فهرست مطالب

مبانی سازنده ها

کلاس زیر را در نظر بگیرید.

?

```
public class BankAccount
{
    private string customerName;
    private decimal originalDeposit;
}
```

پس از تعریف متغیر یک کلاس در برنامه، زمانی که برنامه بالا می آید، compilerبه هر یک از اعضای کلاس به اندازه ی کافی حافظه اختصاص می دهد. فضای ای که به هر متغیر عضو تخصیص داده می شود با توجه به نوع آن متغیر مقداردهی اولیه می شود .

برای شی از نوع(string object) string ، فضای مزبور خالی نگه داشته می شود. برای نوعinteger ، فضای حافظه با ۰ پر می شود. برای شئ ای از این نوع، بهتر است مقداری فراهم شود که متغیرهای عضو را با مقدارهای دلخواه شما مقداردهی اولیه می کند .

متدی که شی ای را مقداردهی (اولیه) می کند، ممکن است هر مقداری برگرداند، اما بهتر است مقدار بازگشتی از نوع void باشد زیرا هدف اصلی آن تنظیم دوباره ی مقادیر است. به این خاطر که متد، مقداری اولیه به تک تک متغیرهای عضو می دهد (متغیرهایی که طبیعتاً باید مقداردهی اولیه شوند)، لازم است آرگومانی معادل برای هر یک از متغیرهایی که مقداردهی اولیه می کند داشته باشد.

?

```
public class BankAccount
{
    private string customerName;
    private decimal originalDeposit;

    public void Initialize(string name, decimal deposit)
    {
    }
}
```

متدی که کلاسی را مقداردهی اولیه می کند، لزومی ندارد حتماً تمامی اعضای یک کلاس را مقداردهی (اولیه) کند. به منظور پیاده سازی متد) منظور متدی است که مقداردهی اولیه می کند)، کافی است آرگومان متد را به متغیر مربوط اختصاص دهید. به مثال های ذیل توجه کنید .

?

```
public class BankAccount
{
    private string customerName;
    private decimal originalDeposit;

```

```
    public void Initialize(string name, decimal deposit)
    {
        customerName = name;
        originalDeposit = deposit;
    }
}
```

سپس می توانید متد را پس از تعریف نمونه ای از کلاس، فراخوانی کنید تا مقادیر اولیه را به فیلدهای کلاس بدهد.

مثال :

?

```
public class BankAccount
{
    private string customerName;
    private decimal originalDeposit;

    public void Initialize(string name, decimal deposit)
    {
        customerName = name;
        originalDeposit = deposit;
    }

    public void Show()
    {
        System.Console.WriteLine("Customer Account Information");
        System.Console.Write("Customer Name:  ");
        System.Console.WriteLine(customerName);
        System.Console.Write("Original Deposit: ");
        System.Console.WriteLine(originalDeposit);
    }
}
public class Exercise
{
    static int Main()
    {
        BankAccount account = new BankAccount();

        account.Initialize("Paul Motto", 450.00M);
        account.Show();

        return 0;
    }
}
Customer Account Information
Customer Name: Paul Motto
Original Deposit: 450.00
Press any key to continue...
```

معرفی سازنده ها

- Microsoft Visual Studioرا اجرا کنید .
- به منظور ایجاد برنامه ی جدید، فهرست گزینه ی اصلی برنامه را باز کرده، سپس گزینه های File -> New Projectرا کلیک کنید .
- گزینه ی Empty Project را از لیست میانی انتخاب کنید .
- اسم را به DepartmentStore3 تغییر داده، روی ok کلیک کنید .
- در پنجره ی Solution Explorer راست کلیک کرده، سپس طبق دستور زیر عمل کنید .
- DeparmentStore3 ->Add ->New Item.
- در لیست میانی پنجره ی محاوره یAdd New Item ، Code Fileرا کلیک کنید .
- حال، اسم را به StoreItem تغییر داده و سپس Add را کلیک کنید .
- در فایل دستورات زیر را وارد کنید .

