паттернам проектирования (1)



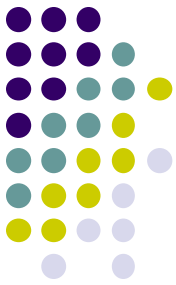
- Вводная группа **OOP0Begin** (12 заданий)
 - Создание классов и объектов, скрытие данных (**инкапсуляция**)
 - Порождение новых классов (**наследование**)
 - Возможность единообразной обработки объектов разных типов, входящих в одну иерархию (**полиморфизм**)
 - Использование **информации о типе времени выполнения** (RTTI)
 - Генерация и обработка **исключений**
 - **Шаблонные классы** и шаблонные функции
 - **Перегрузка операций**, приведение объекта к другим типам, глубокое копирование

Состав электронного задачника по паттернам проектирования (2)



- **OOP1Creat** – порождающие паттерны (10)
 - Factory Method (3), Abstract Factory (2)
 - Singleton (1), Prototype (2), Builder (2)
- **OOP2Struc** – структурные паттерны (11)
 - Adapter (4), Composite (2), Decorator (2)
 - Proxy (1), Bridge (1), Flyweight (1)
- **OOP3Behav** – паттерны поведения (17)
 - Observer (2), Strategy (2), Template Method (2)
 - Iterator (1), Command (2), State (2)
 - Mediator (1), Chain of Responsibility (2), Visitor (1), Interpreter (2)

Окно электронного задачника с сообщением о решении задачи



PT Programming Taskbook - Электронный задачник по программированию [PascalABC.NET]

ADAPTER, COMPOSITE, DECORATOR
Задание: OOP2Struc2*

Выполняет: Иванов Петр

Результаты (F2) Цвет (F3) Режим (F4)

Задание выполнено!

Выход (Esc)

Ввод: 10 из 10 ■

Вывод: 9 из 9 ■

Тесты: 5 из 5 ■

Adapter (Адаптер) – структурный паттерн.

В данном задании рассматривается вариант адаптера, использующий множественное наследование. Такой вариант называется **адаптером класса**, поскольку класс Adapter порождается от **класса** Adaptee, наследуя его реализацию (а также от класса Target, наследуя его интерфейс). Если язык не поддерживает множественное наследование, то адаптер класса можно реализовать с помощью **интерфейсов**; для этого надо преобразовать абстрактный класс Target в интерфейс ITarget, сделать класс Adapter потомком класса Adaptee и добавить к нему интерфейс ITarget.

Задание 2. Выполнить предыдущее задание (см. OOP2Struc1), реализовав Adapter как адаптер класса. Для языков, не поддерживающих множественное наследование, определить интерфейс ITarget с методами GetA, GetB и Request и добавить интерфейс ITarget к классам ConcreteTarget и Adapter (абстрактный класс Target теперь не требуется; для хранения исходных данных надо использовать коллекцию с элементами интерфейсного типа ITarget).

Исходные данные

N = 3

'*' 5 28 '*' 30 11 '+' 24 6

Полученные результаты

24 6 30 30 11 330 5 28 140

Представление формулировки задачи в формате html



PT4Tasks

file:///C:/PABCWork.NET/PT4Tasks.html

Adapter (Адаптер) — структурный паттерн.

Известен также под именем **Wrapper (Обертка)**.

Частота использования: выше средней.

Назначение: преобразует интерфейс одного класса в интерфейс другого, который ожидают клиенты. Адаптер обеспечивает совместную работу классов с несовместимыми интерфейсами, которая без него была бы невозможна.

Участники:

- *Target (Целевой класс)* — определяет зависящий от предметной области интерфейс, которым пользуется Client;
- *Client (Клиент)* — вступает во взаимоотношения с объектами, удовлетворяющими интерфейсу Target;
- *Adaptee (Адаптируемый класс)* — определяет существующий интерфейс, который нуждается в адаптации;
- *Adapter (Адаптер)* — адаптирует интерфейс Adaptee к интерфейсу Target.

