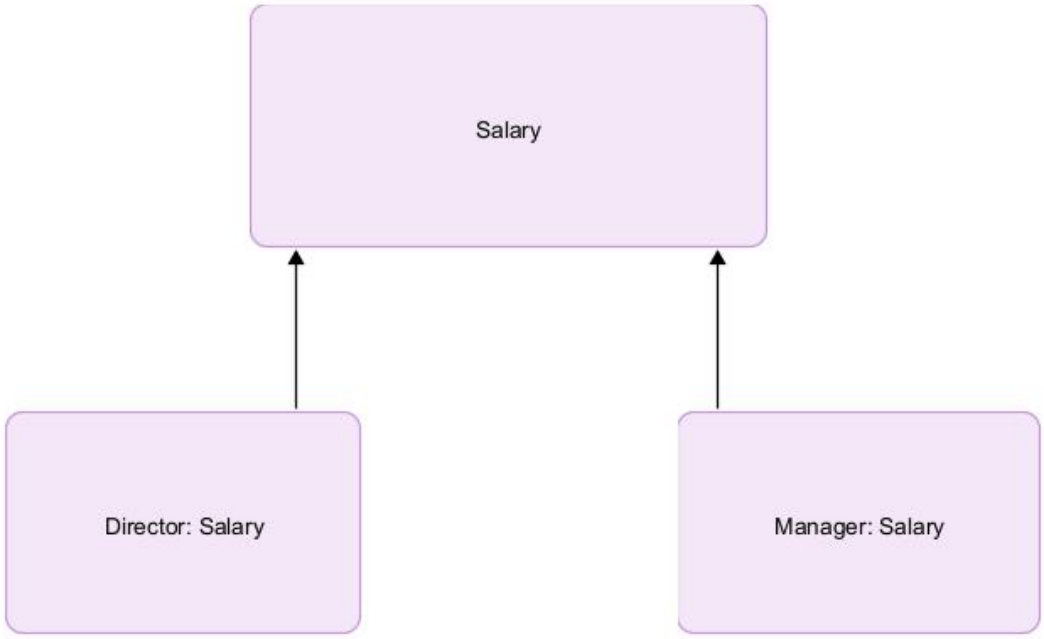
Отчет

Выполнил студент группы 3пк2 Неробеев Алексей

1 проект.

Описание предметной области:
На предприятии есть 2 должности: директор и менеджер. Для каждой из них производится проверка на корректность заработной платы, а также имени
Диаграмма классов:



Краткий вывод:
Паттерн «шаблонный метод» позволяет производить проверку на корректность сразу двух полей, а именно имени и заработной платы
Листинг кода:

