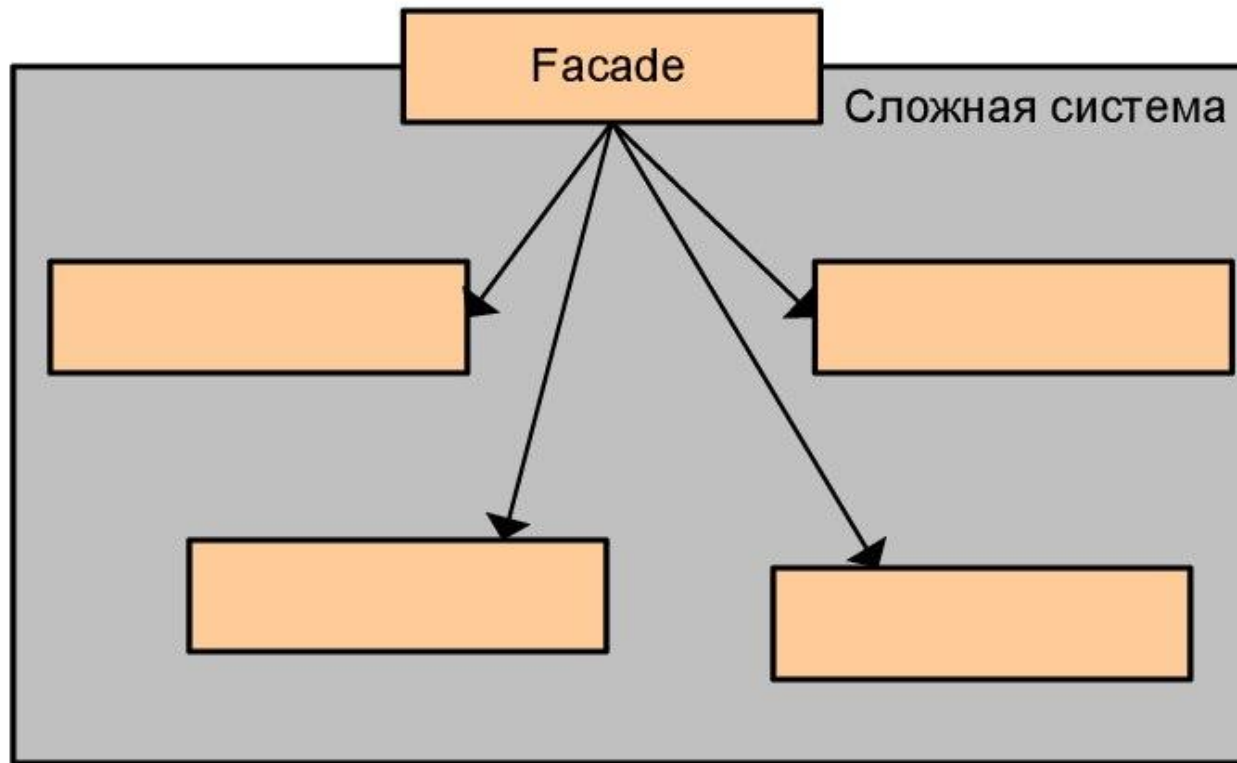


Паттерны проектирования

«Банды Четырех»

Facade



Фасад *Facade*

Тип: Структурный

Что это:

Предоставляет единый интерфейс к группе интерфейсов подсистемы.

Определяет высокоуровневый интерфейс, делая подсистему проще для использования.

Singleton

Одиночка

Singleton

Тип: Порождающий

Что это:

Гарантирует, что класс имеет только один экземпляр и предоставляет глобальную точку доступа к нему.

Singleton
-static uniqueInstance -singletonData
+static instance() +SingletonOperation()

```
Logger.Instance.Write("Preved, Medved");
```



5 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 2 changes

```
public sealed class Logger
```

```
{
```

1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 2 changes

```
private static Lazy<Logger> instance { get; } = new Lazy<Logger>(() => new Logger(), isThreadSafe:true);
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public static Logger Instance => instance.Value;
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public void Write(string message)
```

```
{
```

```
    // Логика логгирования
```

```
}
```

```
}
```

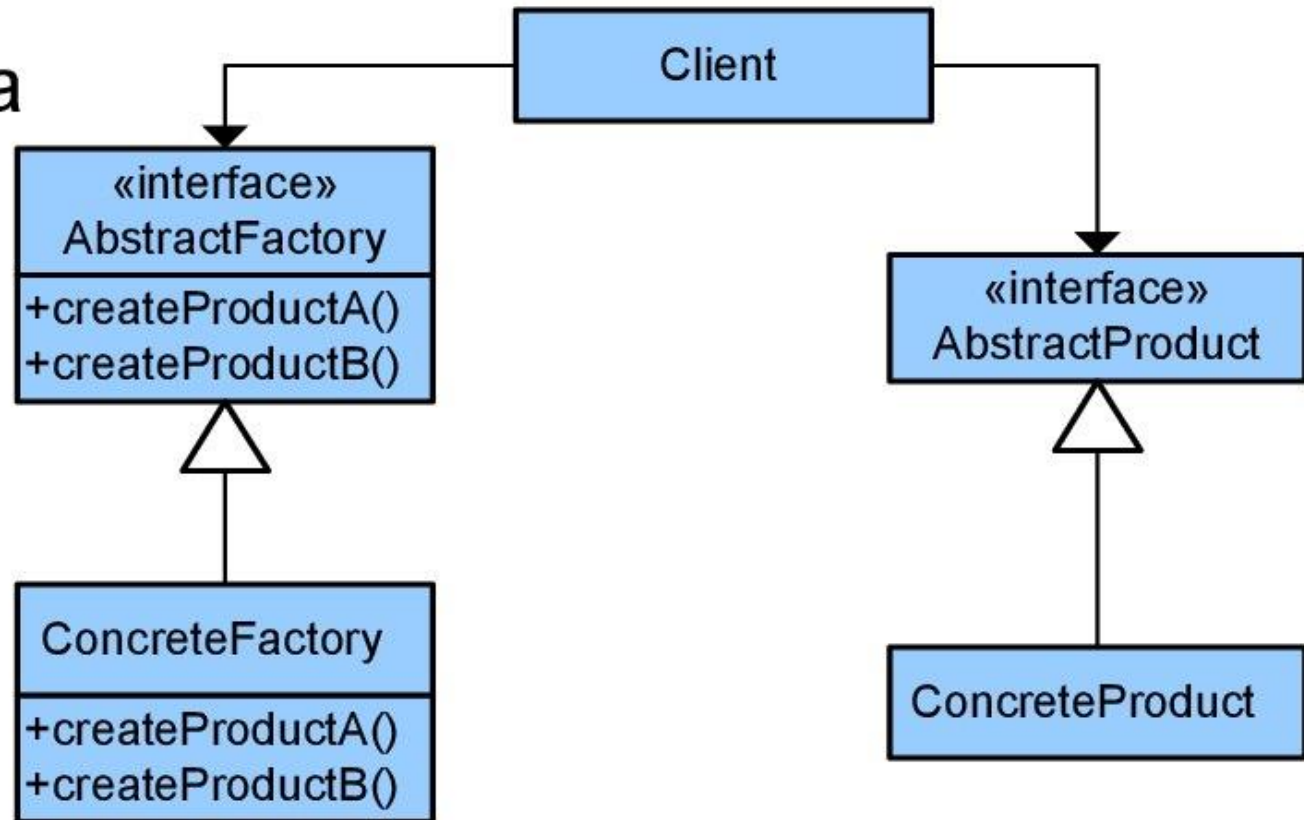
Abstract Factory

Абстрактная фабрика *Abstract factory*

Тип: Порождающий

Что это:

Предоставляет интерфейс для создания групп связанных или зависимых объектов, не указывая их конкретный класс.



```
IGuiFactory guiFactory = new WpfGuiFactory();

IWindow window = guiFactory.CreateWindow();
IButton button = guiFactory.CreateButton();
```

2 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change

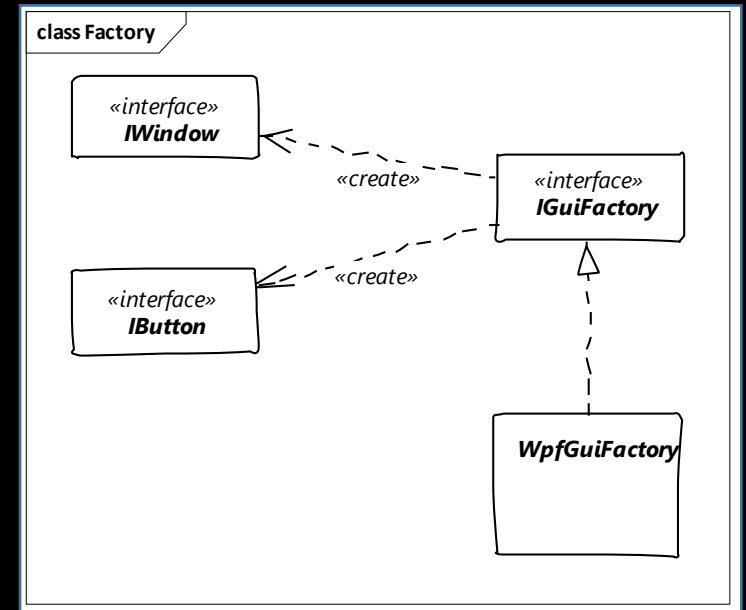
```
public interface IGuiFactory
{
    2 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    IWindow CreateWindow();

    2 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    IButton CreateButton();
}
```

1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change

```
public sealed class WpfGuiFactory : IGuiFactory
{
    2 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public IWindow CreateWindow()
    {
        return new WpfWindow();
    }

    2 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public IButton CreateButton()
    {
        throw new NotImplementedException();
    }
}
```



Factory Method

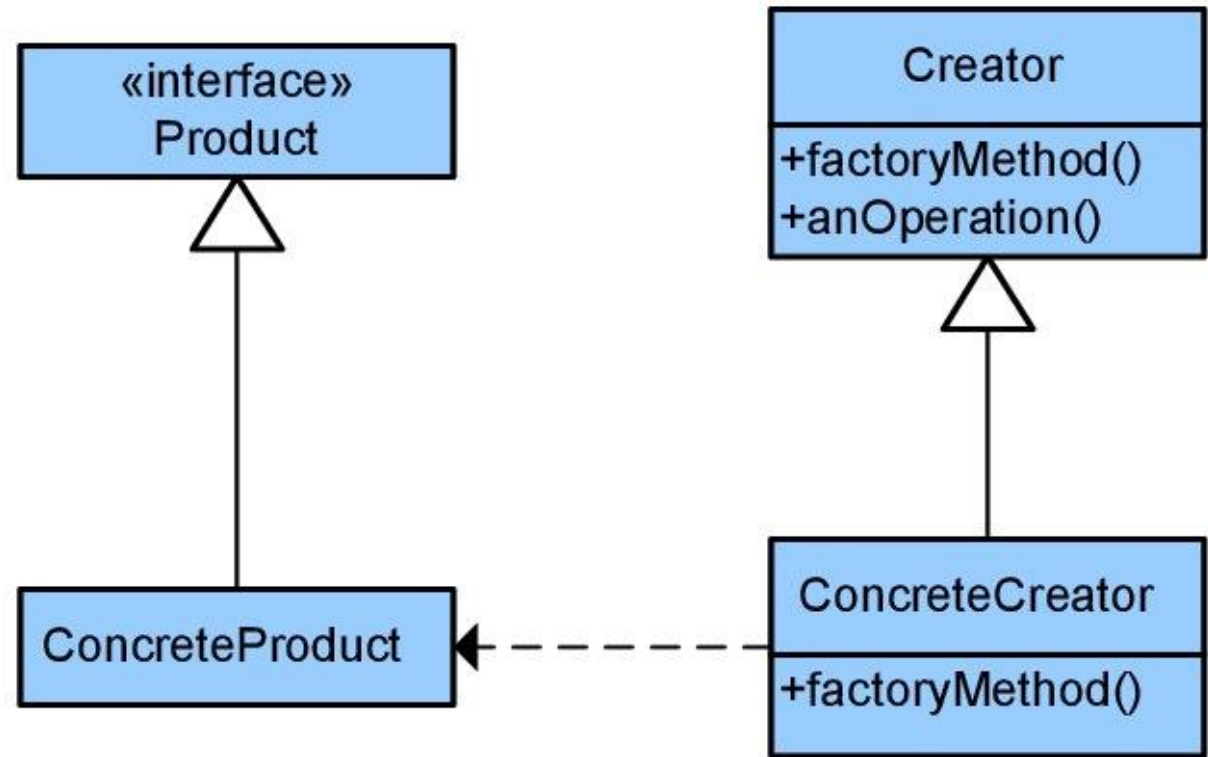
Фабричный метод

Factory method

Тип: Порождающий

Что это:

Определяет интерфейс для создания объекта, но позволяет подклассам решать, какой класс инстанцировать. Позволяет делегировать создание объекта подклассам.



```
var appointment = schedule.CreateAppointment(patient, doctor, from, to);
```

6 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public class Appointment
```

```
{
```

```
    private readonly Schedule schedule;
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
    internal Appointment(Schedule schedule, DateTime from, DateTime to, Doctor doctor, Patient patient)
```

```
    {
```

```
        this.schedule = schedule;
```

```
        From = from;
```

```
        To = to;
```

```
        Doctor = doctor;
```

```
        Patient = patient;
```

```
    }
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
    public DateTime From { get; }
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
    public DateTime To { get; }
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
    public Doctor Doctor { get; }
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
    public Patient Patient { get; }
```

