Інтерфейси в C#. Реалізація інтерфейсів. IEnumerable, IComparable

Клакович Л.М.

Інтерфейси

- Інтерфейси як засіб визначення типу
- FCL-інтерфейси переліку
- FCL-інтерфейси колекцій
- FCL-інтерфейси порівняння
- IComparable<Point>
- Неявна і явна реалізація інтерфейсів
- Варіанти похідного типу
- Використання масивів
- Методи і властивості класу System.Array

Визначення типів

Загальна схема визначення користувацьких типів: класів, структур та інтерфейсів

атрибути $_{opt}$ *модифікатори* $_{opt}$
class *ідентифікатор* : *ідентифікатор базового класу* $_{opt}$, *список інтерфейсів* $_{opt}$
{
визначення членів класу
}

атрибути $_{opt}$ *модифікатори* $_{opt}$
struct *ідентифікатор*: *список інтерфейсів* $_{opt}$
{
визначення членів типу
}

атрибути $_{opt}$ *модифікатори* $_{opt}$
interface *ідентифікатор*: *список інтерфейсів* $_{opt}$
{
оголошення членів інтерфейсу
}

Лише оголошення:
- **методів**
- **властивостей**
- **подій**
- **індексаторів**

Інтерфейси як засіб визначення типу

- **Основне призначення** - Об’єднання в одну іменовану функціональну групу методів, властивостей і подій
  - Гарантія реалізації класом оголошеної інтерфейсом поведінки
  - CLR утворює об'єкт-тип
- **Наслідування інтерфейсів** - обмежений варіант множинного наслідування в CLR
  - Використання об’єктів типів, які реалізують інтерфейс, у контексті інтерфейсного типу як базового
  - Реалізація поліморфної поведінки незалежними типами
  - Наслідування інтерфейсів інтерфейсом – структурування функціональності

- **Інтерфейс може містити лише оголошення методів, властивостей, індексаторів та подій**

method

indexer

property

event

```
interface IMyInterface
{
  void Process(int arg1, double arg2);

  float this [int index] { get; set; }

  string Name { get; set; }

  event MouseEventHandler Mouse;
}
```

- **Всі члени інтерфейсу мають неявний рівень доступу public**
- **Явне вказування рівня доступу - помилка**

error, can not explicitly label as `public`

```
interface IMyInterface
{
  public void Process(int arg1, double arg2);
  ...
}
```

Приклади реалізації інтерфейсів

```
interface IFighter
{
  void Punch(int side);
  void Kick (int side);
  void Block();
}
```

оголошення

Реалізація методів

```
class Soldier : IFighter
{
  public void Punch(int side)
  {
    Move(arms[side], Forward);
  }
  public void Kick (int side)
  {
    Move(legs[side], Forward);
  }
  public void Block()
  {
    Move(arms[Left ], Up);
    Move(arms[Right], Up);
  }
  ...
}
```

```
IFighter f = new Soldier();

f.Punch(Left);
f.Kick(Right);
f.Block();
...
```

При використанні посилання `IFighter`, можна викликати лише методи `IFighter`

Множинне наслідування інтерфейсів

Класи допускають наслідування лише одного базового класу, проте можуть реалізувати більше ніж один інтерфейс

```
interface IFighter
{
  void Punch(int side);
  void Kick (int side);
  void Block();
}
```

```
interface IWrestler
{
  void Takedown(int legs);
  void Escape();
}
```

base class

```
class Soldier : Person, IFighter, IWrestler
{
  public void Punch(int side) { ... }
  public void Kick (int side) { ... }
  public void Block() { ... }

  public void Takedown(int legs) { ... }
  public void Escape() { ... }

  ...
}
```

