الفصل الرابع

البرمجة بلغة VB .NET باستخدام مكتبة الأصناف الجاهزة

VB .NET PROGRAMMING WITH SUPPLIED CLASSES

أهداف الفصل:

- استخدام مكتبة الصنف String.
- استخدام مكتبة الأصناف الجاهزة.
- إنشاء مصفوفات من النوع String.
- إنشاء مصفوفات الصنف ArrayList.
- التعامل مع التواريخ.
- تنسيق الأرقام.
- استخدام الصنف MessageBox.
- إظهار النماذج.

لقد تعرض الفصل السابق إلى أساسيات لغة VB .NET وأسلوب كتابة أوامرها مشتملاً على كيفية تعريف متغيرات وإسناد قيم ابتدائية لها وكتابة التعبيرات الحسابية وكذلك كيفية كتابة كل من الأوامر الشرطية وأوامر التكرار والتعامل مع المصفوفات.

وقد رأينا مدى سهولة كتابة أوامر هذه اللغة والذي يرجع أيضاً إلى وجود كم هائل من الأصناف المعرفة سابقاً داخل اللغة والتي تحتوي على دوال عديدة يمكن استدعائها مباشرة، حيث يركز هذا الفصل على بعض هذه الأصناف وسوف يتعلم القارئ الكثير عن الصنف String (السلاسل النصية) وكيفية إنشاء مصفوفة من كائنات هذا الصنف.

ثم يتعرض الفصل بعد ذلك إلى الصنف ArrayList الذي يُمكّن المبرمج من إنشاء مصفوفة متغيرة الحجم تلقائياً. وبعد ذلك نستكشف سوياً كيفية تنسيق البيانات (النصية والرقمية) لإنشاء مخرجات مقروءة وجذابة.

أيضاً سوف نتعلم كيفية التعامل مع التاريخ من خلال الصنف Date وكيفية إنشاء رسائل متنوعة للمستخدم من خلال الصنف MessageBox والتعرف على استجابة المستخدم لهذه الرسائل.

بعد الانتهاء من هذا الفصل سنرى أهمية استخدام مكتبة الأصناف الجاهزة التي توفر لنا العديد من الدوال سابقة التعريف، أما باقي هذا الكتاب فسوف يركز على كيفية تعريف أصناف جديدة خاصة بك وإنشائها والتي تمثل مكونات التطبيقات المعتمدة على الكائنات.

استخدام المسميات والمجموعات المقدمة مع لغة VB .NET

Using the Namespaces and Classes Supplied with VB .NET

تعتبر لغة VB .NET جزءاً من فيجول ستوديو .نت لمايكروسوفت والتي تعتبر نفسها جزءاً من تقنية أكبر يطلق عليها (تقنية .نت). تحتوي فيجول ستوديو .نت على العديد من لغات البرمجة مثل لغة C# .NET، ولغة C++ .NET، ولغة VB .NET، كما تحتوي فيجول ستوديو .نت على إطار عمل .نت (.NET Framework) الذي يتكون من محرك اللغة المشترك (Common Language Runtime) والذي يتم الإشارة إليه مستخدماً الاختصار CLR وهو المسؤول عن تقديم مجموعة من الخدمات لجميع مبرمجي اللغات السابقة الذكر مثل تقديم أنواع بيانات (Data types) موحدة وتقديم مقاييس موحدة لجميع اللغات. كذلك يحتوي إطار عمل .نت على مكتبة كبيرة من الأصناف التي يمكن استخدامها بواسطة لغات البرمجة التي تعتمد على CLR، ولأن مكتبة .نت تشتمل على كم هائل من الأصناف فقد تم تقسيم هذه الأصناف إلى مجموعات يطلق عليها مجموعات معرفة (Namespaces) حيث تحتوي كل مجموعة على العديد من الأصناف المرتبطة بعلاقة ما. ويمكنك استخدام الكلمة المحجوزة Imports لكي تجعل المترجم يتعامل مع أصناف موجودة في مجموعة معرفة معينة، كما يمكنك إسناد أصناف داخل المجموعة المعرفة التي تقوم بتعريفها. يوضح الجدول رقم (٤.١) أمثلة لعدد من المجموعات المعرفة (Namespaces) مع بعض الأصناف الموجودة داخل كل مجموعة معرفة.

سيتم التعرض لأول مجموعتين معرفتين في هذا الجدول داخل هذا الفصل هما المجموعة System والمجموعة System.Collections مع العلم أن مترجم اللغة يقوم بتحميل المجموعة System بشكل تلقائي، وكما ذكرنا سابقاً إذا أردت أن تتعامل مع أصناف أي مجموعة معرفة أخرى فلابد من استخدام الكلمة المحجوزة Imports متبوعة باسم المجموعة المعرفة المراد التعامل معها وإذا لم تفعل ذلك فإن المترجم سيعطي رسالة خطأ عند كتابة اسم أحد أصناف هذه المجموعة دون استخدام هذه الكلمة لعدم قدرته على إيجاده.

الجدول رقم (٤,٩). بعض المجموعات المعرفة وبعض الأصناف التي تحتوي عليها.

Namespace	Selected Classes	Discussed in Chapter(s)
System	Array Console Convert DateTime Exception TimeSpan String Math	2, 3, 4
System.Collections	ArrayList	4
System.IO	StreamReader StreamWriter	13
System.Data	DataRow DataTable DataSet	13, 14, 15
System.Data.OleDb	OleDbCommand OleDbConnection OleDbDataAdapter OleDbParaneter	13, 14, 15
System.Windows.Forms	Button CheckBox Form Label Menu MenuItem RadioButton TextBox	10, 11, 15
System.Web	System.data System.data.oledb System.Collections System.Web System.Web.UI System.Web.UI.WebControls System.Web.Services	16

سنتعرض للمجموعة المعرفة System.Windows.Forms داخل كل من الفصل العاشر والحادي عشر لتطوير أصناف الواجهة الرسومية لتطبيقات الويندوز، أما أصناف كل من المجموعة System.Data والمجموعة System.IO والمجموعة System.Data.OleDb المسؤولة عن التعامل مع قواعد البيانات فسيتم شرحها داخل كل من الفصل الثالث عشر والفصل الرابع عشر، أما الفصل السادس عشر فسوف تتعرض للمجموعة المعرفة System.Web التي تحتوي على أصناف للتعامل مع نشر مواقع الويب.

