ООП = Объектно-ориентированное программирование

ООП – это парадигма программирования

Парадигма программирования определяет:

- Какие «строительные блоки» мы используем при написании программ
- Как мы структурируем наши программы
- Как мы связываем наши компоненты, как объединяем их в единую систему

Какие бывают парадигмы?

- Императивное программирование
  - Структурное программирование
  - Объектно-ориентированное программирование
- Декларативное программирование
  - Логическое программирование
  - Функциональное программирование

А что с языками?

Обычно языки программирования поддерживают одну или несколько парадигм, но не все сразу

Парадигма программирования	Языки
Императивная	Pascal, Basic, C, C++, C#, Java, Kotlin, Python, Scala…
Структурная	Pascal, Basic, C, C++, C#, Java, Kotlin, Python, Scala…
Объектно-ориентированная	C++, C#, Java, Kotlin, Python, Scala…
Декларативная	Prolog, Planner, Ether, F#, Erlang, R, Wolfram, Kotlin, Python, Scala, Lisp…
Логическая	Prolog, Planner, Ether…
Функциональная	F#, Erlang, R, Wolfram, Kotlin, Python, Scala, Lisp…

История ООП

Очень простые, очень маленькие программы, сотни строк кода

+

Один программист

↓

Императивное программирование

Программы посложнее, тысячи строк кода

+

Несколько программистов

↓

Структурное программирование

Сложные программы с богатым, динамичным функционалом, десятки и сотни тысяч строк кода

+

Десятки, сотни программистов

↓

Объектно-ориентированное программирование

История ООП

Развитие структурного программирования

1967 г. – язык Simula, первый прообраз

1972 г. – язык Smalltalk

1983 г. – языки Objective-C, C++

Так в чём же особенность ООП?

Центральное понятие в ООП – это ***КЛАСС***

Класс – это объединение данных и операции с этими данными

Класс – это описание, схема, в соответствии с которой в программе создаются, работают и уничтожаются экземпляры этого класса – ***ОБЪЕКТЫ***

Так в чём же особенность ООП?

В объектно-ориентированной программе объекты посылают сообщения друг другу, обрабатывают полученные данные. Ответственность за разные операции над разными данными распределяется между объектами разных классов, которые общаются друг с другом

А в чём преимущества?

★ Легче расширять функционал
★ Больше кода используется повторно
★ Изменения функциональности затрагивают только классы, ответственные за этот функционал

Есть и недостатки

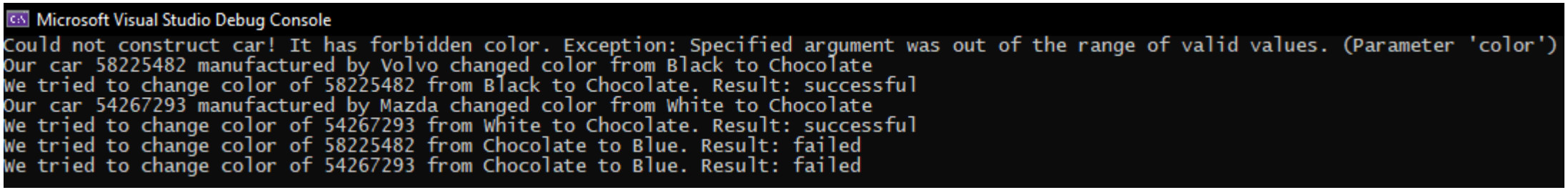
➔Больше времени и сил на проектирование
➔Суммарно пишется больше кода

Классы в C#

```
using System;
using System.Drawing;

namespace ITMO_OOP
{
    13 references
    public class Car
    {
        private Color _color;

        1 reference
        public Car(string manufacturer) : this(Color.White, manufacturer) { }
        3 references
        public Car(Color color, string manufacturer)
        {
            if (!IsColorAllowed(color)) throw new ArgumentOutOfRangeException(nameof(color));
            (_color, Manufacturer) = (color, manufacturer);
        }

        2 references
        public string Manufacturer { get; }

        2 references
        public Color Color
        {
            get => _color;
            private set
            {
                if (!IsColorAllowed(value)) return;

                var oldColor = _color;
                _color = value;
                ColorChanged?.Invoke( car: this, oldColor,  newColor: value);
            }

        }
        1 reference
        public virtual bool ChangeColor(Color newColor)
        {
            var allowed :bool  = IsColorAllowed(newColor);
            if (allowed) Color = newColor;
            return allowed;
        }

        3 references
        private static bool IsColorAllowed(Color color) => color != Color.Blue;

        public delegate void ColorChangedHandler(Car car, Color oldColor, Color newColor);

        public event ColorChangedHandler ColorChanged;
    }
}
```

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

namespace ITMO_OOP
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Car blackVolvo = new(Color.Black, manufacturer: "Volvo");
            Car defaultMazda = new(manufacturer: "Mazda");
            Car blueFiat = null;

            try
            {
                blueFiat = new Car(Color.Blue, manufacturer: "Fiat");
            }
            catch (ArgumentOutOfRangeException e)
            {
                Console.WriteLine($"Could not construct car! It has forbidden color. Exception: {e.Message}");
            }

            blackVolvo.ColorChanged += CarOnColorChanged;
            defaultMazda.ColorChanged += CarOnColorChanged;

            List<Car> cars = new() { blackVolvo, defaultMazda};

            ChangeCarsColor(cars, Color.Chocolate);
            ChangeCarsColor(cars, Color.Blue);
        }

        2 references
        private static void ChangeCarsColor(IEnumerable<Car> cars, Color newColor)
        {
            foreach (var car in cars)
            {
                var oldColor = car.Color;
                var result :bool = car.ChangeColor(newColor);
                Console.WriteLine(
                    $"We tried to change color of {car.GetHashCode()} from {oldColor.Name} to {newColor.Name}. Result: {(result ? "successful" : "failed")}");
            }
        }

        2 references
        private static void CarOnColorChanged(Car car, Color oldColor, Color newColor)
        {
            Console.WriteLine($"Our car {car.GetHashCode()} manufactured by {car.Manufacturer} changed color from {oldColor.Name} to {newColor.Name}");
        }
    }
}
```

Классы в C#


Microsoft Visual Studio Debug Console
Could not construct car! It has forbidden color. Exception: Specified argument was out of the range of valid values. (Parameter 'color')
Our car 58225482 manufactured by Volvo changed color from Black to Chocolate
We tried to change color of 58225482 from Black to Chocolate. Result: successful
Our car 54267293 manufactured by Mazda changed color from White to Chocolate
We tried to change color of 54267293 from White to Chocolate. Result: successful
We tried to change color of 58225482 from Chocolate to Blue. Result: failed
We tried to change color of 54267293 from Chocolate to Blue. Result: failed

Ещё раз

КЛАСС – шаблон для создания **ОБЪЕКТОВ**, описывающий, как объект устроен, из чего состоит, что и как он может делать

ОБЪЕКТ – экземпляр **КЛАССА**. Построенный по указанному в **КЛАССЕ** шаблону, способный выполнять конкретную работу

В общем случае в программе одномоментно может существовать много **ОБЪЕКТОВ** одного **КЛАССА**

Спасибо за внимание!