```
}
```


3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public sealed class Schedule
```

```
{
```

```
    private readonly ICollection<Appointment> allAppointments = new HashSet<Appointment>();
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
    public Appointment CreateAppointment(Patient patient, Doctor doctor, DateTime from, DateTime to)
```

```
    {
```

```
        if (IsValidInterval(from, to))
```

```
        {
```

```
            throw new ArgumentException("Это интервал времени уже занят");
```

```
        }
```

```
        var appointment = new Appointment(this, from, to, doctor, patient);
```

```
        allAppointments.Add(appointment);
```

```
        return appointment;
```

```
    }
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
    private bool IsValidInterval(DateTime from, DateTime to)
```

```
    {
```

```
        return allAppointments.SingleOrDefault(o => Intersect(o, from, to)) != null;
```

```
    }
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
    private bool Intersect(Appointment appointment, DateTime from, DateTime to)
```

```
    {
```

```
        throw new NotImplementedException();
```

```
    }
```

```
}
```

Template Method

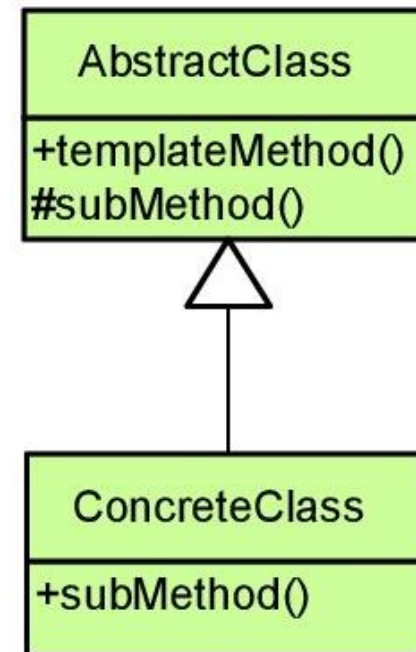
Шаблонный метод

Template method

Тип: Поведенческий

Что это:

Определяет алгоритм, некоторые этапы которого делегируются подклассам. Позволяет подклассам переопределить эти этапы, не меняя структуру алгоритма.




```
1 reference | 0 changes | 0 authors, 0 changes
public abstract class DataProcessor
{
    0 references | 0 changes | 0 authors, 0 changes
    public void Process()
    {
        PrepareData();

        AnalyzeData();
    }

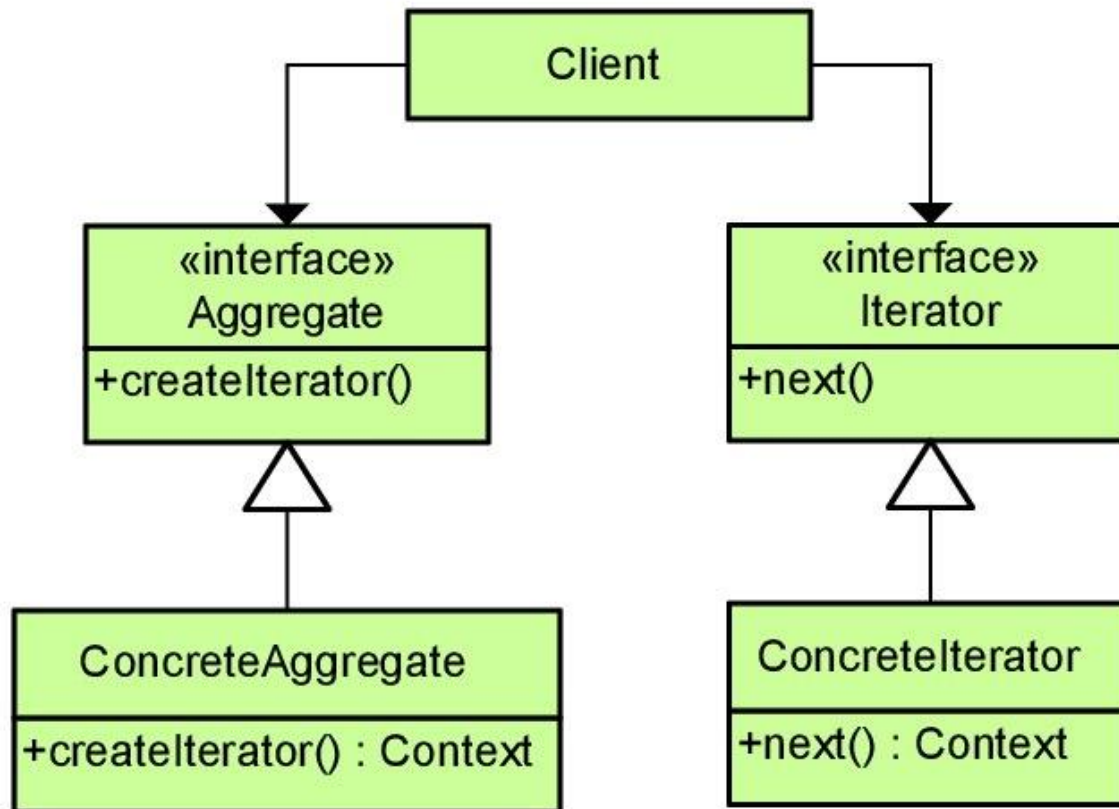
    2 references | 0 changes | 0 authors, 0 changes
    protected abstract void PrepareData();

    1 reference | 0 changes | 0 authors, 0 changes
    protected virtual void AnalyzeData()
    {
        // Алгоритм анализа данных.
    }
}
```



```
0 references | 0 changes | 0 authors, 0 changes
public sealed class SmartDataProcessor : DataProcessor
{
    2 references | 0 changes | 0 authors, 0 changes
    protected override void PrepareData()
    {
        // Реализация шаблонного метода.
    }
}
```

Iterator



Итератор

Iterator

Тип: Поведенческий

Что это:

Предоставляет способ последовательного доступа к элементам множества, независимо от его внутреннего устройства.

```
var str = new[] { "a", "b", "c", "d", "e", "f" };

foreach (var c in str.AsRandom())
{
    Console.WriteLine(c);
}
```



```
0 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
public static class EnumerableExtension
{
    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public static IEnumerable<T> AsRandom<T>(this T[] source)
    {
        var indexes = GenerateRandomSequence(source.Length);

        foreach (var i in indexes)
        {
            yield return source[i];
        }
    }

    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    private static int[] GenerateRandomSequence(int total)
    {
        var random = new Random();

        var sequence = new HashSet<int>();

        while (sequence.Count < total)
        {
            sequence.Add(random.Next(0, total));
        }

        return sequence.ToArray();
    }
}
```

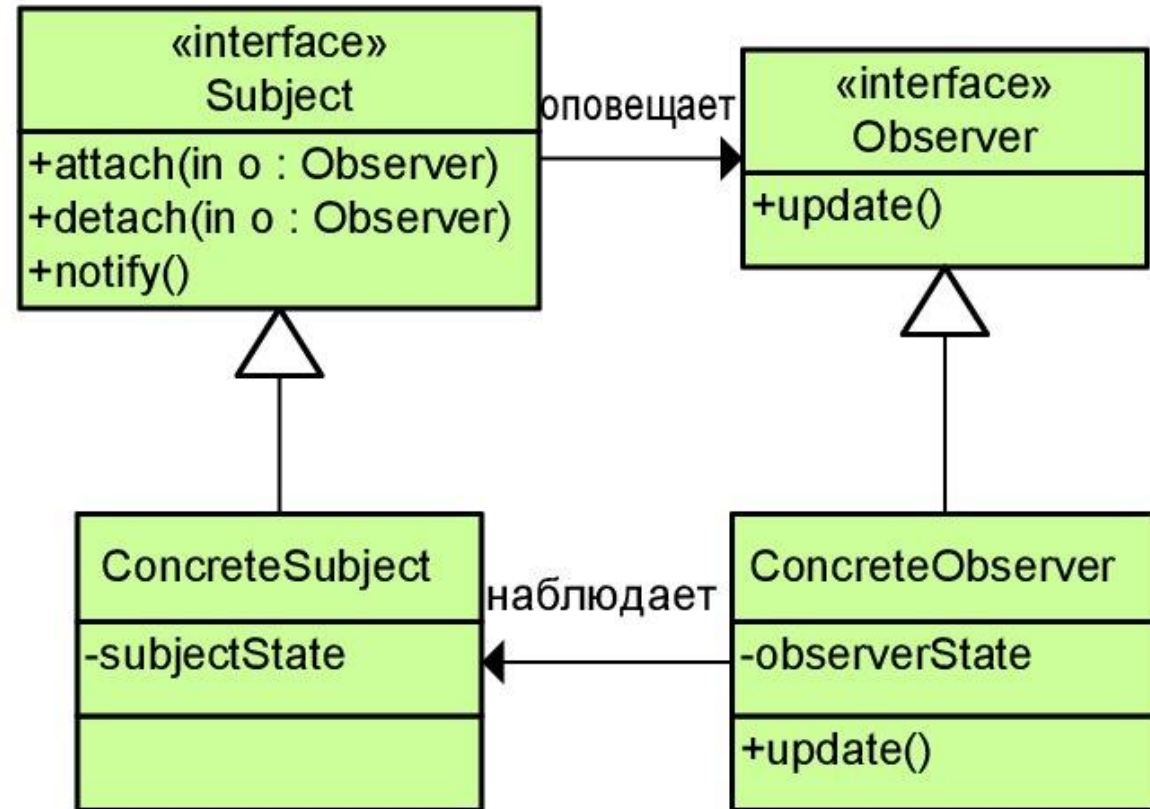
Observer

Наблюдатель *Observer*

Тип: Поведенческий

Что это:

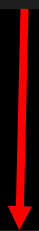
Определяет зависимость “один ко многим” между объектами так, что когда один объект меняет своё состояние, все зависимые объекты оповещаются и обновляются автоматически.



```
var chat = new Chat();

chat.IncomingMessages.Subscribe(msg =>
{
    Console.WriteLine($"Ответ: {msg}");
});

chat.SendMessage(new UserMessage("Привет, Медвед!"));
```

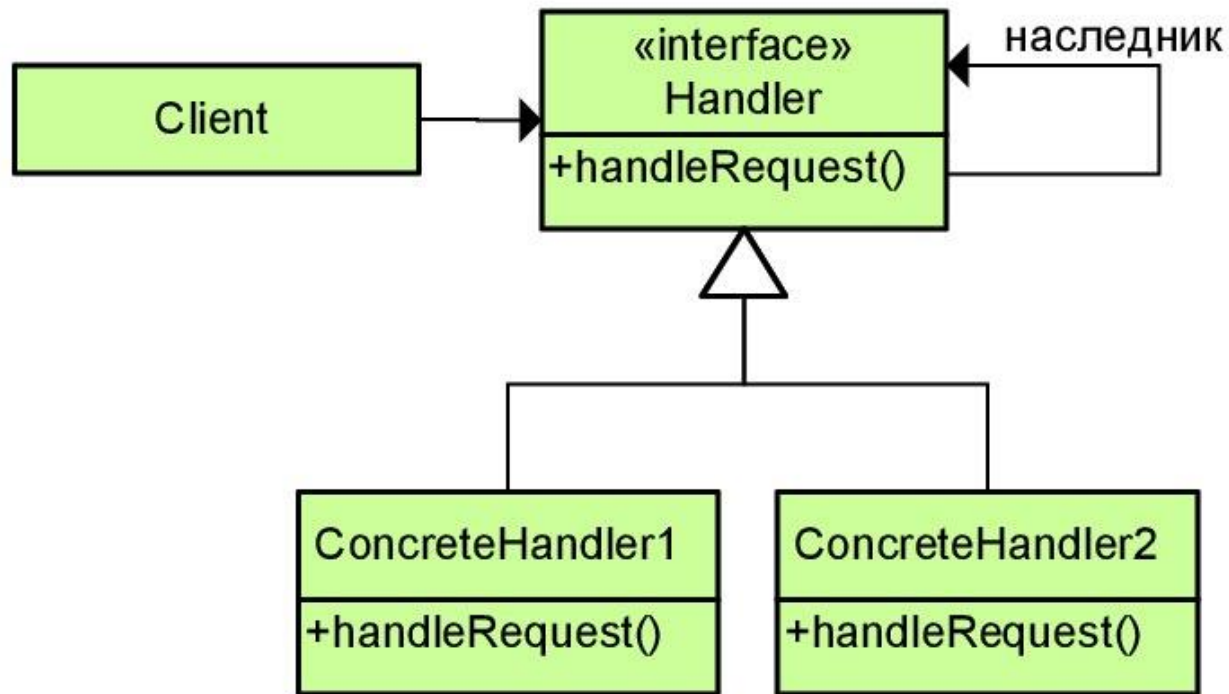


```
1 reference | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
public sealed class Chat
{
    private readonly ISubject<UserMessage> incomingStream = new ReplaySubject<UserMessage>();

    1 reference | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public IObservable<UserMessage> IncomingMessages
    {
        get
        {
            return incomingStream;
        }
    }

    1 reference | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public void SendMessage(UserMessage message)
    {
        // ...
    }
}
```

Chain of Responsibility



Цепочка обязанностей *Chain of responsibility*

Тип: Поведенческий

Что это:

Избегает связывания отправителя запроса с его получателем, давая возможность обработать запрос более чем одному объекту. Связывает объекты-получатели и передаёт запрос по цепочке пока объект не обработает его.


```
2 references | 0 changes | 0 authors, 0 changes
public interface IMessageHandler<TMessage>
{
    2 references | 0 changes | 0 authors, 0 changes
    void Handle(TMessage message);
}
```

```
0 references | 0 changes | 0 authors, 0 changes
public sealed class OrderProcess : IMessageHandler<OrderRequested>, IMessageHandler<OrderComplete>
{
    private readonly IMessagePublisher publisher;

    2 references | 0 changes | 0 authors, 0 changes
    public void Handle(OrderRequested message)
    {
        if (message.PaymentConfirmed)
        {
            publisher.Publish(new OrderComplete());
        }
    }

    2 references | 0 changes | 0 authors, 0 changes
    public void Handle(OrderComplete message)
    {
    }
}
```