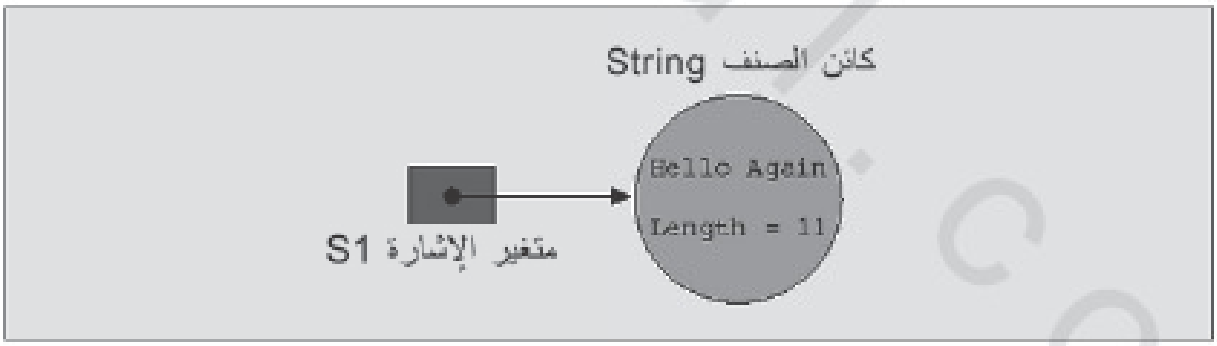
استخدام الصنف String
Using the String Class

يتكون نوع البيانات String من مجموعة من الحروف (Characters) ولإنشاء متغير من هذا النوع تستخدم لغة VB .NET الصنف String لإنشاء كائن وتخزين هذه الحروف بداخله.

يوجد الصنف String داخل المجموعة المعرفة System، وكما رأينا في الفصل السابق أنه من الممكن تعريف متغير من النوع String بطريقة تتشابه مع تعريف أي متغير أولي:

```
Dim s1 As String = "Hello Again"
```

الأمر السابق يقوم بإنشاء كائن (Object) من الصنف String ويطلق على هذه العملية اسم Instantiating بمعنى إنشاء نسخة (Instance) من هذا الصنف حيث يقوم هذا الأمر بتنفيذ ثلاث مهام: إنشاء متغير من النوع String ثم إنشاء كائن (نسخة) من الصنف String وتخزين النص "Hello Again" داخل هذا الكائن وأخيراً إسناد عنوان هذا الكائن للمتغير. واستخدام النوع String لتعريف المتغير s1 يعني أن المتغير s1 سوف يشير إلى كائن من الصنف String ويطلق على المتغير s1 اسم متغير إشارة (Reference Variable). ويجب التنويه إلى أن البرمجة المعتمدة على الكائنات OO Programming تحتوي على الكثير من عملية إنشاء (Instantiating) كائنات من أصناف، ويوضح الشكل رقم (٤,١) على الكائن الذي تم إنشاؤه في الأمر السابق.



الشكل رقم (٤,١). كائن من الصنف String.

يحتوي الصنف String على العديد من الإجراءات (Methods) والخصائص (Properties) مثلها مثل أي صنف آخر. وتمثل الخاصية نوعاً خاصاً من الدوال والتي تبدو كصفة (Attribute) للكائن حيث يمكن التعامل معها مثل التعامل مع أي متغير عام (إسناد قيمة له أو استرجاع قيمة منه). ويصف الفصل السادس كيفية تعريف الخصائص

بشيء من التفصيل وكيفية كتابتها. يسرد الجدول رقم (٤,٢) دوالاً وخصائص مختارة من الصنف String، ويجب أن تلاحظ أن جميع الدوال تم استدعائها من خلال متغير إشارة (المتغير s) عدا الدالة Copy التي تم استدعائها من خلال اسم الصنف مباشرة String.

الجدول رقم (٤,٢). بعض إجراءات الصنف String وخصائصه.

Method/Property Name	Description
`s.Chars(i)`	Method gets the character in the `String` instance `s` at index `i` (relative to zero)
`String.Copy(s)`	Method returns a new `String` instance, which is a copy of `s`
`s.EndsWith("a")`	Method returns Boolean `True` if the `String` instance `s` ends with the string value "a"; otherwise returns `False`
`s.Equals(s1)`	Method compares values character-by-character in `String` instances `s` and `s1` and returns either `True` or `False`
`s.IndexOf("a")`	Method returns the index (relative to zero) of the first occurrence of the character "a" in `String` instance `s`. A return of –1 means not found. A value of 0 means `s` contains nothing
`s.Insert(i, "a")`	Method inserts the string value "a" into `s` beginning at index `i` (relative to zero) and returns the new `String` instance
`s.Length`	Property gets the number of characters in the string, its length
`s.Replace("a","b")`	Method searches `s` for the string value "a", replaces it with the string "b", and returns the new `String` instance
`s.Split("a")`	Method returns a String array containing substrings separated by "a"
`s.StartsWith("a")`	Method returns Boolean `True` if `String` instance `s` begins with the string value "a"; otherwise returns `False`
`s.Substring(i,j)`	Method returns a new `String` instance containing the string of characters in instance `s` beginning at index `i` and ending at `j` (relative to zero)
`s.ToUpper()`	Method returns a new `String` instance containing the contents of instance `s` converted to uppercase
`s.ToLower()`	Method returns a new `String` instance containing the contents of instance `s` converted to lowercase

سنوضح في المناقشة والأمثلة التالية استخدام الدوال المذكورة داخل الجدول السابق، وتتميز كائنات الصنف String بصفتين هامتين وهما:

١- أن قيم كائنات الصنف String غير قابلة للتغير (Immutable)، وهذا يعني أن الدوال التي تبدو دوال تغيير مثل الدالة Insert، والدالة Replace، والدالة ToUpper في الحقيقة تنشئ كائناً جديداً معدلاً وتعيده.

٢- أن كل حرف داخل النص في كائنات الصنف String له فهرس خاص به والذي يشير إلى موقعه، وكما رأينا في الفصل الثالث أن كل عنصر داخل المصفوفة له فهرس خاص به ابتداء من القيمة صفر، وهذا يعني أن أول حرف داخل النص يأخذ الفهرس صفر، والثاني يأخذ الفهرس ١، وهكذا. سابقاً تم إنشاء متغير إشارة من النوع String وأسندنا له القيمة النصية "Hello Again" وذلك بكتابة الأمر التالي:

```
Dim s1 As String = "Hello Again"
```

يوضح الشكل رقم (٤.٢) قيم الفهارس لحروف النص "Hello Again".

H	e	l	l	o		A	g	a	i	n
0	1	2	3	4	5	6	7	8	9	10

الشكل رقم (٤.٢). قيم فهارس الكائن s1 من الصنف String.

