

III. Collections, Delegates, Events, Other

Contents

1. Collections.....	1
1.1. ArrayList	1
1.2. List<T>	1
1.3. Queues, Stacks, and Sets	3
1.4. Custom Collections	3
2. Delegates.....	4
3. Events.....	5
3.1. Custom Events.....	5
3.2. Standard Event Pattern.....	6
4. Other	7
4.1. Method Parameters.....	7
4.2. Static class	8
4.3. Nullable types	8
4.4. Boxing / Unboxing.....	8

1. Collections



Sample code available at <http://online.ase.ro> – “StandardCollections” Sample

Assignment

1. Create a new project with the name “StandardCollections”

1.1. ArrayList

2. Add the following method in the “Program” class and call it from the Main method

```
private static void ArrayListExample()  
{  
    var words = new ArrayList();  
    words.Add("melon");  
    words.Add("avocado");  
    string first = (string)words[0];  
  
    //int first = (int)words[0];  
}
```

1.2. List<T>

2. Add the following method in the “Program” class and call it from the Main method

```
private static void ListExample()  
{
```

```
// New string-typed list
var words = new List<string>();
words.Add("melon");
words.Add("avocado");
words.AddRange(new[] { "banana", "plum" });

// Insert at start
words.Insert(0, "lemon");

// Insert at start
words.InsertRange(0, new[] { "peach", "nashi" });
words.Remove("melon");

// Remove the 4th element
words.RemoveAt(3);

// Remove first 2 elements
words.RemoveRange(0, 2);

// Remove all strings starting in 'n':
words.RemoveAll(x => x.StartsWith("n"));

for (var i=0; i<words.Count; i++)
{
    Console.WriteLine(words[i]);
}

foreach (var word in words)
{
    Console.WriteLine(word);
}
}
```

3. Add the following “Person” class

```
internal class Person
{
    #region Properties
    public string Name { get; set; }
    public int Age { get; set; }
    #endregion

    public Person(string name, int age)
    {
        Name = name;
        Age = age;
    }
}
```

4. Add the following method in the “Program” class and call it from the Main method

```
private static void ListPersonExample()
{
    var personList = new List<Person>();

    var rnd = new Random();
    for (var i = 0; i < 10; i++)
    {
        personList.Add(new Person("Persoana " + i, rnd.Next(100)));
    }
}
```

```
//Which interface is needed for Array.Sort(personList)

foreach (var p in personList) //equivalent to foreach (var p in personList)
    Console.WriteLine(p);
}
```

1.3. Queues, Stacks, and Sets

- [Queue](#), [Stack<T>](#), [LinkedList<T>](#)
- [SortedList<TKey, TValue>](#) and many others: [link](#)

1.4. Custom Collections

C#Sample code available at <http://online.ase.ro> – “CustomCollections” Sample

Assignment

1. Add the following “PersonCollection” class

```
internal class PersonCollection : IEnumerable<Person>
{
    private Person[] _personArray;

    //Indexer [int]
    public Person this[int index]
    {
        get { return _personArray[index]; }
        set { _personArray[index] = value; }
    }

    //Indexer [string]
    public Person this[string name]
    {
        get { return _personArray.FirstOrDefault(x=>x.Name == name); }
    }

    public int Length
    {
        get { return _personArray.Length; }
    }

    public PersonCollection()
    {
        _personArray = new []
        {
            new Person("name1", 1),
            new Person("name2", 2),
            new Person("name3", 3)
        };
    }

    public IEnumerator<Person> GetEnumerator()
    {
        return new PersonEnumerator(this);
    }

    IEnumerator IEnumerable.GetEnumerator()
    {
        return GetEnumerator();
    }
}
```

```
{  
    return GetEnumerator();  
}
```

2. Add the following “PersonEnumerator” class

```
internal class PersonEnumerator : IEnumerator<Person>  
{  
    private int _nIndex;  
    private PersonCollection _personCollection;  
  
    public PersonEnumerator(PersonCollection personCollection)  
    {  
        _personCollection = personCollection;  
        _nIndex = -1;  
    }  
  
    public bool MoveNext()  
    {  
        _nIndex++;  
        return _nIndex < _personCollection.Length;  
    }  
  
    public void Reset()  
    {  
        _nIndex = -1;  
    }  
  
    public Person Current  
    {  
        get { return _personCollection[_nIndex]; }  
    }  
  
    object IEnumerator.Current  
    {  
        get { return Current; }  
    }  
  
    public void Dispose()  
    {  
    }  
}
```

3. Add the following method in the “Program” class and call it from the Main method

```
private static void PersonCollectionExample()  
{  
    var personList = new PersonCollection();  
  
    foreach (var p in personList)  
        Console.WriteLine(p);  
}
```

