**Generic**

جنریک ها در سی شارپ ما را قادر خواهند ساخت کلاس هایی را تعریف کنیم که در زمان استفاده از آنها (و نه در زمان تعریف) نوع داده ای فیلد ها ، پروپرتی ها و متد های آنها را مشخص کنیم. در این فصل با کلاس های جنریک در سی شارپ آشنا میشویم.

**جنریک ها**

جنریک ها در نسخه 2 زبان سی شارپ معرفی شدند. جنریک ها به ما اجازه میدهند کلاس‌هایی را تعریف کنیم که نوع داده ای فیلدها ، پارامتر ها و متد هایشان در زمان استفاده از آنها تعیین شوند. یک کلاس جنریک می‌تواند به وسیله براکت های زاویه ای <> تعریف شود.

class MyGenericClass<T>

{

private T genericMemberVariable;

public MyGenericClass(T value)

{

genericMemberVariable = value;

}

//method

public T genericMethod(T genericParameter)

{

Console.WriteLine("Parameter type: {0}, value: {1}", typeof(T).ToString(),genericParameter);

Console.WriteLine("Return type: {0}, value: {1}", typeof(T).ToString(), genericMemberVariable);

return genericMemberVariable;

}

public T genericProperty { get; set; }

}

همانطور که در مثال بالا مشاهده می کنید کلاس MyGenericClass به همراه <T> تعریف شده است. علامت های براکت زاویه ای <> به ما نشان می دهد که کلاس مورد نظر جنریک بوده و نوع داده‌ی اعضای آن بعداً مشخص خواهد شد. در  اینجا به جای نوع داده ای از کاراکتر T استفاده کرده ایم. اما شما می‌توانید از هر کاراکتر و یا کلمه دیگری به جای آن استفاده کنید.

اکنون می‌توان در زمان استفاده از کلاس جنریک تعریف شده در بالا نوع داده ای آن را مشخص کرد. در نمونه مثال زیر از نوع داده ای int استفاده شده است :

MyGenericClass<int> intGenericClass = new MyGenericClass<int>(10);

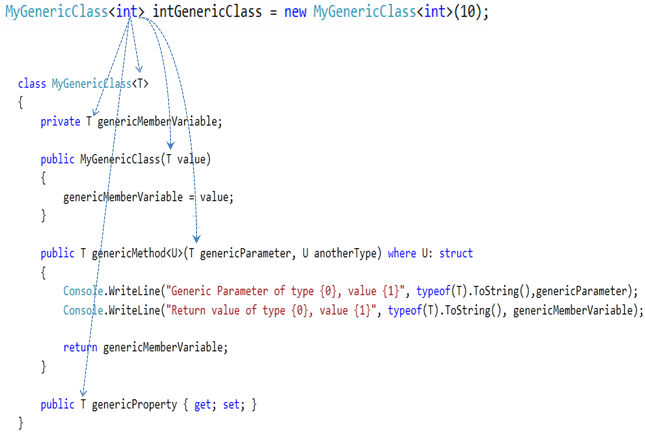
int val = intGenericClass.genericMethod(200);

خروجی نمونه مثال  بالا به شکل زیر است :

Parameter type: int, value: 200

Return type: int, value: 10

شکل زیر نشان می‌دهد که کامپایلر چطور نوع داده ای int را جایگزین کاراکتر T می‌کند :



کلاس <MyGenericClass<int تعریف شده در بالا به شکل زیر کامپایل می شود :

class MyGenericClass

{

private int genericMemberVariable;

public MyGenericClass(int value)

{

genericMemberVariable = value;

}

public int genericMethod(int genericParameter)

{

Console.WriteLine("Parameter type: {0}, value: {1}", typeof(int).ToString(), genericParameter);

Console.WriteLine("Return type: {0}, value: {1}", typeof(int).ToString(), genericMemberVariable);

return genericMemberVariable;

}

public int genericProperty { get; set; }

}

کلاس های جنریک به عنوان کلاس پایه

زمانی که از یک کلاس جنریک ارث بری می کنید باید نوع داده‌ ای جنریک را مشخص کنید :

class MyDerivedClass : MyGenericClass<string>

{

//implementation

}

اما اگر بخواهید کلاس مشتق شده ، یک کلاس جنریک باشد، نیاز نیست نوع داده ای را مشخص کنید :

class MyDerivedClass<U> : MyGenericClass<U>

{

//implementation

}

# **آشنایی با کالکشن جنریک List در سی شارپ**

## کالکشن <List<T

در فصل های قبل مطالبی را در مورد ArrayList آموختید. یک ArrayList زمانی که مقداری به آن افزوده می‌شود، اندازه خود را افزایش می‌دهد. کالکشن <List<T عملکردی شبیه ArrayList دارد. با این حال <List<T یک کالکشن جنریک بوده و ArrayList یک کالکشن غیر جنریک است.

