مقدمه

وقتی که NET Framework 1.0. عرضه شد و با توجه به اینکه مایکروسافت قبلا امکان استفاده از Generic Class ها را در MFC قرار داده بود، همه انتظار داشتند که در NET. هم چنین امکانی وجود داشته باشد ولی نبود! در NET Framework 1.1. هم خبری از آن نبود. اما با عرضه NET Framework 2.0. انتظار به سر آمد! و بالاخره مایکروسافت Generic Class ها را عرضه کرد.

کلاسهای Generic

به زبان ساده کلاسهای Generic کلاسهایی هستند که نوع داده آنها بصورت پارامتریک تعریف می شود. نحوه تعریف یک کلاس Generic بدین نحو می باشد:

```
Public Class GClass<T>
{
}
```

مثال زیر را ببینید:

```
Public Class GClass<T>
{
        Public T Field;
}
```

در زمان تعریف شی از این کلاس به این شکل استفاده می نماییم:

```
GClass<int> obj=new GClass<int>();
obj.Field=5;
```

استفاده از کلاسهای Generic باعث می شود که بتوانیم کلاسهایی با کاربری بالاتر بنویسیم. کلاسهای Generic هم باعث بالا رفتن قابلیت استفاده مجدد کلاس خواهد شد و هم می توان با آن کلاسهای Type Safe نوشت. مثلا تصور کنید که در یک برنامه از ArrayList استفاده کرده اید. هر عضو ArrayList در حقیقت یک شی از نوع Object است و این یعنی هر نوع داده ای را می توان درون آن قرار داد. حالا تصور کنید که یک ArrayList دارید که می خواهید فقط اعداد int را درون آن قرار دهید و در زمان خواندن اطلاعات از آن، اطلاعات درون آن را بافرض int بودن به int ، Cast می کنید. ولی به هر دلیل اگر اعدادی به غیر از نوع داده int داخل ArrayList قرار گرفته باشد، ذخیره خواهد شد ولی در زمان خواندن اطلاعات به Error برخورد می کنید. این نوع ساختمان داده ها Type Safe محسوب نمیشوند. یک راه جلوگیری از چنین اشتباهاتی پیاده سازی این نوع ساختمان داده ها با استفاده از کلاسهای Generic است. چون در زمان تعریف شی از نوع کلاس Generic، نوع داده آن را مشخص می کنید پس شی تولید شده Type Safe محسوب خواهد شد.

ایجاد محدودیت در کلاسهای Generic

وقتی یک کلاس Generic می سازید ممکن است بخواهید محدودیتهایی در انواع داده مورد قبول آن کلاس ایجاد نمایید. دو راه برای این کار دارید: یا اینکه کلاس شما هر نوع داده ای را قبول کرده و سپس خودتان بر اساس کدهایی که می نویسید تشخیص دهید که نوع داده مجاز است یا خیر. راه دوم این است که قواعدی (Constraints) برای انواع داده مجاز برای کلاس مورد نظر تعیین کنید.

فرض کنید کلاسی داریم بنام Person با ساختار زیر:

```
public class Person
{
    private string _name="";
    private int _age=0;

    public Person(string s,int i)
    {
        _name = s;
        _age = i;
    }

    public string Name
    {
        get { return _name; }
        set { _name = value; }
    }

    public int Age
    {
        get { return _age; }
        set { _age = value; }
    }
}
```

حالا می خواهیم یک کلاس Generic بسازیم که فقط نوع Person را به عنوان نوع داده ورودی قبول کند. به این نکته باید توجه کنید که وقتی منظور از این جمله کلاس Person و کلاسهایی که از آن ارث بری دارند، می باشد.

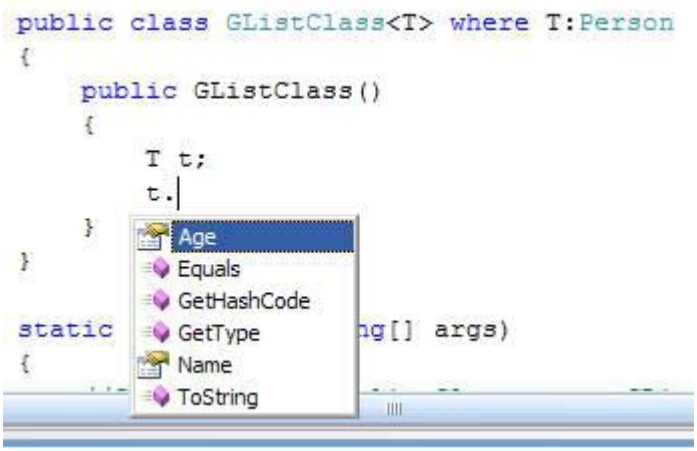
در این مثال می خواهیم کلاسی بسازیم که به عنوان لیستی از Person ها مورد استفاده قرار می گیرد. بنابراین ساختار زیر را برای آن در نظر می گیریم:

```
public class GListClass<T>
{


}
```

اما برای اینکه آن را محدود به کلاس Person کنیم تعریف کلاس جدید را به شکل زیر عوض می کنیم:

```
public class GListClass<T> where T:Person
{


}
```

در اینجا نکته اول این است که نوع داده شما در کلاس Generic شناخته شده است یعنی وقتی که یک متغیر از نوع T تعریف کنید و بعد Propery های آن را ببینید، Name و Age هم در لیست Property ها وجود دارند.



حالا اگر در جایی از برنامه یک شی از کلاس GListClass به شکل زیر بسازید در زمان کمپایل از شما Error خواهد گرفت:

```
GListClass<int> listClass = new GListClass<int>();
```

علتش این است که شما نمی توانید نوع داده کلاس GListClass را چیزی بجز Person و یا کلاسهایی که از Person به ارث بری دارند انتخاب کنید ولی کد زیر کاملا صحیح و قابل اجراست:

```
GListClass<Person> listClass = new GListClass<Person>();
```