2. Delegates



Sample code available at <http://online.ase.ro> – “Delegates” Sample

Assignment

1. Create a new project with the name “Delegates”

```
// This delegate can point to any method, taking two integers and returning an integer.
public delegate int BinaryOp(int x, int y);

//
public class SimpleMath
{
    public static int Add(int x, int y)
    { return x + y; }
    public static int Subtract(int x, int y)
    { return x - y; }
}

internal class Program
{
    private static void Main()
    {
        Console.WriteLine("***** Delegate Example *****\n");

        //Definire si instantiere delegat
        BinaryOp b = new BinaryOp(SimpleMath.Add);
        //BinaryOp b = new BinaryOp(SimpleMath.Subtract);
        //b += new BinaryOp(SimpleMath.Subtract);

        //Apel prin delegat
        Console.WriteLine("10 + 10 is {0}", b(10, 10));
        Console.ReadLine();
    }
}
```

3. Events

3.1. Custom Events



Sample code available at <http://online.ase.ro> – “EventsPropertyTrigger” Sample

Assignment

1. Create a new project with the name “EventsPropertyTrigger”

```
public delegate void PriceChangedHandler(decimal oldPrice, decimal newPrice);

internal class Stock
{
    private string _symbol;
    private decimal _price;

    public Stock(string symbol)
    {
        _symbol = symbol;
    }
}
```

```
public event PriceChangedHandler PriceChanged;

public decimal Price
{
    get { return _price; }
    set
    {
        if (_price == value) return; // Exit if nothing has changed
        decimal oldPrice = _price;
        _price = value;
        if (PriceChanged != null) // If invocation list not
            PriceChanged(oldPrice, _price); // empty, fire event.
    }
}

internal class Program
{
    private static void Main()
    {
        var stock = new Stock("MSFT");
        stock.PriceChanged += Stock_PriceChanged;
        stock.Price = 30;
        stock.Price = 60;
        stock.Price = 90;
    }

    private static void Stock_PriceChanged(decimal oldPrice, decimal newPrice)
    {
        Console.WriteLine("MSFT: {0} {1}", oldPrice, newPrice);
    }
}
```

3.2. Standard Event Pattern



Sample code available at <http://online.ase.ro> – “EventsPropertyTriggerEventArgs” Sample

Assignment

1. Create a new project with the name “EventsPropertyTriggerEventArgs”

```
public class PriceChangedEventArgs : EventArgs
{
    public readonly decimal LastPrice;
    public readonly decimal NewPrice;
    public PriceChangedEventArgs(decimal lastPrice, decimal newPrice)
    {
        LastPrice = lastPrice;
        NewPrice = newPrice;
    }
}

public class Stock
{
    private string _symbol;
    private decimal _price;

    public Stock(string symbol)
```

```

    {
        _symbol = symbol;
    }

    public event EventHandler<PriceChangedEventArgs> PriceChanged;
    protected virtual void OnPriceChanged(PriceChangedEventArgs e)
    {
        if (PriceChanged != null) PriceChanged(this, e);
    }
    public decimal Price
    {
        get { return _price; }
        set
        {
            if (_price == value) return;
            decimal oldPrice = _price;
            _price = value;

            OnPriceChanged(new PriceChangedEventArgs(oldPrice, _price));
        }
    }
}

internal class Program
{
    private static void Main()
    {
        var stock = new Stock("MSFT");
        stock.PriceChanged += Stock_PriceChanged1; ;
        stock.Price = 30;
        stock.Price = 60;
        stock.Price = 90;
    }

    private static void Stock_PriceChanged1(object sender, PriceChangedEventArgs e)
    {
        Console.WriteLine("MSFT: {0} {1}", e.LastPrice, e.NewPrice);
    }
}

```



More event samples available at <http://online.ase.ro>

4. Other

4.1. Method Parameters

The **ref** keyword causes an argument to be passed by reference, not by value. The effect of passing by reference is that any change to the parameter in the called method is reflected in the calling method. For example, if the caller passes a local variable expression or an array element access expression, and the called method replaces the object to which the ref parameter refers, then the caller's local variable or the array element now refer to the new object.



Further reading: [link](#)

The **out** keyword causes arguments to be passed by reference. This is like the ref keyword, except that ref requires that the variable be initialized before it is passed. To use an out parameter, both the method definition and the calling method must explicitly use the out keyword.



Further reading: [link](#)

Assignment



Sample code available at <http://online.ase.ro> – “MethodParameters” Sample

```
private static void IncrementNormal(int i)
{
    i = i + 1;
}

private static void IncrementRef(ref int i)
{
    i = i + 1;
}

private static void IncrementOut(int i, out int result)
{
    result = i + 1;
}

private static void Main()
{
    //Normal
    var val = 1;
    IncrementNormal(val);
    Console.WriteLine(val);

    //Ref
    val = 1;
    IncrementRef(ref val);
    Console.WriteLine(val); // Output: 2

    //out
    val = 1;
    int result;
    IncrementOut(val, out result);
    Console.WriteLine(result); // Output: 2
}
```

4.2. Static class



Further reading: [link](#)

4.3. Nullable types



Further reading: [link](#)

4.4. Boxing / Unboxing



Further reading: [link](#)