МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 4

з дисципліни “Програмування”
тема “Шаблони проєктування”

Виконав студент
ІІ курсу групи КП-01
Тітов Єгор Павлович
Варіант 16

Перевірив
“____” “_____” 20___р
викладач
Заболотня Тетяна Миколаївна

Київ 2021

Постановка задач

1. За допомогою шаблону проєктування реалізувати віртуальний автомат, який виробляє напої. В залежності від того, які параметри задані покупцем (обрано чай/каву/какао, вказано «додатковий цукор», «лимон» і т.ін.) згенерувати напій. Забезпечити виробництво як мінімум трьох різних напоїв.
2. За допомогою шаблону проєктування реалізувати функціональну можливість збереження персонажу комп'ютерної гри в поточному стані для того, щоб мати можливість продовжити гру збереженою копією персонажа в разі, коли оригінал персонажу вбито. Зберегти персонажа можна лише 1 раз.

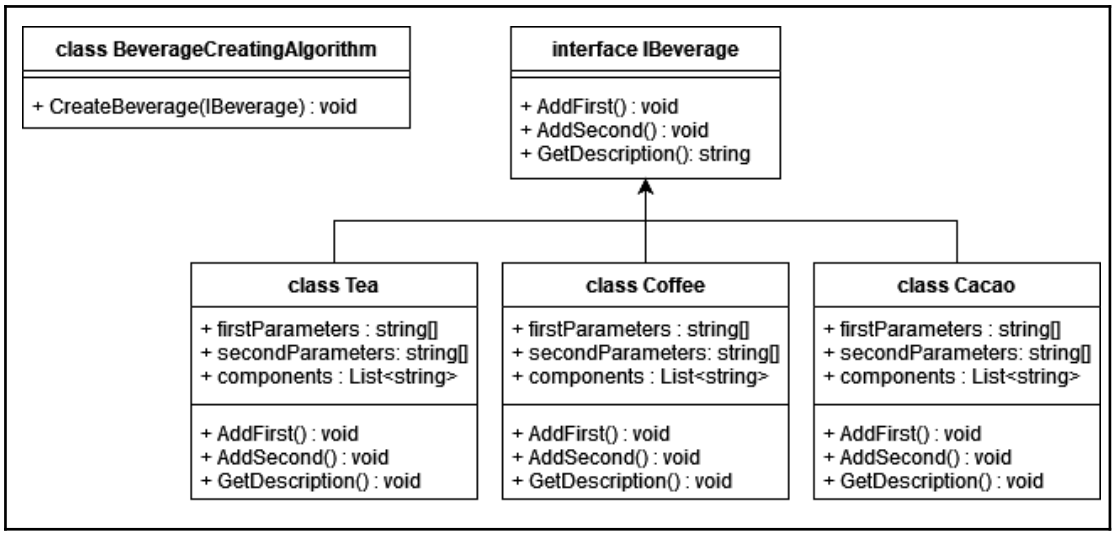
Обгрунтування вибору шаблонів

1. Оскільки маємо декілька класів зі схожим наповненням, кожен з яких слідує певному алгоритму 'створення напою', то можемо обрати такий поведінковий шаблон проектування, як "Template method".

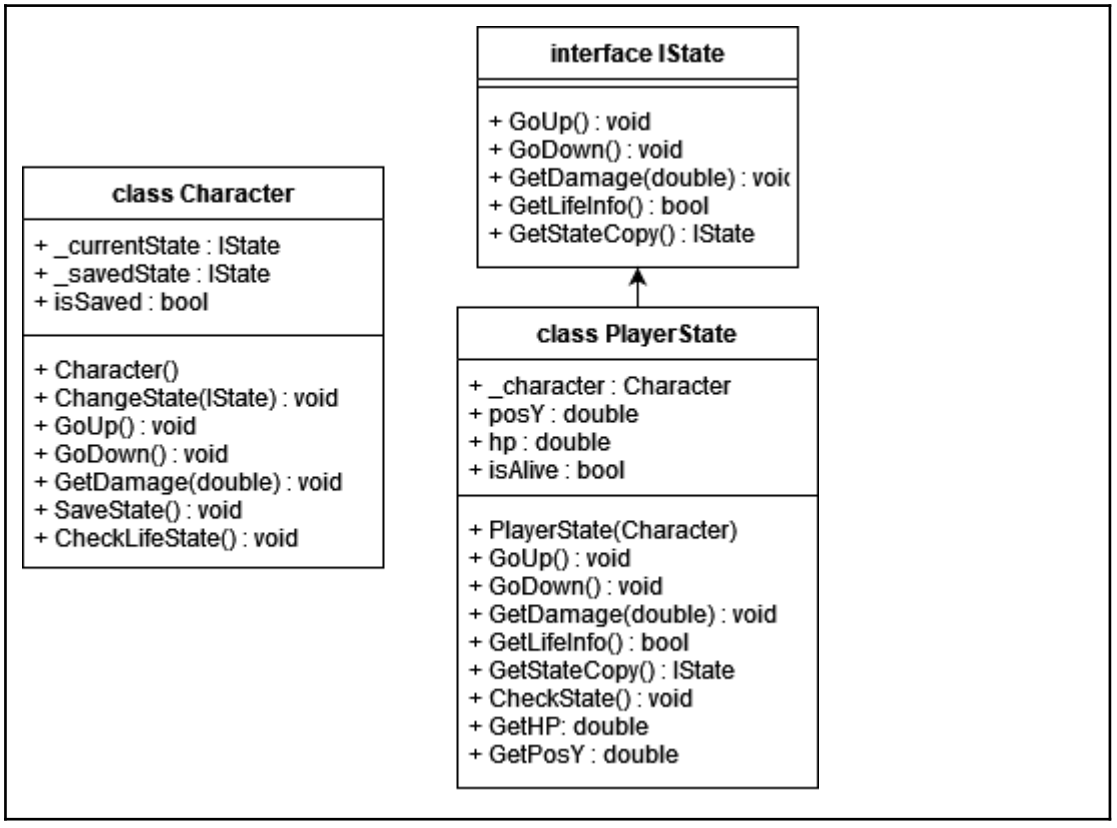
2. Оскільки маємо працювати з безпосередньо 'станами' персонажу, то логічно буде взяти і шаблон "State".