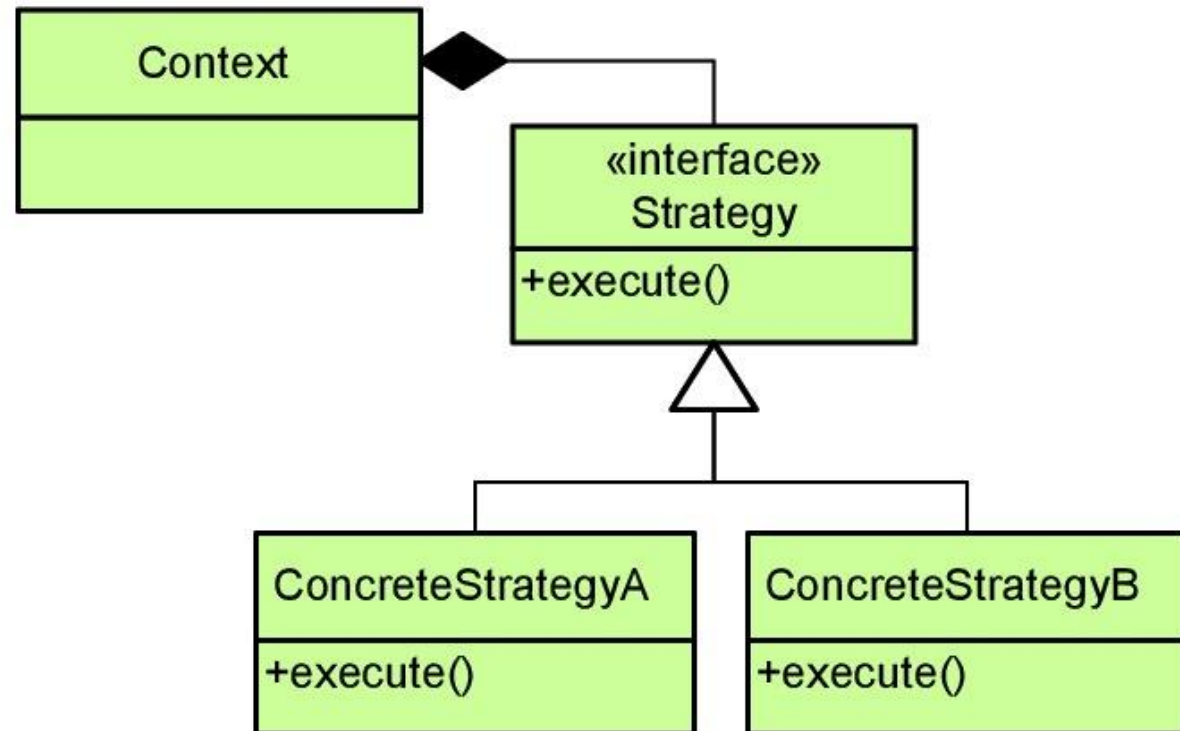
Strategy

Стратегия *Strategy*

Тип: Поведенческий

Что это:

Определяет группу алгоритмов, инкапсулирует их и делает взаимозаменяемыми. Позволяет изменять алгоритм независимо от клиентов, его использующих.



```
var catalog = new ApplicationCatalog();
var container = new CompositionContainer(catalog);

var converter = container.GetExportedValue<Converter>();
var y = "1.5";
var x = converter.Convert<float, string>(y);
```

[Export]

2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

public sealed class Converter

```
{
    private readonly IEnumerable<IConverterStrategy> strategies;

    [ImportingConstructor]
    0 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public Converter([ImportMany] IEnumerable<IConverterStrategy> strategies)
    {
        this.strategies = strategies;
    }
}
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

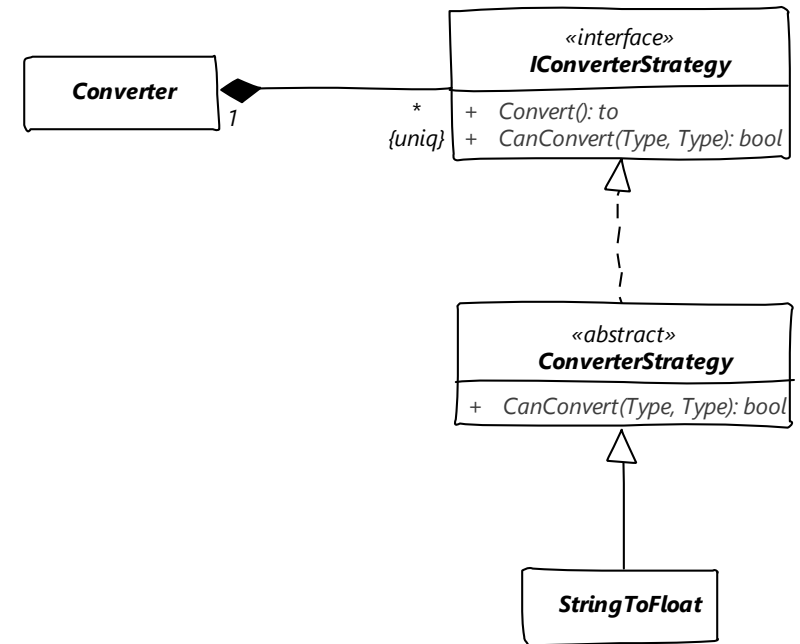
public TTo Convert<TTo, TFrom>(TFrom source)

```
{
    var strategy = strategies.SingleOrDefault(s => s.CanConvert<TFrom, TTo>());

    if (strategy == null)
    {
        throw new ArgumentException($"Нет конвертера {typeof(TFrom)}->{typeof(TTo)}");
    }

    return ((IConverterStrategy<TFrom, TTo>)strategy).Convert(source);
}
```

class Strategy



2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public interface IConverterStrategy<in TFrom, out TTo> : IConverterStrategy
{
    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    TTo Convert(TFrom source);
}
```

4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public interface IConverterStrategy
{
    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    bool CanConvert<TFrom, TTo>();
}
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public abstract class ConverterStrategy<TFrom, TTo> : IConverterStrategy<TFrom, TTo>
{
    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public bool CanConvert<TSource, TDestination>()
    {
        return typeof(TSource) == typeof(TFrom) && typeof(TDestination) == typeof(TTo);
    }

    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public abstract TTo Convert(TFrom source);
}
```

[AttributeUsage(AttributeTargets.Class)]

2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

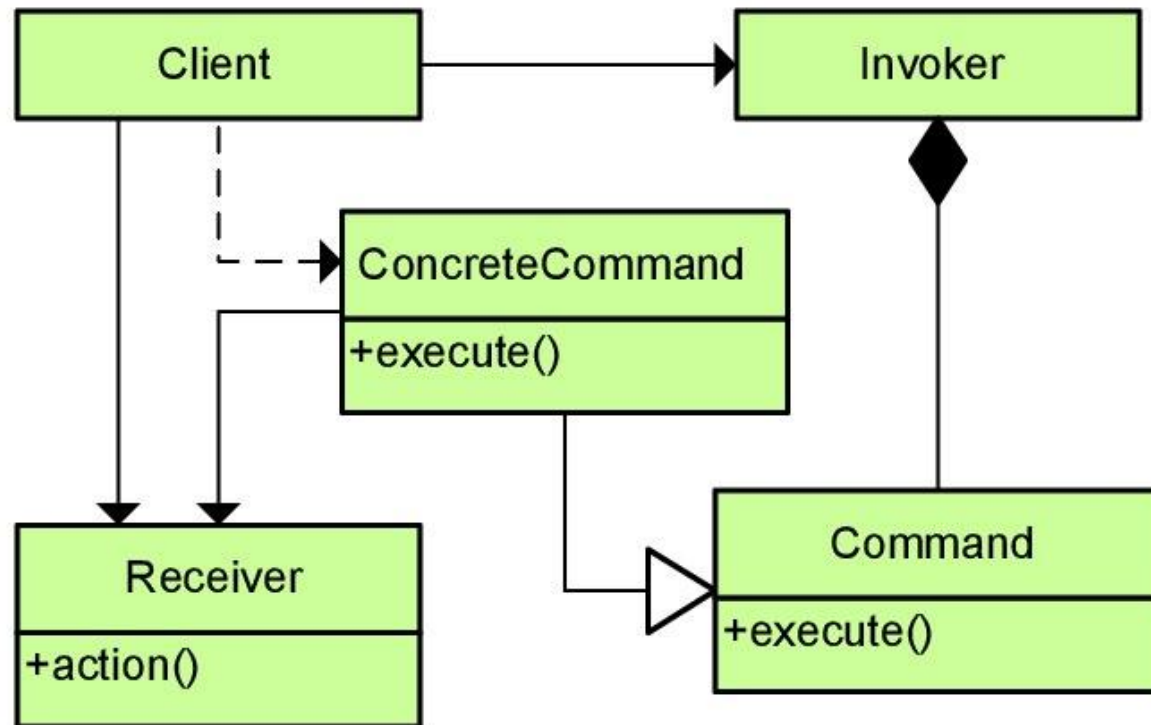
```
public sealed class ConverterStrategyAttribute : ExportAttribute
{
    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public ConverterStrategyAttribute() : base(typeof(IConverterStrategy))
    {
    }
}
```

[ConverterStrategy]

0 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public sealed class StringToFloat : ConverterStrategy<string, float>
{
    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public override float Convert(string source)
    {
        return float.Parse(source, CultureInfo.InvariantCulture);
    }
}
```

Command



Команда *Command*

Тип: Поведенческий

Что это:

Инкапсулирует запрос в виде объекта, позволяя передавать их клиентам в качестве параметров, ставить в очередь, логировать а также поддерживает отмену операций.

```
var filesystemManager = new FilesystemManager(new FilesystemStub());  
filesystemManager.CopyDirectory("c://old", "d://new");
```

```
namespace Command.Commands  
{  
    5 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change  
    public interface ICommand  
    {  
        4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change  
        void Execute();  
  
        4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change  
        void Unexecute();  
    }  
}
```

```
2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change  
public sealed class FilesystemManager  
{  
    private readonly IFfilesystem filesystem;  
  
    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change  
    public FilesystemManager(IFfilesystem filesystem)  
    {  
        this.filesystem = filesystem;  
    }  
  
    0 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change  
    public void CopyFile(string source, string destination)  
    {  
        var command = new FileCopyCommand(filesystem, source, destination);  
        ExecuteOrRecover(command);  
    }  
  
    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change  
    public void CopyDirectory(string source, string destination)  
    {  
        var command = new DirectoryCopyCommand(filesystem, source, destination);  
        ExecuteOrRecover(command);  
    }  
  
    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change  
    private static void ExecuteOrRecover(ICommand command)  
    {  
        try  
        {  
            command.Execute();  
        }  
        catch (IOException)  
        {  
            command.Unexecute();  
  
            throw new IOException("Ошибка и восстановление");  
        }  
    }  
}
```


```
namespace Command.Commands
{
    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public sealed class FileCopyCommand : ICommand
    {
        private readonly IFilesystem filesystem;

        private readonly string source;
        private readonly string destination;

        2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
        public FileCopyCommand(IFilesystem filesystem, string source, string destination)
        {
            this.filesystem = filesystem;
            this.source = source;
            this.destination = destination;
        }

        4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
        public void Execute()
        {
            filesystem.CopyFile(source, destination);
        }

        4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
        public void Unexecute()
        {
            filesystem.DeleteFile(destination);
        }
    }
}
```



```
7 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
public interface IFilesystem
{
    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    void CopyFile(string source, string destination);

    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    IEnumerable<string> GetFiles(string source);

    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    void CreateDirectory(string destination);

    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    void DeleteFile(string destination);

    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    void DeleteEmptyDirectory(string destination);
}
```

```

public sealed class DirectoryCopyCommand : ICommand
{
    private readonly IFilesystem filesystem;
    private readonly Stack<ICommand> children = new Stack<ICommand>();...
    private readonly string source;
    private readonly string destination;

    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public DirectoryCopyCommand(IFilesystem filesystem, string source, string destination)
    {
        this.filesystem = filesystem;

        this.source = source;
        this.destination = destination;
    }

    4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public void Execute()
    {
        var allFiles = filesystem.GetFiles(source);

        foreach (var file in allFiles)
        {
            var command = new FileCopyCommand(filesystem, $"{source}/{file}", $"{destination}/{file}");

            command.Execute();

            children.Push(command);
        }

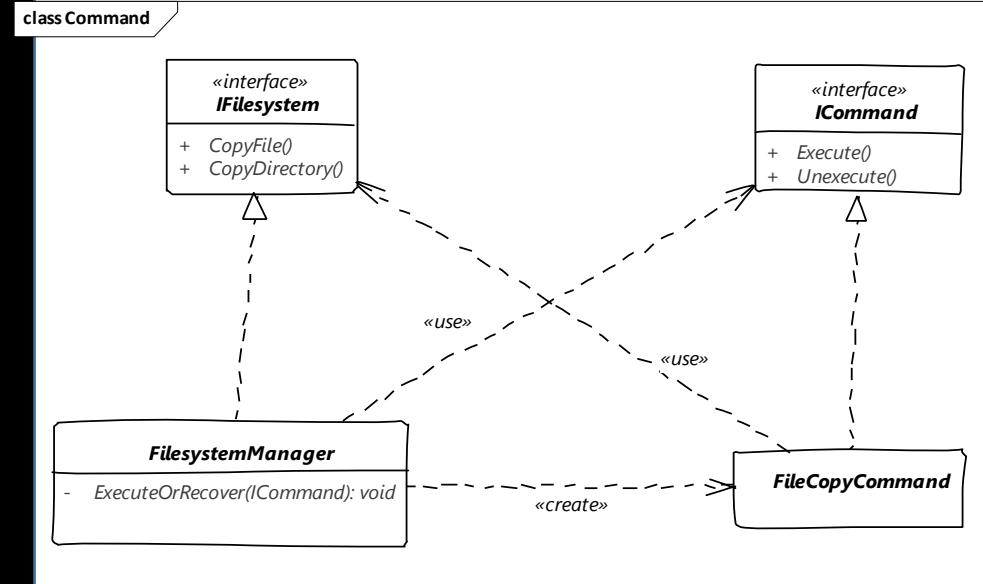
        filesystem.CreateDirectory(destination);
    }

    4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public void Unexecute()
    {
        while (children.Any())
        {
            var command = children.Pop();

            command.Unexecute();
        }

        filesystem.DeleteEmptyDirectory(destination);
    }
}

