



ОНЛАЙН-ОБРАЗОВАНИЕ

Онлайн-образование



Меня хорошо видно && слышно?

Ставьте  , если все хорошо
Напишите в чат, если есть проблемы

Три кита ООП: Абстракция, Наследование и Полиморфизм



Ягур Алексей
Руководитель курса

The background of the image is an aerial view of a city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer that features a network of white lines and dots, resembling a digital or data network. The word "Тестирование" is written in large, white, bold Cyrillic letters across the center of the image.

Тестирование

Не забыть включить запись!



Три кита ООП: Абстракция, Наследование и Полиморфизм



Ягур Алексей
Руководитель курса

Цели занятия

1. Познакомиться с ООП

2. Понять Абстракцию

3. Изучить Наследование и
Полиморфизм

Маршрут занятия

Абстракция

Наследование

Полиморфизм

The image features a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City, with numerous skyscrapers and buildings. A semi-transparent blue band with a white geometric network pattern of dots and lines runs horizontally across the center of the image. Overlaid on this band is the text "ООП" in a large, white, sans-serif font.

ООП

Что такое Объектно-Ориентированное Программирование?

- парадигма программирования, основанная на концепции “объектов”, которые могут содержать данные и код для их обработки.

Основные концепции

- Классы
- Экземпляры
- Методы
- Свойства

The background of the image is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer. A network of thin, light blue lines connects various points across the blue layer, creating a geometric pattern. The word "Абстракция" is written in large, white, sans-serif font across the center of the blue layer.

Абстракция

Абстракция

Абстракция означает скрытие сложных деталей реализации и показ только существенных функций объекта.

Abstract class — класс, который обычно содержит реализацию **некоторых** методов/свойств, а реализацию других методов оставляет за своими «наследниками».

Экземпляры такого класса создавать нельзя.

Interface — контракт (обязательство), который обязуется выполнить класс. Интерфейс может включать в себя, т.е. наследовать другие интерфейсы.

```
abstract class StringHolder
{
    private readonly string _content;

    abstract public string Read();
    abstract public void Add(string content);

    public int Length => _content.Length;
    protected bool CanAdd => Length < 1000;
}

class FileStringHolder : StringHolder
{
    public override string Read()
    {
        // ...
    }
    public override void Add(string content)
    {
        if (!CanAdd)
        {
            throw new Exception("Too much content");
        }
        // ...
    }
    public string Delete()
    {
        // ...
    }
}
```


Используемые модификаторы

В C#

- `abstract` – абстрактный класс
- `virtual` – член класса разрешает, чтобы его переопределяли наследники
- `override` – член класса переопределяет базовое определение
- `new` – член класса скрывает базовое определение

abstract, virtual

```
// Абстрактный класс
public abstract class LivingThing
{
    // Абстрактный
    // Производный класс должен реализовать его
    public abstract void Move();

    // Тут уже какая-то реализация есть
    public virtual void Eat(Thing t)
    {
        Console.WriteLine("I ate");
    }
}
```

```
public class Human : LivingThing
{
    // ОБЯЗАТЕЛЬНО
    // переопределяем метод
    public override void Move()
    {
        Console.WriteLine("One leg");
        Console.WriteLine("Two leg");
    }
}
```


abstract, virtual

```
// Абстрактный класс
public abstract class LivingThing
{
    // Абстрактный
    // Производный класс должен реализовать его
    public abstract void Move();

    // Тут уже какая-то реализация есть
    public virtual void Eat(Thing t)
    {
        Console.WriteLine("I ate");
    }
}
```

```
public class Animal : LivingThing
{
    // ОБЯЗАТЕЛЬНО
    // переопределяем метод
    public override void Move()
    {
        Console.WriteLine("One leg");
        Console.WriteLine("Two leg");
        Console.WriteLine("Three leg");
        Console.WriteLine("Four leg");
    }

    // НЕОБЯЗАТЕЛЬНО
    // переопределяем метод
    public override void Eat(Thing t)
    {
        Console.WriteLine("I bite");
        base.Eat(t);
    }
}
```


The background of the image is a high-angle, aerial photograph of a dense urban skyline, likely New York City, featuring numerous skyscrapers and buildings. The entire image is overlaid with a semi-transparent blue layer. A network of thin, light-blue lines connects various points across the blue area, creating a digital or technological aesthetic. The word "Наследование" is centered in the middle of the image, rendered in a large, bold, white sans-serif font.

