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.

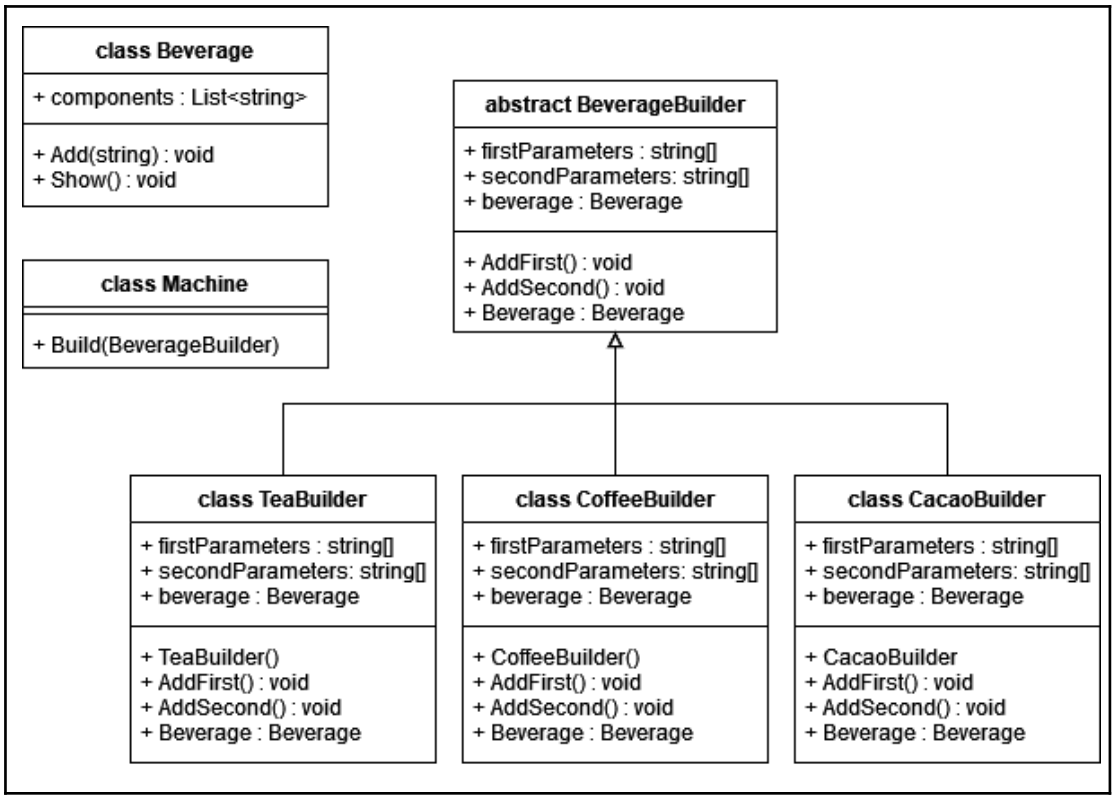
Обгрунтування вибору шаблонів

1. Оскільки маємо різні варіації створення напою, то необхідно забезпечити гнучкість і незалежність створення кожної частини продукту. Саме такий принцип дозволяє втілити шаблон "Builder".

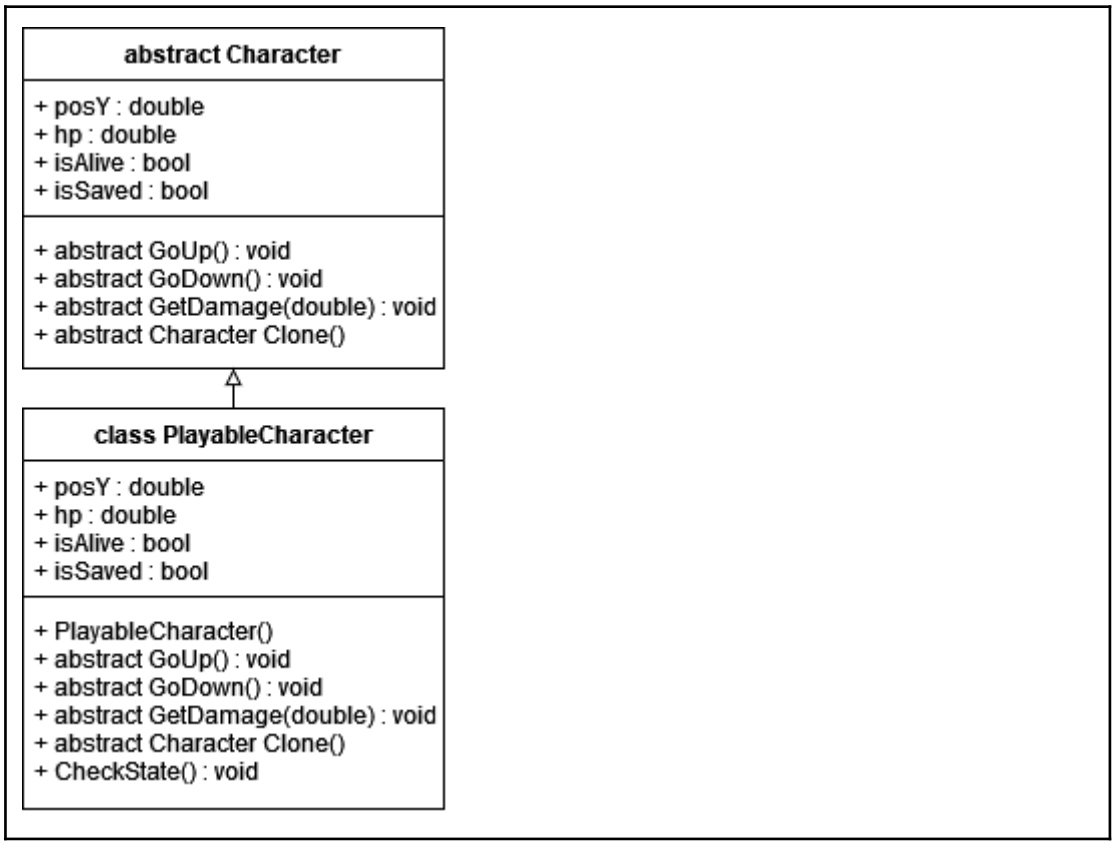
2. Оскільки маємо справу зі зберіганням даних в певному стані, то найлогічніше буде клонувати ці самі дані на певному етапі, що дозволяє шаблон "Prototype".