?

```
1
2public class StoreItem
3{
4      public long itemNumber;
5      public string itemName;
6      public string size;
7      public decimal unitPrice;
8}
9
```

سازنده ی (constructor) پیش فرض

سازنده متدی خاص است که هنگام به وجود آمدن شی جدید (object)ساخته می شود. متد و کلاس هر دو یک اسم دارند. لازم به ذکر است متد، شی را درست در زمان به وجود آمدن آن مقداردهی اولیه می کند. هنگامی که کلاسی می سازید، چنانچه سازنده ای برای آن معرفی نکنید، compilerبه صورت خودکار یک سازنده برای شما ایجاد می کند؛ این کار باعث می شود که کلیه ی شی های کلاس از وجود شی جدید آگاه شوند. به سازنده ای که توسط compiler ایجاد می شود، سازنده ی پیش فرض نیز می گویند. البته شما می توانید سازنده ی خود را ایجاد کنید. برای این منظور

متدی معرفی کنید که با کلاس هم نام است.

داخل کلاس موردنظر راست کلیک کرده ...Insert Snippet را انتخاب کنید، سپس Visual C# را دوبار کلیک کنید.حال، در لیستی که ظاهر می شود روی ctor دوبار کلیک کنید.

Code Editorبرای ایجاد متد جدید از اسم کلاس استفاده می کند

به خاطر داشته باشید که متد فوق نباید هیچ مقداری بازگرداند .

مثال :

?

```
public class BankAccount
{
    public BankAccount()
    {
    }
}
```

زمانی که نمونه ای از کلاس تعریف می کنید، (چه از آن شی استفاده بکنید چه نکنید)، سازنده ای برای شی ایجاد می شود. پس از این که نمونه ای از کلاس تعریف شد، سازنده ی پیش فرض فراخوانده می شود (چه شی نام برده را به کار ببرید و چه آن را نادیده بگیرید). مثال زیر به زیبایی عملیات ذکر شده را تشریح می کند .

?

```

public class BankAccount
{
    public BankAccount()
    {
        System.Console.WriteLine("New Bank Account");
    }
}
public class Exercise
{
    static int Main()
    {
        BankAccount account = new BankAccount();

        return 0;
    }
}

```

نتیجه

?

```
New Bank Account
Press any key to continue...
```

همان طور که در مثال فوق مشاهده کردید، اگرچه شی مذکور مورد استفاده قرار نگرفت، تعریف آن به تنهایی وجود شی جدید را نشان داد .

به کاربردن سازنده ی پیش فرض

- در پنجره یSolution Explorer ، راست کلیک کرده، سپس DepartmentStore3 ->Add ->New Item...
- در لیست میانی پنجره ی محاوره یAdd New Item ، روی Code Fileکلیک کنید. حال، اسم را به DepartmentStoreتغییر داده و Add را کلیک کنید

- فایل را به صورت زیر اصلاح کنید .

?

```
1
2
3public class DepartmentStore
4{
5        static int Main()
6        {
7           StoreItem si = new StoreItem();
8
9            System.Console.WriteLine("Department Store");
10           System.Console.Write("Item #:    ");
11           System.Console.WriteLine(si.itemNumber);
12           System.Console.Write("Item Name: ");
13          System.Console.WriteLine(si.itemName);
14           System.Console.Write("Item Size: ");
15           System.Console.WriteLine(si.size);
16           System.Console.Write("Unit Price: ");
17           System.Console.WriteLine(si.unitPrice);
18           System.Console.ReadKey();
19           return 0;
        }
}
```

- برنامه را اجرا کنید. این نتیجه را به دست می دهد.

?

```
1Department Store
2Item #:    0
3Item Name:
4Item Size:
5Unit Price: 0
```

- پنجره ی DOS را بسته و به محیط برنامه نویسی باز گردید.