<List<T را می توان به دو روش زیر تعریف و نمونه سازی کرد :

List<int> intList = new List<int>();

//Or

IList<int> intList = new List<int>();

در نمونه مثال بالا اولین عبارت از متغیری با نوع List استفاده می کند و این در حالی است که عبارت دوم از متغیری با نوع IList بهره می برد. <List<T اینترفیس <IList<T را پیاده سازی می کند. در برنامه نویسی به روش شی گرایی پیشنهاد میشود از اینترفیس ها به جای نوع های واقعی استفاده شود. این مورد می تواند در تست و نگهداری برنامه ها کمک قابل توجهی به ما بکند. بنابراین از <IList<T برای تعریف شی ایی از نوع <List<T استفاده کنید.

## اضافه کردن عناصر به لیست

از متد ()Add برای افزودن مقادیر به کالکشن مورد نظر خود استفاده کنید. نمونه مثال زیر مقادیر اعداد صحیح را به لیستی از نوع int اضافه می‌کند :

IList<int> intList = new List<int>();

intList.Add(10);

intList.Add(20);

intList.Add(30);

intList.Add(40);

IList<string> strList = new List<string>();

strList.Add("one");

strList.Add("two");

strList.Add("three");

strList.Add("four");

strList.Add("four");

strList.Add(null);

strList.Add(null);

IList<Student> studentList = new List<Student>();

studentList.Add(new Student());

studentList.Add(new Student());

studentList.Add(new Student());

همچنین شما می توانید مقادیر را در زمان نمونه‌سازی به لیست خود اضافه کنید :

IList<int> intList = new List<int>(){ 10, 20, 30, 40 };

//Or

IList<Student> studentList = new List<Student>()

{

new Student(){ StudentID=1, StudentName="Bill"},

new Student(){ StudentID=2, StudentName="Steve"},

new Student(){ StudentID=3, StudentName="Ram"},

new Student(){ StudentID=1, StudentName="Moin"}

};

## متد ()AddRange

با استفاده از متد ()AddRange می توان همه عناصر یک کالکشن را به کالکشن دیگری اضافه کرد. در نمونه مثال زیر مقادیر کالکشن intList1 به کالکشن intList2 اضافه شده است:

IList<int> intList1 = new List<int>();

intList1.Add(10);

intList1.Add(20);

intList1.Add(30);

intList1.Add(40);

List<int> intList2 = new List<int>();

intList2.AddRange(intList1);



نکته : متد ()AddRange تنها بر روی کالکشن هایی با نوع <List<T کاربرد دارد. در اینجا نمی توان از <IList<T استفاده نمود.

## دسترسی به عناصر لیست

با استفاده از متد ForEach و به شکل زیر میتوان آیتم های یک لیست را پیمایش کرد :

List<int> intList = new List<int>() { 10, 20, 30 };

intList.ForEach(el => Console.WriteLine(el));

اگر شما از نوع داده ای <IList<T برای متغیر خود استفاده کرده باشید از حلقه foreach به شکل زیر برای پیمایش استفاده کنید :

IList<int> intList = new List<int>() { 10, 20, 30, 40 };

foreach (var el in intList)

Console.WriteLine(el);

میتوان به عناصر موجود در لیست با استفاده از مقادیر index دسترسی پیدا کرد :

IList<int> intList = new List<int>() { 10, 20, 30, 40 };

int elem = intList[1]; // returns 20

میتوان از حلقه for برای دسترسی به عناصر آرایه به شکل زیر استفاده نمود :

IList<int> intList = new List<int>() { 10, 20, 30, 40 };

for (int i = 0; i < intList.Count; i++)

Console.WriteLine(intList[i]);

خروجی نمونه مثال بالا به شکل زیر است :

10

20

30

40

<List<T اینترفیس <IList<T را پیاده سازی می کند. بنابراین <List<T می تواند به صورت ضمنی به <IList<T تبدیل شود :

static void Print(IList<string> list)

{

Console.WriteLine("Count: {0}", list.Count);

foreach (string value in list)

{

Console.WriteLine(value);

}

}

static void Main(string[] args)