UML діаграма класів

1. Завдання 1



2. Завдання 2



Текст програми

Завдання 1

```
using System;
using System.Collections.Generic;

namespace No_1
{
    class Program
    {
        static void Main(string[] args)
        {
            Dictionary<string, IBeverage> possibleBeverages = new
Dictionary<string, IBeverage>();
            possibleBeverages.Add("tea", new Tea());
            possibleBeverages.Add("coffee", new Coffee());
            possibleBeverages.Add("cacao", new Cacao());

            Console.Write("Choose beverage(tea/coffee/cacao): ");
            string choice = Console.ReadLine();

            IBeverage bev;

            if (possibleBeverages.TryGetValue(choice, out bev))
            {
                BeverageCreatingAlgorithm alg = new
BeverageCreatingAlgorithm();

                alg.CreateBeverage(bev);
            }
            else
            {
                Console.WriteLine($"Beverage '{choice}' does not exist");
            }
        }
    }
}
```

```
namespace No_1
{
    interface IBeverage
```

```
    {
        void AddFirst();

        void AddSecond();

        string GetDescription();
    }
}
```

```
using System;
using System.Collections.Generic;

namespace No_1
{
    class Tea : IBeverage
    {
        string[] firstParameters = {"lemon", "mint"};
        string[] secondParameters = {"1x sugar", "2x sugar", "3x sugar"};

        List<string> components = new List<string>();

        public void AddFirst()
        {
            Console.Write("Enter first additional parameter(lemon/mint): ");
            string choosenParameter = Console.ReadLine();

            foreach(string par in firstParameters)
            {
                if (choosenParameter == par)
                {
                    components.Add(choosenParameter);
                }
            }
        }

        public void AddSecond()
        {
            Console.Write("Enter second additional parameter(1x/2x/3x
sugar): ");
            string choosenParameter = Console.ReadLine();
```

```
            foreach(string par in secondParameters)
            {
                if (choosenParameter == par)
                {
                    components.Add(choosenParameter);
                }
            }
        }

        public string GetDescription()
        {
            string descr = "Tea";

            if (components.Count != 0)
            {
                descr += $" with parameters: {String.Join(',',
components.ToArray())}";
            }

            return descr;
        }
    }
}
```

```
using System;
using System.Collections.Generic;

namespace No_1
{
    class Coffee : IBeverage
    {
        string[] firstParameters = {"milk", "vanilla"};
        string[] secondParameters = {"1x sugar", "2x sugar", "3x sugar"};

        List<string> components = new List<string>();

        public void AddFirst()
        {
            Console.Write("Enter first additional parameter(milk/vanilla):
");
            string choosenParameter = Console.ReadLine();
```

```
            foreach(string par in firstParameters)
            {
                if (choosenParameter == par)
                {
                    components.Add(choosenParameter);
                }
            }
        }

        public void AddSecond()
        {
            Console.Write("Enter second additional parameter(1x/2x/3x
sugar): ");
            string choosenParameter = Console.ReadLine();

            foreach(string par in secondParameters)
            {
                if (choosenParameter == par)
                {
                    components.Add(choosenParameter);
                }
            }
        }

        public string GetDescription()
        {
            string descr = "Coffee";

            if (components.Count != 0)
            {
                descr += $" with parameters: {String.Join(',',
components.ToArray())}";
            }

            return descr;
        }
    }
}
```

```
using System;
```

```
using System.Collections.Generic;

namespace No_1
{
    class Cacao : IBeverage
    {
        string[] firstParameters = {"honey", "vanilla"};
        string[] secondParameters = {"1x sugar", "2x sugar", "3x sugar"};

        List<string> components = new List<string>();

        public void AddFirst()
        {
            Console.Write("Enter first additional parameter(honey/vanilla):
");
            string choosenParameter = Console.ReadLine();

            foreach(string par in firstParameters)
            {
                if (choosenParameter == par)
                {
                    components.Add(choosenParameter);
                }
            }
        }

        public void AddSecond()
        {
            Console.Write("Enter second additional parameter(1x/2x/3x
sugar): ");
            string choosenParameter = Console.ReadLine();

            foreach(string par in secondParameters)
            {
                if (choosenParameter == par)
                {
                    components.Add(choosenParameter);
                }
            }
        }
```

```
        public string GetDescription()
        {
            string descr = "Cacao";

            if (components.Count != 0)
            {
                descr += $" with parameters: {String.Join(',',
components.ToArray())}";
            }

            return descr;
        }
    }
}
```

```
using System;

namespace No_1
{
    class BeverageCreatingAlgorithm
    {
        public void CreateBeverage(IBeverage bev)
        {
            bev.AddFirst();

            bev.AddSecond();

            Console.Write($"Your order: '{bev.GetDescription()}'");
        }
    }
}
```

