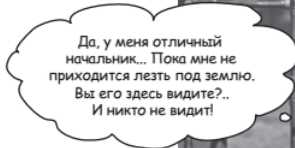


Паттерн Шаблонный метод

Наумов Д.А., доц. каф. КТ

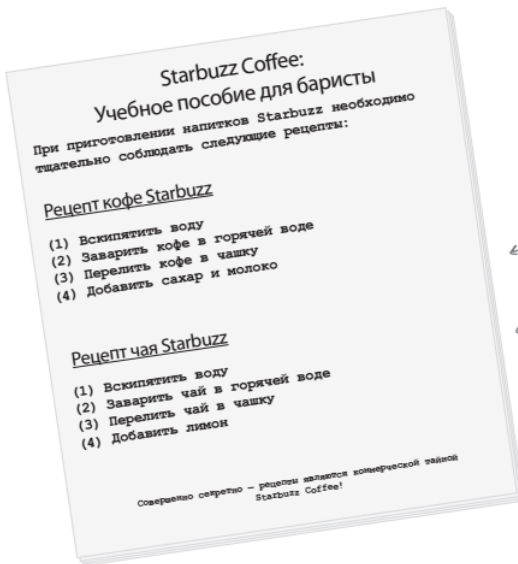
Основы программной инженерии, 2019

Паттерн Шаблонный Метод



Да, у меня отличный
начальник... Пока мне не
приходится лезть под землю.
Вы его здесь видите?..
И никто не видит!





Рецепт приготовления кофе очень похож на рецепт приготовления чая, не правда ли?

Класс Coffee для приготовления кофе:

```

type
  Coffee = class
  public
    procedure prepareRecipe();

    procedure boilWater();
    procedure brewCoffeeGrinds();
    procedure pourInCup();
    procedure addSugarAndMilk();
  end;
procedure Coffee.boilWater();
begin
  writeln('Boiling Water');
end;
procedure Coffee.brewCoffeeGrinds();
begin
  writeln('Dripping Coffee through filter');
end;
procedure Coffee.pourInCup();
begin
  writeln('Pouring into cup');
end;
procedure Coffee.addSugarAndMilk();
begin
  writeln('Adding sugar and milk');
end;
  
```

Рецепт кофе взят прямо из учебного пособия.

Каждый шаг реализован в виде отдельного метода.

Каждый из этих методов реализует один шаг алгоритма: кипячение воды, настаивание кофе, разливание по чашкам, добавление сахара и молока.

```
type
  Tea = class
  public
    procedure prepareRecipe();

    procedure boilWater();
    procedure steepTeaBag();
    procedure pourInCup();
    procedure addLemon();
  end;
  procedure Tea.prepareRecipe();
  begin
    boilWater();
    steepTeaBag();
    pourInCup();
    addLemon();
  end;
end;
```

Реализация очень
похожа на реализацию
Coffee; шаги 2 и 4
различаются, но рецепт
почти не изменился.



```
procedure Tea.boilWater();
begin
  writeln('Boiling Water');
end;

procedure Tea.steepTeaBag();
begin
  writeln('Steeping Tea bag into cup');
end;

procedure Tea.addLemon();
begin
  writeln('Adding a lemon');
end;

procedure Tea.pourInCup();
begin
  writeln('Pouring into cup');
end;
```

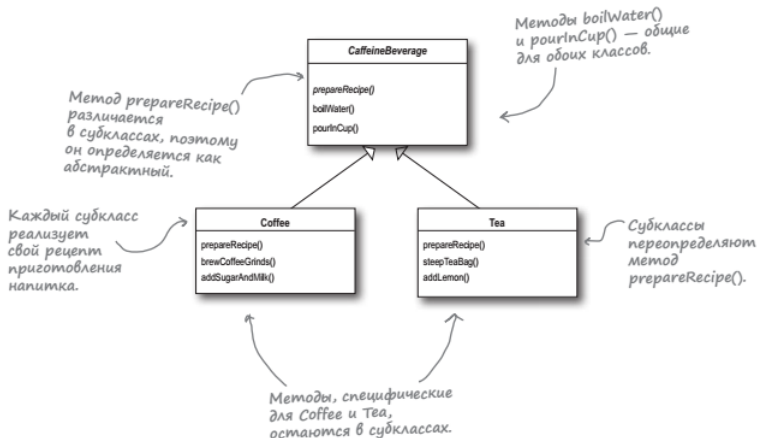
Эти два
метода
специфичны
для класса Tea.

А эти два метода
в точности
совпадают
с методами
Coffee! Имеет
место явное
дублирование кода.



Дублирование кода —
верный признак того, что
в архитектуру необходимо вносить
изменения. Раз чай и кофе так
похожи, может, стоит выделить
их сходные аспекты
в базовый класс?

Сэр, Вам налить абстрактного кофе?



Продолжаем переработку...