لاحظ الخاصية Length في الشكل رقم (٤.١) والتي تحتوي على عدد حروف النص (١١ حرفاً في هذا المثال)، ويمكن التعامل مع محتوى هذه الخاصية بكتابة متغير الإشارة متبوعة بكلمة Length:

```
' Length property contains the number of characters
Console.WriteLine("length of s1 is " & s1.Length)
```

استخدام إجراءات الصنف String

Using String Methods

الإجراء Copy

Copy Method

افترض أنك تريد إنشاء نسخة من الكائن s1 (من النوع String) والإشارة إليه بمتغير إشارة آخر s2، ولعمل ذلك يتم استدعاء الإجراء Copy المسؤول عن إنشاء نسخة من كائن من النوع String المرسل إليها، وإعادة هذه النسخة كما هو واضح في المثال التالي:

```
' create a copy of s1
Dim s2 As String = String.Copy(s1)
Console.WriteLine("s2, a copy of s1, contains " & s2)
```

أما إذا كتبت الأمر التالي، فذلك يعني إنشاء متغير إشارة آخر (وليس إنشاء كائن من النوع String) ليشير إلى نفس الكائن المشار إليه بواسطة المتغير s1، وبمعنى آخر أن كلا المتغيرين s1 و s2 يشاوران على نفس الكائن الذي يحتوي على "Hello Again":

```
Dim s2 As String = s1   ' assign the contents of s1 to s2
```

عند استدعاء الإجراء Copy يتم إنشاء نسخة من الكائن المشار إليه بواسطة المتغير s1 ليشار إليه بواسطة المتغير s2.

يوجد خطأ شائع يقع فيه مبرمجو VB .NET وهو استدعاء إجراء من متغير إشارة يشير إلى لا شيء (أي يحتوي على Nothing)، وبمعنى آخر يتم استدعاء الإجراء من متغير إشارة لم يحتوي على عنوان كائن (أي لا يشير إلى كائن). وعند حدوث هذا الخطأ تُنهي لغة VB .NET تنفيذ البرنامج وإظهار رسالة تفيد بحدوث خطأ من النوع Null Reference Exception (بمعنى التعامل مع Null). فعند ظهور هذه الرسالة لابد من إضافة أوامر لإسناد كائن لمتغير الإشارة المتسبب في الخطأ.

الإجراء Chars
Chars Method

يستخدم هذا الإجراء لإعادة الحرف المناظر لقيمة الفهرس الممررة لهذا الإجراء. ويستدعي الأمر التالي الإجراء Chars من الكائن المشار إليه بالمتغير s1 وتمرير القيمة ٦ لها ثم استقبال الحرف المناظر (الحرف A):

```
Console.WriteLine("char at index 6 of s1 is " & s1.Chars(6))
```

الإجراء Equals
Equals Method

كما رأيت في الفصل الثالث، إن المعامل "=" يستخدم للمقارنة بين قيمتين، ويمكن أن يستخدم هذا المعامل أيضاً للمقارنة بين قيمتين من النوع String. ويوضح المثال التالي مقارنة المتغير s1 والمتغير s2 مستخدماً المعامل "="، وبما أن المتغير s2 تم إنشاؤه كنسخة من s1 فإن القيمتين متساويان:

```
If s1 = s2 Then Console.WriteLine("s1 = s2")
```

يمكن أيضاً استخدام الإجراء Equals بدلاً من المعامل "=" للمقارنة بين قيمتين من النوع String (كما هو واضح في الأمر التالي)، وفي الحقيقة إن مترجم لغة VB .NET يقوم تلقائياً باستبدال معامل المقارنة يساوي "=" بالإجراء Equals عند ترجمة البرنامج. ولاحظ أن لغة VB .NET حساسة لحالة الأحرف عند إجراء المقارنة:

```
Console.WriteLine("s1.Equals(s2) returns " & s1.Equals(s2))
```

الإجراء Substring

Substring Method

يستخلص الإجراء Substring حرفاً أو أكثر من كائن من النوع String وإرجاع كائن آخر من النوع String يحتوي على النص المستخلص. ويستقبل الإجراء Substring فهرس الحرف الأول وعدد الحروف المراد استخلاصها. يستخدم المثال التالي لاستخلاص خمسة حروف بداية من الحرف الأول:

```
Console.WriteLine("s1.Substring(0, 5) returns " & s1.Substring(0, 5))
```

الإجراء Replace

Replace Method

يستخدم الإجراء Replace باستبدال حرف أو أكثر في كائن من النوع String بحرف أو أكثر، ويستقبل الإجراء Replace سلسلة الحروف المراد استبدالها كمعامل أول وسلسلة الحروف الجديدة كمعامل ثاني. يوضح المثال التالي استبدال الكلمة "Hello" بالكلمة "Hi" في الكائن s1. تذكر أن كائنات String غير قابلة للتغيير (Immutable)، لذلك فإن الإجراء Replace ينشئ كائناً معدلاً من النوع String ثم يعيده. في هذه الأمثلة نجد أن كل كائنات String المسترجعة يتم تمريرها للدالة WriteLine.

```
Console.WriteLine("s1.Replace(Hello, Hi) returns " & _
      s1.Replace("Hello", "Hi"))
Console.WriteLine("After Replace s1 contains " & s1)
```

الإجراء Insert

Insert Method

يستخدم الإجراء Insert لإدراج حرف أو أكثر داخل سلسلة نصية بداية من فهرس محدد، حيث يوضح الأمر التالي لإدراج الكلمة "There" داخل الكائن s1 بداية من الفهرس السادس (أي بعد كلمة Hello):

```
Console.WriteLine("s1.Insert(6, There ) returns " & _
      s1.Insert(6, "There "))
```

الإجراء StartsWith والإجراء EndsWith

EndsWith and StartsWith Methods

يستخدم الإجراء StartsWith بمقارنة سلسلة حروف مع بداية حروف كائن String وإعادة True أو False بناء على نتيجة المقارنة. أما الإجراء EndsWith فيقارن سلسلة الحروف الممررة إليه بنهاية سلسلة حروف الكائن وإعادة

True أو False. يستدعي الأمر الأول في البرنامج التالي الإجراء StartsWith لمعرفة هل سلسلة الحروف المخزنة في الكائن s1 تبدأ بالكلمة "Hi" أم لا ، ولأن s1 يحتوي على "Hello Again" لذلك فإن هذا الإجراء سيعيد القيمة False. أما الأمر الثاني فيستدعي الإجراء EndsWith لمعرفة هل سلسلة الحروف المخزنة في s1 تنتهي بالكلمة "Again" والتي ستعيد القيمة True:

```
Console.WriteLine("s1.StartsWith(Hi) returns " & s1.StartsWith("Hi"))
Console.WriteLine("s1.EndsWith(Again) returns " & s1.EndsWith("Again"))
```

الإجراء ToUpper والإجراء ToLower
ToLower and ToUpper Methods

ولتحويل سلسلة حروف إلى حروف كبيرة (Capital) يستخدم الإجراء ToUpper ، ولتحويل الحروف إلى حروف صغيرة (Small) يستخدم الإجراء ToLower ، حيث يعيد الإجراء ToUpper سلسلة الحروف الأصلية بعد تحويلها إلى حروف (Capital) والإجراء ToLower يعيد الحروف الأصلية بعد تحويلها إلى حروف (Small) ، فعلى سبيل المثال يحول الأمر التالي محتويات الكائن s1 إلى حروف (Capital):

```
Console.WriteLine("s1 uppercase is " & s1.ToUpper())
```

