Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине «Архитектура, технологии и инструментальные средства разработки ПО»

**«Порождающие шаблоны»**

Выполнил

студент группы КТбо3-1 Самардак А. В

Принял

Доцент каф. МОП ЭВМ Лутай В. Н

**Задание**

Реализовать программу, которая переводит по запросу входную строку десятичных цифр в строку цифр в системе счисления 2, 4, 8, 16. Создать Фабричный метод., в котором выделен класс, осуществляющий генерацию объектов. Предусмотреть возможность хранения созданных объектов.

*Примечание:* выполнил задание, чтобы можно было использовать все системы счисления от 1-ой системы счисления до 26-ой системы счисления (до 26-ой системы счисления, потому что в английском словаре всего 26 букв).

**Теория**

1. ***Понятие***

Фабричный метод, или виртуальный конструктор — порождающий шаблон проектирования, предоставляющий подклассам (дочерним классам) интерфейс для создания экземпляров некоторого класса. В момент создания наследники могут определить, какой класс создавать. Иными словами, данный шаблон делегирует создание объектов наследникам родительского класса. Это позволяет использовать в коде программы не конкретные классы, а манипулировать абстрактными объектами на более высоком уровне.

1. ***Цель***

Определяет интерфейс для создания объекта, но оставляет подклассам решение о том, на основании какого класса создавать объект. Фабричный метод позволяет классу делегировать создание подклассов. Используется, когда:

1. Классу заранее неизвестно, объекты каких подклассов ему нужно создавать.
2. Класс спроектирован так, чтобы объекты, которые он создаёт, специфицировались подклассами.
3. Класс делегирует свои обязанности одному из нескольких вспомогательных подклассов, и планируется локализовать знание о том, какой класс принимает эти обязанности на себя.
4. ***Структура***
5. *Product* — продукт

* определяет интерфейс объектов, создаваемых абстрактным методом.

1. *ConcreteProduct* — конкретный продукт

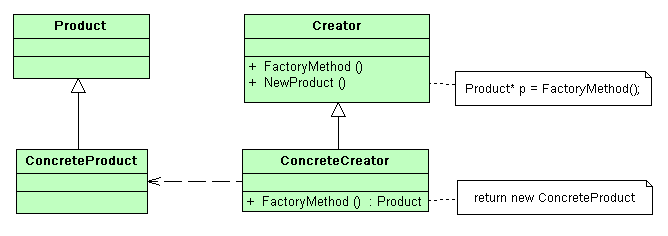
* реализует интерфейс Product.

1. *Creator* — создатель

* объявляет фабричный метод, который возвращает объект типа Product. Может также содержать реализацию этого метода «по умолчанию».
* вызывать фабричный метод для создания объекта типа Product.

1. *ConcreteCreator* — конкретный создатель

* переопределяет фабричный метод таким образом, чтобы он создавал и возвращал объект класса ConcreteProduct.



1. ***Достоинства***
2. Позволяет сделать код создания объектов более универсальным, не привязываясь к конкретным классам (ConcreteProduct), а оперируя лишь общим интерфейсом (Product).
3. Позволяет установить связь между параллельными иерархиями классов.
4. ***Недостатки***
5. Необходимость создавать наследника Creator для каждого нового типа продукта (ConcreteProduct).

**Результат выполнения программы**

Результат правильности выполнения программы проверял на разных доступных Web-ресурсах. Программа корректно выполняла поставленные ей задачи.

(Рис. 1) Показывает выполнение программы при вычислении двоичной системы счисления.

Изображение выглядит как текст

Автоматически созданное описание

*Рис.1*

(Рис. 2) Показывает выполнение программы при вычислении четверичной системы счисления.

Изображение выглядит как текст

Автоматически созданное описание

*Рис.2*

(Рис. 3) Показывает выполнение программы при вычислении восьмеричной системы счисления.

Изображение выглядит как текст

Автоматически созданное описание

*Рис.3*

(Рис. 4) Показывает выполнение программы при вычислении шестнадцатеричной системы счисления.