2. Завдання 2

```
namespace No_2
{
    class Program
    {
        static void Main(string[] args)
```

```
        {
            Character character = new Character();

            character.GoUp();
            character.GoUp();
            character.GetDamage(66);

            character.SaveState();

            character.GetDamage(44);

            character.GoUp();
        }
    }
}
```

```
namespace No_2
{
    interface IState
    {
        void GoUp();
        void GoDown();

        void GetDamage(double damage);
        bool GetLifeInfo();
        IState GetStateCopy();
    }
}
```

```
using System;

namespace No_2
{
    class PlayerState : IState
    {
        private Character _character;
        private double posY = 0;
        private double hp = 100;
        private bool isAlive = true;

        public PlayerState(Character character)
```

```
        {
            _character = character;
        }

        public void GoDown()
        {
            if (isAlive)
            {
                posY -= 10;

                Console.WriteLine($"Character went down. Current position:
'{posY}'");
            }
        }

        public void GoUp()
        {
            if (isAlive)
            {
                posY += 10;

                Console.WriteLine($"Character went up. Current position:
'{posY}'");
            }
        }

        public void GetDamage(double damage)
        {
            if (isAlive)
            {
                hp -= damage;

                Console.WriteLine($"Character get damaged. Current hp:
'{hp}'");

                CheckState();
            }
        }

        private void CheckState()
        {
```

```
            if (hp <= 0)
            {
                isAlive = false;

                Console.WriteLine($"Character died");
            }
        }

        public bool GetLifeInfo()
        {
            return isAlive;
        }

        public double GetHP()
        {
            return hp;
        }

        public double GetPosY()
        {
            return posY;
        }

        public IState GetStateCopy()
        {
            PlayerState copy = new PlayerState(_character);

            copy.posY = posY;
            copy.hp = hp;
            copy.isAlive = isAlive;

            return copy;
        }
    }
}
```

```
using System;

namespace No_2
{
    class Character
```

```
{
    private IState _currentState;
    private IState _savedState;
    private bool isSaved = false;

    public Character()
    {
        _currentState = new PlayerState(this);
    }

    public void ChangeState(IState state)
    {
        _currentState = state;
    }

    public void GoUp()
    {
        if (_currentState != null)
        {
            _currentState.GoUp();
        }
    }

    public void GoDown()
    {
        if (_currentState != null)
        {
            _currentState.GoDown();
        }
    }

    public void GetDamage(double damage)
    {
        if (_currentState != null)
        {
            _currentState.GetDamage(damage);
        }

        CheckLifeState();
    }
```

```
        public void SaveState()
        {
            if (isSaved == false && _currentState.GetLifeInfo() != false)
            {
                Console.WriteLine("Character state saved");

                isSaved = true;

                _savedState = _currentState.GetStateCopy();
            }
        }

        private void CheckLifeState()
        {
            if (_currentState.GetLifeInfo() == false)
            {
                if (_savedState != null)
                {
                    Console.WriteLine("Loading saved state...");

                    _currentState = _savedState;
                }
                else
                {
                    Console.WriteLine("No saved states. Game over.");
                }
            }
        }
    }
}
```

Приклади результатів

Завдання 1:

```
Choose beverage(tea/coffee/cacao): tea
Enter first additional parameter(lemon/mint): lemon
Enter second additional parameter(1x/2x/3x sugar): 1x sugar
Your order: 'Tea with parameters: lemon,1x sugar'
```

Завдання 2:

```
Character went up. Current position: '10'
Character went up. Current position: '20'
Character get damaged. Current hp: '34'
Character state saved
Character get damaged. Current hp: '-10'
Character died
Loading saved state...
Character went up. Current position: '30'
```

Висновки

В результаті виконання лабораторної роботи я ознайомився і використав для вирішення поставлених завдань такі шаблони як “State” і “Template method”. Внаслідок цього я зрозумів переваги використання поведінкових шаблонів проєктування і як важливо правильно їх застосовувати.