الإجراء IndexOf
IndexOf Method

يستخدم الإجراء IndexOf للبحث عن قيمة نصية داخل محتويات كائن من النوع String ، ثم يعيد رقم فهرس أول حرف داخل الكائن أو يعيد ١- إذا لم يجد هذه القيمة. فعلى سبيل المثال يستخدم الإجراء IndexOf للبحث عن الكلمة "Again" داخل الكائن s1 والذي سيعيد القيمة ٦ (فهرس الحرف A داخل الكائن):

```
Console.WriteLine("s1.indexof(Again) returns " & s1.IndexOf("Again"))
```

إجراءات التحويل
Convert Methods

في بعض الأحيان نريد أن نقوم بتحويل قيمة رقمية إلى قيمة نصية. فعلى سبيل المثال لو كان لديك متغير صحيح (Integer) اسمه i والذي يخزن القيمة ٥ ، تستطيع أن تحول هذه القيمة إلى قيمة نصية بواسطة استدعاء الإجراء ToString من الصنف Convert ، ويوضح المثال التالي تحويل قيمة صحيحة إلى قيمة نصية:

```
' convert an integer to String
Dim i As Integer = 5
Console.WriteLine("value of Convert.ToString(i) is " & _
      Convert.ToString(i))
```

تستطيع أيضاً تحويل كائنات من النوع String إذا كانت تحتوي على أرقام إلى أنواع بيانات رقمية مثل Single وInteger وذلك بواسطة استدعاء إجراءات الصنف Convert، ويوضح المثال التالي تحويل محتويات الكائن s3 (من النوع String) الذي يحتوي على القيمة ٥ إلى قيمة من النوع Integer بواسطة الإجراء ToInt32 المعرف داخل الصنف Convert:

```
Dim s3 As String = "5"
Console.WriteLine("value of Convert.ToInt32(s3) is " & _
      Convert.ToInt32(s3))
```

يحتوي البرنامج StringDemo.vb على جميع الأمثلة التي تم شرحها في هذه الفقرة. ويعرض الشكل رقم (٤,٣) محتويات هذا البرنامج، ويوضح الشكل رقم (٤,٤) مخرجات هذا البرنامج.

```
' Chapter 4 StringDemo Example 1
Option Strict On
Module StringDemo
   Sub Main()

      Dim i As Integer
      ' create a String instance
      Dim s1 As String = "Hello Again"
      Console.WriteLine("s1 contains " & s1)

       ' Length property contains the number of characters
      Console.WriteLine("length of s1 is " & s1.Length)
      ' create a copy of s1
      Dim s2 As String = String.Copy(s1)
      Console.WriteLine("s2, a copy of s1, contains " & s2)
      Console.WriteLine("char at index 6 of s1 is " & s1.Chars(6))
      If s1 = s2 Then Console.WriteLine("s1 = s2")
      Console.WriteLine("s1.Equals(s2) returns " & s1.Equals(s2))
      Console.WriteLine("s1.Substring(0, 5) returns " & _
         s1.Substring(0, 5))
      Console.WriteLine("s1.Replace(Hello, Hi) returns " & _
         s1.Replace("Hello", "Hi"))
      Console.WriteLine("After Replace s1 contains " & s1)
      Console.WriteLine("s1.Insert(6, There ) returns " & _
         s1.Insert(6, "There "))
      Console.WriteLine("s1.StartsWith(Hi) returns " & _
         s1.StartsWith("Hi"))
      Console.WriteLine("s1.EndsWith(Again) returns " & _
         s1.EndsWith("Again"))
      Console.WriteLine("s1 uppercase is " & s1.ToUpper())
      Console.WriteLine("s1.indexof(Again) returns " & _
         s1.IndexOf("Again"))
      ' convert an integer to String
      i = 5
      Console.WriteLine("value of Convert.ToString(i) is " & _
         Convert.ToString(i))

      Dim s3 As String = "5"
      Console.WriteLine("value of Convert.ToInt32(s3) is " & _
         Convert.ToInt32(s3))
   End Sub
End Module
```

الشكل رقم (٤,٣). شفرة البرنامج StringDemo.vb.

```
s1 contains Hello Again
length of s1 is 11
s2, a copy of s1, contains Hello Again
char at index 6 of s1 is A
s1 = s2
s1.Equals(s2) returns True
s1.Substring(0, 5) returns Hello
s1.Replace(Hello, Hi) returns Hi Again
After Replace s1 contains Hello Again
s1.Insert(6, There ) returns Hello There Again
s1.StartsWith(Hi) returns False
s1.EndsWith(Again) returns True
s1 uppercase is HELLO AGAIN
s1.IndexOf(Again) returns 6
value of Convert.ToString(i) is 5
value of Convert.ToInt32(s3) is 5
```

الشكل رقم (٤,٤). مخرجات البرنامج StringDemo.vb.

إنشاء مصفوفة من النوع String
Creating a String Array

لقد أنشأنا في الفصل الثالث مصفوفة من النوع Integer بواسطة الأمر التالي:

```
' declare an integer array with 5 elements
Dim testScores(4) As Integer
```

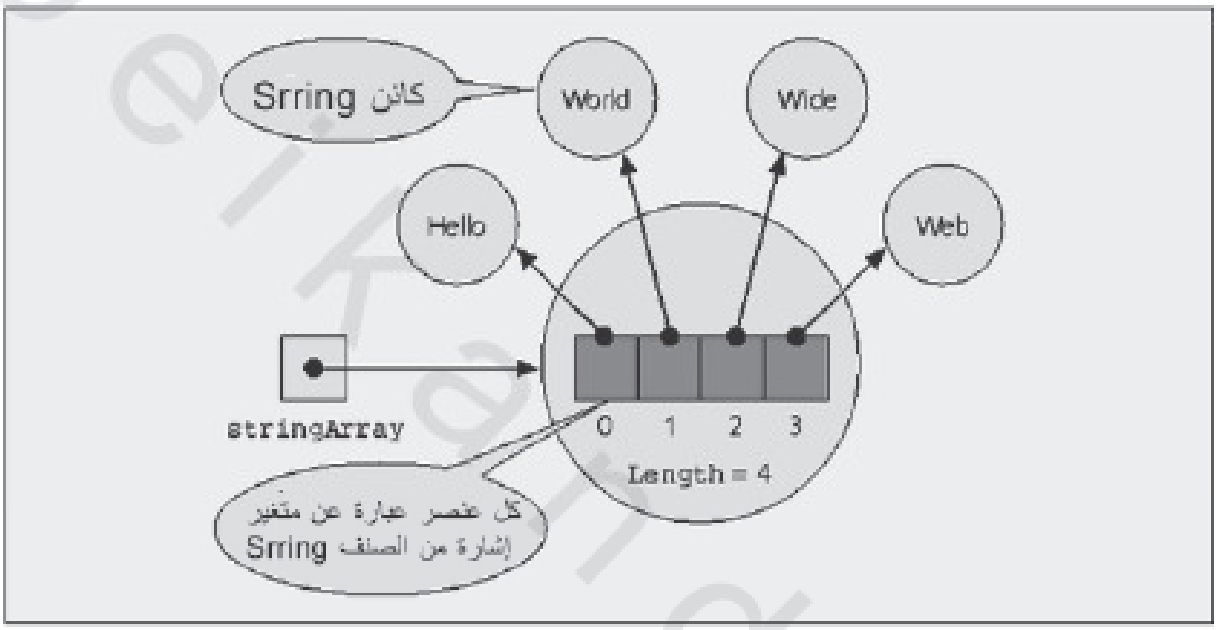
هذا الأمر ينشئ متغير الإشارة testScores ثم إنشاء كائن من النوع Array الذي يحتوي على خمس عناصر من النوع Integer، مع إسناد عنوان هذا الكائن داخل متغير الإشارة (الآن المتغير testScores يشاور على كائن من النوع Array). يمكن التعامل مع عناصر المصفوفة بواسطة الفهرس (تذكر أن فهرس عناصر المصفوفة يبدأ دائماً بصفر). وبنفس الأسلوب يمكن إنشاء مصفوفة من النوع String بواسطة الأمر التالي:

```
' declare an String array with 4 elements
Dim stringArray(3) As String
```