Что еще общего у классов Coffee и Tea? Начнем с рецептов.

Рецепт кофе Starbuzz

- (1) Вскипятить воду
- (2) Заварить кофе в горячей воде
- (3) Перелить кофе в чашку
- (4) Добавить сахар и молоко

Рецепт чая Starbuzz

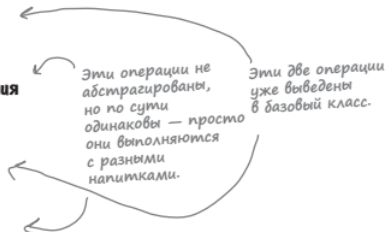
- (1) Вскипятить воду
- (2) Заварить чай в горячей воде
- (3) Перелить чай в чашку
- (4) Добавить лимон

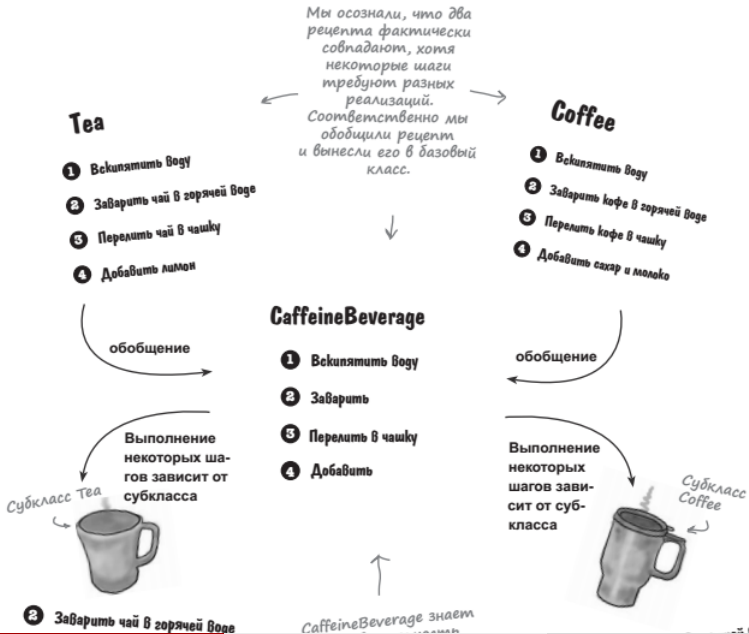
Обратите внимание: оба рецепта следуют одному алгоритму:

- ❶ Вскипятить воду.
- ❷ Использовать горячую воду для настаивания кофе или чая.
- ❸ Перелить напиток в чашку.
- ❹ Добавить соответствующие дополнения в напиток.

Эти операции не абстрагированы, но по сути одинаковы — просто они выполняются с разными напитками.

Эти две операции уже выведены в базовый класс.





Паттерн Шаблонный Метод

```
type
  CaffeineBeverage = class
  public
    procedure prepareRecipe();

    procedure brew(); virtual; abstract;
    procedure addCondiments(); virtual; abstract;
    procedure boilWater(); //реализация
    procedure pourInCup(); //реализация
  end;
  procedure Tea.prepareRecipe();
begin
  boilWater();
  brew();
  pourInCup();
  addCondiments ();
end;
  procedure CaffeineBeverage.boilWater();
begin
  writeln('Boiling Water');
end;
  procedure CaffeineBeverage.pourInCup();
begin
  writeln('Pouring into cup');
end;
```

prepareRecipe() — шаблонный метод. Почему?

Потому что:

- (1) Бесспорно, это метод.
- (2) Он служит шаблоном для алгоритма — в данном случае алгоритма приготовления напитка.

В шаблоне каждый шаг алгоритма представлен некоторым методом.

Реализация одних методов предоставляется этим классом...

...реализация других предоставляется subclasses.

Методы, которые должны предоставляться subclasses, объявляются абстрактными.

Готовим чай...

Давайте последовательно разберем процедуру приготовления чая, обращая особое внимание на то, как работает шаблонный метод. Вы увидите, что шаблонный метод управляет алгоритмом; в некоторых точках алгоритма он дает возможность subclasses предоставить свою реализацию...

За сценкой 



Что дает Шаблонный Метод?



Тривиальная реализация Tea и Coffee

Алгоритм определяется классами
Coffee и Tea.

Частичное дублирование кода
в классах Coffee и Tea.

Модификация алгоритма требует
открытия субклассов и внесения
множественных изменений.

Добавление новых классов в такой
структуре требует значительной
работы.

Знание алгоритма и его реализации



Новый класс CaffeineBeverage на базе Шаблонного Метода

Алгоритм определяется классом
CaffeineBeverage.

Класс CaffeineBeverage обеспечи-
вает повторное использование кода
между субклассами.

Алгоритм находится в одном месте,
в котором вносятся изменения в коде
алгоритма.