سازنده ای که مقداردهی اولیه می کند

می توان از سازنده (constructor) برای مقداردهی اولیه ی فیلدهای یک کلاس نیز استفاده کرد. به همین دلیل، می تواند جایگزین مناسبی برای متدی باشد که مقداردهی اولیه می کند. برای استفاده از سازنده به منظور مقداردهی اولیه ی فیلدهای یک کلاس، متغیرهایی را که می خواهید مقداردهی اولیه کنید را به عنوان آرگومان تعریف کنید. نیازی نیست تمامی متغیرهای عضو را تعریف کنید، بلکه تنها آن دسته متغیرهایی که ملزوم به مقداردهی اولیه هستید کفایت می کند. در حقیقت، تنها باید آن عضوهایی را مقداردهی (اولیه) کنید که مطمئن هستید، دیگر شی ها به هنگام استفاده از این شئ به آن ها نیاز دارند. به عبارت روشن تر، objectشما فیلدهایی دارد که اشیا خارجی (external objects)نیاز ندارند تغییر دهند (یا به آن ها دسترسی پیدا کنند) یا متغیرهای عضو بعده ها هنگامی که از شئ مورد نیاز فراخوانده می شوند، مقداردهی اولیه می شوند .

به منظور ایجاد سازنده ی پیش فرض، می توانید فقط فیلدهای دلخواه کلاس را مقداردهی اولیه کنید. برای متغیری از نوع عددی، کافی است ثابت دلخواه را به هر متغیر عضو اختصاص دهید. در صورتی که متغیر یک کاراکتر بود، فقط یک تک کوتیشن (')به آن اضافه می کنیم. حال، چنانچه متغیر یک رشته بود، باید مقداری با (") دابل کوتیشن به متغیر اختصاص داد .

?

```
1
2
3public class BankAccount
4{
5    private string customerName;
6    private decimal originalDeposit;
7    public BankAccount()
8    {
9        customerName = "John Doe";
10       originalDeposit = 0M;
11   }
12   public void Show()
13   {
14       System.Console.WriteLine("Customer Account Information");
15       System.Console.Write("Customer Name:    ");
16       System.Console.WriteLine(customerName);
17       System.Console.Write("Original Deposit: ");
18       System.Console.WriteLine(originalDeposit);
19   }
20}
21public class Exercise
22{
23   static int Main()
24   {
25       BankAccount account = new BankAccount();
26       account.Show();
27       return 0;
28   }
29}
30
31
32
```

مثالی از برنامه در حال اجرا

?

```
1Customer Account Information
2Customer Name:    John Doe
3Original Deposit: 0
4Press any key to continue...
```

- فهرست گزینه ی اصلی را باز کرده، روی گزینه ی Window ->StoreItem.cs کلیک کنید.
- فایل را به صورت زیر اصلاح کنید.

?

```
1public class StoreItem
2{
3       public long itemNumber;
       public string itemName;
```

```
4         public string size;
5         public decimal unitPrice;
6         public StoreItem()
7         {
8             itemNumber = 0;
9             itemName = "Unknown";
10            size = "0";
11            unitPrice = 0.00M;
12        }
13    }
14
```

- 3.برنامه را اجرا کنید. نتیجه ی زیر حاصل می گردد.

?

```
1Department Store
2Item #:    0
3Item Name: Unknown
4Item Size: 0
5Unit Price: 0.00
```

- پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید.

اضافه بارگذاری سازنده (constructor overloading)

مناسب ترین مکان برای تعریف اولیه ی متغیر ها درون سازنده می باشد (مناسب ترین جا برای تخصیص مقادیر پیش فرض به اعضای کلاس، سازنده ی پیش فرض می باشد). علاوه بر سازنده ی پیش فرض، می توانید هر تعداد سازنده که ملزوم می دانید اضافه کنید. این جنبه به شما اجازه می دهد برای اهداف مختلف، سازنده های متفاوت تولید کنید. بنابراین، یک سازنده نیز (مثل متد) امکان overload شدن دارد.

