

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №6

Выполнила
Радиончик С.С.,
студентка группы ПО-5

Проверил
Крощенко А.А.,
ст. преп. Кафедры ИИТ,
«__» _____ 2021 г.

Брест, 2021

Цель работы: приобрести навыки применения паттернов проектирования при решении практических задач.

Вариант 6.

Общее задание.

- Прочитать задания, взятые из каждой группы.
- Определить паттерн проектирования, который может использоваться при реализации задания. Пояснить свой выбор.
- Реализовать фрагмент программной системы, используя выбранный паттерн. Реализовать все необходимые дополнительные классы.

Задание 1. Музыкальный магазин. Должно обеспечиваться одновременное обслуживание нескольких покупателей. Магазин должен предоставлять широкий выбор товаров различных музыкальных направлений.

Задание 2. Учетная запись покупателя книжного интернет-магазина. Предусмотреть различные уровни учетки в зависимости от активности покупателя. Дополнительные уровни добавляют функциональные возможности и открывают доступ к уникальным предложениям.

Задание 3. Проект «Принтер». Предусмотреть выполнение операций (печать, загрузка бумаги, извлечение зажатой бумаги, заправка картриджа), режимы – ожидание, печать документа, зажатие бумаги, отказ – при отсутствии бумаги или краски, атрибуты – модель, количество листов в лотке, % краски в картридже, вероятность зажатия.

Выполнение:

*1) **Фабричный метод** — это порождающий паттерн проектирования, который решает проблему создания различных продуктов, без указания конкретных классов продуктов.*

Код программы.

AcademicProduct.cs

```
using System;
using System.Threading;

namespace Lab6_1_1
{
    class AcademicProduct : IProduct
    {
        public void ChooseInstrument(string name)
        {
            Thread.Sleep(8000);
            Console.WriteLine(name);

            Console.WriteLine("Выберите музыкальный инструмент:\n1 - блокфлейта\n2 - гlockеншпиль\n3 - синтезатор\n4 - скрипка");
            bool choice = true;
        }
    }
}
```

```

        while (choice)
        {
            var n = Console.ReadLine();

            switch (n)
            {
                case "1":
                    Console.WriteLine("Блокфлейта. ");
                    choice = false;
                    break;
                case "2":
                    Console.WriteLine("Глокеншпиль. ");
                    choice = false;
                    break;
                case "3":
                    Console.WriteLine("Синтезатор. ");
                    choice = false;
                    break;
                case "4":
                    Console.WriteLine("Скрипка. ");
                    choice = false;
                    break;
                default:
                    Console.WriteLine("Вы ввели неправильное значение.
Попробуйте ещё");
                    break;
            }
        }
    }
}

```

AcademicSelector.cs

```

namespace Lab6_1_1
{
    class AcademicSelector : Selector
    {
        public override void FactoryMethod(string name)
        {
            AcademicProduct p1 = new AcademicProduct();
            p1.ChooseInstrument(name);
        }
    }
}

```

Client.cs

```

using System;

namespace Lab6_1_1
{
    class Client
    {
        public void Main(string name)
        {
            Console.WriteLine(name);
            Choice(name);
        }

        public void Choice(string name)
        {
            Console.WriteLine("Выберите музыкальное направление:\n1 - народная музыка\n2 -
духовная музыка\n3 - академическая музыка\n4 - популярная музыка");
        }
    }
}

```

```

        bool choice = true;

        while (choice)
        {
            var n = Console.ReadLine();

            switch (n)
            {
                case "1":
                    choice = false;
                    ClientCode(new FolkSelector(), name);
                    break;
                case "2":
                    choice = false;
                    ClientCode(new ScaredSelector(), name);
                    break;
                case "3":
                    choice = false;
                    ClientCode(new AcademicSelector(), name);
                    break;
                case "4":
                    choice = false;
                    ClientCode(new PopSelector(), name);
                    break;
                default:
                    Console.WriteLine("Вы ввели неправильное значение.
Попробуйте ещё");
                    break;
            }
        }
    }

    public void ClientCode(Selector selector, string name)
    {
        selector.FactoryMethod(name);
        selector.Buy(name);
    }
}

```

FolkProduct.cs

```

using System;
using System.Threading;

namespace Lab6_1_1
{
    class FolkProduct : IProduct
    {
        public void ChooseInstrument(string name)
        {
            Thread.Sleep(8000);
            Console.WriteLine(name);

            Console.WriteLine("Выберите музыкальный инструмент:\n1 - гусли\n2 - бубен\n3 -
волынка\n4 - свирель");
            bool choice = true;

            while (choice)
            {
                var n = Console.ReadLine();

                switch (n)
                {
                    case "1":