```



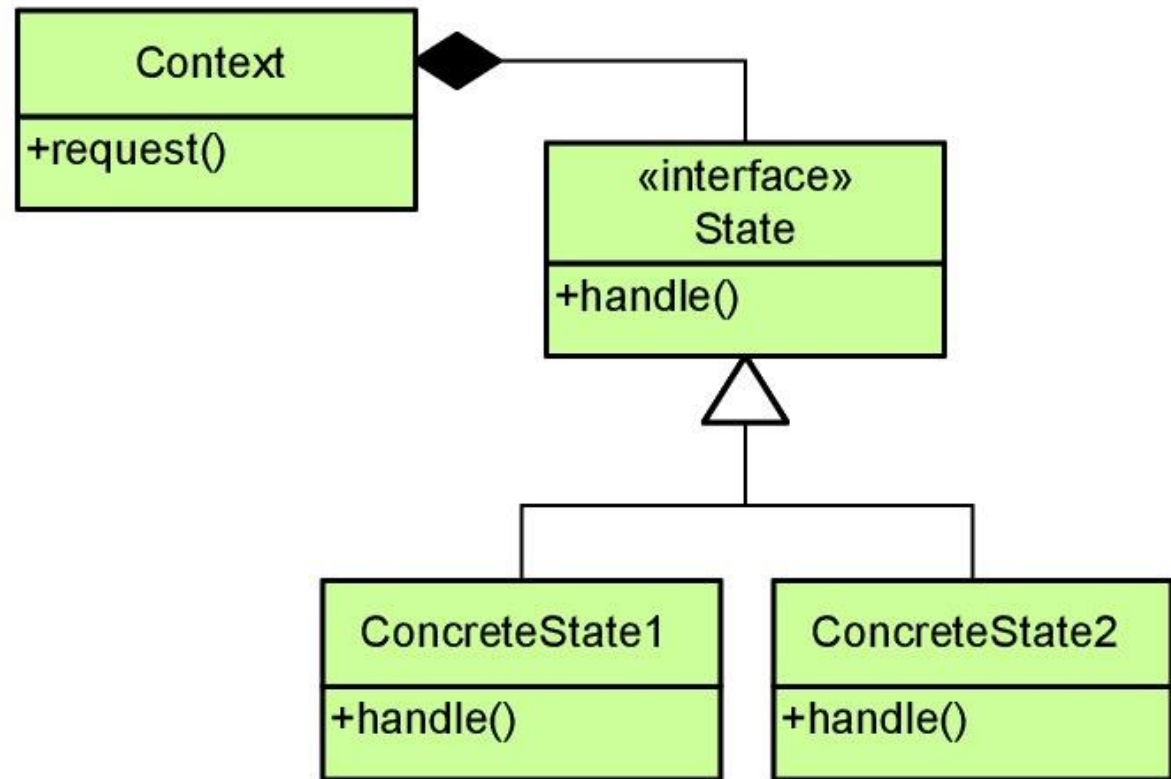
State

Состояние *State*

Тип: Поведенческий

Что это:

Позволяет объекту изменять своё поведение в зависимости от внутреннего состояния.



```
var wizard = new InstallationWizard();

wizard.GoNext();

wizard.GoNext();

wizard.Cancel();
```

```
2 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
public interface IWizard
{
    7 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    string Text { get; }

    6 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    void GoNext();

    4 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    void GoPrev();

    6 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    void Cancel();

    3 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    void Finish();
}
```

```
7 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
public interface IStateable<in TState>
{
    3 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    void SetNextState(TState nextState);
}
```

```
2 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
public sealed class InstallationWizard : IWizard, IStateable<IWizardState>
{
    private IWizardState currentState;

    1 reference | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public InstallationWizard()
    {
        SetNextState(new WelcomeState(this));...
    }

    #region Implementation of IStateable<IWizard>

    3 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public void SetNextState(IWizardState nextState)
    {
        currentState?.Exit();

        currentState = nextState;

        currentState.Enter();
    }

    #endregion

    #region Implementation of IWizard

    7 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public string Text => currentState.Text;

    6 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public void GoNext() => currentState.GoNext();

    4 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public void GoPrev() => currentState.GoPrev();

    6 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public void Cancel() => currentState.Cancel();

    3 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public void Finish() => currentState.Finish();

    #endregion
}
```

5 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change

```
public abstract class WizardState : IWizardState
```

```
{  
    protected readonly IStateable<IWizardState> stateHolder;
```

4 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change

```
protected WizardState(IStateable<IWizardState> stateHolder)  
{  
    this.stateHolder = stateHolder;  
}
```

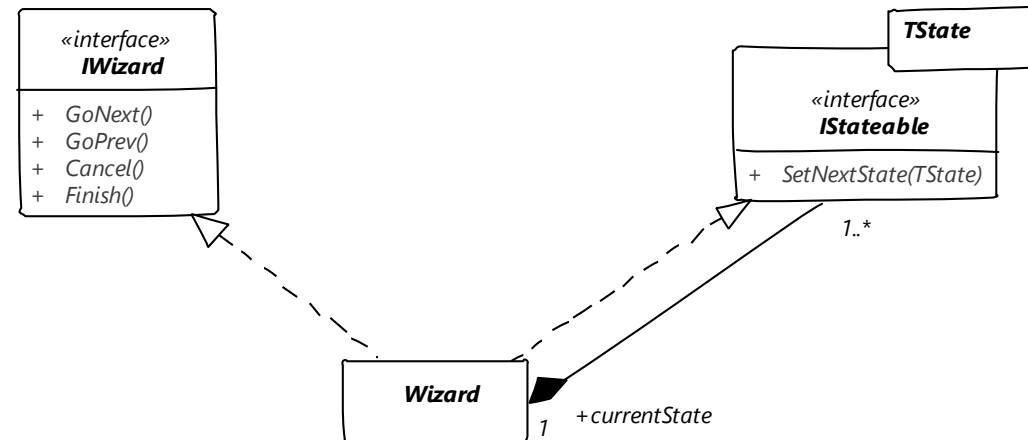
Implementation of IWizard

Implementation of IWizardState

5 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change

```
protected void SetNextState(IWizardState nextState)  
{  
    stateHolder.SetNextState(nextState);...  
}
```

class State



3 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change

```
public sealed class WelcomeState : WizardState
```

```
{  
    2 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change  
    public WelcomeState(IStateable<IWizardState> stateHolder) : base(stateHolder)  
    {  
    }  
  
    7 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change  
    public override string Text { get; set; } = "Welcome!";  
  
    6 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change  
    public override void Cancel()  
    {  
        Console.WriteLine("Отмена инсталляции...");  
  
        SetNextState(new CanceledState(stateHolder));  
    }  
  
    6 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change  
    public override void Enter()  
    {  
        Console.WriteLine("Привет! Это суперинсталлятор для мегапрограммы.");  
    }  
}
```

2 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change

```
public sealed class LicenseConfirmationState : WizardState
```

```
{  
    private bool userConfirm;  
  
    1 reference | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change  
    public LicenseConfirmationState(IStateable<IWizardState> stateHolder) : base(stateHolder)  
    {  
    }  
  
    7 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change  
    public override string Text { get; set; } = "License";  
  
    6 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change  
    public override void GoNext()  
    {  
        if (userConfirm)  
        {  
            SetNextState(new CopyingFilesState(stateHolder));  
        }  
    }  
  
    4 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change  
    public override void GoPrev()  
    {  
        Console.WriteLine("Возврат к предыдущей странице");  
  
        SetNextState(new WelcomeState(stateHolder));  
    }  
  
    6 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change  
    public override void Enter()  
    {  
        Console.WriteLine("Согласитесь с условиями?");  
  
        userConfirm = true;  
    }  
}
```

Visitor

Посетитель

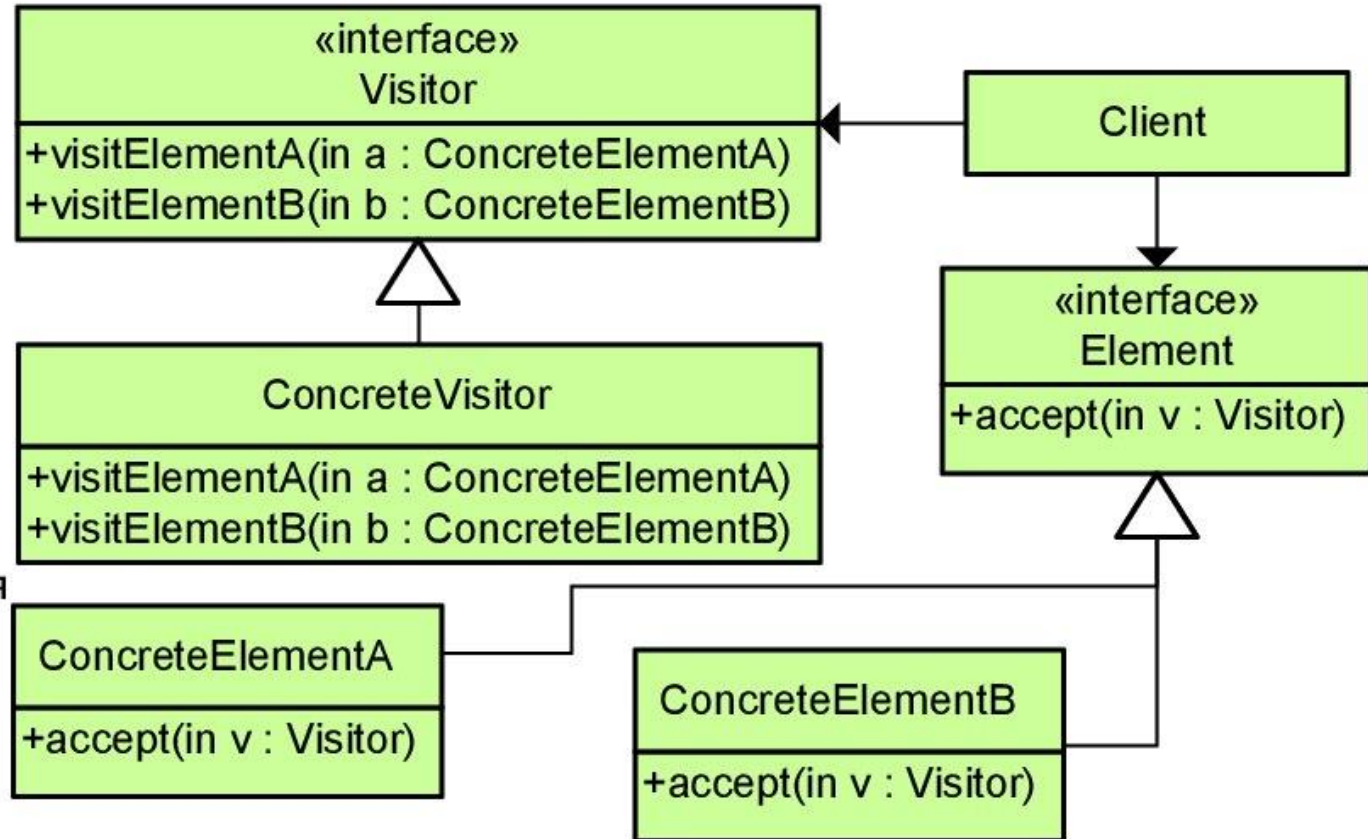
Visitor

Тип: Поведенческий

Что это:

Представляет собой операцию, которая будет выполнена над объектами группы классов.

Даёт возможность определить новую операцию без изменения кода классов, над которыми эта операция проводится.



```
var visitor = new RenderVisitor();

var element = new BoldText { Content = "это html тег" };
element.Accept(visitor);
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public interface IHtmlVisitable
{
    5 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    void Accept(IHtmlVisitor visitor);
}
```

6 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public interface IHtmlVisitor
{
    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    void Visit(PlainText element);

    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    void Visit(BoldText element);

    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    void Visit(StrikeoutText element);
}
```

3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public abstract class HtmlElement : IHtmlVisitable
{
    4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public string Content { get; set; }

    5 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public abstract void Accept(IHtmlVisitor visitor);
}
```

```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
public sealed class RenderVisitor : IHtmlVisitor
{
    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public void Visit(PlainText element)
    {
        Console.Write(element.Content);
    }

    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public void Visit(BoldText element)
    {
        Console.Write($"<b>{element.Content}</b>");
    }

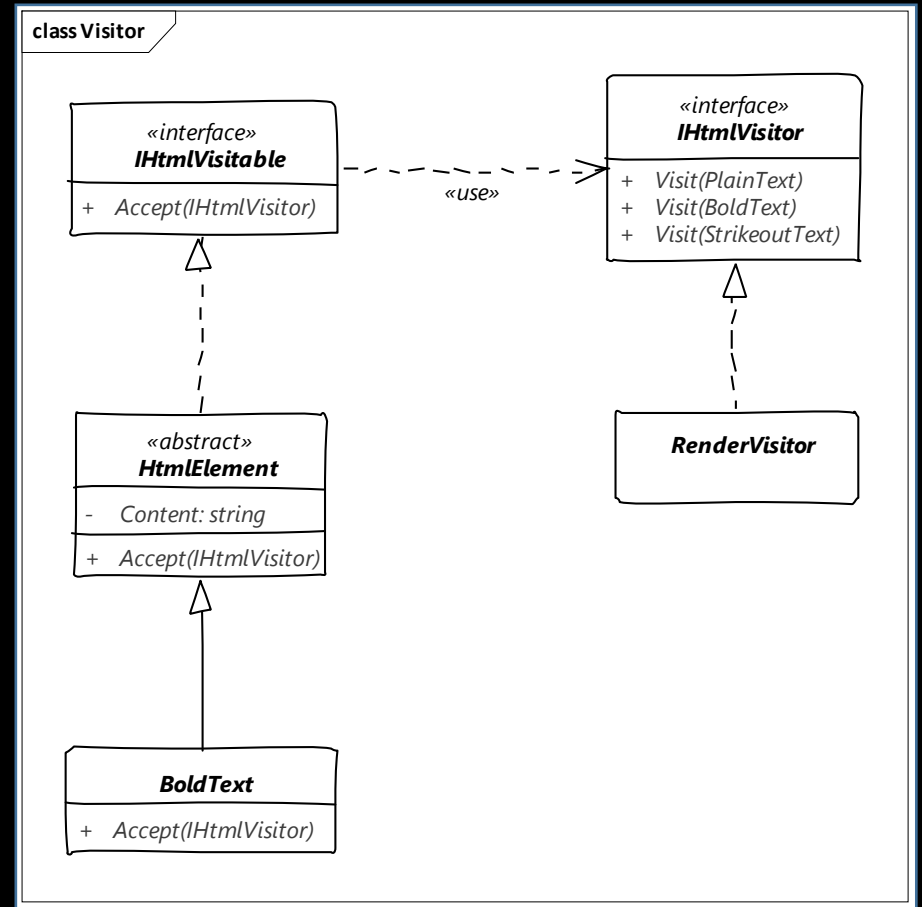
    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public void Visit(StrikeoutText element)
    {
        Console.Write($"<strike>{element.Content}</strike>");
    }
}