در مورد کلیه ی قواعد اضافه بارگذاری متد قبلاً مفصل بحث کردیم. ابتدایی ترین سازنده ای که ایجاد می کنید، قابلیت استفاده از یک آرگومان را دارد. هنگام ایجاد سازنده ای که تنها یک آرگومان می گیرد، باید آن عضوی را مقداردهی اولیه کنید که با آرگومان مذکور مطابقت دارد، و دیگر اعضا را با مقادیر پیش فرض مقداردهی کنید. به مثال زیر توجه کنید.

?

```
1public class BankAccount
2{
3     private string accountNumber;
4     private string customerName;
5     private decimal originalDeposit;
6     public BankAccount(string number)
7     {
8         accountNumber = number;
9         customerName = "John Doe";
10        originalDeposit = 0M;
11    }
    public void Show()
    {
```

```
        System.Console.WriteLine("Customer Account Information");
       System.Console.Write("Account N#:      ");
        System.Console.WriteLine(accountNumber);
        System.Console.Write("Customer Name:   ");
        System.Console.WriteLine(customerName);
        System.Console.Write("Original Deposit: ");
        System.Console.WriteLine(originalDeposit);
    }
}
```

در صورتی که کلاسی با تنها یک سازنده ایجاد کردید (مانند مثال فوق)، باید هنگام معرفی نمونه ای از کلاس، سازنده ی نام برده را به کار ببرید: توجه داشته باشید که نمی توان از سازنده ی پیش فرض ای که آرگومان نمی گیرد استفاده کرد. هنگام تعریف متغیر، آن را با سازنده مقداردهی اولیه کنید، سپس مقادیر را داخل پرانتز سازنده قرار دهید.

?

```
public class BankAccount
{
        private string accountNumber;
        private string customerName;
        private decimal originalDeposit;
        public BankAccount(string number)
        {
            accountNumber = number;
            customerName = "John Doe";
            originalDeposit = 0M;
        }
        public void Show()
        {
            System.Console.WriteLine("Customer Account Information");
            System.Console.Write("Account #:       ");
            System.Console.WriteLine(accountNumber);
            System.Console.Write("Customer Name:   ");
            System.Console.WriteLine(customerName);
           System.Console.Write("Original Deposit: ");
            System.Console.WriteLine(originalDeposit);
        }
}
public class Exercise
{
        static int Main()
        {
            BankAccount account = new BankAccount("27-940025-17");
            account.Show();
            return 0;
        }
}
```

نتیجه ی ذیل حاصل می گردد.

?

```
Customer Account Information
Account #:       27-940025-17
Customer Name:   John Doe
```

```
Original Deposit: 0
Press any key to continue...
```

به همین ترتیب، می توان برای مقداردهی های اولیه ی متفاوت، سازنده های متفاوت ایجاد کرد، البته این امکان وجود ندارد که برای هر متغیر سازنده ای متفاوت ایجاد کرد. چنانچه سازنده ای متفاوت با آرگومان های مختلف (برای مقداردهی اولیه) ایجاد کنید، هنگام تعریف متغیرهای کلاس، دقت کنید که هر نمونه را با تعداد صحیح آرگومان مقداردهی اولیه کنید، در غیر این صورت، compilerایراد می گیرد .

چنانچه کلاسی با تنها یک سازنده ایجاد کنید، و آن سازنده نیز حداقل یک آرگومان داشته باشد، سازنده ی پیش فرض دیگر قابل استفاده و در دسترس نخواهد بود. برای دسترسی به سازنده ی پیش فرض یک شی، دو گزینه پیش رو دارید.