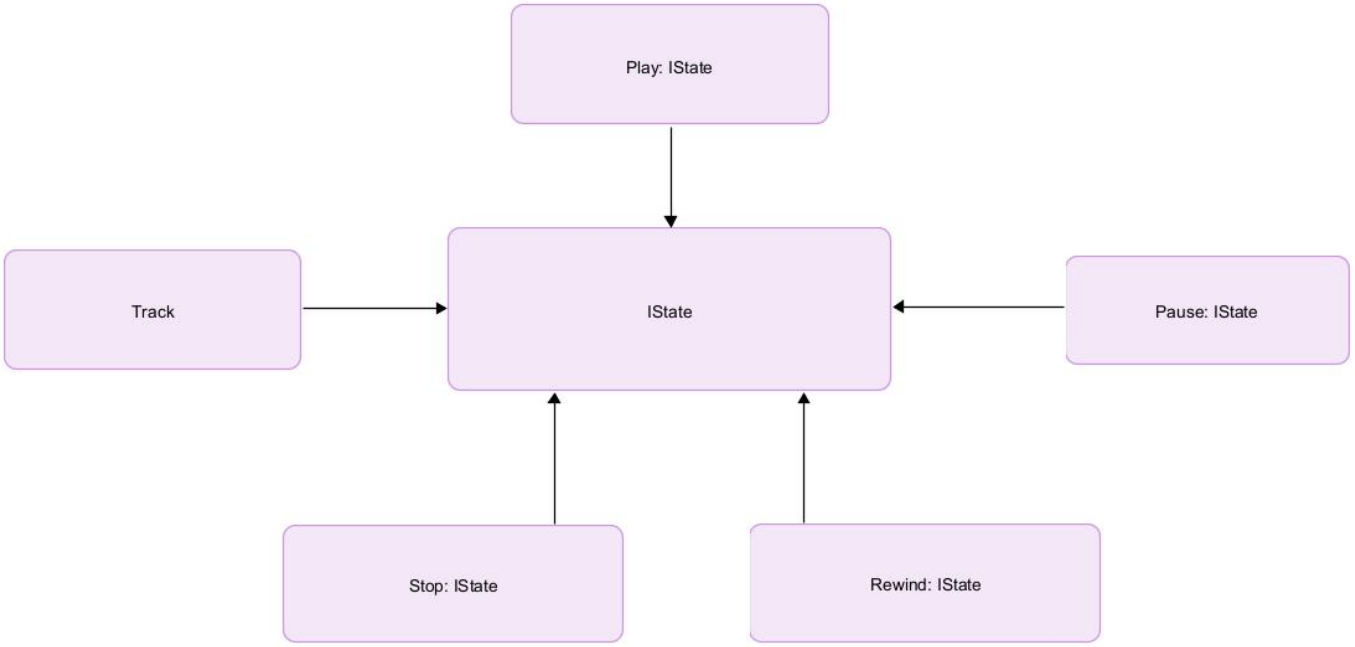
```
internal abstract class Salary
{
    private string _name { get; set; }
    private int _salary { get; set; }

    public string Name
    {
        get => _name;
        set => _name = value;
    }

    public int EmployeeSalary
    {
        get => _salary;
        set => _salary = value;
    }

    public abstract void CheckName(Salary salary);
    public abstract void CheckSalary(Salary salary);
    public abstract void PrintInfo();
    public void Check(Salary salary)
```

```
    {
        PrintInfo();
        CheckName(salary);
        CheckSalary(salary);
    }
}

internal class Director: Salary
{
    public override void PrintInfo()
    {
        Console.WriteLine($"Должность: {Name}, зарплата:
{EmployeeSalary}");
    }

    public override void CheckName(Salary salary)
    {
        if (!salary.Name.Any(char.IsUpper))
        {
            Console.WriteLine("Имя должно начинаться с заглавной буквы");
        }
    }

    public override void CheckSalary(Salary salary)
    {
        if(salary.EmployeeSalary < 0)
        {
            Console.WriteLine("Зарплата не может быть отрицательной");
        }
        else if(salary.EmployeeSalary < 50000 || salary.EmployeeSalary >
100000)
        {
            Console.WriteLine("Зарплата не входит в требуемый диапазон");
        }
    }
}

internal class Manager: Salary
{
    public override void PrintInfo()
    {
        Console.WriteLine($"Должность: {Name}, зарплата:
{EmployeeSalary}");
    }

    public override void CheckName(Salary salary)
    {
        if (!salary.Name.Any(char.IsUpper))
        {
            Console.WriteLine("Имя должно начинаться с заглавной буквы");
        }
    }

    public override void CheckSalary(Salary salary)
    {
        if (salary.EmployeeSalary < 0)
        {
            Console.WriteLine("Зарплата не может быть отрицательной");
        }
        else if (salary.EmployeeSalary < 30000 || salary.EmployeeSalary >
50000)
        {
            Console.WriteLine("Зарплата не входит в требуемый диапазон");
```

```
        }
    }
}

internal class Program
{
    static void Main(string[] args)
    {
        Director director = new Director();
        Manager manager = new Manager();
        director.Name = "Иван";
        director.EmployeeSalary = 55000;
        manager.Name = "андрей";
        manager.EmployeeSalary = 10000;

        director.Check(director);
        Console.WriteLine("------------------------");
        manager.Check(manager);
    }
}
```

2 проект.