В данном задании рассматривается вариант адаптера, использующий композицию. Такой вариант называется *адаптером объекта*, поскольку адаптируется *объект* типа Adapter, входящий в виде ссылочного поля в класс Adapter.

Задание 1. Абстрактный класс Target содержит три абстрактных метода: GetA, GetB и Request (не имеют параметров, возвращают значение целого типа). Класс ConcreteTarget является потомком класса Target; он содержит поля a и b целого типа, которые инициализируются в конструкторе, имеющем одноименные параметры. Методы GetA и GetB класса ConcreteTarget возвращают значения полей a и b соответственно, а метод Request возвращает сумму этих полей.

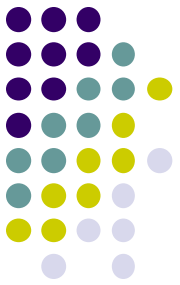
Класс Adaptee содержит два целочисленных поля a и b, конструктор с параметрами a и b, задающий значения этих полей, метод GetAll, возвращающий текущие значения полей (либо с помощью выходных параметров, либо с помощью возвращаемого значения — массива или кортежа), и метод SpecificRequest без параметров, возвращающий произведение полей a и b.

Реализовать класс Adapter, адаптирующий класс Adaptee к интерфейсу класса Target. Класс должен быть адаптером объекта: он порождается от класса Target и включает ссылку ad на экземпляр адаптируемого объекта Adaptee. Ссылка ad инициализируется в конструкторе класса Adapter путем вызова конструктора класса Adaptee с параметрами a и b, совпадающими с одноименными параметрами конструктора класса Adapter. Метод Request класса Adapter должен вызывать метод SpecificRequest объекта ad, а методы GetA и GetB — возвращать значения полей a и b объекта ad, используя его метод GetAll.

Тестирование разработанной системы классов. Дано целое число $N (\leq 6)$ и набор из N троек (C, A, B) , где C является символом «+» или «*», а элементы A и B являются целыми числами. Создать структуру данных (например, массив) с элементами-ссылками типа Target и заполнить ее объектами типа ConcreteTarget (для троек с символом «+») и Adapter (для троек с символом «*») с полями, равными A и B . Перебирая элементы полученного набора в *обратном порядке*, вывести для каждого из них значения полей a, b и результат выполнения метода Request.

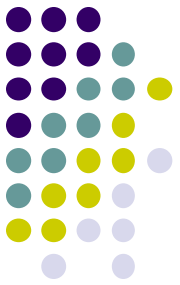
Request()

8



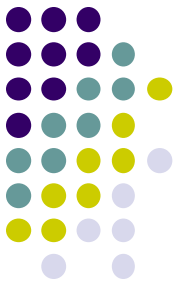
Особенности заданий

- Первое задание содержит описание **модельной системы классов**, связанной с обработкой простых наборов данных.
- Для большинства паттернов предусмотрено два задания, причем во втором задании обычно рассматривается задача с **реальным содержанием**.
- Например, для паттерна **Decorator** строятся классы для генерации суперпозиции различных функций и вычисления их значений, а для паттерна **Strategy** – классы-валидаторы, обеспечивающие проверку правильности различных наборов текстовых данных.



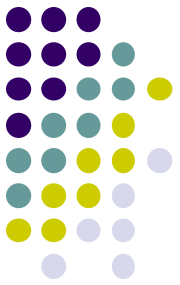
Файлы дополнений

- **Файлы дополнений** – это вспомогательные файлы задачника, содержащие дополнительные примечания и тексты индивидуальных программ-заготовок на различных языках программирования для каждого задания.
- Файлы дополнений разработаны для всех групп задачника PT for OOP.
- Они включают тексты программ-заготовок на языках PascalABC.NET, C++, C#, Java, Python, Ruby, Julia.



