

# МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»**

# МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»**

# МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»**

# МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»**

**Факультет управления и информатики в технологических системах**

**Кафедра**[**корпоративных информационных систем и программирования**](https://vsuet.ru/obuchenie/faculties/uits/k_kisip)

**Направление подготовки 09.03.02**

***(шифр)***

**Информационные системы и технологии**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(наименование направления подготовки)*

**Отчет**

**по практическому заданию № 2**

Выполнил студент гр. У-213

Боровкова Анастасия Дмитриевна

(ф.и.о)

**\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***(****подпись****)***

Проверил(а):

Демичев С.Е.

**\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***(****подпись****)***

Воронеж – 2023

**using** System;

**namespace** RefactoringGuru.DesignPatterns.AbstractFactory.Conceptual

{

*// Интерфейс Абстрактной Фабрики объявляет набор методов, которые возвращают*

*// различные абстрактные продукты.  Эти продукты называются семейством и*

*// связаны темой или концепцией высокого уровня. Продукты одного семейства*

*// обычно могут взаимодействовать между собой. Семейство продуктов может*

*// иметь несколько вариаций,  но продукты одной вариации несовместимы с*

*// продуктами другой.*

**public** **interface** IAbstractFactory

    {

        IAbstractProductA CreateProductA();

        IAbstractProductB CreateProductB();

    }

*// Конкретная Фабрика производит семейство продуктов одной вариации. Фабрика*

*// гарантирует совместимость полученных продуктов.  Обратите внимание, что*

*// сигнатуры методов Конкретной Фабрики возвращают абстрактный продукт, в то*

*// время как внутри метода создается экземпляр  конкретного продукта.*

**class** ConcreteFactory1 : IAbstractFactory

    {

**public** IAbstractProductA CreateProductA()

        {

**return** new ConcreteProductA1();

        }

**public** IAbstractProductB CreateProductB()

        {

**return** new ConcreteProductB1();

        }

    }

*// Каждая Конкретная Фабрика имеет соответствующую вариацию продукта.*

**class** ConcreteFactory2 : IAbstractFactory

    {

**public** IAbstractProductA CreateProductA()

        {

**return** new ConcreteProductA2();

        }

**public** IAbstractProductB CreateProductB()

        {

**return** new ConcreteProductB2();

        }

    }

*// Каждый отдельный продукт семейства продуктов должен иметь базовый*

*// интерфейс. Все вариации продукта должны реализовывать этот интерфейс.*

**public** **interface** IAbstractProductA

    {

**string** UsefulFunctionA();

    }

*// Конкретные продукты создаются соответствующими Конкретными Фабриками.*

**class** ConcreteProductA1 : IAbstractProductA

    {

**public** **string** UsefulFunctionA()

        {

**return** "Результат работы продукта A1.";

        }

    }

**class** ConcreteProductA2 : IAbstractProductA

    {

**public** **string** UsefulFunctionA()

        {

**return** "Результат работы продукта A2.";

        }

    }

*// Базовый интерфейс другого продукта. Все продукты могут взаимодействовать*

*// друг с другом, но правильное взаимодействие возможно только между*

*// продуктами одной и той же конкретной вариации.*

**public** **interface** IAbstractProductB

    {

*// Продукт B способен работать самостоятельно...*

**string** UsefulFunctionB();

*// ...а также взаимодействовать с Продуктами А той же вариации.*

*//*

*// Абстрактная Фабрика гарантирует, что все продукты, которые она*

*// создает, имеют одинаковую вариацию и, следовательно, совместимы.*

**string** AnotherUsefulFunctionB(IAbstractProductA collaborator);

    }

*// Конкретные Продукты создаются соответствующими Конкретными Фабриками.*

**class** ConcreteProductB1 : IAbstractProductB

    {

**public** **string** UsefulFunctionB()

        {

**return** "Результат работы продукта B1.";

        }

*// Продукт B1 может корректно работать только с Продуктом A1. Тем не*

*// менее, он принимает любой экземпляр Абстрактного Продукта А в*

*// качестве аргумента.*

**public** **string** AnotherUsefulFunctionB(IAbstractProductA collaborator)

        {

**var** result = collaborator.UsefulFunctionA();

**return** $"Результат сотрудничества B1 с ({result})";

        }

    }

**class** ConcreteProductB2 : IAbstractProductB

    {

**public** **string** UsefulFunctionB()

        {

**return** "Результат работы продукта B2.";

        }

*// Продукт B2 может корректно работать только с Продуктом A2. Тем не*

*// менее, он принимает любой экземпляр Абстрактного Продукта А в качестве*

*// аргумента.*

**public** **string** AnotherUsefulFunctionB(IAbstractProductA collaborator)

        {

**var** result = collaborator.UsefulFunctionA();

**return** $"Результат сотрудничества B2 с ({result})";

        }

    }

*// Клиентский код работает с фабриками и продуктами только через абстрактные*

*// типы: Абстрактная Фабрика и Абстрактный Продукт. Это позволяет передавать*

*// любой подкласс фабрики или продукта клиентскому коду, не нарушая его.*

**class** Client

    {

**public** **void** Main()

        {

*// Клиентский код может работать с любым конкретным классом фабрики.*

            Console.WriteLine("Клиент: Тестирование клиентского кода с использованием первого заводского типа...");

            ClientMethod(new ConcreteFactory1());

            Console.WriteLine();

            Console.WriteLine("Клиент: Тестирование того же кода клиента со вторым заводским типом...");

            ClientMethod(new ConcreteFactory2());

        }

**public** **void** ClientMethod(IAbstractFactory factory)

        {

**var** productA = factory.CreateProductA();

**var** productB = factory.CreateProductB();

            Console.WriteLine(productB.UsefulFunctionB());

            Console.WriteLine(productB.AnotherUsefulFunctionB(productA));;

        }

    }

**class** Program

    {

**static** **void** Main(**string**[] args)

        {

            new Client().Main();

            Console.ReadKey();

        }

    }

}