{

string[] strArray = new string[2];

strArray[0] = "Hello";

strArray[1] = "World";

Print(strArray);

List<string> strList = new List<string>();

strList.Add("Hello");

strList.Add("World");

Print(strList);

}

خروجی نمونه مثال بالا به شکل زیر است :

Hello

World

Hello

World

## افزودن عناصر به لیست

با استفاده از متد ()Insert می توان مقادیر را به لیست مورد نظر اضافه کرد :

IList<int> intList = new List<int>(){ 10, 20, 30, 40 };

intList.Insert(1, 11);// inserts 11 at 1st index: after 10.

foreach (var el in intList)

Console.Write(el);

خروجی نمونه مثال بالا به شکل زیر است :

10

11

20

30

40

## حذف عناصر از لیست

با استفاده از متدهای ()Remove و ()RemoveAt می توان مقادیر را از یک کالکشن <List<T حذف کرد :

IList<int> intList = new List<int>(){ 10, 20, 30, 40 };

intList.Remove(10); // removes the 10 from a list

intList.RemoveAt(2); //removes the 3rd element (index starts from 0)

foreach (var el in intList)

Console.Write(el);

خروجی نمونه مثال بالا :

20

30

## متد ()TrueForAll

()TrueForAll متدی از کلاس <List<T است. این متد مقدار true را در صورتی که شرط موردنظر صحیح باشد بر می گرداند و در غیر این صورت مقدار بازگشتی false خواهد بود :

List<int> intList = new List<int>(){ 10, 20, 30, 40 };

bool res = intList.TrueForAll(el => el%2 == 0);// returns true

در نمونه مثال بالا در صورتی که مقادیر موجود در لیست همگی بر ۲ بخش پذیر باشند، مقدار true برگردانده میشود. این متد همچنین می تواند یک متد را به عنوان delegate دریافت کند :

static bool isPositiveInt(int i)

{

return i > 0;

}

static void Main(string[] args)

{

List<int> intList = new List<int>(){10, 20, 30, 40};

bool res = intList.TrueForAll(isPositiveInt);

}

# **آشنایی با کالکشن جنریک SortedList در سی شارپ**

با استفاده از کالکشن جنریک SortedList میتوان مقادیر را به شکل کلید و مقدار ذخیره سازی کرد. در فصول قبلی با نوع غیر جنریک آن آشنا شدید. اما در این فصل به SortedList  جنریک خواهیم پرداخت.

## کالکشن <SortedList<TKey,TValue

کالکشن SortedList مقادیر را به صورت key و value در خود ذخیره میکند. این مقادیر به صورت پیش فرض بر اساس key مرتب سازی می‌شوند. سی شارپ دارای دو نوع SortedList است یکی جنریک و دیگری غیر جنریک ([SortedList غیر جنریک](http://learnsource.net/post/171/%DA%A9%D9%84%D8%A7%D8%B3-sortedlist-%D8%AF%D8%B1-%D8%B3%DB%8C-%D8%B4%D8%A7%D8%B1%D9%BE)) در این فصل با نوع جنریک آن آشنا خواهیم شد.

نوع جنریک کالکشن SortedList به شکل <SortedList<TKey,TValue بوده که TKey نوع کلید و TValueنوع مقدار را مشخص میکند.

## تعریف و نمونه سازی یک SortedList

می توان یک SortedList جنریک را به شکل زیر تعریف کرد :

SortedList<int,string> mySortedList = new SortedList<int,string>();

در نمونه مثال بالا از مقدار int برای نوع key و از نوع string برای نوع value استفاده شده است.

## افزودن مقادیر به SortedList

از متد ()Add برای افزودن مقادیر به SortedList استفاده کنید. key نمیتواند null باشد اما value می‌تواند شامل null باشد.

SortedList<int,string> sortedList1 = new SortedList<int,string>();

sortedList1.Add(3, "Three");

sortedList1.Add(4, "Four");

sortedList1.Add(1, "One");

sortedList1.Add(5, "Five");

sortedList1.Add(2, "Two");

SortedList<string,int> sortedList2 = new SortedList<string,int>();

sortedList2.Add("one", 1);

sortedList2.Add("two", 2);

sortedList2.Add("three", 3);

sortedList2.Add("four", 4);

// Compile time error: cannot convert from <null> to <int>

// sortedList2.Add("Five", null);

SortedList<double,int?> sortedList3 = new SortedList<double,int?>();

sortedList3.Add(1.5, 100);