```

```

        Console.WriteLine("Гусли. ");
        choice = false;
        break;
    case "2":
        Console.WriteLine("Бубен. ");
        choice = false;
        break;
    case "3":
        Console.WriteLine("Волянка. ");
        choice = false;
        break;
    case "4":
        Console.WriteLine("Свирель. ");
        choice = false;
        break;
    default:
        Console.WriteLine("Вы ввели неправильное значение.
        Попробуйте ещё");
        break;
    }
}
}
}

```

FolkSelector.cs

```

namespace Lab6_1_1
{
    class FolkSelector : Selector
    {
        public override void FactoryMethod(string name)
        {
            FolkProduct p1 = new FolkProduct();
            p1.ChooseInstrument(name);
        }
    }
}

```

Iproduct.cs

```

namespace Lab6_1_1
{
    interface IProduct
    {
        public void ChooseInstrument(string name);
    }
}

```

PopProduct.cs

```

using System;
using System.Threading;

namespace Lab6_1_1
{
    class PopProduct : IProduct
    {
        public void ChooseInstrument(string name)
        {
            Thread.Sleep(8000);
            Console.WriteLine(name);

            Console.WriteLine("Выберите музыкальный инструмент:\n1 - гитара\n2 -
            барабан\n3 - мелодика\n4 - MIDI-клавиатура");
        }
    }
}

```

```

        bool choice = true;

        while (choice)
        {
            var n = Console.ReadLine();

            switch (n)
            {
                case "1":
                    Console.WriteLine("Гитара. ");
                    choice = false;
                    break;
                case "2":
                    Console.WriteLine("Барабан. ");
                    choice = false;
                    break;
                case "3":
                    Console.WriteLine("Мелодика. ");
                    choice = false;
                    break;
                case "4":
                    Console.WriteLine("MIDI-клавиатура. ");
                    choice = false;
                    break;
                default:
                    Console.WriteLine("Вы ввели неправильное значение.
Попробуйте ещё");
                    break;
            }
        }
    }
}

```

PopSelector.cs

```

namespace Lab6_1_1
{
    class PopSelector : Selector
    {
        public override void FactoryMethod(string name)
        {
            PopProduct p1 = new PopProduct();
            p1.ChooseInstrument(name);
        }
    }
}

```

Program.cs

```

using System.Threading;
using System.Threading.Tasks;

namespace Lab6_1_1
{
    class Program
    {
        static void Main(string[] args)
        {
            Parallel.Invoke(() => new Client().Main("Customer 1"), () =>
            { Thread.Sleep(3000); new Client().Main("Customer 2"); }, () =>
            { Thread.Sleep(6000); new Client().Main("Customer 3"); });
        }
    }
}

```

ScaredProduct.cs

```
using System;
using System.Threading;

namespace Lab6_1_1
{
    class ScaredProduct : IProduct
    {
        public void ChooseInstrument(string name)
        {
            Thread.Sleep(8000);
            Console.WriteLine(name);

            Console.WriteLine("Выберите музыкальный инструмент:\n1 - флейта\n2 - кларнет\n3 - труба\n4 - тромбон");
            bool choice = true;

            while (choice)
            {
                var n = Console.ReadLine();

                switch (n)
                {
                    case "1":
                        Console.WriteLine("Флейта. ");
                        choice = false;
                        break;
                    case "2":
                        Console.WriteLine("Кларнет. ");
                        choice = false;
                        break;
                    case "3":
                        Console.WriteLine("Труба. ");
                        choice = false;
                        break;
                    case "4":
                        Console.WriteLine("Тромбон. ");
                        choice = false;
                        break;
                    default:
                        Console.WriteLine("Вы ввели неправильное значение. Попробуйте ещё");
                        break;
                }
            }
        }
    }
}
```

ScaredSelector.cs

```
namespace Lab6_1_1
{
    class ScaredSelector : Selector
    {
        public override void FactoryMethod(string name)
        {
            ScaredProduct p1 = new ScaredProduct();
            p1.ChooseInstrument(name);
        }
    }
}
```