```

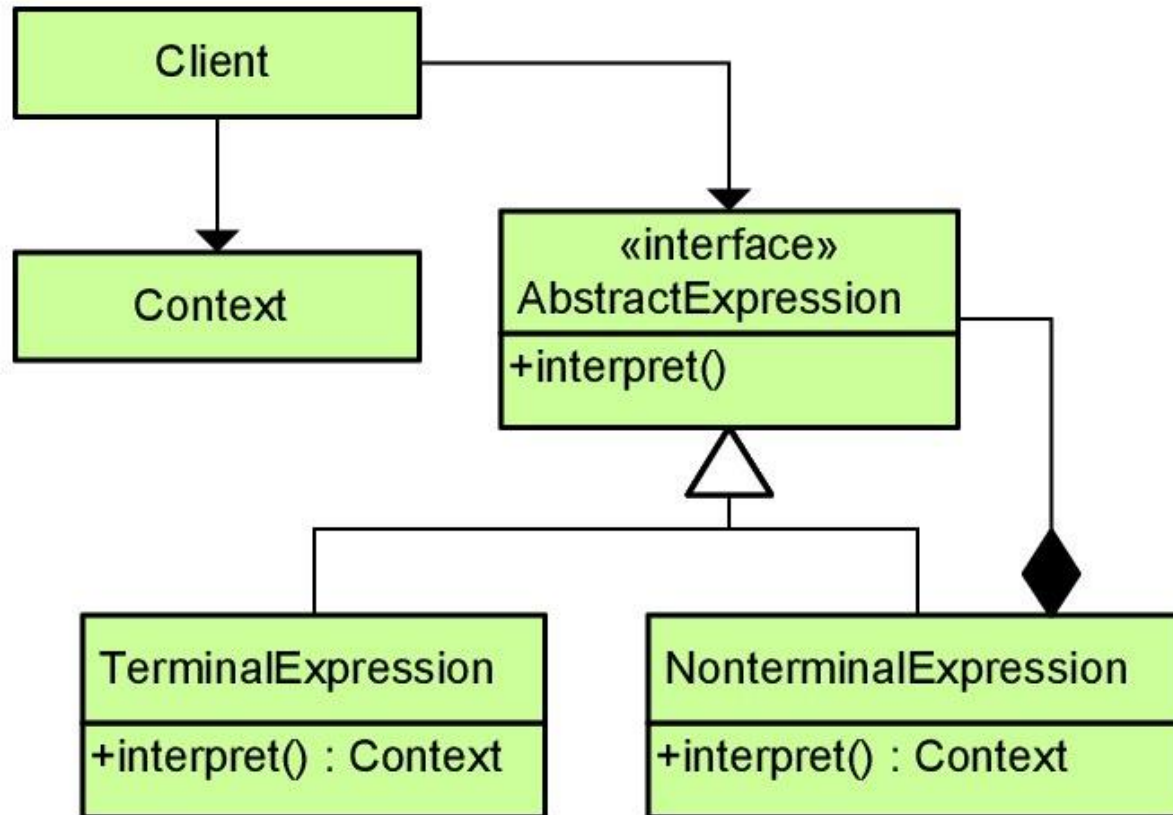
```

3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
public sealed class BoldText : HtmlElement
{
    5 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public override void Accept(IHtmlVisitor visitor)
    {
        visitor.Visit(this);
    }
}

```



Interpreter



Интерпретатор *Interpreter*

Тип: Поведенческий

Что это:

Получая формальный язык, определяет представление его грамматики и интерпретатор, использующий это представление для обработки выражений языка.


```
7 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
public sealed class ConstantExpression : Expression
{
    private readonly int value;

    6 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public ConstantExpression(int value)
    {
        this.value = value;
    }

    9 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public override int Evaluate()
    {
        return value;
    }
}
```

```
16 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
public abstract class Expression
{
    9 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public abstract int Evaluate();

    1 reference | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public static Expression operator +(Expression left, Expression right)
    {
        return new SumExpression(left, right);
    }

    1 reference | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public static Expression operator *(Expression left, Expression right)
    {
        return new MulExpression(left, right);
    }
}
```

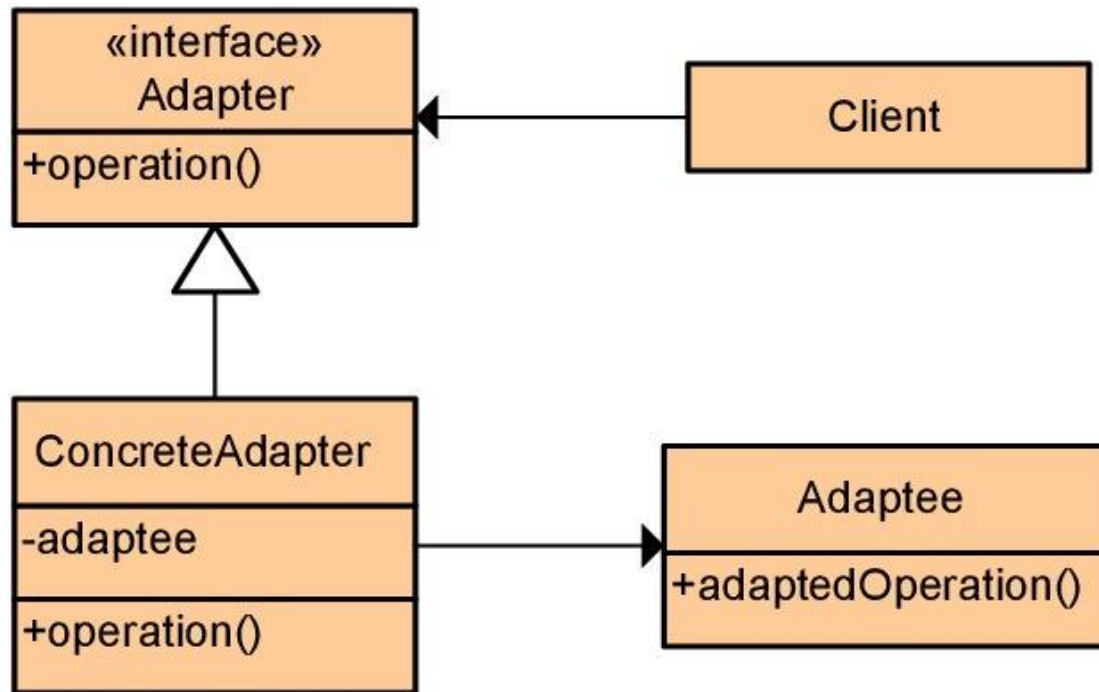
```
namespace Interpreter
{
    3 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public sealed class SumExpression : BinaryExpression
    {
        2 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
        public SumExpression(Expression left, Expression right) : base(left, right)
        {
        }

        9 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
        public override int Evaluate()
        {
            return left.Evaluate() + right.Evaluate();
        }
    }
}
```

```
3 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
public sealed class MulExpression : BinaryExpression
{
    2 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public MulExpression(Expression left, Expression right) : base(left, right)
    {
    }

    9 references | SPBHELIX\karinskiy.a, 4 days ago | 1 author, 1 change
    public override int Evaluate()
    {
        return left.Evaluate() * right.Evaluate();
    }
}
```

Adapter



Адаптер *Adapter*

Тип: Структурный

Что это:

Конвертирует интерфейс класса в другой интерфейс, ожидаемый клиентом. Позволяет классам с разными интерфейсами работать вместе.

2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 2 changes

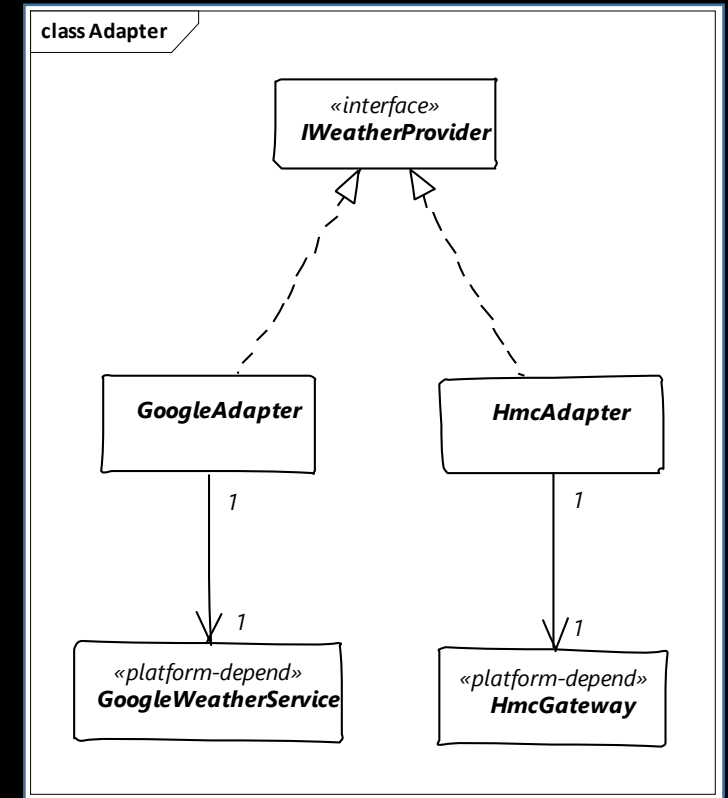
```
public interface IWeatherProvider
```

```
{  
    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change  
    IEnumerable<WeatherInfo> GetWeather(float altitude, float longitude, DateTime from, DateTime to);  
}
```

4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public sealed class WeatherInfo
```

```
{  
    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change  
    public float Temperature { get; set; }  
  
    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change  
    public float Humidity { get; set; }  
  
    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change  
    public float Pressure { get; set; }  
}
```



namespace Adapter.Google

```
{
    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public sealed class GoogleWeatherDto
    {
        1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
        public float Temper { get; set; }

        1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
        public float Press { get; set; }

        1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
        public float Humid { get; set; }
    }
}
```

namespace Adapter.Google

```
{
    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public sealed class GoogleWeatherService
    {
        1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
        public GoogleWeatherService(string url) {...}

        1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
        public GoogleWeatherDto[] GetWeather(float x, float y, DateTime fromTime, DateTime toTime) {...}
    }
}
```

4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
public sealed class WeatherInfo

```
{
    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public float Temperature { get; set; }

    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public float Humidity { get; set; }

    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public float Pressure { get; set; }
}
```

namespace Adapter.Google

```
{
    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public sealed class GoogleAdapter : IWeatherProvider
    {
        private readonly GoogleWeatherService adaptee;

        0 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
        public GoogleAdapter()
        {
            adaptee = new GoogleWeatherService("http://google.weather.com");
        }

        2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
        public IEnumerable<WeatherInfo> GetWeather(float altitude, float longitude, DateTime from, DateTime to)
        {
            // Получение данных из конкретного провайдера погоды.
            var googleWeathers = adaptee.GetWeather(x: altitude, y: longitude, fromTime: from, toTime: to);