IFighter методи

IWrestler методи

FCL-інтерфейси переліку

```
public interface IEnumerator                    //System.Collections
{//підтримує просте переміщення по колекції
    Object Current { get; } //поточний елемент колекції
    bool MoveNext(); //true – при успішному переміщенні на наступний ел.
                     //false – досягнуто кінець колекції
    void Reset();    // встановлення в початкову позицію, перед першим ел. колекції
 }
public interface IEnumerable                    //System.Collections
{//надає перелічувач
IEnumerator GetEnumerator();
}
```

```
//бібліотека надає також інтерфейси переліку для узагальнень
public interface IEnumerable<T> : IEnumerable //System.Collections.Generic
{
   IEnumerator<T> GetEnumerator();
}
public interface IEnumerator<T> : IDisposable, IEnumerator{
   T Current { get; }
}
```

Реалізація інтерфейсу IEnumerable<T> дозволяє використовувати об”єкти цього класу в інструкції foreach:

class Polygon: IEnumerable {…}

```
Polygon triangle=new Polygon(new Point(),new Point(1,2),
                             new Point(3,5));
foreach(Point p in triangle)
        Console.WriteLine(p);
```

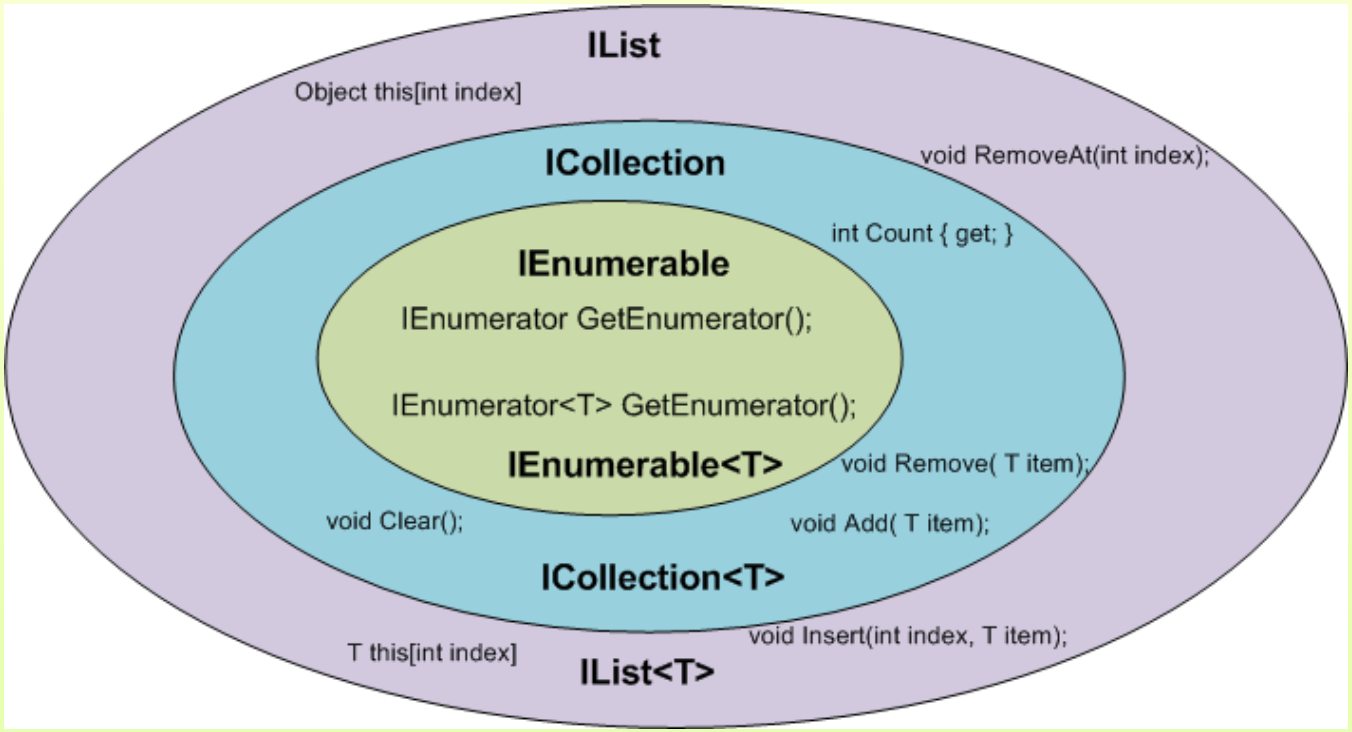
FCL-інтерфейси колекцій

```
public interface ICollection<T> : IEnumerable<T>, IEnumerable
{ //оголошує методи, що використовуються для керування колекціями
        void Add(T item);
        void Clear();
        Boolean Contains(T item);
        void CopyTo(T[] array, Int32 arrayIndex);
        Boolean Remove(T item);
        Int32 Count { get; }
        Boolean IsReadOnly { get; }
}
```

```
public interface IList<T> : ICollection<T>, IEnumerable<T>,
IEnumerable
{
     T this[int index] { get; set; }
      int IndexOf(T item);
    void Insert(int index, T item);
    void RemoveAt(int index);
}
```

```
public abstract class Array :ICloneable, IList<T>,
                              ICollection<T>,IEnumerable<T>
{//одновимірні масиви, початок нумерації з нуля
        ...
```