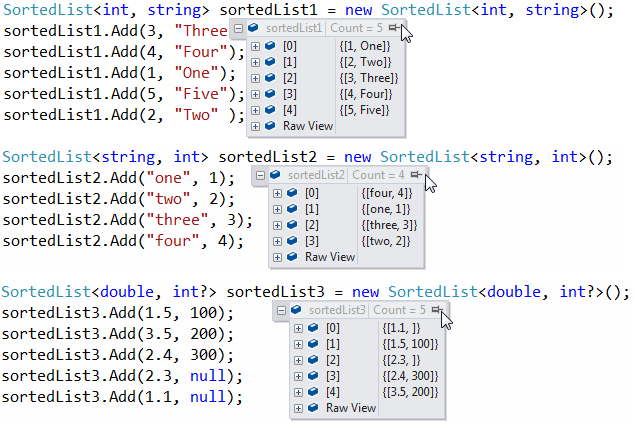
sortedList3.Add(3.5, 200);

sortedList3.Add(2.4, 300);

sortedList3.Add(2.3, null);

sortedList3.Add(1.1, null);

کالکشن SortedList زمانی که عنصری به آن افزوده می شود عناصر خود را مرتب سازی می کند. این مرتب سازی بر اساس key ها انجام می‌شود. اگر شما در محیط دیباگ ویژوال استادیو لیست ها را بررسی کنید می توانید این مورد را به وضوح مشاهده کنید :

**همانطور که در شکل بالا مشاهده می کنید sortedList1 بر اساس مقادیر key (در اینجا اعداد صحیح) مرتب سازی شده است. در sortedList2 از آنجایی که نوع key رشته است ، این مرتب سازی بر اساس حروف الفبا انجام شده است.**

## دسترسی به عناصر SortedList

مقادیر در لیست SortedList می‌توانند با استفاده از مقدار index و key مورد دسترسی قرار بگیرند :

SortedList<string,int> sortedList2 = new SortedList<string,int>();

sortedList2.Add("one", 1);

sortedList2.Add("two", 2);

sortedList2.Add("three", 3);

sortedList2.Add("four", 4);

Console.WriteLine(sortedList2["one"]);

Console.WriteLine(sortedList2["two"]);

Console.WriteLine(sortedList2["three"]);

//Following will throw runtime exception: KeyNotFoundException

Console.WriteLine(sortedList2["ten"]);

خروجی نمونه مثال بالا به شکل زیر است :

1

2

3

نمونه مثال زیر نحوه دسترسی به مقادیر SortedList با استفاده از حلقه for را نشان می دهد :

SortedList<string,int> sortedList2 = new SortedList<string,int>();

sortedList2.Add("one", 1);

sortedList2.Add("two", 2);

sortedList2.Add("three", 3);

sortedList2.Add("four", 4);

for (int i = 0; i < sortedList2.Count; i++)

{

Console.WriteLine("key: {0}, value: {1}", sortedList2.Keys[i], sortedList2.Values[i]);

}

خروجی نمونه مثال بالا به شکل زیر است :

key: four, value: 4

key: one, value: 1

key: three, value: 3

key: two, value: 2

میتوان به شکل زیر و با استفاده از حلقه foreach و ساختار KeyValuePair به عناصر لیست دسترسی داشت :

SortedList<string,int> sortedList2 = new SortedList<string,int>();

sortedList2.Add("one", 1);

sortedList2.Add("two", 2);

sortedList2.Add("three", 3);

sortedList2.Add("four", 4);

foreach(KeyValuePair<string,int> kvp in sortedList2 )

Console.WriteLine("key: {0}, value: {1}", kvp.Key , kvp.Value );

خروجی نمونه مثال بالا به شکل زیر است :

key: four, value: 4

key: one, value: 1

key: three, value: 3

key: two, value: 2

## حذف عناصر از SortedList

با استفاده از متدهای (Remove(key و (RemoveAt(index می توان مقادیر را از SortedList حذف کرد :

SortedList<string,int> sortedList2 = new SortedList<string,int>();

sortedList2.Add("one", 1);

sortedList2.Add("two", 2);

sortedList2.Add("three", 3);

sortedList2.Add("four", 4);

sortedList2.Remove("one");//removes the element whose key is 'one'

sortedList2.RemoveAt(0);//removes the element at zero index i.e first element: four

foreach(KeyValuePair<string,int> kvp in sortedList2 )

Console.WriteLine("key: {0}, value: {1}", kvp.Key , kvp.Value );

خروجی نمونه مثال بالا به شکل زیر است :

key: three, value: 3

key: two, value: 2