

МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова)

Кафедра	<u>О7</u>	Информационные системы и программная инженерия
	шифр	наименование кафедры, по которой выполняется работа
Дисциплина		Информационные технологии
		наименование дисциплины

УЧЕБНО-ПРАКТИЧЕСКАЯ РАБОТА

1

номер задания (при наличии)

РАЗРАБОТКА ИЕРАРХИИ КЛАССОВ С
ИСПОЛЬЗОВАНИЕМ
ИНТЕРФЕЙСОВ, АБСТРАКТНЫХ КЛАССОВ И
ДРУГИХ
МЕХАНИЗМОВ НАСЛЕДОВАНИЯ

Вариант 17.

при наличии указать тему учебно-практической работы и (или) номер варианта

ОБУЧАЮЩИЙСЯ

группы Е123Б

Усов Д. А.

подпись

фамилия и инициалы

дата сдачи

ПРОВЕРИЛ

ученая степень, ученое звание, должность

Землянская Е.Р

подпись

фамилия и инициалы

Оценка / балльная оценка

дата проверки

Санкт-Петербург
20 23 г.

СОДЕРЖАНИЕ

1	Постановка задачи	3
1.1	Задание 1	3
2	Реализация	4
2.1	Файл ProgramFirst	4
3	Результаты работы программы	10
3.1	Задание 1	10
	ЗАКЛЮЧЕНИЕ	11

1 Постановка задачи

1.1 Задание 1

Разработать класс, согласно индивидуальному варианту, содержащий:

- элементы разного уровня доступа (public и private);
- не менее четырех свойств;
- не менее трех методов;
- перегрузку метода ToString();
- статический метод;
- константное или поле только для чтения;
- не менее трех конструкторов;
- перегрузку операции присваивания и одной любой арифметической.

Рекомендуемые поля и методы указаны в варианте. Также необходимо написать программу с меню, позволяющую протестировать разработанный класс. Обязательные пункты меню:

- задание параметров конструируемого объекта;
- вывод свойств объекта;
- выполнение статического метода;
- выполнение методов объекта.

2 Реализация

2.1 Файл ProgramFirst

```
namespace Pr1
{

    public abstract class ProgramFirst
    {
        private static EarlGreyTea? Tea { get; set; }

        public static void InitMenu()
        {
            while (true)
            {
                Console.Clear();
                Console.Write("""
                    1. Creating a mug of tea
                    2. Display properties
                    3. Execute a static method
                    4. Perform the action on the mug
                    999. Exit
                    Enter your action:
                    """);

                int action;
                while (!int.TryParse(Console.ReadLine(), out action))
                    Console.Write("Wrong entry! \nEnter the action number:");

                switch (action)
                {
                    case 1:
                    {
                        CreateMag();
                        break;
                    }
                    case 2:
                    {
                        DisplayProperties();
                        break;
                    }
                    case 3:
                    {
                        ExecuteStaticMethod();
                        break;
                    }
                    case 4:
                    {
                        ActionWithTea();
                        break;
                    }
                    case 999:
                    {
                        return;
                    }
                    default:
                    {
                        Console.WriteLine("You have entered a non-existent action!");
                        break;
                    }
                }
            }
        }
    }
}
```

```

    }

    Console.WriteLine("Press any button to continue...");
    Console.ReadKey();
}

private static void ActionWithTea()
{
    while (true)
    {
        Console.Clear();
        if (Tea == null)
        {
            Console.WriteLine("You haven't created a mug of tea yet");
            return;
        }

        Console.Write("""
                        1. Brew
                        2. Brew with milk
                        3. Drink
                        4. Microwave
                        999. Back
                        Enter your action:
                        """);

        int action;
        while (!int.TryParse(Console.ReadLine(), out action))
            Console.Write("Wrong entry! \nEnter the action number:");

        switch (action)
        {
            case 1:
            {
                Tea.Brew();
                break;
            }
            case 2:
            {
                Tea.BrewWithMilk();
                break;
            }
            case 3:
            {
                Tea.Drink();
                break;
            }
            case 4:
            {
                Tea.Microwave();
                break;
            }
            case 999:
            {
                return;
            }
            default:
            {
                Console.WriteLine("You have entered a non-existent action!");
            }
        }
    }
}

```

```

        break;
    }
}

    Console.WriteLine("Press any button to continue...");
    Console.ReadKey();
}
}

private static void ExecuteStaticMethod()
{
    Console.Clear();
    EarlGreyTea.FiveClock();
}

private static void DisplayProperties()
{
    Console.Clear();
    if (Tea == null)
    {
        Console.WriteLine("You haven't created a mug of tea yet");
        return;
    }

    Console.WriteLine(Tea.ToString());
}

private static void CreateMag()
{
    Console.Clear();
    Console.Write("Enter the manufacture: ");
    var manufacture = Console.ReadLine();

    int bergamot;
    Console.Write("Enter the amount of bergamot: ");
    while (!int.TryParse(Console.ReadLine(), out bergamot) || bergamot <= 0)
        Console.Write("Wrong entry! \nTry again:");

    int volume;
    Console.Write("Enter the volume: ");
    while (!int.TryParse(Console.ReadLine(), out volume) || volume <= 5 ||
↪ volume < bergamot)
        Console.Write("Wrong entry! \nTry again:");

    Tea = new EarlGreyTea(manufacture, bergamot, volume);
}

public class EarlGreyTea
{
    public EarlGreyTea(string? manufacture)
    {
        Manufacture = manufacture;
        Console.WriteLine("A mug of tea has been created");
    }

    public EarlGreyTea(string? manufacture, int bergamot)
    {
        Manufacture = manufacture;
        Bergamot = bergamot;
    }
}