ينشئ هذا الأمر متغير الإشارة stringArray يشير على كائن من النوع Array الذي سوف يحتوي على أربعة عناصر، كل عنصر من هذه العناصر عبارة عن متغير إشارة سيشير إلى كائن من النوع String. لاحظ أن عناصر المصفوفة testScores عبارة عن متغيرات أولية (Primitive Variables) من النوع Integer، بينما عناصر المصفوفة stringArray عبارة عن متغيرات إشارة لكائنات من النوع String، ولاحظ أيضاً أنه عند تعريف المصفوفة يتم كتابة فهرس آخر عنصر في المصفوفة وليس عدد عناصر المصفوفة.

الأوامر التالية مسؤولة عن إنشاء كائنات من النوع String وإسنادها لعناصر المصفوفة. ويوضح الشكل رقم (٤.٥) علاقة هذه المصفوفة والكائنات التي تم إنشاءها من النوع String:

```
stringArray(0) = "Hello"
stringArray(1) = "World"
stringArray(2) = "Wide"
stringArray(3) = "Web"
```



الشكل رقم (٤.٥). مصفوفة من متغيرات الإشارة لكائنات String.

وكما هو مبين في الشكل رقم (٤.٥) فإن المتغير stringArray عبارة عن متغير إشارة يشير إلى كائن من النوع Array والذي يحتوي على أربعة عناصر، كل عنصر عبارة عن متغير إشارة يشير إلى كائن من النوع String، وكل كائن من النوع String يحتوي على سلسلة من الحروف. كما توضح الصفة Length لكائن المصفوفة والتي تحتوي على رقم ٤ (عدد عناصر المصفوفة).

كما يمكن تقسيم نص (مخزن في كائن من النوع String) إلى كلمات وتكوين كائن مصفوفة يحتوي على هذه الكلمات وذلك باستدعاء الإجراء Split. فعلى سبيل المثال، يمكن تكوين مصفوفة تحتوي على أربع كائنات من النوع String من الجملة "Hello World Wide Web" وذلك بواسطة استدعاء الإجراء Split الذي يقسم هذه الجملة باستخدام المسافات الفاصلة بينها. ويسمى الحرف مسافة في هذه الحالة فاصل (Delimiter). لاحظ عند استدعاء الإجراء Split فلابد من تمرير الفاصل المناسب (في هذه الحالة الفاصل هو مسافة).

```
' declare a String
Dim s As String = "Hello World Wide Web"
' extract words delimited by space to populate a String array
Dim stringArray() As String = s.Split(" ")
```

وللتعامل مع عناصر المصفوفة، يتم استخدام فهرس المصفوفة كما فعلنا سابقاً. فعلى سبيل المثال لطباعة عناصر المصفوفة السابقة يتم كتابة المثال التالي:

```
// display each String value
Console.WriteLine(stringArray(0))
Console.WriteLine(stringArray(1))
Console.WriteLine(stringArray(2))
Console.WriteLine(stringArray(3))
```

وبالطبع يمكن استبدال شفرة المثال السابق مستخدماً التكرار For كما هو واضح في المثال التالي:

```
' display the values using a loop
Dim i As Integer
For i = 0 To stringArray.Length - 1
      Console.WriteLine(stringArray(i))
Next
```

لاحظ في المثال السابق أنه يتم استخدام الصفة Length لاختبار نهاية المصفوفة، ولأن الصفة Length تحتوي على ٤، لذلك فإن الأمر stringArray.Length - 1 يساوي ٣.

يجب أن تفرق بين الصفة Length الخاصة بكائنات الصنف Array والصفة Length الخاصة بكائنات الصنف String. ويعرض المثال التالي عدد عناصر المصفوفة (أربع عناصر) مستخدماً الصفة Length:

```
' display the number of elements
Console.WriteLine(stringArray.Length & " elements")
```

ولكي نعرض عدد حروف كائن من النوع String، فلابد من استخدام الصفة Length. ويوضح المثال التالي عدد حروف العنصر الرابع في المصفوفة (كائن من النوع String) والذي يحتوي على الكلمة "Web"، حيث سيعرض هذا الأمر القيمة ٣:

```
' display the number of characters in the last element
Console.WriteLine("length of Web is " & stringArray(3).Length)
```

يمكن أن تستدعي إجراءات الصنف String من عناصر المصفوفة المعرفة من النوع String، كما يتم استدعائها من كائنات الصنف String. على سبيل المثال، يمكن استدعاء الإجراء ToUpper من أول عنصر داخل المصفوفة السابقة كما هو واضح في المثال التالي:

```
' invoke ToUpper method for the first element
Console.WriteLine(stringArray(0).ToUpper())
```

يمكنك أيضاً البحث عن قيمة محددة داخل مصفوفة من النوع String، فعلى سبيل المثال افترض أنك تريد أن تبحث داخل المصفوفة stringArray عن القيمة "Web"، ولعمل ذلك عرِّف كائناً من النوع String مشار إليه بالمتغير searchValue والذي يحتوي على القيمة المراد البحث عنها كما هو واضح من الأمر التالي:

```
' search for the value "Web"
Dim searchValue As String = "Web"
```

ثم يتم تعريف المتغير i وإسناد قيمة ابتدائية "صفر" له، وكذلك تعريف المتغير found من النوع Boolean وإسناد قيمة ابتدائية False له. وسوف تتغير قيمة المتغير found إلى True عندما نجد القيمة التي نبحث عنها.

```
Dim found As Boolean = False
Dim i As Integer = 0
```