Содержание файлов дополнений

- Для языков со статической типизацией (PascalABC.NET, C++, C#, Java) в программы-заготовки включались описания **абстрактных базовых классов**, для языков с динамической типизацией (Python, Ruby) приводились фрагменты описаний **конкретных классов**.
- В ряде случаев добавлялись комментарии, связанные с **особенностями реализации паттерна** на конкретном языке.
- Для всех заданий вводной группы OOP0Begin в файлы дополнений были включены комментарии, описывающие **особенности объектных моделей** каждого языка.



Формат файлов дополнений

- Файлы дополнений включают тексты примечаний, фрагменты программного кода и управляющие команды.

[2]

[text]

%{S}Указание (PascalABC.NET).%{s} При определении конструктора класса %{M}ConcreteProduct2%{m} вызывайте в его начале конструктор предка следующим образом: %{M}inherited Create(info)%{m}, при модификации метода %{M}Transform%{m} класса %{M}ConcreteProduct2%{m} аналогичным образом вызывайте одноименный метод предка: %{M}inherited Transform()%{m}.

[code]

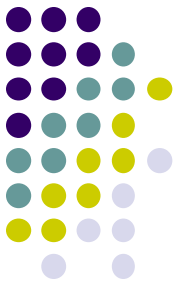
type

ConcreteProduct1 = class

protected

info: string;

Файлы дополнений в режиме html-документа



Transform созданного продукта и с помощью его метода GetInfo возвращает полученный результат.

Тестирование разработанной системы классов. Даны пять строк. Используя конкретных создателей 1 и 2, применить к каждой из данных строк метод AnOperation и вывести возвращаемый результат этого метода (вначале выводятся результаты для первой строки, затем для второй и т. д.).

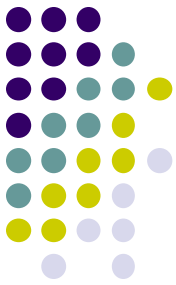
Указание (PascalABC.NET). При определении конструктора класса ConcreteProduct2 вызывайте в его начале конструктор предка следующим образом: inherited Create(info), при модификации метода Transform класса ConcreteProduct2 аналогичным образом вызывайте одноименный метод предка: inherited Transform().

```
type
    ConcreteProduct1 = class
    protected
        info: string;
    public
        function GetInfo: string;
        begin
            Result := info;
        end;
        constructor Create(info: string);
        begin
            // Реализуйте конструктор
        end;
        procedure Transform; virtual;
        begin
            // Реализуйте метод
        end;
    end;

    // Реализуйте класс-потомок ConcreteProduct2

    ConcreteCreator1 = class
    protected
```

Заготовка на языке PascalABC.NET



PascalABC.NET 3.10

Файл Правка Вид Программа Сервис Модули Помощь

•OOP1Creat2.pas

```
uses PT4;

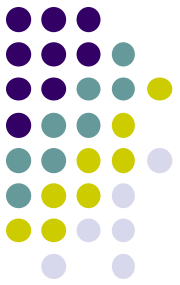
type
  ConcreteProduct1 = class
  protected
    info: string;
  public
    function GetInfo: string;
    begin
      Result := info;
    end;
    constructor Create(info: string);
    begin
      // Реализуйте конструктор
    end;
    procedure Transform; virtual;
    begin
      // Реализуйте метод
    end;
  end;

  // Реализуйте класс-потомок ConcreteProduct2

  ConcreteCreator1 = class
  protected
    function FactoryMethod(info: string): ConcreteProduct1; virtual;
```

Компиляция прошла успешно (45 строк)

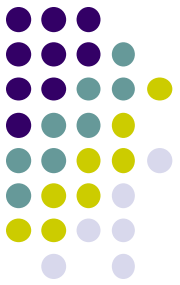
Строка 44 Столбец 3



Дополнительные возможности

- Файлы дополнений являются обычными текстовыми файлами, поэтому любой преподаватель может их корректировать, добавляя или изменяя указания или тексты программ-заготовок для различных языков.
- В варианте файлов дополнений для языка C++ предполагается, что для работы с объектами используются **умные указатели** `shared_ptr<T>` и `weak_ptr<T>`.