```

```

        Console.WriteLine("A mug of tea has been created");
    }

    public EarlGreyTea(string? manufacture, int bergamot, int volume)
    {
        Manufacture = manufacture;
        Bergamot = bergamot;
        Volume = volume;
        Console.WriteLine("A mug of tea has been created");
    }

    private string? Manufacture { get; set; }
    private int Bergamot { get; set; } = 10;
    private string? Date { get; set; } = "08.12.2022";
    private int Volume { get; set; } = 200;

    private bool _isBrew = false;
    private bool _isDrank = false;

    private const string Name = "Earl grey tea";

    public void Brew()
    {
        if (_isDrank)
        {
            Console.WriteLine("You've already had your " + Name);
            return;
        }

        if (!_isBrew)
        {
            _isBrew = true;
            Console.WriteLine(Name + " is brewing");
            return;
        }

        Console.WriteLine(Name + "'s already brewed");
    }

    public void Drink()
    {
        if (!_isBrew)
        {
            Console.WriteLine(Name + " is still brewing");
            return;
        }

        if (_isDrank)
        {
            Console.WriteLine("You've already had your " + Name);
            return;
        }

        Console.WriteLine("You drank the " + Name);
        _isDrank = true;
        _isBrew = false;
    }

    public void BrewWithMilk()
    {

```

```

        if (!_isDrank)
        {
            Console.WriteLine("You've already had your " + Name);
            return;
        }

        if (!_isBrew)
        {
            _isBrew = true;
            Console.WriteLine(Name + " is brewed with milk");
            return;
        }

        Console.WriteLine(Name + "'s already brewed");
    }

    public void Microwave()
    {
        if (!_isDrank)
        {
            Console.WriteLine("You've already had your " + Name);
            return;
        }

        if (!_isBrew)
        {
            Console.WriteLine(Name + " is still brewing");
            return;
        }

        Console.WriteLine(Name + " was heated");
    }

    public static void FiveClock()
    {
        Console.WriteLine("Time for tea!!!");
    }

    public override string ToString()
    {
        return
            $"{Name}: Manufacture: {Manufacture}, Bergamot: {Bergamot} g, Date:
→ {Date}, Volume: {Volume} ml";
    }

    public static bool operator <(EarlGreyTea tea1, EarlGreyTea tea2)
    {
        return tea1.Volume < tea2.Volume;
    }

    public static bool operator >(EarlGreyTea tea1, EarlGreyTea tea2)
    {
        return tea1.Volume > tea2.Volume;
    }

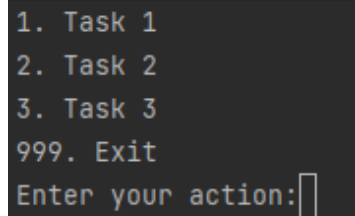
    public static EarlGreyTea operator +(EarlGreyTea tea1, EarlGreyTea tea2)
    {
        return new EarlGreyTea($"{tea1.Manufacture}&{tea2.Manufacture}",
            tea1.Bergamot + tea2.Bergamot,
            tea1.Volume + tea2.Volume);
    }

```


}
}
}

3 Результаты работы программы

Внутри программы реализовано меню с выбором нужного задания, это мы можем увидеть на рисунке 3.1

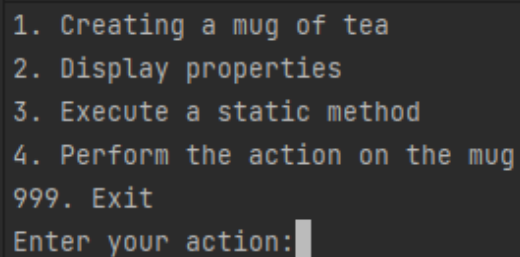


```
1. Task 1
2. Task 2
3. Task 3
999. Exit
Enter your action:█
```

Рисунок 3.1 — Меню выбора задания

3.1 Задание 1

На рисунке 3.2 мы можем видеть меню выбора действия



```
1. Creating a mug of tea
2. Display properties
3. Execute a static method
4. Perform the action on the mug
999. Exit
Enter your action:█
```

Рисунок 3.2 — Меню выбора действия

ЗАКЛЮЧЕНИЕ

Выполняя учебно-практическую работу была разработана программа реализующая работу с классами и интерфейсом.