Selector.cs

```
using System;
using System.Threading;

namespace Lab6_1_1
{
    abstract class Selector
    {
        public abstract void FactoryMethod(string name);

        public void Buy(string name)
        {
            Thread.Sleep(10000);
            Console.WriteLine(name);
            Console.WriteLine($"Купить?\n 1 - Да\n 2 - Нет");
            bool choice = true;

            while (choice)
            {
                var n = Console.ReadLine();

                switch (n)
                {
                    case "1":
                        choice = false;
                        Console.WriteLine("Спасибо за покупку!");
                        break;
                    case "2":
                        choice = false;
                        Console.WriteLine("Ждём в следующий раз");
                        break;
                    default:
                        Console.WriteLine("Вы ввели неправильное значение. Попробуйте ещё");
                        break;
                }
            }
        }
    }
}
```

2) **Декоратор** — структурный паттерн проектирования, который позволяет динамически добавлять объектам новую функциональность, оборачивая их в полезные «обёртки».

Код программы.

Level.cs

```
namespace Lab6_2
{
    public abstract class Level
    {
        public abstract string GetLevel();
        public abstract string AddToFavourites();
        public abstract string GetDiscont();
    }
}
```

ConcreteLevel.cs

```
namespace Lab6_2
{
    class ConcreteLevel : Level
    {
    }
}
```



```

{
    public override string AddToFavourites()
    {
        return $"You can't add to favourites at {this.GetLevel()}";
    }

    public override string GetDiscont()
    {
        return $"You can't get discount at {this.GetLevel()}";
    }

    public override string GetLevel()
    {
        return "level 0";
    }
}
}

```

Decorator.cs

```

namespace Lab6_2
{
    class Decorator : Level
    {
        protected Level _level;

        public Decorator (Level level)
        {
            this._level = level;
        }

        public void SetLevel(Level level)
        {
            this._level = level;
        }

        public override string GetLevel()
        {
            if (this._level != null)
            {
                return this._level.GetLevel();
            }
            else
            {
                return string.Empty;
            }
        }

        public override string AddToFavourites()
        {
            if (this._level != null)
            {
                return this._level.AddToFavourites();
            }
            else
            {
                return string.Empty;
            }
        }

        public override string GetDiscont()
        {
            if (this._level != null)
            {
                return this._level.GetDiscont();
            }
        }
    }
}

```

```

        }
        else
        {
            return string.Empty;
        }
    }
}

```

LevelOne.cs

```

namespace Lab6_2
{
    class LevelOne : Decorator
    {
        public LevelOne(Level lev) : base(lev) { }

        public override string GetLevel()
        {
            return "level 1";
        }

        public override string AddToFavourites()
        {
            return $"You can add to favourites at {this.GetLevel()}";
        }

        public override string GetDiscont()
        {
            return $"You can't get discount at {this.GetLevel()}";
        }
    }
}

```

LevelTwo.cs

```

namespace Lab6_2
{
    class LevelTwo : Decorator
    {
        public LevelTwo(Level lev) : base(lev) { }

        public override string GetLevel()
        {
            return "level 2";
        }

        public override string AddToFavourites()
        {
            return $"You can add to favourites at {this.GetLevel()}";
        }

        public override string GetDiscont()
        {
            return $"You can get discount at {this.GetLevel()}";
        }
    }
}

```

Customer.cs

```

using System;

namespace Lab6_2

```

```

{
    public class Customer
    {
        protected string _nickName;

        public Customer(string nickName)
        {
            _nickName = nickName;
        }

        public void MyLevel(Level level)
        {
            Console.WriteLine($"{_nickName}, you have {level.GetLevel()}");
        }

        public void GetData(Level level)
        {
            Console.WriteLine(level.AddToFavourites());
            Console.WriteLine(level.GetDiscont());
            Console.WriteLine();
        }
    }
}

```

Program.cs

```

namespace Lab6_2
{
    class Program
    {
        static void Main(string[] args)
        {
            Customer customer = new Customer("Grof");
            var component = new ConcreteLevel();
            customer.MyLevel(component);
            customer.GetData(component);

            LevelOne levelOne = new LevelOne(component);
            customer.MyLevel(levelOne);
            customer.GetData(levelOne);

            LevelTwo levelTwo = new LevelTwo(levelOne);
            customer.MyLevel(levelTwo);
            customer.GetData(levelTwo);
        }
    }
}

```

3) Состояние — поведенческий паттерн проектирования, который позволяет объектам менять поведение в зависимости от своего состояния. Извне создаётся впечатление, что изменился класс объекта.

Код программы.