Описание предметной области:
Приложение для воспроизведения музыки, где пользователь может управлять состоянием трэка: воспроизвести, поставить на паузу, остановить, перемотать
Диаграмма классов:



Краткий вывод:
Паттерн позволяет отслеживать текущее состояние трэка, чтобы пользователь мог понимать воспроизводится ли сейчас трэк и какие функции могут быть доступны
Листинг кода:

```
internal interface IState
{
    void Play();
    void Stop();
    void Pause();
    void Rewind();
}

internal class Track
{
    private string _name;

    public string Name
    {
        get => _name;
        set => _name = value;
    }

    private IState state;
    public void Play() =>
        state.Play();

    public void Pause() =>
        state.Pause();

    public void Stop() =>
        state.Stop();

    public void Rewind() =>
        state.Rewind();

    public Track(string name)
    {
        Name = name;
        state = new StopState();
    }

    public void ChangeState(IState state) => this.state = state;

    public override string ToString()
    {
        return $"трэк {Name} {state}";
    }
}

internal class PauseState: IState
{
    public void Stop()
    {
        Console.WriteLine("остановка трэка");
    }

    public void Play()
    {
        Console.WriteLine("воспроизведение трэка");
    }

    public void Pause()
    {
        Console.WriteLine("трэк уже на паузе");
    }
```