            // Адаптация модели провайдера к абстрактной модели.
            foreach (var sourceDto in googleWeathers)
            {
                yield return new WeatherInfo
                {
                    Temperature = sourceDto.Temper,
                    Humidity = sourceDto.Humid,
                    Pressure = sourceDto.Press
                };
            }
        }
    }
}
```

Proxy

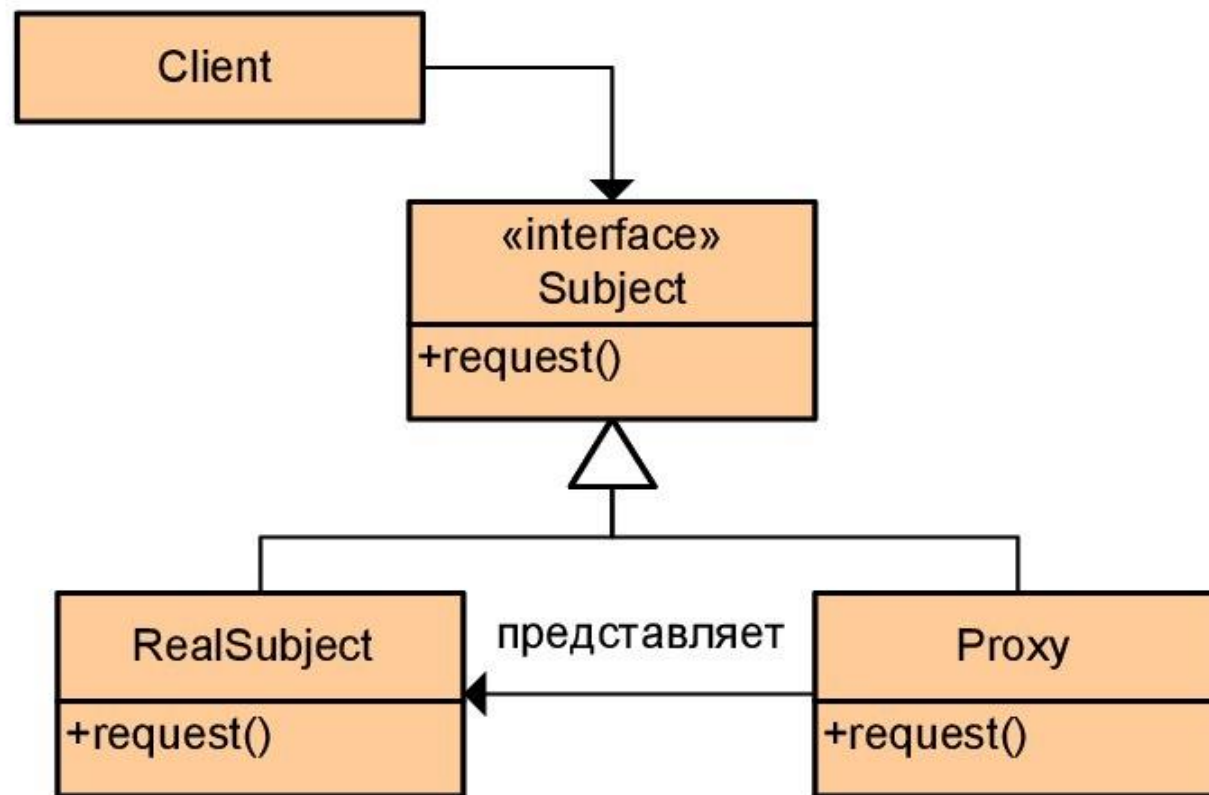
Прокси

Proxy

Тип: Структурный

Что это:

Предоставляет замену другого объекта для контроля доступа к нему.



3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public class Employee
{
    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public virtual string FirstName { get; set; }

    0 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public virtual string LastName { get; set; }

    0 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public virtual ICollection<Company> Companies { get; set; }
}
```

```
internal class EmployeeDataAccessProxy : Employee
{
    private readonly PersonalityContext db;
    private readonly IEquatable<object> primaryKey;

    private Employee state;

    0 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public EmployeeDataAccessProxy(PersonalityContext db, IEquatable<object> primaryKey) ...

    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public override string FirstName
    {
        get
        {
            LoadIfNeeded();
            return state.FirstName;
        }

        set
        {
            LoadIfNeeded();
            state.FirstName = value;
        }
    }

    2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    private void LoadIfNeeded()
    {
        if (NotLoaded())
        {
            Load();
        }
    }

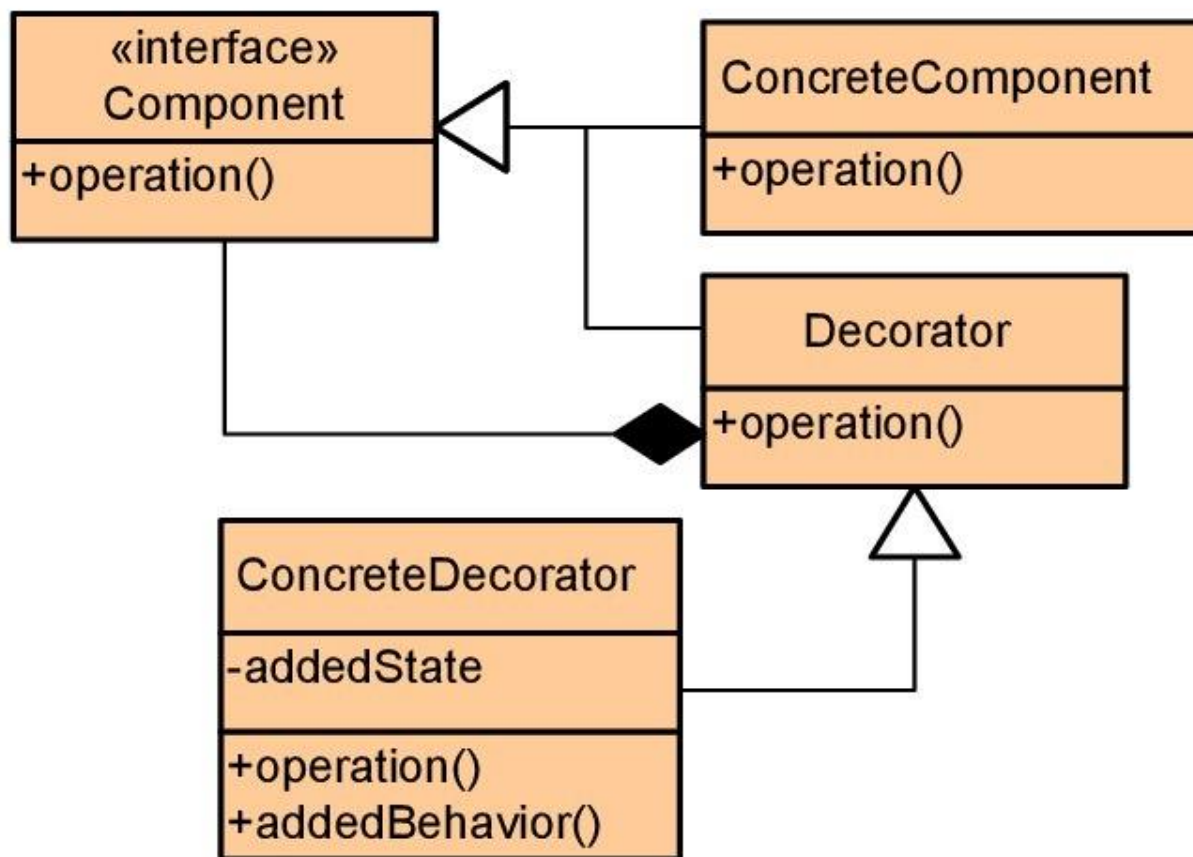
    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    private void Load()
    {
        state = db.Employees.Find(primaryKey);

        MarkAsLoaded();
    }

    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    private bool NotLoaded()

```

Decorator



Декоратор

Decorator

Тип: Структурный

Что это:

Динамически предоставляет объекту дополнительные возможности.

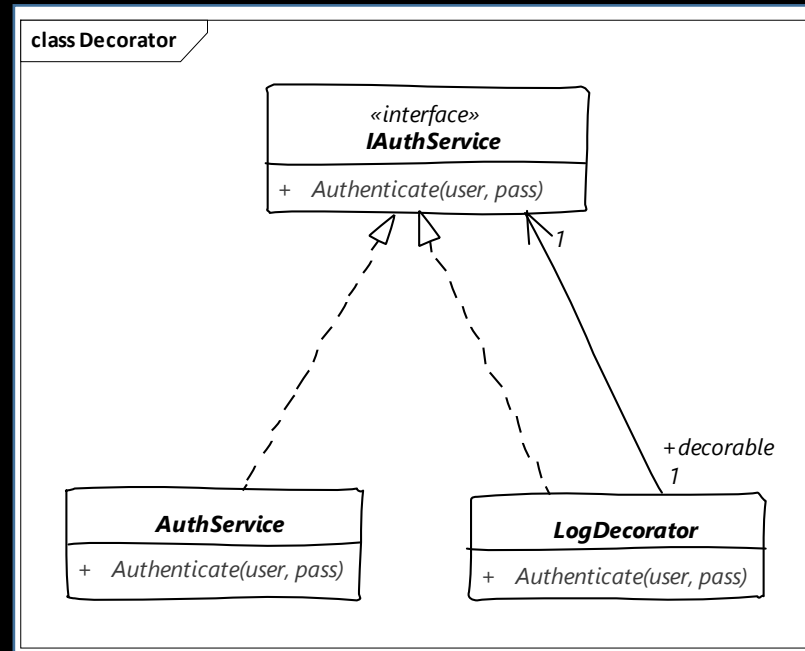
Представляет собой гибкую альтернативу наследованию для расширения функциональности.


```

IAuthenticationService service = new AuthenticationService();
IAuthenticationService loggingDecorator = new AuthenticationLogDecorator(service);

loggingDecorator.Authenticate("Василий", "12345");

```



6 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```

public interface IAuthService
{
    4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    void Authenticate(string user, string password);
}

```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```

public sealed class AuthenticationService : IAuthService
{
    4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public void Authenticate(string user, string password)
    {
        // Логика аутентификации...
    }
}

```

2 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public sealed class AuthenticationLogDecorator : IAuthenticationService
```

```
{
```

```
    private readonly IAuthenticationService decorable;
```

1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
    public AuthenticationLogDecorator(IAuthenticationService decorable)
```

```
    {
```

```
        this.decorable = decorable;
```

```
    }
```

4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
    public void Authenticate(string user, string password)
```

```
    {
```

```
        Trace.TraceInformation("Запрос аутентификации");
```

```
        try
```

```
        {
```

```
            // Делегирование вызова.
```

```
            decorable.Authenticate(user, password);
```

```
            Trace.TraceInformation("Пользователь аутентифицирован");
```

```
        }
```

```
        catch (AuthenticationException exception)
```

```
        {
```

```
            Trace.TraceError($"Ошибка аутентификации: {exception.Message}");
```

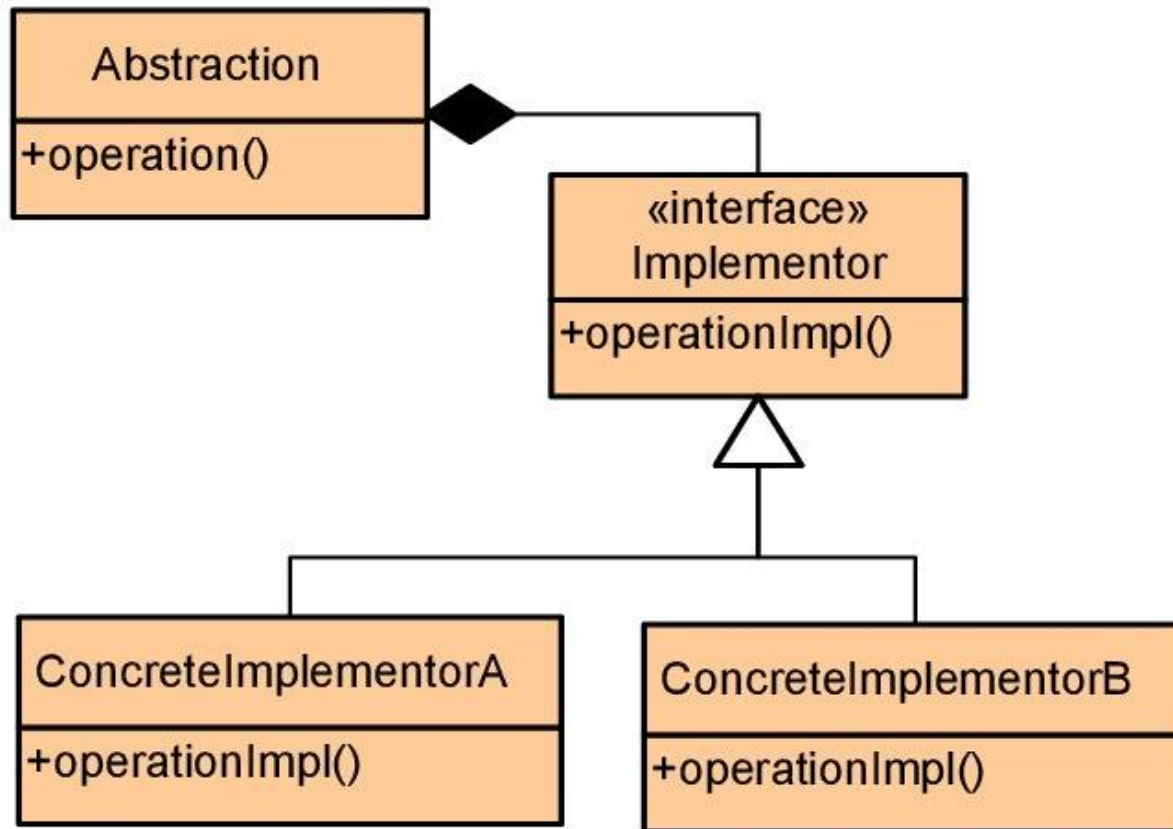
```
            throw;
```

```
        }
```

```
    }
```

```
}
```

Bridge



Мост *Bridge*

Тип: Структурный

Что это:

Разделяет абстракцию и реализацию так, чтобы они могли изменяться независимо.