اكتب داخل التكرار Do While الأمر If الشرطي الذي يستدعي الإجراء Equals لمقارنة قيمة المتغير searchValue مع قيمة عنصر المصفوفة الحالي (وذلك بناء على قيمة الحالية للفهرس i). فإذا تساوت القيمتان تتغير قيمة المتغير Found إلى True ومن ثم تنهي عمل التكرار، أما إذا لم تتساوى القيمتين يزداد قيمة الفهرس i ويستمر تنفيذ أوامر التكرار طالما قيمة الفهرس أقل من عدد عناصر المصفوفة.

```
Do While i < stringArray.Length And Not found
      If stringArray(i).Equals(searchValue) Then
            found = True
      Else
            i += 1
      End If
Loop
If found Then Console.WriteLine("found " & searchValue)
```

لاحظ أن أمر If الشرطي استدعى الإجراء Equals للمقارنة بين القيم النصية بدلاً من استخدام المعامل =. كذلك يمكنك إضافة أوامر لفحص قيمة المتغير Found لمعرفة هل تم إيجاد القيمة المراد البحث عنها ومن ثم إظهار رسالة تفيد ذلك. وأيضاً يحتوي الصنف Array على الإجراء Sort والذي يستخدم لفرز عناصر المصفوفة كما هو واضح من المثال التالي والمسؤول عن ترتيب عناصر المصفوفة stringArray التي ستظهر بشكل مرتب عند طباعة عناصر المصفوفة.

```
' sort the string array into ascending sequence and then display
Array.Sort(stringArray)
```

يوضح الشكل رقم (٤.٦) الشفرة كاملة للبرنامج StringArrayDemo.vb، كما يوضح الشكل رقم (٤.٧) مخرجات هذا البرنامج.

```
' Chapter 4 StringArrayDemo Example 2
Module StringArrayDemo
    Sub Main()
    ' declare a String
    Dim s As String = "Hello World Wide Web"
    ' extract words delimited by space to populate a String array
    Dim stringArray() As String = s.Split(" ")

    ' display the values using a loop
    Dim i As Integer
    For i = 0 To stringArray.Length - 1
       Console.WriteLine(stringArray(i))
    Next

    ' display the number of elements
    Console.WriteLine(stringArray.Length & " elements")
    ' display the number of characters in the last element
    Console.WriteLine("length of Web is " & stringArray(3).Length)
    ' invoke ToUpper method for the first element
    Console.WriteLine(stringArray(0).ToUpper())

    ' search for the value "Web"
    Dim searchValue As String = "Web"
    Dim found As Boolean = False
    i = 0
    Do While i < stringArray.Length And Not found
       If stringArray(i).Equals(searchValue) Then
          found = True
       Else
          i += 1
       End If
    Loop
    If found Then Console.WriteLine("found " & searchValue)

    Console.WriteLine() ' blank line
    ' sort the string array into ascending sequence and then display
    Array.Sort(stringArray)
    For i = 0 To stringArray.Length - 1
       Console.WriteLine(stringArray(i))
    Next
    End Sub
End Module
```

الشكل رقم (٤.٦). شفرة البرنامج StringArrayDemo.vb.

```
Hello
World
Wide
Web
4 elements
length of Web is 3
HELLO
found Web

Hello
Web
Wide
World
```

الشكل رقم (٤,٧). مخرجات البرنامج StringArrayDemo.vb.

استخدام الصنف ArrayList
Using the ArrayList Class

كما رأيت أن عناصر المصفوفة تتكون من متغيرات والتي تتنوع من متغيرات أولية (Primitive Variables) أو متغيرات إشارة (Reference Variable) وذلك بناء على نوع بيانات المصفوفة المستخدم. ولقد قمنا بتعريف مصفوفات كثيرة من المتغيرات الأولية خلال الفصل الثالث وتعريف مصفوفة من النوع String (متغير إشارة) خلال الموضوع الفقرة السابقة، ولكن توجد هناك مشكلة رئيسة يعاني منها المبرمجون عند التعامل مع المصفوفات وهي أن المصفوفة تحتوي على حجم ثابت من العناصر التي من الصعب تغييره أثناء تنفيذ البرنامج.

ولكي نتغلب على هذه المشكلة نستخدم الصنف ArrayList (المعرف داخل الفضاء المسمى System.Collections) لتعريف مصفوفة يمكن تغيير عدد العناصر بها أثناء تنفيذ البرنامج بسهولة (أي تغيير عدد عناصر المصفوفة بشكل ديناميكي أثناء تشغيل البرنامج). كما يقدم هذا الصنف مجموعة من الإجراءات المفيدة التي نوضح بعضاً منها خلال هذه الفقرة. ويسرد الجدول رقم (٤,٣) العديد من هذه الإجراءات التي سيتم شرحها بشيء من التفصيل خلال هذه الفقرة.

الجدول رقم (٤,٣). بعض إجراءات الصنف ArrayList وصفاته.

الإجراء	الوصف
Add(O)	يضيف هذا الإجراء مؤشراً للكائن O إلى المصفوفة بعد آخر عنصر.
Contains(O)	يبحث هذا الإجراء عن الكائن O داخل المصفوفة وتعيد True أو False طبقاً لنتيجة البحث.
Capacity	تعيد أو تغير هذه الصفة حجم العناصر الممكن تخزينها بالمصفوفة.
Count	تعيد هذه الصفة عدد عناصر المصفوفة المتواجدة فعلياً داخل المصفوفة.
Item(i)	تعيد أو تستد كائن عند الفهرس i.
IndexOf(O)	تعيد فهرس الكائن O إذا كان متواجداً داخل المصفوفة أو تعيد =١ إذا لم يكن موجوداً.
Remove(O)	يقوم هذا الإجراء بمسح أول حدوث للكائن O من المصفوفة.
Reverse( )	يقوم هذا الإجراء بترتيب المصفوفة ترتيباً عكسياً.