- در صورتی که هیچ سازنده ای در کلاس ایجاد نکنید، سازنده ی پیش فرض همیشه هنگام فراخوانی کلاس آماده هست .
- اگر حداقل یک سازنده برای کلاس ایجاد کنید و حداقل یک آرگومان برای سازنده ی مزبور فراهم کنید، باید خودتان (صراحتا) یک سازنده ی پیش فرض برای کلاس ایجاد کنید .

اضافه بارگذاری سازنده

- به فهرست گزینه ی اصلی مراجعه کرده، روی گزینه های Window ->StoreItem.cs کلیک کنید .
- فایل را به صورت زیر اصلاح کنید .

?

```
public class StoreItem
{
    public long itemNumber;
    public string itemName;
    public string size;
    public decimal unitPrice;
    public StoreItem()
    {
        itemNumber = 0;
        itemName = "Unknown";
        size = "0";
        unitPrice = 0.00M;
    }

    public StoreItem(long number)
    {
        itemNumber = number;
        itemName = "Unknown";
        size = "0";
        unitPrice = 0.00M;
    }
    public StoreItem(long number, string name, string itemSize,
decimal price)
    {
        itemNumber = number;
        itemName  = name;
```

```
        size      = itemSize;
        unitPrice = price;
    }
}
```

- به فایل DepartmentStore.cs دسترسی پیدا کرده، سپس آن را به صورت زیر تغییر دهید.

?

```
public class DepartmentStore
{
    static int Main()
    {
        // Using the default contructor to create an unknown object
        StoreItem unknown = new StoreItem();
        System.Console.WriteLine("Department Store");
        System.Console.Write("Item #:    ");
        System.Console.WriteLine(unknown.itemNumber);
        System.Console.Write("Item Name: ");
        System.Console.WriteLine(unknown.itemName);
        System.Console.Write("Item Size: ");
        System.Console.WriteLine(unknown.size);
        System.Console.Write("Unit Price: ");
        System.Console.WriteLine(unknown.unitPrice);
        System.Console.WriteLine("=-------------------------------
----------=");

        // Using the contructor that takes one argument create an
object
        // whose only known information is the item number
       StoreItem knownNumber = new StoreItem(227174);
        System.Console.WriteLine("Department Store");
        System.Console.Write("Item #:    ");
        System.Console.WriteLine(knownNumber.itemNumber);
        System.Console.Write("Item Name: ");
        System.Console.WriteLine(knownNumber.itemName);
        System.Console.Write("Item Size: ");
        System.Console.WriteLine(knownNumber.size);
        System.Console.Write("Unit Price: ");
        System.Console.WriteLine(knownNumber.unitPrice);
        System.Console.WriteLine("=-------------------------------
----------=");

        // Using the contructor that has all the information
        // necessary to create an object
        StoreItem complete = new StoreItem(180318,"V-Neck Cardigan
with Ruffle Trim", "M", 74);
        System.Console.WriteLine("Department Store");
        System.Console.Write("Item #:    ");
        System.Console.WriteLine(complete.itemNumber);
        System.Console.Write("Item Name: ");
        System.Console.WriteLine(complete.itemName);
        System.Console.Write("Item Size: ");
        System.Console.WriteLine(complete.size);
        System.Console.Write("Unit Price: ");
        System.Console.WriteLine(complete.unitPrice);
```

```
        System.Console.WriteLine("=----------------------------------
----------=");
        System.Console.ReadKey();
        return 0;
    }
}
```

- برنامه را اجرا کنید. نتیجه ی زیر به دست می آید .

?

```

Department Store
Item #:    0
Item Name: Unknown
Item Size: 0
Unit Price: 0.00
=----------------------------------------=
Department Store
Item #:    227174
Item Name: Unknown
Item Size: 0
Unit Price: 0.00
=----------------------------------------=
Department Store
Item #:    180318
Item Name: V-Neck Cardigan with Ruffle Trim
Item Size: M
Unit Price: 74
=----------------------------------------=

```

- پنجره ی DOS را بسته و به محیط برنامه نویسی بازگردید.