```

0 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
public sealed class Program
{
    0 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public static void Main(string[] args)
    {
        IMessageQueue queue = new RabbitQueue();
        MessagePublisher publisher = new SyncMessagePublisher(queue);
        publisher.Publish("Preved!");
    }
}

```

```

namespace Bridge.Implementation
{
    7 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public interface IMessageQueue
    {
        4 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
        void Enqueue(byte[] messageContent);
    }
}

```

```

namespace Bridge.Abstraction
{
    4 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public abstract class MessagePublisher
    {
        protected readonly IMessageQueue messageQueue;

        2 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
        protected MessagePublisher(IMessageQueue messageQueue)
        {
            this.messageQueue = messageQueue;
        }

        3 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
        public abstract void Publish(object message);

        2 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
        protected byte[] SerializeMessage(object message) ...
    }
}

```

```

namespace Bridge.Abstraction
{
    1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public sealed class AsyncMessagePublisher : MessagePublisher
    {
        0 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
        public AsyncMessagePublisher(IMessageQueue messageQueue) : base(messageQueue)
        {
        }

        3 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
        public override void Publish(object message)
        {
            Task.Factory.StartNew(() =>
            {
                var messageContent = SerializeMessage(message);
                messageQueue.Enqueue(messageContent);
            });
        }
    }
}

```

```

namespace Bridge.Abstraction
{
    2 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public sealed class SyncMessagePublisher : MessagePublisher
    {
        1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
        public SyncMessagePublisher(IMessageQueue messageQueue) : base(messageQueue)
        {
        }

        3 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
        public override void Publish(object message)
        {
            var messageContent = SerializeMessage(message);

            messageQueue.Enqueue(messageContent);
        }
    }
}

```

```

namespace Bridge.Implementation
{
    0 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public sealed class MsmqQueue : IMessageQueue
    {
        4 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
        public void Enqueue(byte[] messageContent) ...
    }
}

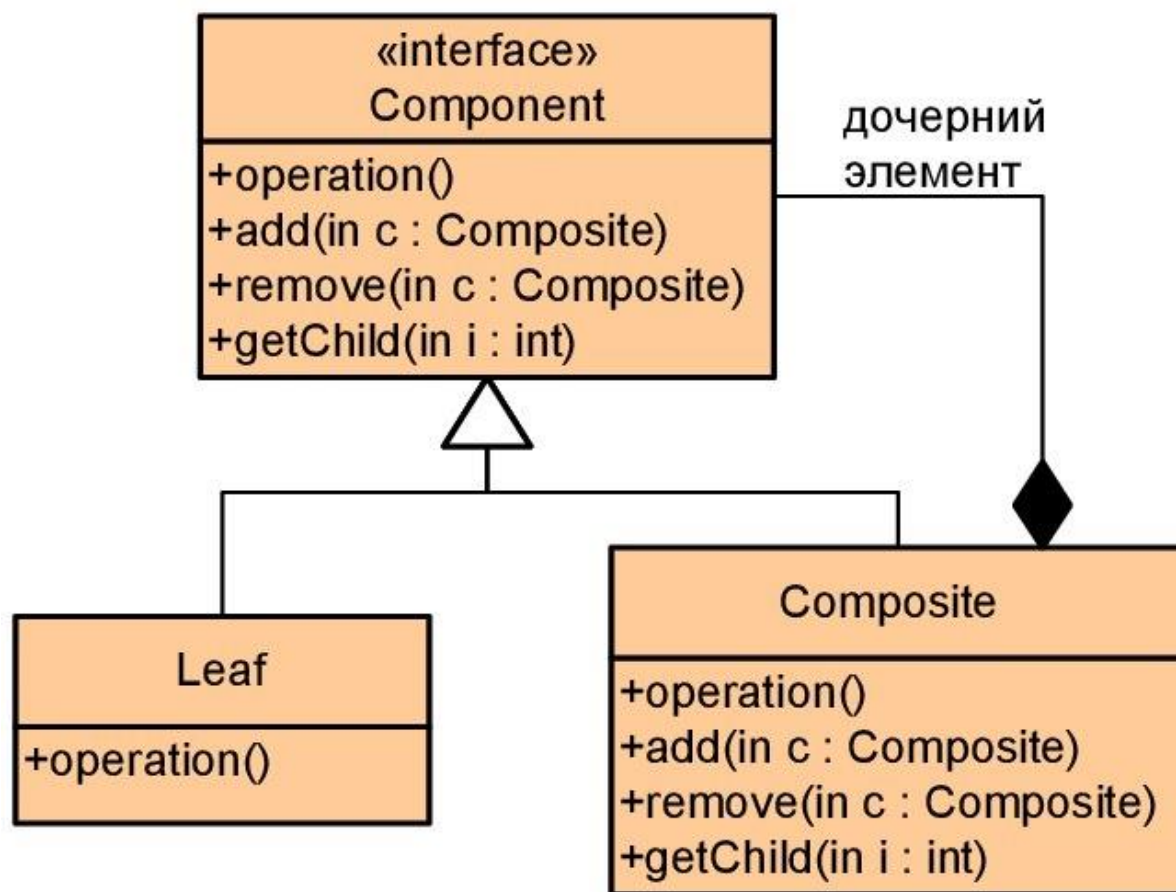
```

```

namespace Bridge.Implementation
{
    1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public sealed class RabbitQueue : IMessageQueue
    {
        4 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
        public void Enqueue(byte[] messageContent) ...
    }
}

```

Composite



КОМПОЗОВЩИК *Composite*

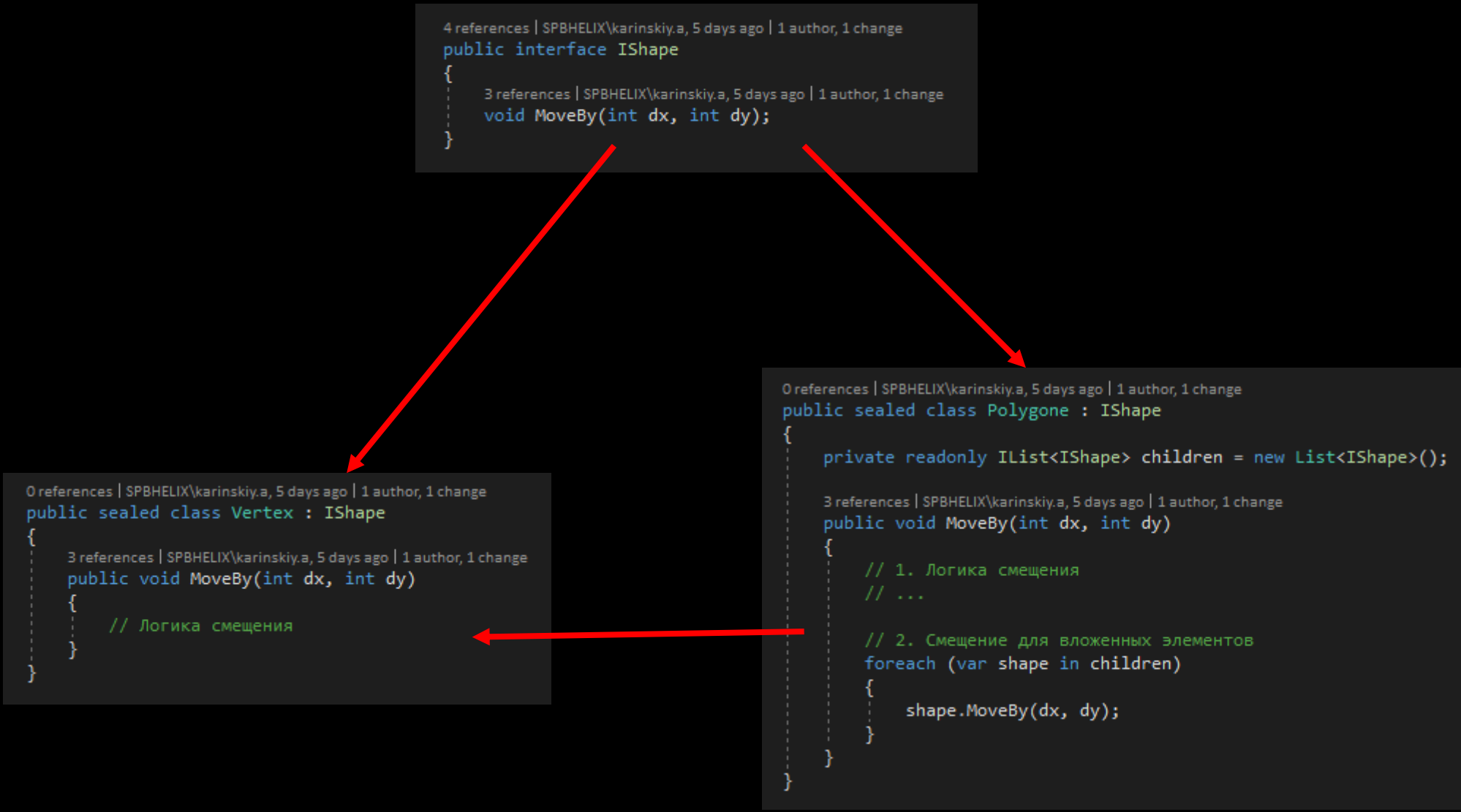
Тип: Структурный

Что это:

Компонуется объекты в древовидную структуру, представляя их в виде иерархии. Позволяет клиенту одинаково обращаться как к отдельному объекту, так и к целому поддереву.

4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public interface IShape
{
    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    void MoveBy(int dx, int dy);
}
```



0 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public sealed class Vertex : IShape
{
    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public void MoveBy(int dx, int dy)
    {
        // Логика смещения
    }
}
```

0 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change

```
public sealed class Polygone : IShape
{
    private readonly IList<IShape> children = new List<IShape>();

    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public void MoveBy(int dx, int dy)
    {
        // 1. Логика смещения
        // ...

        // 2. Смещение для вложенных элементов
        foreach (var shape in children)
        {
            shape.MoveBy(dx, dy);
        }
    }
}
```

Builder

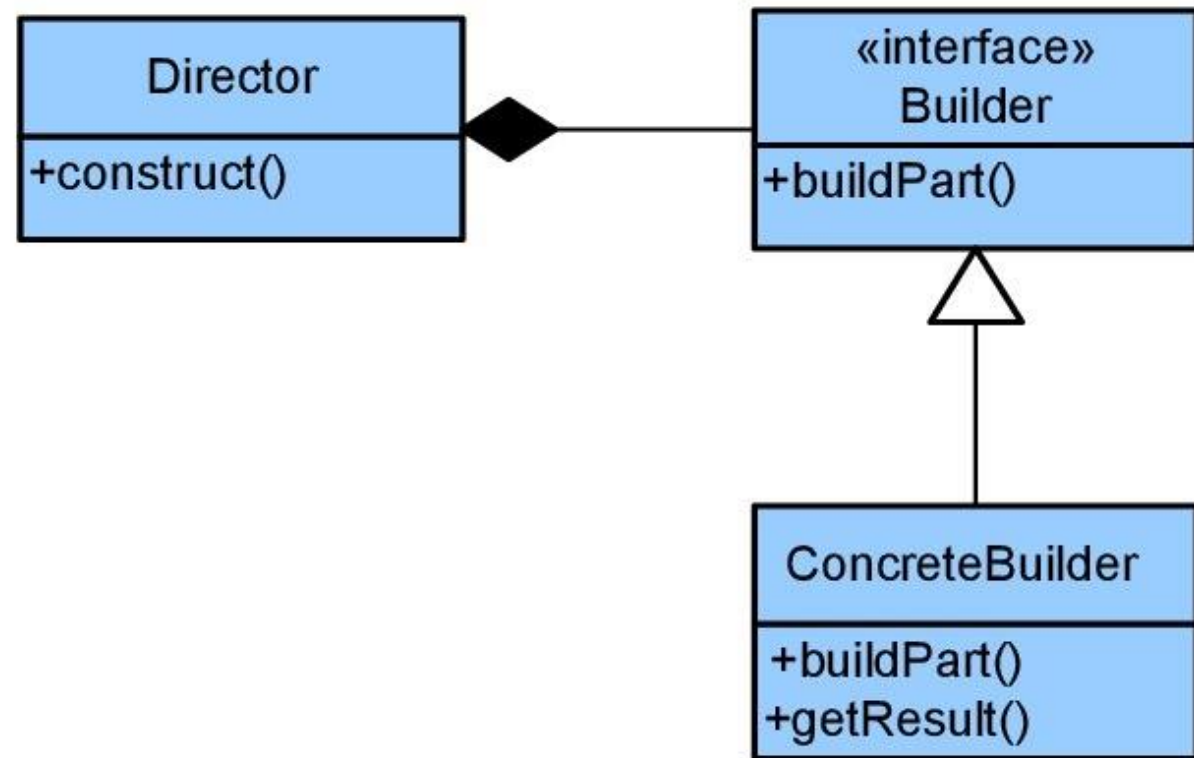
Строитель

Builder

Тип: Порождающий

Что это:

Разделяет создание сложного объекта и инициализацию его состояния так, что одинаковый процесс построения может создать объекты с разным состоянием.



4 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change

```
public sealed class Address
{
    2 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public Country Country { get; set; }

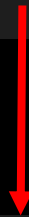
    1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public City City { get; set; }

    2 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public Street Street { get; set; }

    1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public string Building { get; set; }

    1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public int Apartment { get; set; }

    1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    public bool IsActual { get; set; }
}
```



```
var builder = new AddressBuilder();

var address = builder
    .Country("Russia")
    .City("Spb")
    .Street("Medicov")
    .LivesAt(building: 10, apart: 365)
    .Actual()
    .Build();
```