لكي يتم إنشاء مصفوفة من النوع ArrayList يجب إنشاء كائن من الصنف ArrayList كما هو واضح داخل الأوامر التالية والمسؤولة عن إنشاء الكائن anArrayList من الصنف ArrayList وإسناد ثلاثة عناصر داخل هذه المصفوفة، مع العلم أن حجم هذه المصفوفة سيكبر عند إضافة عناصر أخرى لاحقاً. ولاحظ أنه تم إنشاء الكائن عن طريق استخدام الكلمة المحجوزة New وذلك لاستدعاء إجراء إنشاء الكائن Constructor الذي يتم استدعاؤه تلقائياً عند إنشاء الكائن. وكما هو واضح من البرنامج، فقد تم تحديد الحجم الابتدائي للمصفوفة ثلاثة عناصر وذلك بتمرير الرقم ٣ إلى إجراء الإنشاء الذي يمثل عدد العناصر الفعلي المراد حجزه.

```
' create an ArrayList instance with 3 elements
Dim anArrayList As ArrayList = New ArrayList(3)
```

والآن قم بإنشاء أربعة كائنات من الصنف String كما فعلنا في البرنامج السابق وذلك لتخزينها داخل المصفوفة هكذا.

```
' create String instances
Dim s1 As String = New String("Hello")
Dim s2 As String = New String("World")
Dim s3 As String = New String("Wide")
Dim s4 As String = New String("Web")
```

لاحظ أنه يمكن دمج الخطوتين السابقتين في أمر واحد وذلك بإنشاء كائن من النوع String وإضافته مباشرة داخل المصفوفة هكذا.

```
' populate the first two elements
anArrayList.Add(s1)
anArrayList.Add(s2)
```

كما يمكنك إنشاء كائن من الصنف String وتمريره مباشرة داخل الإجراء ()Add دون الحاجة إلى تعريف متغير إشارة () ثم تمرير هذا المتغير إلى الإجراء ()Add كما هو واضح في الأمر التالي الذي يوفر خطوة زائدة.

```
anArrayList.Add("Hello")
```

والآن يمكننا استخدام الصفة Capacity لاسترجاع الحجم الكلي للمصفوفة والصفة Count لاسترجاع عدد العناصر الموجودة داخل المصفوفة كما هو واضح داخل الأوامر التالية ؛ ولذلك سوف تظهر الرسالة رقم ٣ معبراً عن حجم المصفوفة anArrayList والرقم ٢ معبراً عن عدد العناصر المتواجدة داخل هذه المصفوفة.

```
Console.WriteLine("number of elements = " & anArrayList.Capacity)
Console.WriteLine(anArrayList.Count & " are populated")
```

كما يمكن استخدام الإجراء ()Contains للبحث عن كائن داخل المصفوفة وإرجاع القيمة True إذا كان موجوداً أو إرجاع القيمة False إذا لم يكن موجوداً. ويوضح المثال القادم استخدام هذا الإجراء للبحث عن النص "Hello" داخل المصفوفة anArrayList والذي يظهر القيمة True لأن هذا العنصر يتواجد في بداية المصفوفة.

```
' search for "Hello"
Console.WriteLine("the ArrayList contains Hello: " & _
   anArrayList.Contains("Hello"))
```

وفيما يلي نقوم بإضافة نصين آخرين إلى المصفوفة ، ومع أن حجم المصفوفة ثلاثة عناصر فإن العنصر الرابع سيتم إضافته بشكل آلي إلى المصفوفة.

```
' populate two more elements
anArrayList.Add(s3)
anArrayList.Add(s4)
Console.WriteLine("number of elements = " & anArrayList.Capacity)
Console.WriteLine(anArrayList.Count & " are populated")
```

يستخدم الإجراء IndexOf() للحصول على رقم فهرس عنصر معين داخل المصفوفة، يوضح المثال التالي استرجاع فهرس النص "Wide" الذي يمثل العنصر الثالث داخل المصفوفة ومن ثم يسند له الفهرس الثاني (لاحظ أن الفهرس يبدأ من صفر وليس من واحد).

```
' get the index of "Wide"
Console.WriteLine("the index of Wide is " & _
    anArrayList.IndexOf("Wide"))
```

وأخيراً نقوم باستخدام الإجراء Reverse() لإعادة ترتيب المصفوفة من النهاية إلى البداية، كما يمكننا أن نكتب برنامجاً للتجوال داخل المصفوفة وطباعة عناصرها كما فعلنا سابقاً في مثال الموضوع السابق لطباعة عناصر المصفوفة العادية (واضح من أوامر البرنامج التالية).

```
' display the values using a loop
Dim i As Integer
For i = 0 To StringArray.length - 1
     Console.WriteLine(StringArray(i))
Next
```

توضح أوامر البرنامج التالية كيفية عكس ترتيب المصفوفة anArrayList وطباعة عناصره.

```
' reverse the elements, then loop to display the contents
anArrayList.Reverse()
Dim i As Integer
For i = 0 To anArrayList.Count - 1
     Console.WriteLine(anArrayList.Item(i))
Next
```

يوجد هناك فرقان بين أوامر البرامج السابقة. أولاً، يتم استخدام الخاصية Length مع المصفوفة العادية ويتم استخدام الخاصية Count مع مصفوفة ArrayList عند التجوال داخل المصفوفة. وثانياً، يتم استرجاع العنصر من المصفوفة العادية إما مع مصفوفة ArrayList يتم استخدام الخاصية item متبوعة بالفهرس. يوضح الشكل رقم (٤.٨) برنامج هذا الموضوع كاملاً (ArrayListDemo.vb)، كما يوضح الشكل رقم (٤.٩) مخرجات هذا البرنامج. ولاحظ أن أوامر البرنامج تحتوي على الأمر Import System.Collections لاستدعاء أصناف الفضاء المسمى "System.Collections" داخل هذا البرنامج.

```
' Chapter 4 ArrayListDemo Example 3
Imports System.Collections
Module ArrayListDemo
   Sub Main()
      ' create an ArrayList instance with 3 elements
      Dim anArrayList As ArrayList = New ArrayList(3)
      ' create String instances
      Dim s1 As String = New String("Hello")
      Dim s2 As String = New String("World")
      Dim s3 As String = New String("Wide")
      Dim s4 As String = New String("Web")

      ' populate the first two elements
      anArrayList.Add(s1)
      anArrayList.Add(s2)
      Console.WriteLine("number of elements = " & anArrayList.Capacity)
      Console.WriteLine(anArrayList.Count & " are populated")

      ' search for "Hello"
      Console.WriteLine("the ArrayList contains Hello: " & _
         anArrayList.Contains("Hello"))

      ' populate two more elements
      anArrayList.Add(s3)
      anArrayList.Add(s4)

      Console.WriteLine("number of elements = " & anArrayList.Capacity)
      Console.WriteLine(anArrayList.Count & " are populated")
      ' get the index of "Wide"
      Console.WriteLine("the index of Wide is " & _
         anArrayList.IndexOf("Wide"))
      ' reverse the elements, then loop to display the contents
      anArrayList.Reverse()
      Dim i As Integer
      For i = 0 To anArrayList.Count - 1
         Console.WriteLine(anArrayList.Item(i))
      Next
   End Sub
End Module
```

الشكل رقم (٤,٨). برنامج ArrayListDemo.vb كاملاً.

```
number of elements = 3
2 are populated
the ArrayList contains Hello: True
number of elements = 6
4 are populated
the index of Wide is 2
Web
Wide
World
Hello
```

الشكل رقم (٤,٩). مخرجات برنامج ArrayListDemo.vb.