Наследование

Наследование

```
// Базовый класс
public class Vehicle
{
    public long Speed { get; set; }
}
```

```
// Car наследует свойства Vehicle
public class Car : Vehicle
{
    public string Name { get; set; }
}
```

```
var v = new Vehicle();
v.Speed = 10;
```

```
var car = new Car();
// у car есть СВОЙСТВО Speed
// от Vehicle
car.Speed = 100;
// И собственное Name
car.Name = "Nissan";
```


Глубина наследования

Количество уровней наследования не ограничено, но не рекомендуется делать иерархию чрезмерно глубокой

```
class Stuff { }  
class Equipment : Stuff { }  
class Computer : Equipment { }  
class Laptop : Computer { }  
class Macbook : Laptop { }
```

Неявно в корне любой иерархии класс Object

```
class Staff : Object { }
```


Нет множественного наследования класса от классов

```
public class Robot  
{  
}
```

// Можно

```
public class Transformer : Car  
{  
}
```

// Нельзя

```
public class Transformer : Car, Robot  
{  
  
}
```


The background of the entire image is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. A semi-transparent blue overlay covers the entire image. In the center, there is a horizontal band with a gradient from teal on the left to dark blue on the right. Overlaid on this band is a white network pattern of dots and lines. The title text is centered within this band.

Модификаторы доступа

Модификаторы

Модификаторы доступа помогают защититься от ошибок разработчиков (т.н. защитное программирование)

private — доступ только для членов этого же класса

protected — доступ только для членов этого же класса + для наследников этого класса (как прямых, так и опосредованных)

internal — доступ только для членов этого проекта

public — доступ для всех

Некоторые другие модификаторы:

const — значение должно быть рассчитано на этапе компиляции

readonly — значение может быть записано только один раз (при инициализации класса)

static — не требует создания экземпляра класса для обращения к этому члену

Модификаторы доступа: private

```
class Parent
{
    private int a;

    public void PrintParent()
    {
        Console.WriteLine(a);
    }
}
```

```
class Child : Parent
{
    public void PrintChild()
    {
        Console.WriteLine("Нельзя " + a);
    }
}
```

```
class Alien
{
    public void PrintAlien()
    {
        Parent parent = new Parent();
        Console.WriteLine("Нельзя " + parent.a);
    }
}
```


Модификаторы доступа: protected

```
class Parent
{
    protected int a;

    public void PrintParent()
    {
        Console.WriteLine(a);
    }
}

class Child : Parent
{
    public void PrintChild()
    {
        Console.WriteLine("Можно " + a);
    }
}
```

```
class Alien
{
    public void PrintAlien()
    {
        Parent parent = new Parent();
        Console.WriteLine("Нельзя " + parent.a);
    }
}
```

Модификаторы доступа: public

```
class Parent
{
    public int a;

    public void PrintParent()
    {
        Console.WriteLine(a);
    }
}
```

```
class Child : Parent
{
    public void PrintChild()
    {
        Console.WriteLine("Можно " + a);
    }
}
```

```
class Alien
{
    public void PrintAlien()
    {
        Parent parent = new Parent();
        Console.WriteLine("Можно " + parent.a);
    }
}
```


*Инкапсуляция

Инкапсуляция — это процесс отделения друг от друга *элементов объекта*, *определяющих его устройство и поведение*; инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации

Г. Буч

Модификаторы доступа
«private»

```
enum ServerState
{
    Starting,
    Started,
    Stopping,
    Stopped
}
```

```
class Server
{
    private ServerState state = ServerState.Stopped;

    public void Run() { DoRun(); }
    public void Stop() { DoStop(); }

    private void DoRun()
    {
        state = ServerState.Starting;

        // выполнить логику по запуску сервера
        // ...

        state = ServerState.Started;
    }

    private void DoStop()
    {
        state = ServerState.Stopping;

        // выполнить логику по остановке сервера
        // ...

        state = ServerState.Stopped;
    }
}
```

sealed

```
public class FaunaThing : LivingThing
{
    public override void Move()
    {
    }

    // sealed не позволяет переопределять дальше
    public override sealed void Eat(Thing t)
    {
        Console.WriteLine("absorb air");
    }
}
```

```
public class Tree : FaunaThing
{
    // Ошибка компиляции
    public override void Eat(Thing t)
    {
    }
}
```




Полиморфизм

Полиморфизм

Полиморфизм (*Polymorphe* греч.) Дословно - множественность форм. *Способность объекта или оператора ссылаться на объекты разных классов на стадии выполнения.* И. Грэхем

- это способность объектов принимать множество форм, что позволяет одному интерфейсу представлять различные подлежащие типы данных.

Выделяют два основных типа полиморфизма:

- Времени компиляции (перегрузка методов)
- Времени выполнения (переопределение методов)



LIVE



Цели занятия


1. Познакомиться с ООП

2. Понять Абстракцию

3. Изучить Наследование и
Полиморфизм

The background of the image is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this is a semi-transparent blue band that contains a white network diagram of interconnected nodes and lines. Centered within this band is the word "Тестирование" in a large, bold, white sans-serif font.