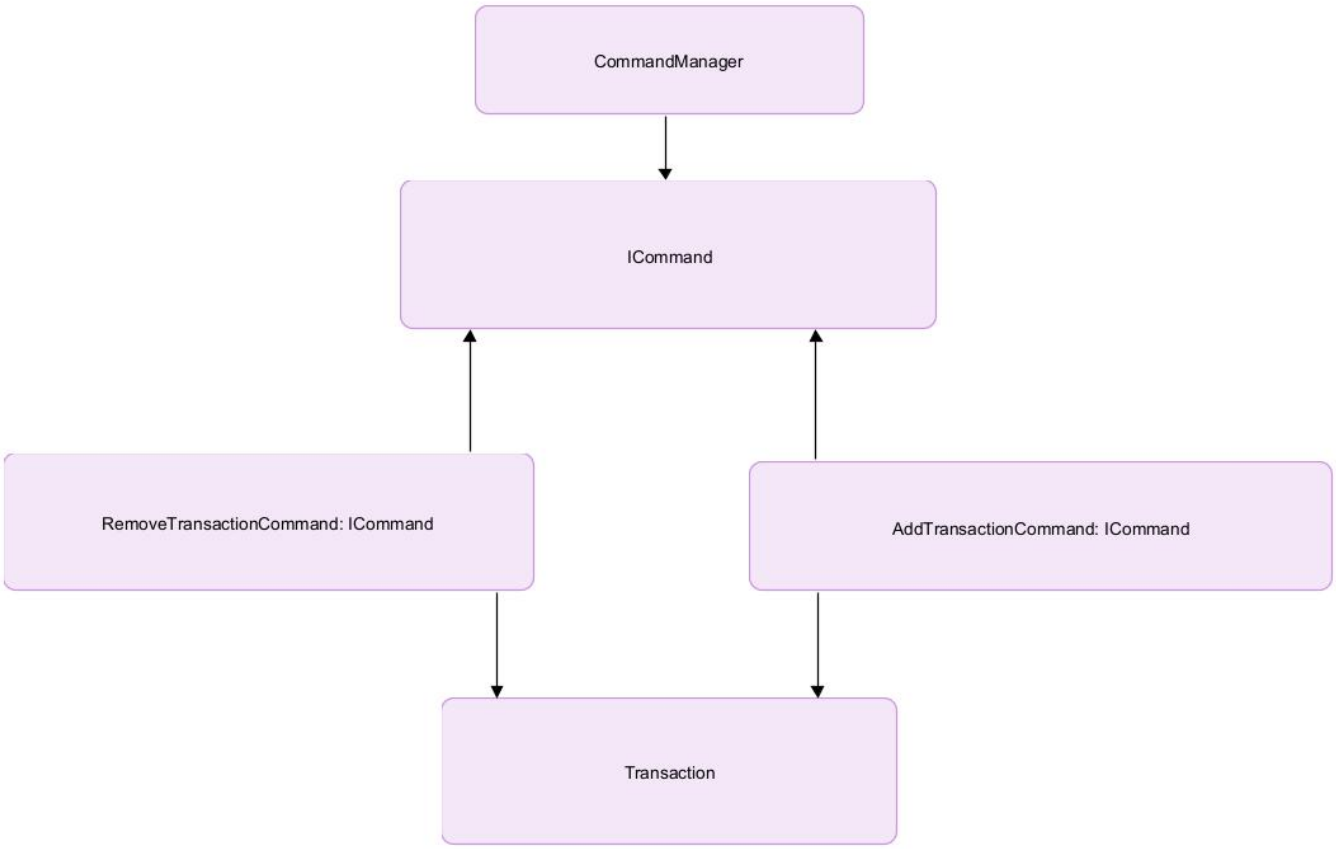
```
    public void Rewind()
    {
        Console.WriteLine("перемотка трэка");
    }

    public override string ToString()
    {
        return "на паузе";
    }
}

internal class PlayState: IState
{
    public void Stop()
    {
        Console.WriteLine("остановка трэка");
    }

    public void Play()
    {
        Console.WriteLine("трэк уже воспроизведен");
    }

    public void Pause()
    {
        Console.WriteLine("постановка трэка на паузу");
    }

    public void Rewind()
    {
        Console.WriteLine("перемотка трэка");
    }

    public override string ToString()
    {
        return "воспроизведен";
    }
}

internal class RewindState: IState
{
    public void Stop()
    {
        Console.WriteLine("остановка трэка");
    }

    public void Play()
    {
        Console.WriteLine("воспроизведение трэка");
    }

    public void Pause()
    {
        Console.WriteLine("постановка трэка на паузу");
    }

    public void Rewind()
    {
        Console.WriteLine("трэк уже перематывается");
    }

    public override string ToString()
```

```
    {
        return "перематывается";
    }
}

internal class StopState: IState
{
    public void Stop()
    {
        Console.WriteLine("трэк уже остановлен");
    }

    public void Play()
    {
        Console.WriteLine("воспроизведение трэка");
    }

    public void Pause()
    {
        Console.WriteLine("постановка трэка на паузу");
    }

    public void Rewind()
    {
        Console.WriteLine("перемотка трэка");
    }

    public override string ToString()
    {
        return "остановлен";
    }
}

internal class Program
{
    static void Main(string[] args)
    {
        Track track = new Track("Lucid Dreams");
        Console.WriteLine(track);
        track.Stop();
        track.Play();
        track.Pause();
        track.Rewind();
        Console.WriteLine("-----------------");

        while (true)
        {
            Console.WriteLine("\nКоманды: play, pause, stop, rewind,
exit");
            string command = Console.ReadLine();

            switch (command.ToLower())
            {
                case "play":
                    track.ChangeState(new PlayState());
                    Console.WriteLine(track);
                    track.Stop();
                    track.Play();
                    track.Pause();
                    track.Rewind();
                    Console.WriteLine("-----------------");
                    break;
                case "pause":
```

```
                    track.ChangeState(new PauseState());
                    Console.WriteLine(track);
                    track.Stop();
                    track.Play();
                    track.Pause();
                    track.Rewind();
                    Console.WriteLine("-----------------");
                    break;
                case "stop":
                    track.ChangeState(new StopState());
                    Console.WriteLine(track);
                    track.Stop();
                    track.Play();
                    track.Pause();
                    track.Rewind();
                    Console.WriteLine("-----------------");
                    break;
                case "rewind":
                    track.ChangeState(new RewindState());
                    Console.WriteLine(track);
                    track.Stop();
                    track.Play();
                    track.Pause();
                    track.Rewind();
                    Console.WriteLine("-----------------");
                    break;
                case "exit":
                    return;
                default:
                    Console.WriteLine("Неизвестная команда.");
                    break;
            }
        }
    }
}
```

3 проект.