Інтерфейси IEnumerable, ICollection, IList



FCL-інтерфейси порівняння

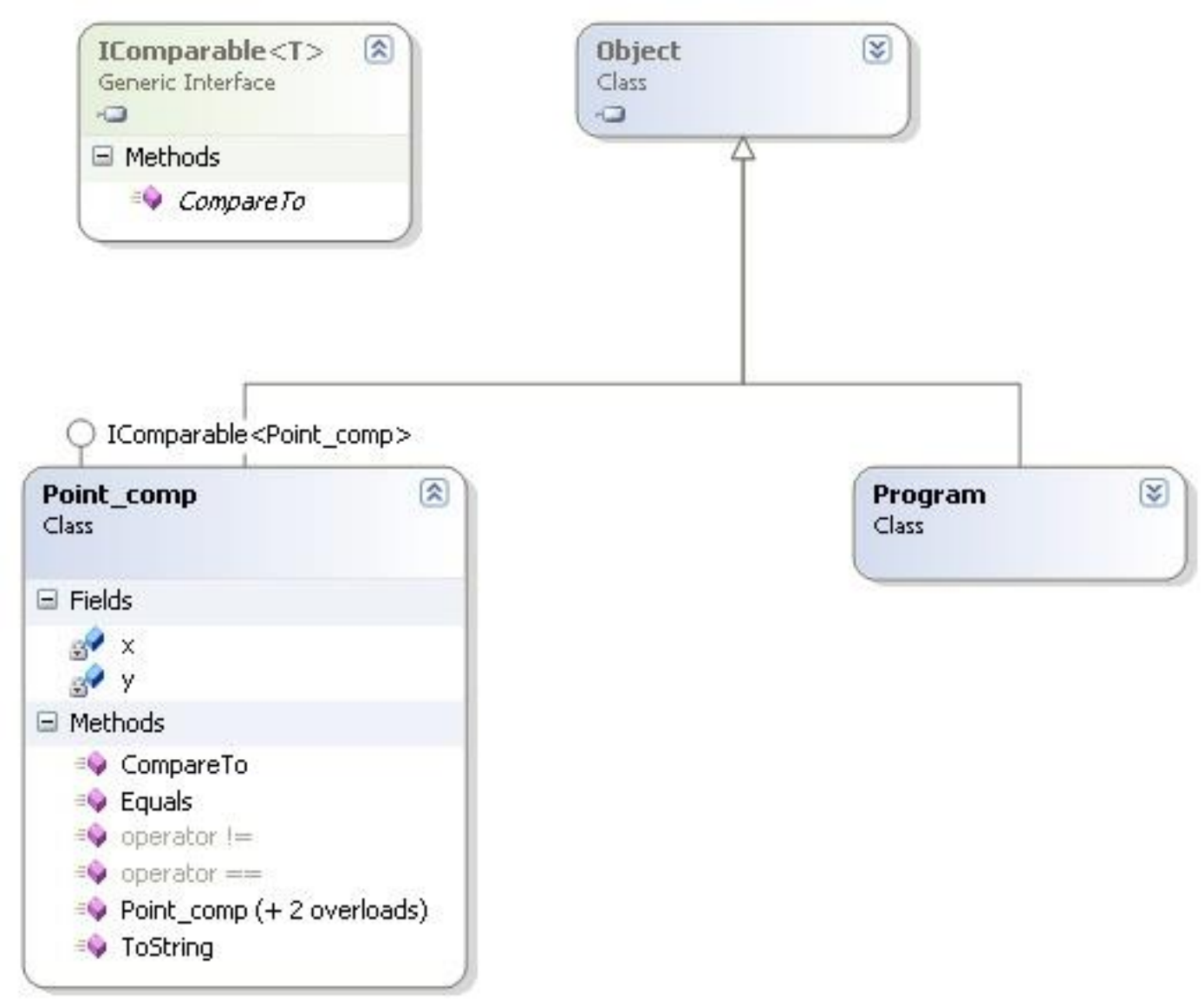
```
public interface IComparable<T> { //System
//Визначає метод порівняння, тип значення чи клас якого використовується для створення методу
//порівняння впорядкованих екземплярів.
      Int32 CompareTo(T other);
}

public interface IComparable { //System
      Int32 CompareTo(Object other);
}
```