**Изображение выглядит как текст

Автоматически созданное описание**

*Рис.4*

**Листинг программы**

1. ***Main.cs***

using System;

namespace Soft\_Laba1

{

internal class main

{

static void Main(string[] args)

{

ConsoleIt console = new ConsoleIt();

console.StartConsole();

}

}

}

1. ***ConsoleIt.cs***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Soft\_Laba1

{

internal class ConsoleIt

{

public void StartConsole()

{

int count = GetCount();

int[] array = CreateArray(count);

int numberSystem = GetNumberСalcSystem();

CreateFactory(numberSystem);

string[] convertArray = ConvertArray(array, numberSystem);

PrintArray(array, convertArray);

}

private int GetCount()

{

Console.WriteLine("Сколько чисел вы хотите ввести?:");

int count = Convert.ToInt32(Console.ReadLine());

return count;

}

private int[] CreateArray(int count)

{

Console.WriteLine("Введите числа:");

int[] array = new int[count];

for(int i = 0; i < count; ++i)

{

array[i] = Convert.ToInt32(Console.ReadLine());

}

return array;

}

private int GetNumberСalcSystem()

{

Console.WriteLine("Ввидете число, систему счисления которого хотите использовать:");

int nymberSystem = Convert.ToInt32(Console.ReadLine());

return nymberSystem;

}

private void CreateFactory(int numberSystem)

{

Factory factory = new Factory();

NumberSystem = factory.createSystem(numberSystem);

}

private string[] ConvertArray(int[] array, int numberSystem)

{

string[] convertArray = new string[array.Length];

for(int i = 0; i < array.Length; ++i)

{

convertArray[i] = NumberSystem.Convert(array[i], numberSystem);

}

return convertArray;

}

private void PrintArray(int[] array, string[] convertArray)

{

Console.WriteLine("Выходные данные:");

for(int i = 0; i < convertArray.Length; ++i)

{

Console.WriteLine(array[i] + " : " + convertArray[i]);

}

}

NumberSystem NumberSystem;

}

}

1. ***NumberSystem.cs***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Soft\_Laba1

{

abstract class NumberSystem

{

public abstract string Convert(int number, int numberSystem);

public virtual string ReverseNumber(string number)

{

char[] charArray = number.ToCharArray();

Array.Reverse(charArray);

number = new string(charArray);

return number;

}

}

}

1. ***SimpleNumSys.cs***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Soft\_Laba1

{

internal class SimpleNumSys : NumberSystem

{

public override string Convert(int number, int numberSystem)

{

string binaryNumber = "";

while (true)

{

binaryNumber += number % numberSystem;

if (number / numberSystem == 0)

{

break;

}

number = number / numberSystem;

}

return ReverseNumber(binaryNumber);

}

}

}

1. ***ComplexNumSys.cs***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Soft\_Laba1

{

internal class ComplexNumSys : NumberSystem

{

public override string Convert(int number, int numberSystem)

{

string binaryNumber = "";

while (true)

{

int i = number % numberSystem;

if (i > 9 && i < 26)

{

binaryNumber += CharArray[i - 10];

}

else

{

binaryNumber += i;

}

if (number / numberSystem == 0)

{

break;

}

number = number / numberSystem;

}

return ReverseNumber(binaryNumber);

}

char[] CharArray =

{'A', 'B', 'C', 'D', 'E',

'F', 'G', 'H', 'I', 'J',

'K', 'L', 'M', 'N', 'O',

'P', 'Q', 'R', 'S', 'T',

'U', 'V', 'W', 'X', 'Y', 'Z'};

}

}

1. ***Factory.cs***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Soft\_Laba1

{

internal class Factory

{

public NumberSystem createSystem(int numberSystem)

{

if(numberSystem < 10)

{

return new SimpleNumSys();

}

else

{

return new ComplexNumSys();

}

}

}

}

**Вспомогательные ссылки**

1. [Объяснение реализации Фабричного метода простыми словами.](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B1%D1%80%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))
2. [Объяснение реализации Фабричного метода на примерах. Более углублённое погружение на реальные примеры использования данного метода.](https://refactoring.guru/ru/design-patterns/factory-method)
3. [Мой проект на github.](https://github.com/input-love/Soft_Laba1)