Структура классов на базе Шаблонно-
го Метода обеспечивает простое до-
бавление новых классов — они лишь
должны реализовать пару методов.

Вся информация об алгоритме

Определение паттерна Шаблонный Метод

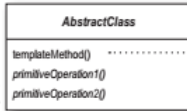
Паттерн Шаблонный Метод задает «скелет» алгоритма в методе, оставляя определение реализации некоторых шагов subclasses. Subclasses могут переопределять некоторые части алгоритма без изменения его структуры.

Шаблонный метод использует методы `primitiveOperation` в реализации алгоритма. Он изолирован от фактической реализации этих операций.

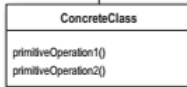
Класс `AbstractClass` содержит шаблонный метод...

...и абстрактные версии операций, используемых в шаблонном методе.

В схеме может быть задействовано много классов `ConcreteClass`, каждый из которых реализует полный набор операций, необходимых для работы шаблонного метода.



`primitiveOperation1();`
`primitiveOperation2();`



`ConcreteClass` реализует абстрактные операции, вызываемые в ходе выполнения `templateMethod()`.



Код под увеличительным стеклом

Давайте повнимательнее присмотримся к определению AbstractClass, включая шаблонный метод и примитивные операции.

Абстрактный класс: он должен subclass'иться классами, предоставляющими реализацию операций.

Шаблонный метод; объявляется с ключевым словом final, чтобы subclasses не могли изменить последовательность шагов в алгоритме.

```
type
  AbstractClass = class
  public
    procedure templateMethod();

    procedure primitiveMethod1(); virtual; abstract;
    procedure primitiveMethod2(); virtual; abstract;
    procedure concreteOperation();
  end;

  procedure AbstractClass.templateMethod();
begin
  primitiveMethod1();
  primitiveMethod2();
  concreteOperation();
end;
```

В данном примере две примитивные операции должны реализоваться конкретными subclasses.

Шаблонный метод определяет последовательность шагов, каждый из которых представлен методом.

Конкретная операция, определенная в абстрактном классе.



Код под микроскопом

А сейчас мы еще подробнее рассмотрим методы, которые могут определяться в абстрактном классе:

*В templateMethod()
включен вызов нового
метода.*

```
procedure AbstractClass.templateMethod()
begin
    primitiveMethod1();
    primitiveMethod2();
    concreteOperation();
    hook();
end;
```

```
type
    AbstractClass = class
    public
        procedure templateMethod();
        procedure primitiveMethod1(); virtual; abstract;
        procedure primitiveMethod2(); virtual; abstract;
        procedure concreteOperation();
        procedure hook();
    end;
```

*Примитивы-методы никуда
не делись; они объявлены
абстрактными и реализуются
конкретными subclasses.*

```
procedure primitiveMethod1(); virtual; abstract;
procedure primitiveMethod2(); virtual; abstract;
```

*Конкретная операция определяется
в абстрактном классе.*

```
procedure concreteOperation();
begin
    //какая-то реализация
end;
```

*Она может использоваться
как напрямую шаблонным методом,
так и subclasses.*

```
procedure AbstractClass.hook();
begin
    //пустой метод
end;
```

*Subclasses могут переопределять такие
методы (называемые «перехватчиками»),
но не обязаны это делать.*

*Конкретный
метод, который
не делает ничего!*

Перехватчики в паттерне Шаблонный Метод

```

type
  CaffeineBeverage = class
    procedure prepareRecipe();
    ...
    procedure addCondiments(); virtual; abstract;
    function customerWantsCondiments(): boolean;
  end;
procedure CaffeineBeverage.prepareRecipe();
begin
  boilWater();
  brew();
  pourInCup();
  if customerWantsCondiments then addCondiments();
end;
function CaffeineBeverage.customerWantsCondiments(): boolean;
begin
  Result := true;
end
  
```

Добавляем условную конструкцию, результатом которой определяется вызов конкретного метода customerWantsCondiments(). Только если вызов вернем true, мы вызываем addCondiments().

Метод с (почти) пустой реализацией по умолчанию: просто возвращает true, и не делает ничего более.

Перехватчик: subclasses может переопределить этот метод, но не обязан этого делать.



```

type
  CaffeineWithHook = class(CaffeineBeverage)
  ...
    procedure addCondiments();
    function customerWantsCondiments(): boolean; override;
  end;
function CaffeineBeverage.customerWantsCondiments(): boolean;
var
  answer: string;
begin
  writeln('Would you like sugar and milk (y/n)?');
  readln(answer);
  Result := (answer='y');
end;
procedure CaffeineWithHook.addCondiments();
begin
  writeln('Adding Sigar and Milk.');
```

Здесь вы
переопределяете
перехватчик
и задаете нужную
функциональность.

.Предлагаем поль-
зователю принять
решение и возвра-
щаем true или false
в зависимости от
полученных данных.