Value	Meaning
Від'ємне	Даний об'єкт є менший ніж параметр.
Нуль	Даний об'єкт дорівнює параметру.
Додатнє	Даний об'єкт є більший ніж параметр.

IComparable<Point>

```
class Point_comp: IComparable<Point_comp> {
   private int x;
   private int y;
      ...
   public int CompareTo(Point_comp other)
   {
     return Math.Sign(Math.Sqrt(x * x + y * y)
     - Math.Sqrt(other.x * other.x + other.y * other.y));
   }
   public override string ToString()
   {return " (" + x + ", " + y + ")";   }
}
```

Point_comp Class Diagram



Порівняння об'єктів Point_comp

```
Point_comp pt = new Point_comp();
Point_comp pt1 = new Point_comp(6, 16);
Point_comp pt2 = new Point_comp(10);

if (pt1.CompareTo(pt2) > 0)
   Console.WriteLine("pt1 is far from origin then pt2");
else Console.WriteLine("pt1 is not far from origin then pt2");

Point_comp[] parr = { pt, pt1, pt2, new Point_comp(1, 2) };
Console.WriteLine("array:");
for (int i = 0; i < parr.Length; ++i)
  Console.WriteLine("parr[{0}]={1}", i,
  parr[i]);

Array.Sort(parr);
//Sort вимагає реалізації IComparable<T>
//для кожного елемента масиву
Console.WriteLine("array after sorting:");
int ind = 0;
foreach (Point_comp p in parr)
{
 Console.WriteLine("parr[{0}]={1}",ind, p)
  ++ind;
}
```

```
pt1 is far from origin then pt2

array:
parr[0]= (0, 0)
parr[1]= (6, 16)
parr[2]= (10, 0)
parr[3]= (1, 2)

array after sorting:
parr[0]= (0, 0)
parr[1]= (1, 2)
parr[2]= (10, 0)
parr[3]= (6, 16)
```

Неявна і явна реалізація інтерфейсів

Таблиця методів класу:

1) методи базових класів
2) методи інтерфейсів
3) власні методи

member-name for implementation:

identifier

interface-type . identifier

Неявна реалізація методів інтерфейсу

- Назва методу при реалізації в класі не включає назву інтерфейсу
- Рівень доступу – **`public`**, за замовчуванням **`sealed`**; можлива специфікація **`virtual`**.
- посилання на метод дублюється в секціях 2) і 3) таблиці методів
- виклик - як через змінну типу класу, так і змінну інтерфейсного типу

```
class Point_comp: IComparable<Point_comp>
{       ...
   public int CompareTo(Point_comp other)
   {
        ...
   }
}
```

```
Point_comp pt = new Point_comp(5,6);
pt.CompareTo(new Point_comp(3,4));

IComparable<Point_comp> cp = pt;
cp.CompareTo(new Point_comp(3, 4));
```

Неявна і явна реалізація інтерфейсів

Таблиця методів класу:

1) методи базових класів
2) методи інтерфейсів
3) власні методи

member-name for implementation:

identifier

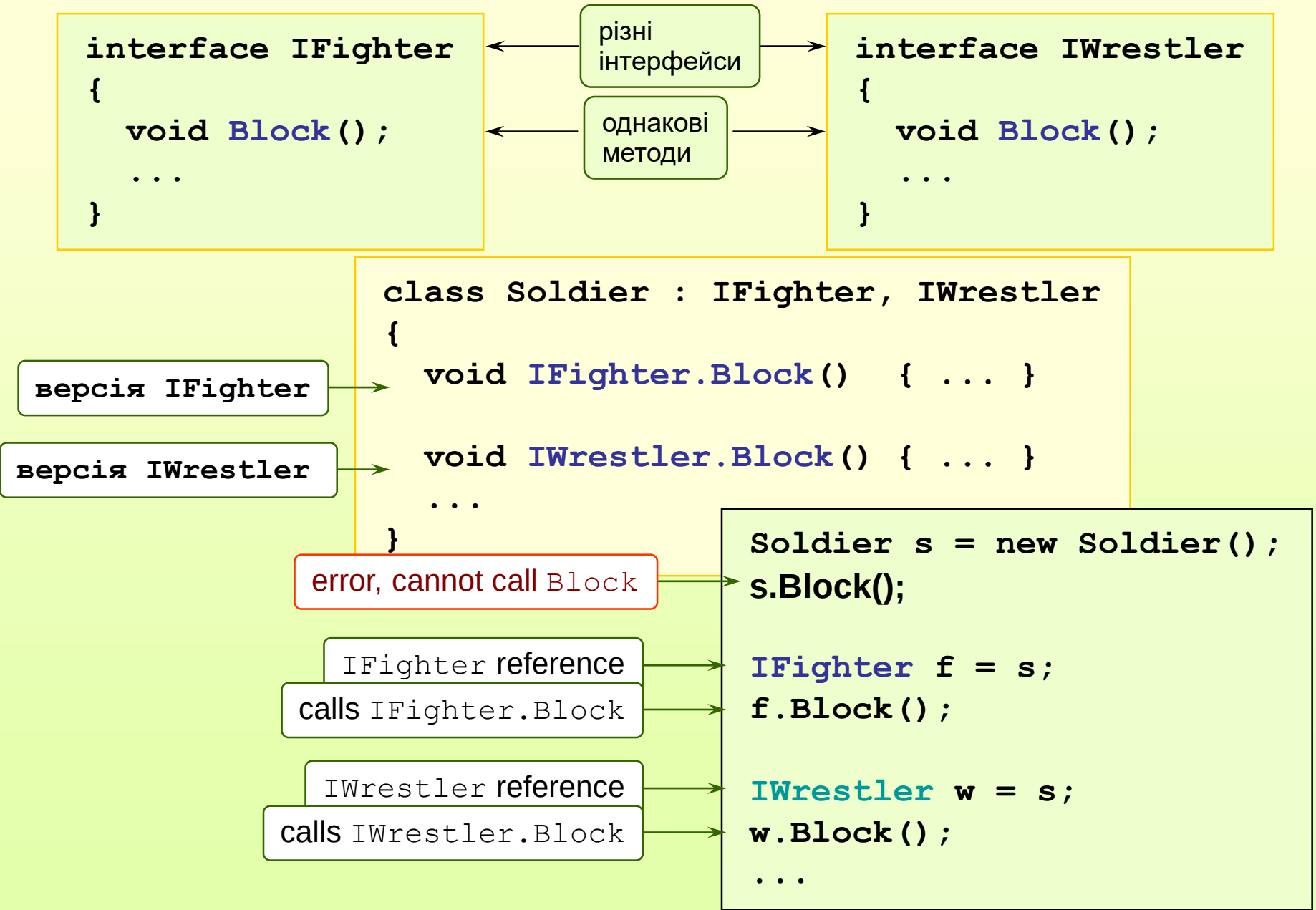
interface-type . identifier

Явна реалізація методів інтерфейсу

(EIMI - Explicid Interface Method Implementation)