سازنده ای با مقادیر پیش فرض

به این خاطر که یک سازنده پیش از هر چیز یک متد است و می تواند آرگومان بگیرد،متعاقباً آرگومان های آن می توانند مقادیر پیش فرض به کاربرند .

به منظور تخصیص مقدار پیش فرض به آرگومان یک سازنده، مقدار مناسب را هنگام ایجاد سازنده به آرگومان اختصاص دهید.

مثال :

?

```
public class Rectangle
{
```

```
    private double len;
    private double hgt;

    public Rectangle(double side = 10.00D)
    {
    }
}
```

می توان از سازنده برای مقداردهی اولیه ی فیلدهای یک کلاس استفاده کرد.

اگر سازنده ای ایجاد کنید که یک آرگومان می گیرد، هنگام ایجاد نمونه ای از کلاس، آن تک سازنده هم به عنوان یک سازنده ی پیش فرض عمل می کند، هم به عنوان سازنده ای که یک آرگومان می گیرد. به این معنا که شما می توانید متغیری تعریف کرده و از سازنده ای استفاده کنید که پرانتزهای آن خالی است. به مثال زیر توجه کنید .

?

```
public class Rectangle
{
        private double len;
        private double hgt;

        // Length = Height: Square
        public Rectangle(double side = 10.00D)
        {
            len = side;
            hgt = side;
        }

        public void Describe()
        {
            System.Console.WriteLine("Square Characteristics");
            System.Console.Write("Side: ");
            System.Console.WriteLine(len);
        }
}
    public class Exercise
    {
        static int Main()
        {
            Rectangle rect = new Rectangle();
            rect.Describe();
            return 0;
        }
}
```

نتیجه ی زیر به دست می آید.

?

```
Square Characteristics
Side: 10
Press any key to continue...
```

به همین ترتیب می توانید سازنده های مختلفی ایجاد کنید که آرگومان های متفاوتی می گیرند، و برخی از آرگومان های ذکر شده می توانند مقادیر پیش فرض داشته باشند. سازنده های گوناگون می توانند آرگومان های از نوع متفاوت (مختلفی) داشته باشند، برخی آرگومان ها می توانند مقادیر پیش فرض داشته باشند، در حالی که برخی دیگر (از این آرگومان ها) مقادیر پیش فرض نداشته باشند. هنگام ایجاد شی، لازم است به نوع سازنده ای که به کار می برید دقت کنید. مثال زیر را در نظر بگیرید.

?

```
public class Rectangle
{
    private double len;
    private double hgt;

    public Rectangle(double length, double height = 10.00D)
    {
       len = length;
        hgt = height;
    }

    public void Describe()
    {
        System.Console.WriteLine("Square Characteristics");
        System.Console.Write("Side: ");
        System.Console.WriteLine(len);
    }

    public void Describe(int rect)
    {
        System.Console.WriteLine("Rectangle Characteristics");
        System.Console.Write("Length: ");
        System.Console.WriteLine(len);
        System.Console.Write("Height: ");
        System.Console.WriteLine(hgt);
    }
}

public class Exercise
{
    static int Main()
    {
       Rectangle rect = null;

        rect = new Rectangle(24.72);
        rect.Describe();

        rect = new Rectangle(24.72, 20.64);
        rect.Describe(1000);

        return 0;
    }
}
```

نتیجه ی زیر حاصل می شود.

