|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет экономики, менеджмента и бизнес-информатики* |
|  |
| Демина Дарья Сергеевна  **ПАТТЕРН «СТРОИТЕЛЬ»**  Лабораторная работа  по направлению подготовки *09.03.04 Программная инженерия*  образовательная программа «Программная инженерия»   |  |  | | --- | --- | |  | Руководитель  Преподаватель кафедры информационных технологий в бизнесе.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  А.В. Кычкин | |

Пермь, 2017 год

# Назначение и цель применения паттерна Builder

Строитель – паттерн проектирования, который позволяет создавать сложные объекты пошагово. Паттерн позволяет создавать один и тот же объект в различных представлениях.

Преимущества:

* Более тонкий контроль надо процессом.
* Код конструирования находится отдельно от кода представления.
* Возможность изменить внутреннее представление продукта.

# UML-диаграмма



Director – распорядитель; определяет порядок вызова шагов строительства.

Builder – определяет интерфейс для создания частей объекта Product, объявляет шаги конструирования, общие для всех видов строителей.

ConcreteBuilder – Создает объект Product и определяет интерфейс для доступа к нему, реализуют шаги каждый по-своему.

Product – объект, который должен быть создан (все части объекта заключены в списке parts); Продукты, сделанные разными строителями, не обязаны иметь общий интерфейс.

# Области применения

Паттерн «Строитель» применяется, когда:

1. Алгоритм создания сложного объекта не должен зависеть от того, из каких частей состоит объект и как они стыкуются между собой.
2. Процесс конструирования должен обеспечивать различные представления конструируемого объекта.

# Особенности паттерна

Особенность паттерна заключается в том, что в нем нет необходимость указывать большое количество параметров, основная часть которых не будет использоваться при каждом вызове, как это приходится делать в конструкторе.

# Пример реализации

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication2

{

class Program

{

static void Main(string[] args)

{

// содаем объект пекаря

Baker baker = new Baker();

// создаем билдер для ржаного хлеба

BreadBuilder builder = new RyeBreadBuilder();

// выпекаем

Bread ryeBread = baker.Bake(builder);

Console.WriteLine("Ржаной хлеб");

Console.WriteLine(ryeBread.ToString());

// оздаем билдер для пшеничного хлеба

builder = new WheatBreadBuilder();

Bread wheatBread = baker.Bake(builder);

Console.WriteLine("Пшеничный хлеб");

Console.WriteLine(wheatBread.ToString());

Console.Read();

}

}

// абстрактный класс строителя

abstract class BreadBuilder

{

public Bread Bread { get; private set; }

public void CreateBread()

{

Bread = new Bread();

}

public abstract void SetFlour();

public abstract void SetSalt();

public abstract void SetAdditives();

}

// пекарь

class Baker

{

public Bread Bake(BreadBuilder breadBuilder)

{

breadBuilder.CreateBread();

breadBuilder.SetFlour();

breadBuilder.SetSalt();

breadBuilder.SetAdditives();

return breadBuilder.Bread;

}

}

// строитель для ржаного хлеба

class RyeBreadBuilder : BreadBuilder

{

public override void SetFlour()

{

this.Bread.Flour = new Flour { Sort = "Ржаная мука 1 сорт" };

}

public override void SetSalt()

{

this.Bread.Salt = new Salt();

}

public override void SetAdditives()

{

// не используется

}

}

// строитель для пшеничного хлеба

class WheatBreadBuilder : BreadBuilder

{

public override void SetFlour()

{

this.Bread.Flour = new Flour { Sort = "Пшеничная мука высший сорт" };

}

public override void SetSalt()

{

this.Bread.Salt = new Salt();

}

public override void SetAdditives()

{

this.Bread.Additives = new Additives { Name = "улучшитель хлебопекарный" };

}

}

//мука

class Flour

{

// какого сорта мука

public string Sort { get; set; }

}

// соль

class Salt

{ }

// пищевые добавки

class Additives

{

public string Name { get; set; }

}

class Bread

{

// мука

public Flour Flour { get; set; }

// соль

public Salt Salt { get; set; }

// пищевые добавки

public Additives Additives { get; set; }

public override string ToString()

{

Console.WriteLine("Состав:");

StringBuilder sb = new StringBuilder();

if (Flour != null)

sb.Append(Flour.Sort + "\n");

if (Salt != null)

sb.Append("Соль \n");

if (Additives != null)

sb.Append("Добавки: " + Additives.Name + " \n");

return sb.ToString();

}

}

}