

# Интерфейсы

**Интерфейс** —это ссылочный тип, который может определять некоторый функционал - набор методов и свойств без реализации.

Этот функционал реализуют классы и структуры, которые применяют данные интерфейсы.

**Интерфейсы определяют:**

- Методы
- Свойства
- Индексаторы
- События
- Статические поля и константы

Члены интерфейса (методы и свойства) не имеют модификаторов доступа, по умолчанию доступ **public**

# Объявление интерфейсов

```
interface IPurse
{
    int Sum
    {
        get;
        set;
    }
    void AddMoney(int sum);
    int DecMoney(int sum);
}
```

Объявление интерфейса начинается со слова interface;

Интерфейс не может иметь переменных;

Интерфейсы не наследуют никакого класса;

Для самого интерфейса можно указать модификатор доступа;

# Реализация интерфейсов

```
internal class Person : IPurse
{
    // реализация класса
}
```

```
internal class Person : Object, IPurse
{
    // реализация класса
}
```

Поддерживается множественное  
наследование интерфейсов

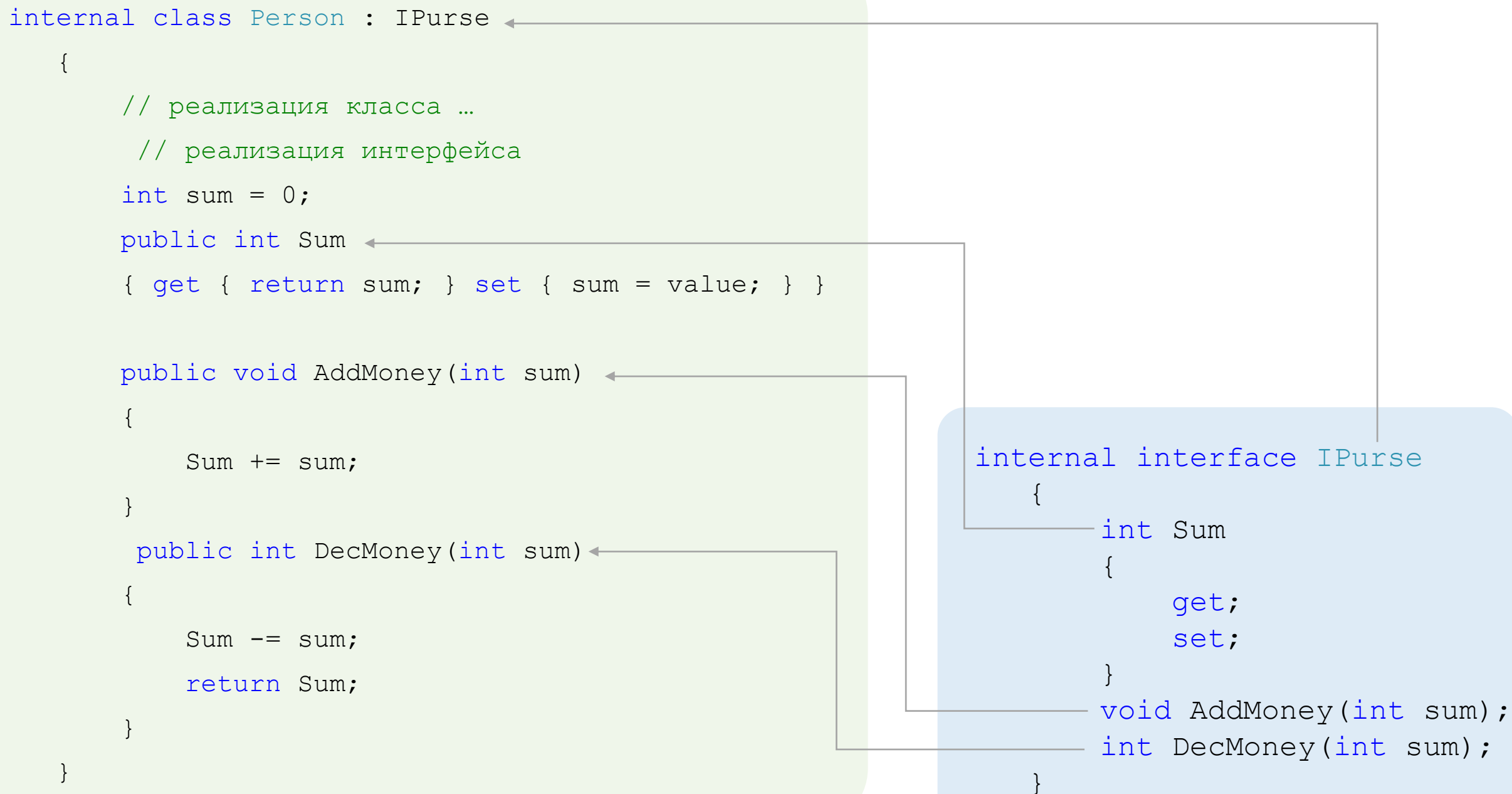
Невозможна частичная реализация  
интерфейса

```
internal class Person : IPurse
{
    // реализация класса ...
    // реализация интерфейса
    int sum = 0;
    public int Sum
    { get { return sum; } set { sum = value; } }

    public void AddMoney(int sum)
    {
        Sum += sum;
    }

    public int DecMoney(int sum)
    {
        Sum -= sum;
        return Sum;
    }
}
```

```
internal interface IPurse
{
    int Sum
    {
        get;
        set;
    }
    void AddMoney(int sum);
    int DecMoney(int sum);
}
```