?

```
Square Characteristics
Side: 24.72
Rectangle Characteristics
Length: 24.72
Height: 20.64
Press any key to continue...
```

استفاده از سازنده های یک کلاس

مخرب های کلاس(class destructor)

درست برخلاف سازنده ها، یک مخرب زمانی فراخوانده می شود که برنامه دیگر کارش با شی تمام شده و نیازی به آن ندارد. مخرب فرایند پاک سازی را پشته صحنه انجام می دهد. درست مثل سازنده ی پیش فرض، در صورتی که مخربی برای کلاس تعریف نکنید، compilerخود یک مخرب پیش فرض ایجاد می کند .برخلاف سازنده، مخرب را نمی توان اضافه بارگذاری (overload)کرد. به عبارت دیگر، شما نمی توانید بیش از یک مخرب داشته باشید. اما درست مثل سازنده، مخرب با کلاس هم اسم است. این بار اسم مخرب با علامت "~" آغاز می شود .

برای ایجاد مخرب، علامت ~ و به دنبال آن اسم کلاس را تایپ کنید.

مثال :

?

```
public class BankAccount
{
        private string accountNumber;
        private string customerName;
        private decimal originalDeposit;

        public BankAccount(string number)
        {
            accountNumber = number;
            customerName = "John Doe";
            originalDeposit = 0M;
        }

        public void Show()
        {
            System.Console.WriteLine("Customer Account Information");
            System.Console.Write("Account #:       ");
            System.Console.WriteLine(accountNumber);
            System.Console.Write("Customer Name:   ");
            System.Console.WriteLine(customerName);
            System.Console.Write("Original Deposit: ");
            System.Console.WriteLine(originalDeposit);
        }
```

```
20
21          ~BankAccount()
22          {
23          }
24  }
```

متغیرهای خواندنی (read-only variables)

هنگام ایجاد متغیر عضو کلاس، یکی از تصمیماتی که اتخاذ می کنید این است که فیلد کلاس چگونه مقدارش را دریافت کند. گاهی اوقات به آن اعضایی (clients of a class) که از کلاس استفاده می کنند، اجازه ی تغییر مقادیر فیلدها داده می شود. گاهی اوقات هم شما می خواهید فیلد، فقط مقدار مزبور را نگه دارد یا ارائه بدهد، ولی در عین حال نتواند آن را تغییر دهد .این کار باز هم به اعضا اجازه می دهد که به فیلد دسترسی پیدا کنند ولی تنها در حد خواندن آن .

به منظور ایجاد فیلدی که مقدار آن را فقط بتوان خواند، پیش از نوع داده ی آن باید کلیدواژه ی readonly را تایپ کرد. به مثال زیر توجه کنید.

?

```
1public readonly double PI;
```

پس از تعریف متغیر، لازم است آن را مقداردهی اولیه کنید. دو گزینه اصلی پیش رو دارید، می توانید فیلد را هنگام تعریف مقداردهی اولیه کنید.

مثال :

?

```
 public class Circle
1{
2       public double radius;
3
4       public Circle(double rad)
5       {
6           radius = rad;
7       }
8
9       public readonly double PI = 3.14159;
10  }
11
12     public class Exercise
13     {
14         static int Main()
15         {
16             var circ = new Circle(24.72);
17
18             System.Console.WriteLine("Circle Characteristics");
               System.Console.Write("Radius: ");
               System.Console.WriteLine(circ.radius);
               System.Console.Write("PI:     ");
```

```
            System.Console.WriteLine(circ.PI);

            return 0;
        }
}
```

نتیجه ی زیر را به دست می دهد.

?

```
Circle Characteristics
Radius: 24.72
PI:    3.14159
Press any key to continue...
```

گزینه ی دوم این است که فیلد را داخل سازنده ی یک کلاس مقداردهی اولیه کنید. این کار به صورت زیر انجام می پذیرد.