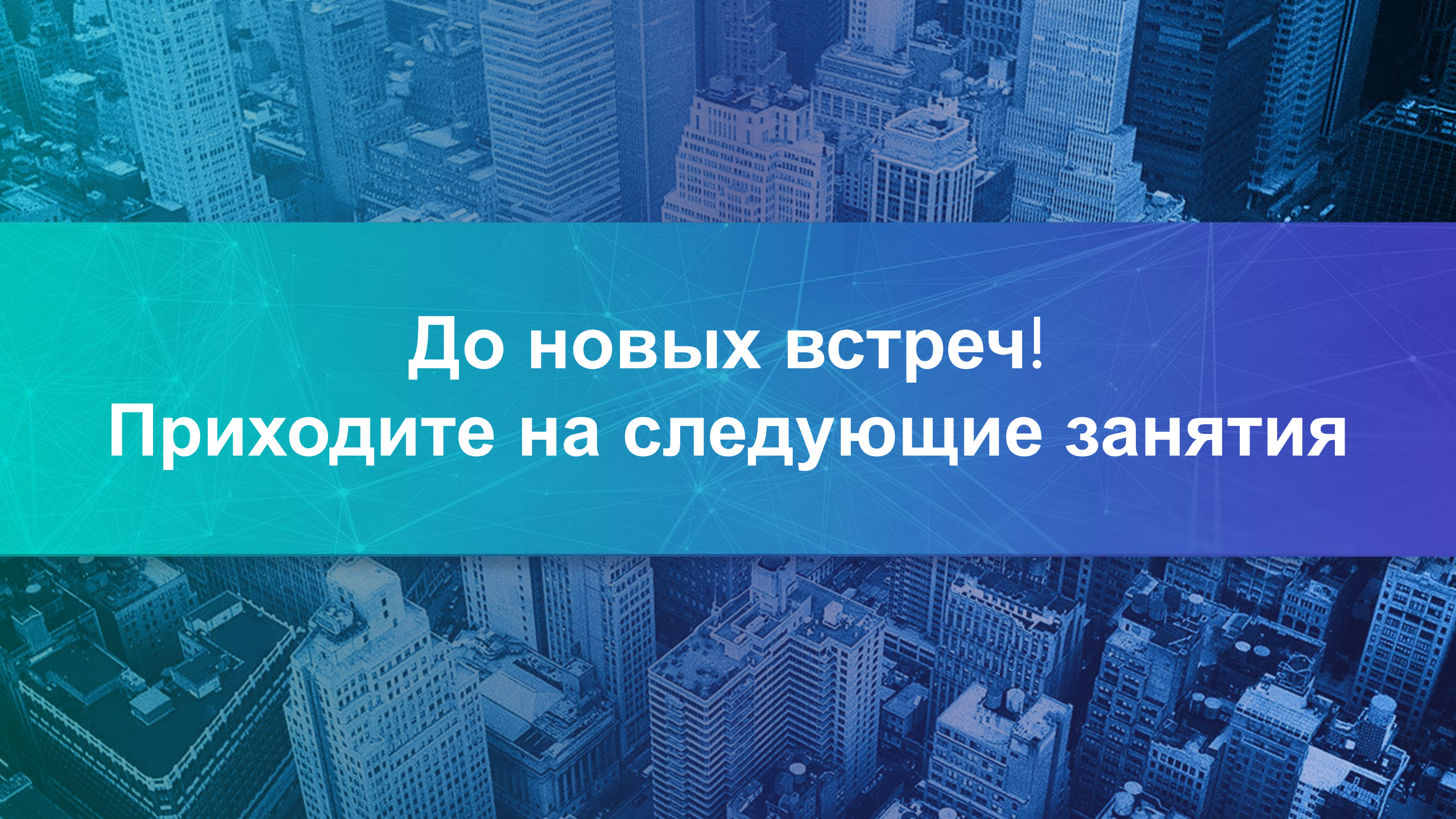
Тестирование

The background of the image is an aerial view of a dense city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer that features a white network pattern of interconnected dots and lines. The text is centered within this blue layer.

Заполните, пожалуйста,
опрос о занятии по ссылке в чате

The background of the entire image is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. A semi-transparent blue overlay covers the entire image. In the center, there is a horizontal band with a gradient from teal on the left to dark blue on the right. Overlaid on this band is a white network pattern of dots and lines. The text "ОТВЕТЫ НА ВОПРОСЫ" is centered in this band in a large, white, bold, sans-serif font.

ОТВЕТЫ НА ВОПРОСЫ



До новых встреч!
Приходите на следующие занятия