State.cs

```

using System;

namespace Lab6_3
{
    class State
    {
        public void Expectation()
        {

```

```

        Console.WriteLine("Ожидание..");
    }

    public void Print()
    {
        Console.WriteLine("Печатаем..");
    }

    public void Clamping()
    {
        Console.WriteLine("Зажатие бумаги..");
    }

    public void Refusal()
    {
        Console.Write("Отказ..\nПополните ресурсы: ");
    }
}
}

```

Printer.cs

```

using System;

namespace Lab6_3
{
    class Printer
    {
        private string _model;
        private int _paperNum;
        private double _occupancyPercentage;
        private double _clampingPercentage;
        private State _state;

        public Printer(string model, int paperNum, double occupancyPercentage, double
clampingPercentage)
        {
            this._model = model;
            this._paperNum = paperNum;
            this._clampingPercentage = clampingPercentage;
            this._state = new State();

            if (occupancyPercentage < 100)
                this._occupancyPercentage = occupancyPercentage;
            else
                this._occupancyPercentage = 100;
        }

        public void GetName()
        {
            Console.WriteLine("\n"+this._model);
        }

        public void Print()
        {
            if (this._paperNum < 1)
            {
                this._state.Refusal();
                Console.WriteLine("бумага");
                this._state.Expectation();
                this.LoadPaper();
            }

            if (this._occupancyPercentage < 20)
            {

```

```

        this._state.Refusal();
        Console.WriteLine("краска");
        this._state.Expectation();
        this.CatridgeRefuling();
    }

    this._state.Print();

    if(this._clampingPercentage > 40)
    {
        this._state.Clamping();
        this.RecoveryPaperClamping();
        this._state.Print();
    }
}

private void RecoveryPaperClamping()
{
    Console.WriteLine("Можем продолжить работу!");
}

private void LoadPaper()
{
    this._paperNum = 50;
}

private void CatridgeRefuling()
{
    this._occupancyPercentage = 100;
}
}
}

```

Program.cs

```

namespace Lab6_3
{
    class Program
    {
        static void Main(string[] args)
        {
            Printer printer = new Printer("XM-43", 10, 60, 20);
            printer.GetName();
            printer.Print();


            Printer printer1 = new Printer("JG-21", 0, 10, 20);
            printer1.GetName();
            printer1.Print();

            Printer printer2 = new Printer("FH-58", 10, 50, 60);
            printer2.GetName();
            printer2.Print();
        }
    }
}

```

Результаты работы программы:

1)

 Консоль отладки Microsoft Visual Studio

Customer 1

Выберите музыкальное направление:

- 1 - народная музыка
- 2 - духовная музыка
- 3 - академическая музыка
- 4 - популярная музыка

1

Customer 2

Выберите музыкальное направление:

- 1 - народная музыка
- 2 - духовная музыка
- 3 - академическая музыка
- 4 - популярная музыка

3

Customer 3

Выберите музыкальное направление:

- 1 - народная музыка
- 2 - духовная музыка
- 3 - академическая музыка
- 4 - популярная музыка

4

-

Customer 1

Бубен. Купить?

- 1 - Да
- 2 - Нет

2

Ждём в следующий раз

Customer 2

Скрипка. Купить?

- 1 - Да
- 2 - Нет

1

Спасибо за покупку!

Customer 3

Мелодика. Купить?

- 1 - Да
- 2 - Нет

2

Ждём в следующий раз

Customer 1

Выберите музыкальный инструмент:

- 1 - гусли
- 2 - бубен
- 3 - волынка
- 4 - свирель

2

Customer 2

Выберите музыкальный инструмент:

- 1 - блокфлейта
- 2 - гlockenspiel
- 3 - синтезатор
- 4 - скрипка

4


Customer 3

Выберите музыкальный инструмент:

- 1 - гитара
- 2 - барабан
- 3 - мелодика
- 4 - MIDI-клавиатура

3

2)

 Консоль отладки Microsoft Visual Studio

Grof, you have level 0

You can't add to favourites at level 0

You can't get discount at level 0

Grof, you have level 1

You can add to favourites at level 1


You can't get discount at level 1

Grof, you have level 2

You can add to favourites at level 2

You can get discount at level 2

3)

 Консоль отладки Microsoft Visual Studio

ХМ-43

Печатаем..

ЈG-21

Отказ..

Пополните ресурсы: бумага

Ожидание..

Отказ..

Пополните ресурсы: краска

Ожидание..

Печатаем..

FN-58

Печатаем..

Зажатие бумаги..

Можем продолжить работу!

Печатаем..

Вывод: приобрела навыки применения паттернов проектирования при решении практических задач.