?

```

public class Circle
{
    public double radius;

    public Circle(double rad)
    {
        radius = rad;
        PI = 3.14159;
    }

    public readonly double PI;
}

public class Exercise
{
    static int Main()
    {
        var circ = new Circle(24.72);

        System.Console.WriteLine("Circle Characteristics");
        System.Console.Write("Radius: ");
        System.Console.WriteLine(circ.radius);
        System.Console.Write("PI:    ");
        System.Console.WriteLine(circ.PI);

        return 0;
    }
}
```

چنانچه مقدار یک فیلد خواندنی (read-only field) از یک عبارت گرفته شده باشد، فیلد نام برده باید در سازنده و با عبارت دلخواه مقداردهی اولیه شود. بر این اساس، کد زیر ترجمه (compile) نمی شود.

?

```
1
2
3                       .syntaxhighlighter {
4public class Circle
5{
6    public double radius;
7
8    public Circle(double rad)
9    {
10       radius = rad;
11       PI = 3.14159;
12   }
13
14   public readonly double PI;
15   public readonly double Diameter = radius * 2;
16}
17public class Exercise
18{
19   static int Main()
20   {
21       var circ = new Circle(24.72);
22
23       System.Console.WriteLine("Circle Characteristics");
24       System.Console.Write("Radius:   ");
25       System.Console.WriteLine(circ.radius);
26       System.Console.Write("PI:       ");
27       System.Console.WriteLine(circ.PI);
28       System.Console.Write("Diameter: ");
29       System.Console.WriteLine(circ.Diameter);
30
31       return 0;
   }
}
```

نتیجه ی زیر به دست می آید.

?

```
Error          1        A field initializer cannot reference the non-
1static
2or property 'Circle.radius'     ،method،field
3C:\Exercise1\Exercise.cs              12        39        Bank
```

یکی از راه حل های رفع خطای فوق، این است که فیلد مزبور را به عنوان متغیری فقط خواندنی (read-only) در کلاس تعریف کرده، سپس آن را در سازنده و با عبارت موردنظر مقداردهی اولیه کنید . نمونه های آن را زیر مشاهده می کنید.

?

```
1
2
3
4  public class Circle
5  {
6      public double radius;
7
8      public Circle(double rad)
9      {
10         radius = rad;
11         Diameter = radius * 2;
12         Circumference = Diameter * PI;
13         Area = radius * radius * PI;
14     }
15
16     public readonly double PI = 3.14159;
17     public readonly double Diameter;
18     public readonly double Circumference;
19     public readonly double Area;
20 }
21 public class Exercise
22 {
23     static int Main()
24     {
25         Circle circ = new Circle(24.72);
26
27         System.Console.WriteLine("Circle Characteristics");
28         System.Console.Write("Radius:       ");
29         System.Console.WriteLine(circ.radius);
30         System.Console.Write("Diameter:     ");
31         System.Console.WriteLine(circ.Diameter);
32         System.Console.Write("Circumference: ");
33         System.Console.WriteLine(circ.Circumference);
34         System.Console.Write("Area:         ");
35         System.Console.WriteLine(circ.Area);
36
         return 0;
     }
 }
```

نتیجه ی زیر را به دست می دهد.

?

```
1 Circle Characteristics
2 Radius:       24.72
3 Diameter:     49.44
4 Circumference: 155.3202096
5 Area:         1919.757790656
6 Press any key to continue...
```

می دانیم که یک متغیر ثابت (constant variable) را باید به محض ایجاد کردن، مقداردهی کرد. لازم به ذکر است که متغیر (فقط) خواندنی از این قاعده پیروی نمی کند. به خاطر داشته باشید که نیازی نیست متغیر خواندنی را هنگام ایجاد آن مقداردهی

اولیه کرد، زیرا این کار داخل سازنده ی کلاس صورت می گیرد. هم چنین به دلیل این که سازنده قابلیت overload شدن را دارد، می تواند مقادیر متفاوتی داشته باشد، اما مقدار متغیر ثابت هیچ گاه تغییر نمی کند : یک بار در کلاس (یا متد) مقداردهی می شود و آن مقدار را برای همیشه سرتاسر کلاس (یا متد) نگه می دارد (در تمام بخش های کلاس از همان مقدار استفاده می کند .(