**Dumi Soft**

Coding Conventions (قراردادهای برنامه نویسی)

1. برای Class ، record ، struct از شیوه PascalCasing استفاده گردد .

public class DataService

public record PhysicalAddress

public struct ValueCoordinate

1. برای interface ، علاوه بر پیشوند نام با I از شیوه PascalCasing استفاده گردد .

public interface IWorkerQueue

1. برای fields, properties, events, methods,local functions از شیوه PascalCasing استفاده گردد .

public bool IsValid;

public IWorkerQueue WorkerQueue { get; init; }

public event Action EventProcessing;

public void StartEventProcessing();

public record PhysicalAddress(

string Street,

string City,

string StateOrProvince,

string ZipCode);

1. برای فیلد های Private یا internal از شیوه camelCasing به همراه \_ در شروع نام متغیر استفاده گردد .

private IWorkerQueue \_workerQueue;

1. برای فیلد های static بصورت Private یا internal از شیوه camelCasing به همراه s\_ در شروع نام متغیر و در صورتی که thread باشد از t\_ استفاده گردد .

private static IWorkerQueue s\_workerQueue;

[ThreadStatic]

private static TimeSpan t\_timeSpan;

1. برای پارامترهای متد ها از شیوه camelCasing استفاده گردد .

public T SomeMethod<T>(int someNumber, bool isValid)

قراردادهای چیدمان (Layout conventions)

1. در هر خط فقط یک دستور نوشته شود.
2. در هر خط فقط یک اعلان تعریف شود .
3. تو رفتگی (indent )در کد نویسی و خطوط رعایت گردد .
4. در تعریف کلاس ها حداقل یک خط خالی بین تعریف خصوصیات و تعریف متد ها باشد

قرارداد های توضیحات (Commenting conventions)

1. کامنت ها در خطوط جداگانه نوشته شوند و نه در آخر خط کد .
2. شروع متن کامنت با حرف بزرگ شروع شود.
3. پایان متن کامنت نقطه گذاشته شود.
4. بین comment delimiter (//) و متن کامنت یک فاصله باشد .

// The following declaration creates a query. It does not run

// the query.

1. از بلوک های قالب بندی قالب ، با ستاره و خط تیره خودداری کنید .
2. برای تمام اعضای عمومی در کل پروژه از کامنت مناست استفاده گردد .

دستورالعمل های زبان (Language guidelines)

1. برای الحاق رشته ها از $ استفاده شود .

string displayName = $"{nameList[n].LastName}, {nameList[n].FirstName}";

1. برای الحاق رشته های بزرگ از StringBuilder استفاده گردد .
2. در زمان تعریف متغیر در صورتی که تکلیف نوع داده ای مشخص است و یا نوع دقیق آن مهم نیست متغیر ضمنی (implicit typing) تعریف گردد .

var var1 = "This is clearly a string.";

var var2 = 27;

1. در زمان تعریف متغیر در زمانی که تکلیف نوع داده ای سمت راست مشخص نیست ، بصورت ضمنی تعریف نگردد .

int var3 = Convert.ToInt32(Console.ReadLine());

int var4 = ExampleClass.ResultSoFar();

1. برای تعیین نوع متغیر به نام متغیر تکیه نکنید (گمراه کننده است)

var inputInt = Console.ReadLine();

Console.WriteLine(inputInt);

1. برای تعریف متغییر در حلقه ها forوforeach از نوع صریح استفاده گردد .

for (var i = 0; i < 10000; i++)

{

manyPhrases.Append(phrase);

}

/////////////////////////

for (int i = 0; i < 10000; i++)

{

manyPhrases.Append(phrase);

}

1. به طور کلی ، به جای انواع بدون علامت ، از int استفاده گردد. استفاده از int در سرتاسر C#رایج است و هنگام استفاده از int راحت تر می توانید با کتابخانه های دیگر تعامل داشته باشید.
2. برای رسیدگی به خطا همیشه از try/catch استفاده گردد .
3. برای تعریف instantiation از object از یک سینتکس در کل پروژه استفاده گردد .

var instance1 = new ExampleClass();

ExampleClass instance2 = new();

پیشنهادی

ExampleClass instance2 = new ExampleClass();

1. مقداردهی به object ها در زمان تعریف آنها انجام گردد .

var instance3 = new ExampleClass { Name = "Desktop", ID = 37414,

Location = "Redmond", Age = 2.3 };

var instance4 = new ExampleClass();

instance4.Name = "Desktop";

instance4.ID = 37414;

instance4.Location = "Redmond";

instance4.Age = 2.3;

1. از نامهای معنی دار و بصورت camelCasing در متغیر های جستجو استفاده گردد .

var seattleCustomers = from customer in customers

where customer.City == "Seattle"

select customer.Name;

لیست مشتریانی که در Seattle زندگی میکنند

1. هیچ متدی بیشتر از سه پارامتر ورودی نداشته باشد و در اینصورت پارامتر ها در قالب یک کلاس dto به متد ارسال گردند .

نسخه : 1.0.0

تاریخ : 30/05/1400

تهیه کننده : پیمان پردل

سمت : برنامه نویس