- назва методу при реалізації в класі включає назву інтерфейсу
- посилання на метод лише в секції 2) таблиці методів
- доступ не вказується, за замовчуванням **`private`**
- виклик лише через змінну інтерфейсного типу.
- необхідна при співпадінні назв методів інтерфейсів

Приклад явної реалізації



Використання масивів

```
public abstract class Array : ICloneable, IList<T>,
                              ICollection<T>, IEnumerable<T>
{//одновимірні масиви, початок нумерації з нуля
        ...
}
```

```
int[] numbers = { 4, 5, 6, 1, 2, 3, -2, -1, 0 };
Point_comp[] parr = new Point_comp[3];

Point_comp[] parr1 ={pt, pt1, pt2, new Point_comp(10,20)};

Point_comp[,] parr2 = new Point_comp[2, 3]
 { {pt, pt1, new Point_comp(10, 20) },
 {pt2, new Point_comp(30, 40), new Point_comp(40, 50) } };
```

Read-only Instance Properties of System.Array

Member	Description
`Rank`	Повертає розмірність масиву
`Length`	Повертає загальну кількість елементів в масиві
`IsReadOnly`	Повідомляє коли масив є read-only. Для масивів це завжди false.
`IsSynchronized`	Indicates whether the array access is thread-safe. Для масивів це завжди false.
`SyncRoot`	Retrieves an object that can be used to synchronize access to the array. Для масивів це посилання на самого себе.
`IsFixedSize`	Повідомляє коли масив є фіксованого розміру (для масивів завжди true).

Instance Methods of System.Array

Member	Description
`GetLength`	Returns the number of elements in the specified dimension of the array.
`GetLowerBound`	Returns the lower bound of the specified dimension. This is almost always 0.
`GetUpperBound`	Returns the upper bound of the specified dimension. This is almost always the number of elements in the dimension minus 1.
`GetValue`	Returns a reference to the element located at the specified position in the array. If the array contains value types, the return value refers to a boxed copy of the element. This rarely used method is required only when you don't know at design time the number of dimensions in an array.
`SetValue`	Sets the element located at the specified position in the array. This rarely used method is required only when you don't know at design time the number of dimensions in an array.
`GetEnumerator`	Returns an `IEnumeratorfor` the array. This allows using C#'s `foreach` statement (or an equivalent in another language). For multidimension arrays, the enumerator iterates through all the elements, with the right-most dimension changing the fastest.
`Clone`	Creates a new array that's a shallow copy of the source array.
`CopyTo`	Copies elements from one array to another array.
`Initialize`	Calls the default constructor for each element in an array of value types. This method does nothing if the elements in the array are reference types. C# doesn't allow you to define default constructors for value types, so this method has no use for arrays of C# structures. This method is primarily for compiler vendors.

Static Methods of System.Array

Member	Description
Sort	Сортування елементів в одновимірному масиві, пар з двох масивів чи в частині масиву. Елементи масиву повинні реалізувати інтерфейс **ICompable.** Можливесортування з використанням об'єкту I**Comparer**
BinarySearch	Відшукання у вказаному одновимірному масиві заданого елемента алгоритмом бінарного пошуку.
IndexOf	Returns the index of the first occurrence of a value in a one-dimension array or in a portion of it.
LastIndexOf	Returns the index of the last occurrence of a value in a one-dimension array or in a portion of it.
Reverse	Reverses the order of the elements in the specified one-dimension array or in a portion of it.
Copy	Copies a section of one array to another array, performing any appropriate casting required.
Clear	Sets a range of elements in the array to 0 or to a nullobject reference.
CreateInstance	Creates an instance of an array. This rarely used method allows you to dynamically (at run time) define arrays of any type, rank, and bounds.

Використання методів System.Array

```
Point_comp[] parr = new Point_comp[3];
…
for (int i = 0; i < parr.Length; ++i)
      Console.WriteLine("parr[{0}]={1}",i, parr[i]);

Array.Sort(parr);

int ind = 0;
foreach (Point_comp p in parr){…}
```