# Использование реализации интерфейса

- **Простой вызов методов:**

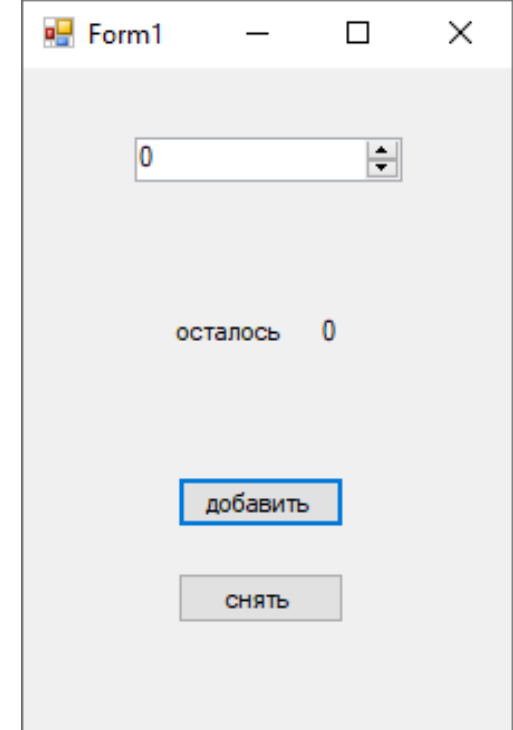
```
Person person = new Person("Иванов Иван");  
person.AddMoney(10000);
```

- **Унификация доступа к методам (листинг 1)**
- **Использование интерфейса в качестве параметра**

## Листинг 1. унификация доступа к методам

```
public partial class Form1 : Form
{
    //переменные класса
    Person person = new Person { Name = "Иванов Иван" };
    Object personObject;
    IPurse purse;

    public Form1 ()
    {
        InitializeComponent();
        //инициализация переменных
        personObject = person;
        purse = person;
    }
}
```



Интерфейсные переменные нельзя инициализировать интерфейсами, но объектами классов, которые реализуют интерфейс, можно!

## Продолжение листинга 1.

```
private void addButton_Click(object sender, EventArgs e)
{ //приведение к интерфейсу и вызов его метода
    if (personObject is IPurse)
    {
        ((IPurse)personObject).AddMoney((int)numericUpDown1.Value);
        sumLabel1.Text = ((IPurse)personObject).Sum.ToString();
    }
}

private void decButton_Click(object sender, EventArgs e)
{ //метод интерфейса вызывается напрямую через интерфейсную переменную
    purse.DecMoney((int)numericUpDown1.Value);
    sumLabel1.Text = purse.Sum.ToString();
}

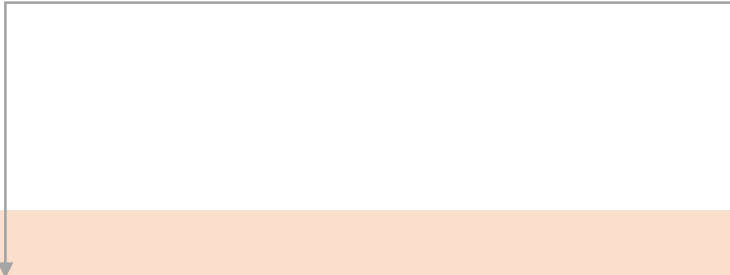
}
```



## Использование интерфейса в качестве параметра

-позволяет работать как с универсальным типом данных, который можно передавать как переменные в другие методы