Описание предметной области:
Пользователь может добавлять и удалять расходы или доходы, чтобы отслеживать их. Также он можеть отменять команды
Диаграмма классов:



Краткий вывод:
Паттерн предоставляет возможность выполнения команд добавления и удаления расходов или доходов, а также отменять последнее действие
Листинг кода:

```
internal interface ICommand
{
    void Execute();
    void Undo();
}

internal class Transaction
{
    public string Description { get; }
    public decimal Amount { get; }

    public Transaction(string description, decimal amount)
    {
        Description = description;
        Amount = amount;
    }

    public override string ToString()
    {
        return $"{Description} - {Amount}";
    }
}

internal class CommandManager
{
```

```
    private readonly Stack<ICommand> _commandHistory = new
Stack<ICommand>();

    public void ExecuteCommand(ICommand command)
    {
        command.Execute();
        _commandHistory.Push(command);
    }

    public void Undo()
    {
        if (_commandHistory.Count > 0)
        {
            var command = _commandHistory.Pop();
            command.Undo();
        }
    }
}

internal class RemoveTransactionCommand: ICommand
{
    private readonly List<Transaction> _transactions;
    private readonly Transaction _transaction;

    public RemoveTransactionCommand(List<Transaction> transactions,
Transaction transaction)
    {
        _transactions = transactions;
        _transaction = transaction;
    }

    public void Execute()
    {
        _transactions.Remove(_transaction);
    }

    public void Undo()
    {
        _transactions.Add(_transaction);
    }
}

internal class AddTransactionCommand: ICommand
{
    private readonly List<Transaction> _transactions;
    private readonly Transaction _transaction;

    public AddTransactionCommand(List<Transaction> transactions,
Transaction transaction)
    {
        _transactions = transactions;
        _transaction = transaction;
    }

    public void Execute()
    {
        _transactions.Add(_transaction);
    }

    public void Undo()
    {
        _transactions.Remove(_transaction);
    }
```

```
}

internal class Program
{
    static void Main(string[] args)
    {
        List<Transaction> transactions = new List<Transaction>();
        CommandManager commandManager = new CommandManager();

        while (true)
        {
            Console.WriteLine("1. Добавить доход/расход");
            Console.WriteLine("2. Удалить доход/расход");
            Console.WriteLine("3. Показать все транзакции");
            Console.WriteLine("4. Отменить последнюю команду");
            Console.WriteLine("5. Выход");
            Console.Write("Выберите действие: ");
            var choice = Console.ReadLine();

            switch (choice)
            {
                case "1":
                    Console.Write("Введите описание: ");
                    var description = Console.ReadLine();
                    Console.Write("Введите сумму: ");
                    var amount = decimal.Parse(Console.ReadLine());

                    var transaction = new Transaction(description, amount);
                    var addCommand = new
AddTransactionCommand(transactions, transaction);
                    commandManager.ExecuteCommand(addCommand);
                    break;

                case "2":
                    Console.Write("Введите индекс транзакции для удаления:
");
                    var index = int.Parse(Console.ReadLine());

                    if (index >= 0 && index < transactions.Count)
                    {
                        var removeCommand = new
RemoveTransactionCommand(transactions, transactions[index]);
                        commandManager.ExecuteCommand(removeCommand);
                    }
                    else
                    {
                        Console.WriteLine("Некорректный индекс.");
                    }
                    break;

                case "3":
                    Console.WriteLine("Транзакции:");
                    for (int i = 0; i < transactions.Count; i++)
                    {
                        Console.WriteLine($"{i}. {transactions[i]}");
                    }
                    break;

                case "4":
                    commandManager.Undo();
                    break;

                case "5":
```

```
                    return;

                default:
                    Console.WriteLine("Некорректный выбор. Попробуйте
снова.");
                    break;
            }
        }
    }
}
```