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, BeverageBuilder> possibleBeverageBuild = new
Dictionary<string, BeverageBuilder>();
            possibleBeverageBuild.Add("tea", new TeaBuilder());
            possibleBeverageBuild.Add("coffee", new CoffeeBuilder());
            possibleBeverageBuild.Add("cacao", new CacaoBuilder());

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

            BeverageBuilder bevBuild;

            if (possibleBeverageBuild.TryGetValue(choice, out bevBuild))
            {
                Machine machine = new Machine();

                machine.Build(bevBuild);
            }
            else
            {
                Console.WriteLine($"Beverage '{choice}' does not exist");
            }
        }
    }
}
```

```
namespace No_1
{
    class Machine
    {
```

```
        public void Build(BeverageBuilder bev)
        {
            bev.AddFirst();
            bev.AddSecond();

            bev.Beverage.Show();
        }
    }
}
```

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

namespace No_1
{
    class Beverage
    {
        private List<string> _parts = new List<string>();

        public void Add(string part)
        {
            _parts.Add(part);
        }

        public void Show()
        {
            Console.Write($"Builded beverage: '{_parts[0]}' with additives:
");


            Console.Write(String.Join(',', _parts.ToArray(), 1, _parts.Count
- 1));
        }
    }
}
```

```
namespace No_1
{
    abstract class BeverageBuilder
    {
        protected string[] firstParameters;
```

```
        protected string[] secondParameters;
        protected Beverage beverage;

        public Beverage Beverage
        {
            get
            {
                return beverage;
            }
        }

        public abstract void AddFirst();
        public abstract void AddSecond();
    }
}
```

```
using System;

namespace No_1
{
    class TeaBuilder : BeverageBuilder
    {
        public TeaBuilder()
        {
            firstParameters = new string[]{"lemon", "mint"};

            secondParameters = new string[]{"1x sugar", "2x sugar", "3x
sugar"};

            beverage = new Beverage();
            beverage.Add("tea");
        }

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

            foreach(string par in firstParameters)
            {
                if (choosenParameter == par)
```

```
                {
                    beverage.Add(choosenParameter);
                }
            }
        }

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

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

```
using System;

namespace No_1
{
    class CoffeeBuilder : BeverageBuilder
    {
        public CoffeeBuilder()
        {
            firstParameters = new string[]{"milk", "vanilla"};

            secondParameters = new string[]{"1x sugar", "2x sugar", "3x
sugar"};

            beverage = new Beverage();
            beverage.Add("coffee");
        }

        public override void AddFirst()
```

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

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

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

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

```
using System;

namespace No_1
{
    class CacaoBuilder : BeverageBuilder
    {
        public CacaoBuilder()
        {
            firstParameters = new string[]{"honey", "vanilla"};
```

```
            secondParameters = new string[]{"1x sugar", "2x sugar", "3x
sugar"};

            beverage = new Beverage();
            beverage.Add("cacao");
        }

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

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

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

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

2. Завдання 2

```
namespace No_2
{
    class Program
    {
        static void Main(string[] args)
        {
            Character character = new PlayableCharacter();

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

            Character savedCopyCharacter = character.Clone();

            character.GetDamage(34);

            savedCopyCharacter.GoUp();
        }
    }
}
```

```
namespace No_2
{
    abstract class Character
    {
        protected double posY;
        protected double hp;
        protected bool isAlive;
        protected bool isSaved;

        public abstract void GoUp();
        public abstract void GoDown();
        public abstract void GetDamage(double damage);

        public abstract Character Clone();
    }
}
```

```
using System;
```

```
namespace No_2
{
    class PlayableCharacter : Character
    {
        public PlayableCharacter()
        {
            hp = 100;
            posY = 0;
            isAlive = true;
            isSaved = false;
        }

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

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

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

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

        public override Character Clone()
        {
            if (isSaved)
            {
                Console.WriteLine("Character copy already saved.");

                return null;
            }
            else
            {
                isSaved = true;

                Console.WriteLine("Character copy saved");

                return (Character)this.MemberwiseClone();
            }
        }
    }
}
```

Приклади результатів

Завдання 1:

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

Завдання 2:

```
Character went up. Current position: '10'
Character went up. Current position: '20'
Character get damaged. Current hp: '34'
Character copy saved
Character get damaged. Current hp: '0'
Character died
Character went up. Current position: '30'
```

Висновки

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