Результаты использования задачника в учебных курсах



- Задачник использовался при проведении **лабораторных занятий** по дисциплинам:
 - «**Объектно-ориентированное программирование**» (C++, Python) и «**Языки программирования**» (C#), 3 курс факультета вычислительной математики и кибернетики Университета МГУ-ППИ в Шэньчжэне)
 - «**Языки программирования**» (C#), 1 год магистратуры мехмата ЮФУ, программа «Разработка компьютерных игр и мобильных приложений»
- Это позволило существенно увеличить количество выполненных заданий и упростило их проверку.
- Задачник также применялся на **лекциях**: обсуждение формулировок задач, программ-заготовок и тестовых данных позволило осветить различные аспекты рассматриваемого паттерна и продемонстрировать его работу на примерах.

Видеозаписи занятий, на которых использовался задачник PTforOOP



- Ссылка на облачное хранилище (ссылка имеется на сайте задачника): <https://owncloud.nano.sfedu.ru/index.php/s/9c59zf3LG6Bb45n>
- Общая продолжительность видеозаписей – около 15 часов
- Часть 1. Знакомство с ООП на примерах языков C++ и Python
 - Инкапсуляция и наследование в C++ (OOP0Begin1,2)
 - Полиморфизм в C++: виртуальные функции и позднее связывание (OOP0Begin3). Объектная модель языка Python (OOP0Begin1,2,3)
 - Получение информации о типе времени выполнения (OOP0Begin4). Паттерны проектирования: обзор. Паттерн «Фабричный метод» (OOP1Creat1)
 - Паттерны «Фабричный метод» и «Адаптер» (OOP1Creat1, OOP1Struc2)
 - Паттерн «Наблюдатель» (OOP3Behav1)
- Часть 2. Объектная модель языков платформы .NET на примере C#
 - Основы объектной модели платформы .NET на примере паттерна «Фабричный метод»
 - Работа с интерфейсами на примере паттерна «Адаптер». Свойства и делегаты на примере паттерна «Наблюдатель»
 - События в паттерне «Наблюдатель», обзор паттернов «Посредник» и «Цепочка обязанностей»
 - Обзор паттернов «Посетитель» и «Интерпретатор»
 - Паттерн «Наблюдатель», модель проталкивания (OOP3Behav2), паттерн «Итератор», интерфейсы IEnumerator и IEnumerable (OOP3Behav7)

Страница облачного хранилища с видеозаписями занятий



Files - Nextcloud

owncloud.nano.sfedu.ru/index.php/s/9c59zf3LG6Bb45n

PTforOOP-2025 Download all files

Name	Size	Modified
1-1. Инкапсуляция и наследование в C++. Использование умных указателей shared_ptr. OOP0Begin1,2.mp4	229.8 MB	a year ago
1-2. Полиморфизм в C++; виртуальные функции и позднее связывание. OOP0Begin3. Объектная модель языка Python. OOP0Begin1,2,3(Py).mp4	252.7 MB	a year ago
1-3. Получение информации о типе времени выполнения. OOP0Begin4(C++,Py). Паттерны проектирования. Паттерн Factory Method. OOP1Creat1.mp4	260.1 MB	a year ago
1-4. Паттерны Factory Method и Adapter. OOP1Creat1(C++,Py), OOP1Struc2(C++,Py).mp4	253.5 MB	a year ago
1-5. Паттерн Observer. OOP3Behav1(C++,Py).mp4	267.6 MB	a year ago
2-1. Основы объектной модели платформы NET на примере паттерна Фабричный метод (OOP1Creat1).mp4	245 MB	a year ago
2-2. Работа с интерфейсами на примере паттерна Адаптер (OOP2Struc2). Свойства и делегаты на примере паттерна Наблюдатель (OOP3Behav1).mp4	261.1 MB	a year ago
2-3. События и делегаты в паттерне Наблюдатель (OOP3Behav1), обзор паттернов Посредник и Цепочка обязанностей.mp4	272.5 MB	10 months ago
2-4. Обзор сложных паттернов поведения (Посетитель, Интерпретатор).mp4	177.5 MB	10 months ago
3-1. Паттерн Наблюдатель, модель проталкивания (OOP3Behav2), паттерн Итератор, интерфейсы IEnumerator и IEnumerable (OOP3Behav7).mp4	262.9 MB	a year ago
3-2. Паттерн Итератор, часть 2 (OOP3Behav7).mp4	159.5 MB	a year ago