```
1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
public interface IBuilder<T>
{
    2 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change
    T Build();
}
```

namespace Builder

{

6 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change

public sealed class AddressBuilder : IBuilder<Address>

{

private readonly Address address = new Address();

#region Builder

2 references | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change

public Address Build()

{

return address;

}

1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change

public AddressBuilder Country(string countryAlias)

{

address.Country = FindCountryByAlias(countryAlias);

return this;

}

1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change

public AddressBuilder City(string cityAlias)

{

address.Street = FindCityByAlias(address.Country, cityAlias);

return this;

}

1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change

public AddressBuilder Street(string streetAlias)

{

address.Street = FindStreetByAlias(address.City, streetAlias);

return this;

}

1 reference | SPBHELIX\karinskiy.a, 3 days ago | 1 author, 1 change

public AddressBuilder LivesAt(int building, int apart)

{

ValidateBuildingAndApartment(building, apart);

address.Building = building.ToString();

address.Apartment = apart;

return this;

}

Prototype

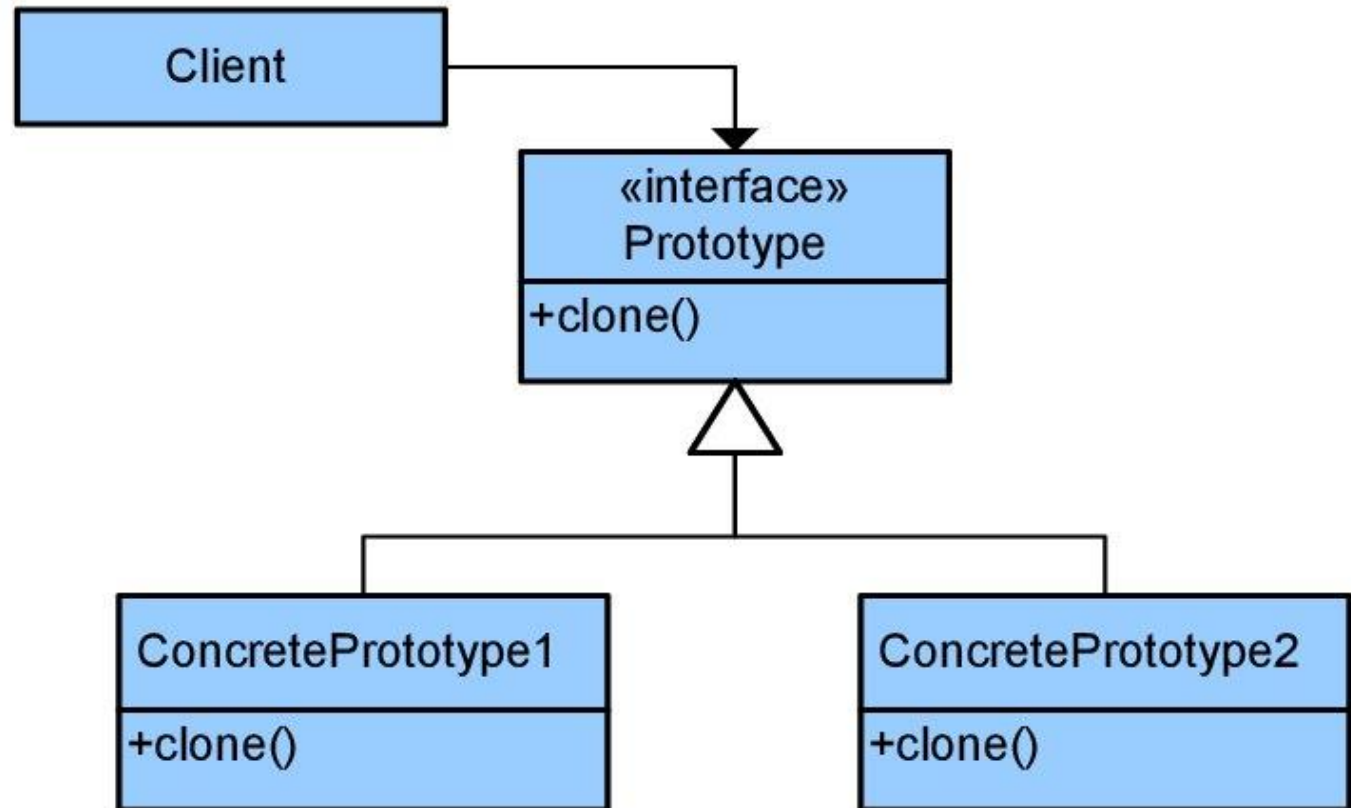
Прототип

Prototype

Тип: Порождающий

Что это:

Определяет несколько видов объектов, чтобы при создании использовать объект-прототип и создаёт новые объекты, копируя прототип.



```
var person1 = new Employee
{
    FirstName = "Василий",
    LastName = "Пупкин",
    Organization = "Рога и Копыта",
    WorkPhone = "322223223322",
    WorkAddress = "Красная Площадь"
};

var person2 = (Employee)person1.Clone();
person2.FirstName = "Абрам";
person2.LastName = "Шниперсон";
```

```
3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
public sealed class Employee : ICloneable
{
    4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public string FirstName { get; set; }

    4 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public string LastName { get; set; }

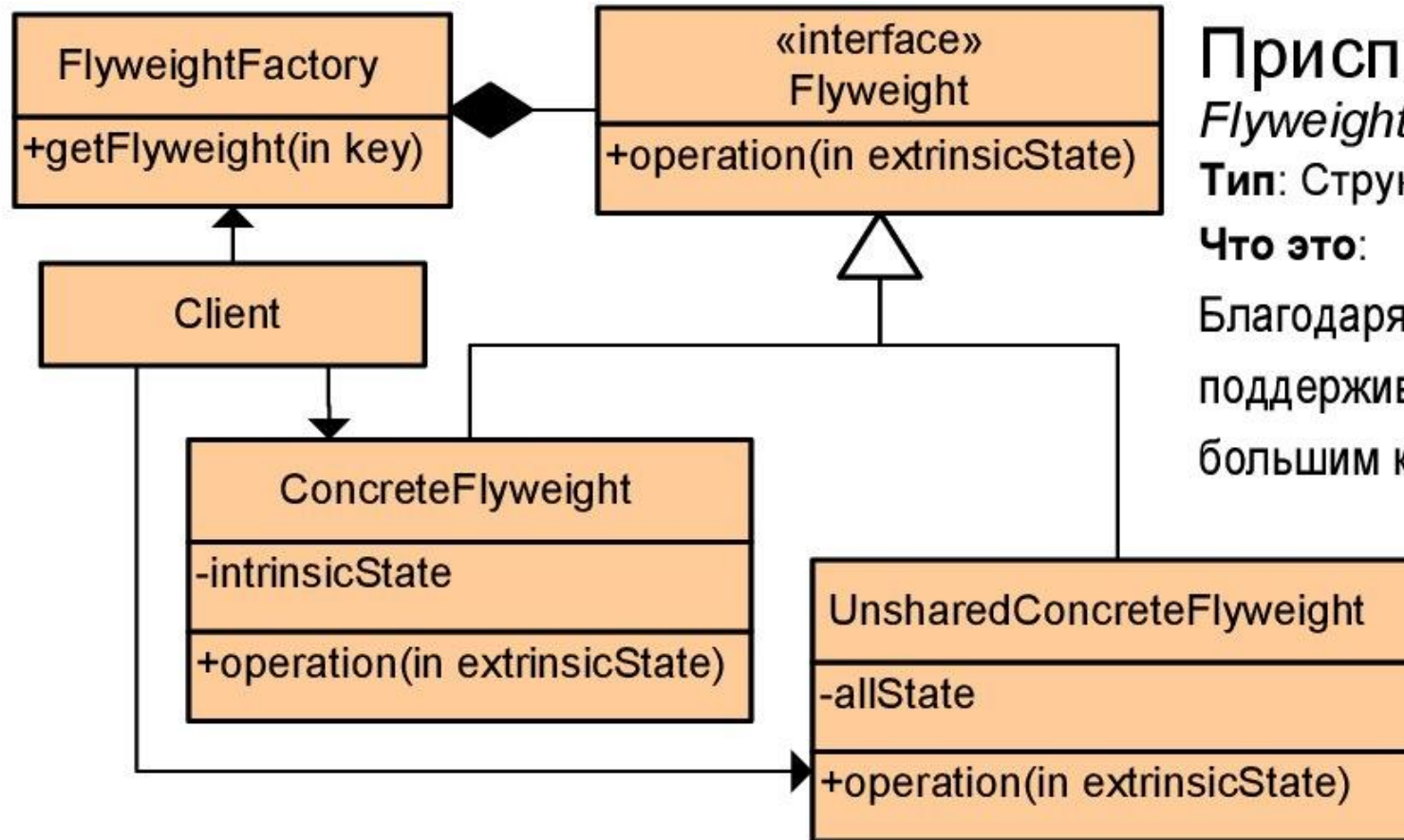
    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public string Organization { set; get; }

    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public string WorkPhone { get; set; }

    3 references | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public string WorkAddress { get; set; }

    1 reference | SPBHELIX\karinskiy.a, 5 days ago | 1 author, 1 change
    public object Clone()
    {
        return new Employee
        {
            FirstName = FirstName,
            LastName = LastName,
            Organization = Organization,
            WorkAddress = WorkAddress,
            WorkPhone = WorkPhone
        };
    }
}
```

Flyweight



Приспособленец

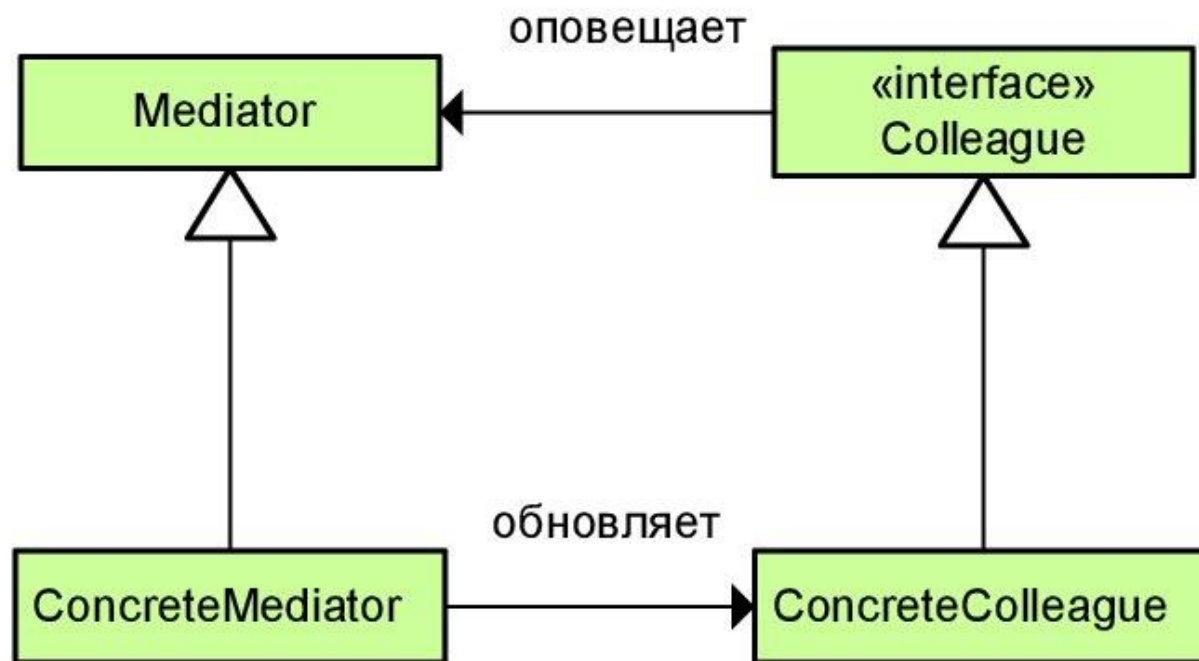
Flyweight

Тип: Структурный

Что это:

Благодаря совместному использованию, поддерживает эффективную работу с большим количеством объектов.

Mediator



Посредник *Mediator*

Тип: Поведенческий

Что это:

Определяет объект, инкапсулирующий способ взаимодействия объектов.

Обеспечивает слабую связь, избавляя объекты от необходимости прямо ссылаться друг на друга и даёт возможность независимо изменять их взаимодействие.

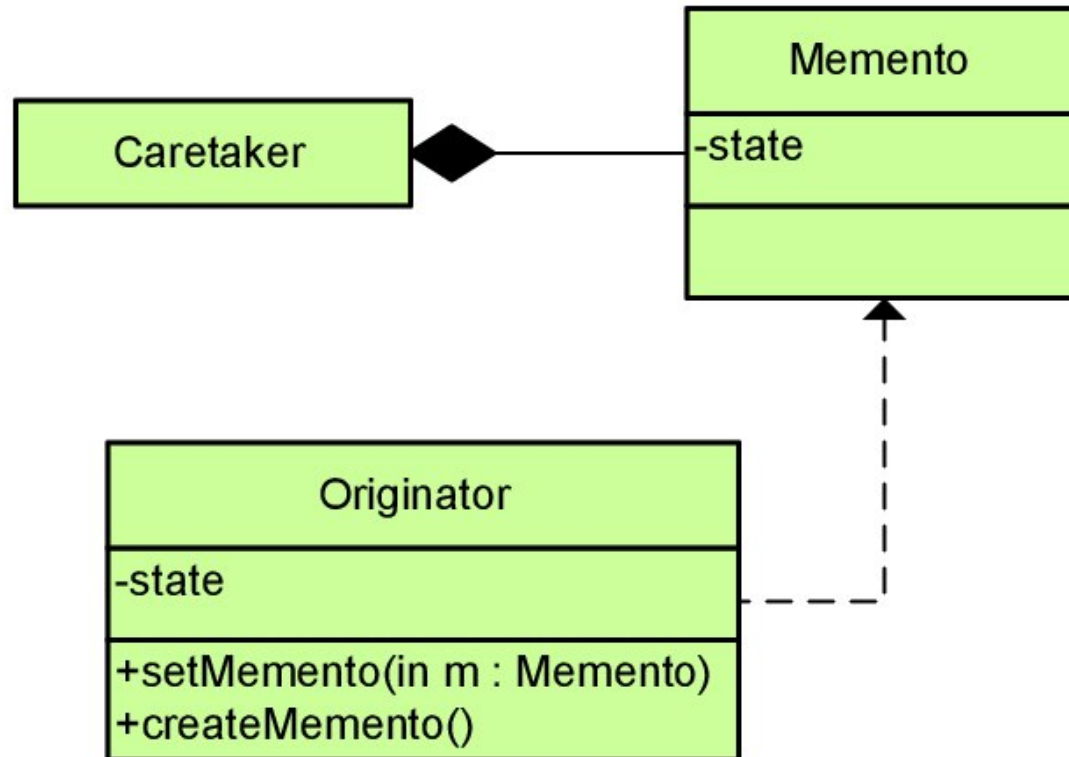
Memento

Хранитель *Memento*

Тип: Поведенческий

Что это:

Не нарушая инкапсуляцию,
определяет и сохраняет внутреннее
состояние объекта и позволяет позже
восстановить объект в этом состоя-
нии.



???