14 Принципы объектно-ориентированного программирования. Классы. Доступ к членам класса. Конструктор класса. Абстрактные классы. Интерфейсы

принципы ООП

- абстракция - моделирование атрибутов и поведения сущностей для определения абстрактного представления системы
- инкапсуляция - скрытие внутреннего устройства, предоставление доступа только через открытый набор функций
- наследование - возможность создания новых абстракций на основе уже существующих
- полиморфизм - возможность реализации одинаковых методов разными способами в рамках множества абстракций

классы

объект - предмет или явление, имеющее четко определенные характеристики, поведение.

класс - множество объектов, связанных общностью структуры и поведения.

у классов есть атрибуты (переменные, характеристики) и методы (функции).

класс - абстракция, которая описывает структуру объектов.

доступ к членам класса

- public - общедоступные
- protected - доступные классу и дочерним классам
- private - доступные только внутри класса

в зависимости от языка программирования могут быть и другие уровни доступа (например, readonly).

конструктор класса

метод класса, позволяющий создать экземпляр класса с определенными параметрами.

абстрактный класс

класс, который можно наследовать, но нельзя реализовать.

абстрактными называются классы, содержащие один и более абстрактных методов. абстрактный метод - объявленный, но не реализованный метод.

интерфейс

похож на абстрактный класс тем, что его нельзя реализовать. разница - один класс может наследовать несколько интерфейсов. методы интерфейса не нужно реализовывать с ключевым словом override. интерфейсы не могут определять не статичные переменные; его методы и свойства не имеют модификатор доступа (они public по умолчанию).

пример в C#:

C#

```
interface IFigureInfo
{
    double area();
    double perimeter();
}

abstract class RegularPoligon
{
    public double length{get;set;}
    public abstract double area();
    public abstract double perimeter();
    public RegularPoligon(double len)
    {
        this.length = len;
    }
}

class Square.RegularPoligon
{
    public override double area()
    {
        return length*length;
    }
}

class Circle:IFigureInfo
{ ... }
// ^ обязательно определить методы area() и
perimeter()
```