11 files 2.6 GB

Nextcloud – a safe home for all your data
Get your own free account

Раздел сайта ptaskbook.com с описанием задачника PT for OOP



Programming Taskbook

ptaskbook.com/ru/ptforoop/index.php

Programming Taskbook

Электронный задачник по программированию

© М. Э. Абрамян (Южный федеральный университет, Университет МГУ-ППИ в Шэньчжэне), 1998–2025

Главная

Задания

Решения

Teacher Pack

PT for MPI

PT for Bio

PT for LINQ

PT for Exam

PT for STL

PT for MPI-2

PT for OOP

Общее описание

Указатель паттернов

Видеозаписи занятий

Группы заданий

OOP0Begin

OOP1Creat

OOP2Struc

OOP3Behav

Выполнение задания

OOP1Creat1

Знакомство с заданием

Решение на языке C++

Решение на языке Python

Решение на языках C# и PascalABC.NET

Решение на языках Ruby, Java и Julia

PT for OOP | Общее описание

Скачать дистрибутив электронного задачника Programming Taskbook for OOP (русская версия 1.1)

Общее описание

Программный комплекс «Электронный задачник по паттернам проектирования Programming Taskbook for OOP» (PT for OOP) содержит дополнительные компоненты электронного задачника Programming Taskbook, предназначенные для изучения классических паттернов проектирования, описанных в книге Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования (издание на русском языке: СПб.: Питер, 2015. — 368 с.). В дальнейшем ссылки на эту книгу будут даваться в виде GoF («Gang of Four», как часто называют команду ее авторов).

Для возможности использования данного комплекса его следует установить в системный каталог базового варианта электронного задачника Programming Taskbook версии не ниже 4.22 (обычно системным каталогом задачника является каталог C:\Program Files (x86)\PT4).

Задания могут выполняться на любых языках и в любых программных средах, поддерживаемых электронным задачником Programming Taskbook.

Комплекс PT for OOP является свободно распространяемым программным продуктом (freeware); он может использоваться как с полным вариантом задачника PT4Complete, так и со свободно распространяемым мини-вариантом PT4Mini.

В состав комплекса входит гипертекстовая справочная система PTforOOP.chm, содержащая ту же информацию, что и раздел сайта ptaskbook.com, посвященный задачнику PT for OOP.

Программный комплекс «Электронный задачник по паттернам проектирования Programming Taskbook for OOP» зарегистрирован в Реестре программ для ЭВМ 9 августа 2022 г.

Росси́йская Федера́ция

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2022665039

«Электронный задачник по паттернам проектирования Programming Taskbook for OOP»

Примолвилец: Абрамян Михаил Эдуардович (RU)

Автор(ы) Абрамян Михаил Эдуардович (RU)

Заявка № 2022663785

Дата поступления 21 июля 2022 г.

Дата государственной регистрации 9 августа 2022 г.

Реестр программ для ЭВМ

Спасибо за внимание

Электронный задачник Programming Taskbook for OOP:
<http://ptaskbook.com/ru/ptforoop/>