В качестве параметра можно передать любой класс, реализующий интерфейс




```
void DecMoney(IPurse purse)
{
    purse.DecMoney((int)numericUpDown1.Value);
    sumLabel.Text = purse.Sum.ToString();
}
```

# Интерфейс как возвращаемый тип

```
internal interface IMailClient
{
    // содержимое интерфейса
}
```

```
IMailClient GetEmailClient()
{
    if (something)
    {
        return new GmailClient();
    }
    if (somethingelse)
    {
        return new YandexClient();
    }
}
```



Конкретные реализации интерфейса

# Перегрузка интерфейсных методов

```
class Person : IPurse, ITripplPurse, IMailClient
{
    // реализация класса
    public string Name { get; set; }

    // реализация интерфейса IPurse
    //...
    public void AddMoney(int sum)
    {
        Sum += sum;
    }

    //...
    // реализация интерфейса ITripplPurse

    void ITripplPurse.AddMoney(int sum)
    {
        Sum += sum*3;
    }
}
```

Вызов метода из нужного интерфейса

```
((ITripplPurse)person).AddMoney(10);
```

Указание на явную  
реализацию интерфейса

# Наследование

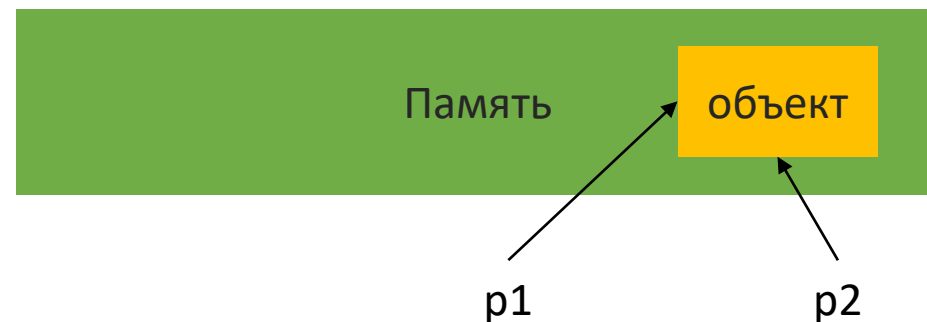
Если интерфейс наследует какой-либо другой интерфейс, то он наследует все его методы и свойства

```
interface ISafe : IPurse
{
    bool Locked
    {
        get;
    }
    void Lock();
    void Unlock();
}
```

# Клонирование объектов

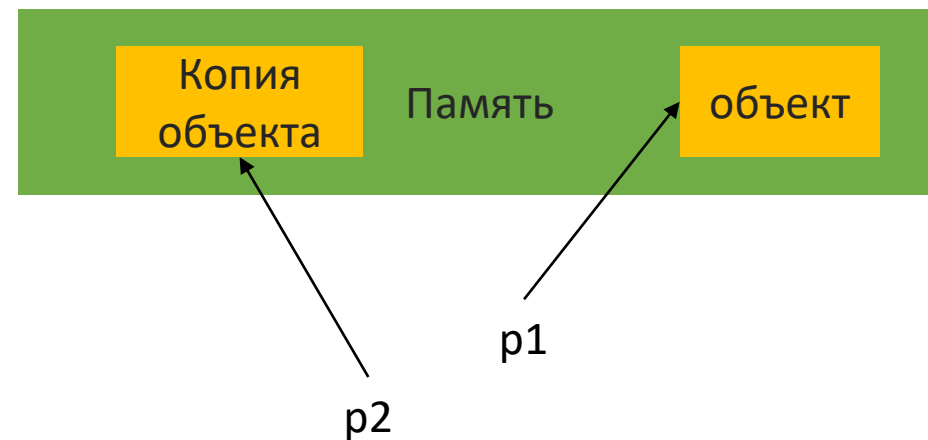
## Простое присваивание

```
Person p1 = new Person();  
Person p2 = p1;
```



## Создание копии объекта

```
Person p2 = new Person(p1.Name);
```



# Клонирование через интерфейс

```
internal class Unit : ICloneable
{
    public string Name { get; set; }
    public Unit(string name)
    {
        Name = name;
    }
    public Object Clone()
    {
        Unit u = new Unit(this.Name);
        return u;
    }
}
```

```
internal interface ICloneable
{
    Object Clone();
}
```

Метод Clone() возвращает копию текущего объекта через Object

```
Person p1 = new Person();
Person p2 = (Person)p1.Clone();
```