التعامل مع التاريخ
Working with Dates

إن معظم برامج الحاسب الآلي تحتاج أن تتعامل مع التاريخ، فمثلاً ربما تريد أن تتعامل مع تاريخ تعيين موظف، وتاريخ الميلاد، وتاريخ اليوم، وتاريخ انتهاء الصلاحية، وهكذا. تقدم لغة VB .NET العديد من الأصناف والإجراءات الجاهزة لاسترجاع تاريخ الجهاز ولتنسيق قيم التاريخ ولتنفيذ عمليات حسابية على التاريخ ولمقارنة قيم التاريخ وستتعرض لمعظم هذه الإجراءات خلال هذه الفقرة.

يوضح المثال التالي استخدام صنفين: الصنف DateTime والصنف TimeSpan حيث يحتوي كائن الصنف DateTime على تاريخ بينما كائن الصنف TimeSpan فيحتوي على حاصل طرح تاريخين، ويتواجد كل من الصنفين داخل المجموعة المعرفة System والتي يتم جلبها بشكل آلي بواسطة مترجم لغة VB .NET، ولذلك لا يستوجب استخدام الأمر Imports في بداية البرنامج. يبدأ المثال بتعريف متغير الإشارة Today من النوع DateTime ثم استدعاء الصفة Now من الصنف DateTime لتعيد كائناً من الصنف DateTime يحتوي على التاريخ والوقت الحالي ومن ثم إسناده لمتغير الإشارة Today. وإذا أردت استرجاع التاريخ الحالي فقط، يتم استخدام الصفة Today فقط:

```
' create an instance of DateTime populated with the system date & time
Dim today As DateTime = DateTime.Now
Console.WriteLine("Today is " & today)
```

كما يقدم الصنف DateTime الإجراءات اللازمة لإجراء عمليات حسابية على التاريخ، فمثلاً لإضافة أيام أو أشهر أو سنوات على تاريخ ما تستخدم الإجراءات AddDays و AddMonths و AddYears بالترتيب لعمل ذلك حيث تمرر القيمة المراد إضافتها لأي من هذه الإجراءات، فعلى سبيل المثال إذا أردنا إضافة شهر واحد يستدعى الإجراء AddMonths(1). حيث يستدعي المثال التالي الإجراء AddMonths لإضافة شهر على التاريخ الحالي وإسناد كائن التاريخ الجديد إلى متغير الإشارة aMonthFromToday، وأيضاً يستدعى الإجراء AddYears(1) لإضافة سنة إلى التاريخ الحالي وإسناد كائن التاريخ الناتج إلى متغير إشارة aYearFromToday.

```
' add 1 to today's month
Dim aMonthFromToday As DateTime = today.AddMonths(1)
Console.WriteLine("One Month From Today is " & aMonthFromToday)
' add 1 to today's year
Dim aYearFromToday As DateTime = today.AddYears(1)
Console.WriteLine("One Year From Today is " & aYearFromToday)
```

يعرض الإجراء WriteLine التاريخ على الشكل "mm/dd/yyyy" حيث يمثل mm الشهر، ويمثل dd اليوم، بينما يمثل yyyy السنة، فإذا افترضنا أن المتغير Today يحتوي على التاريخ February 15, 2003 فإن الأوامر السابقة ستظهره هكذا 02/15/2003. ولتغيير شكل التاريخ، يمكن استخدام الإجراء ToString وإرسال القيم اللازمة لتوصيف شكل التاريخ المراد. فعلى سبيل المثال، إذا أردت إظهار التاريخ على الشكل "February 15, 2003"، اكتب الأمر التالي:

```
Console.WriteLine("MMMM dd, yyyy) is " & today.ToString("MMMM dd,yyyy"))
```

يسرد الجدول رقم (٤.٤) حروف التنسيق المختلفة التي يمكنك استخدامها للحصول على شكل التاريخ المناسب لك. لاحظ الفرق بين استخدام الحروف الكبيرة (Capital) واستخدام الحروف الصغيرة (Small)، فمثلاً الحرف M يمثل الشهر بينما الحرف m فيمثل الدقائق. ويمكنك أيضاً وضع مسافات وفواصل وحروف أخرى لتنسيق التاريخ.

الجدول رقم (٤.٤). حروف التنسيق المستخدمة مع الإجراء ToString.

Format Characters	Description
d dd ddd dddd	Day as a number (1–7) Day as a number with leading zero (01–07) Three-character day name (Sun) Full day name (Sunday)
M MM MMM MMMM	Month as a number (1–12) Month as a number with leading zero for single digit (01–12) Three-character month name (Feb) Full month name (February)
m mm s ss	Minute without leading zeros Minute with leading zeros Second without leading zeros Second with leading zeros
t tt	Displays "A" for AM and "P" for PM Displays "AM" or "PM"
y yy yyyy	Single-digit year number without leading zeros (3) Two-digit year number (03) Four-digit year number (2003)

توضح الأوامر التالية استخدام هذه الحروف للحصول على تنسيقات مختلفة للتاريخ:

```
' illustrate various date formats
Console.WriteLine("MMMM dd, yyyy " & today.ToString("MMMM dd, yyyy"))
Console.WriteLine("MM/dd/yy hh:mm:ss tt " & _
      today.ToString("MM/dd/yy hh:mm:ss tt"))
Console.WriteLine("dddd, MMMM dd, yyyy " & _
      today.ToString("dddd, MMMM dd, yyyy"))
Console.WriteLine("MMMM yy " & today.ToString("MMMM yy"))
```

كما يمكنك إنشاء كائن من الصنف DateTime يحتوي على تاريخ محدد، فمثلاً ينشئ البرنامج التالي كائنين من النوع DateTime. اسم الأول eleanorsBirthday ويحتوي على التاريخ December 15, 1998 واسم الثاني emilysBirthday ويحتوي على التاريخ April 6, 2002. ويظهر الأمران الآخران هذان التاريخين في الشكل "monthname day, year".

```
' create specific dates
Dim eleanorsBirthday As DateTime = New DateTime(1998, 12, 15)
Dim emilysBirthday As DateTime = New DateTime(2002, 4, 6)
Console.WriteLine("Eleanor's Birthday is " & _
      eleanorsBirthday.ToString("MMMM dd, yyyy"))
Console.WriteLine("Emily's Birthday is " & _
      emilysBirthday.ToString("MMMM dd, yyyy"))
```

ولطرح تاريخ من تاريخ آخر، يتم استدعاء الإجراء Subtract والذي يعيد كائناً من النوع TimeSpan يحتوي على الفرق الزمني بين التاريخين (يمكن أن يقاس بالأيام أو بالساعات أو بالدقائق أو بالثواني). يعرف المثال التالي متغير إشارة من النوع TimeSpan اسمه ageDifference، ثم يستدعى الإجراء Subtract من كائن التاريخ emilysBirthday ويمرر له الكائن eleanorsBirthday كمعامل، حيث يحسب الإجراء Subtract الفرق بين التاريخين ويعيد كائناً من النوع TimeSpan والذي يسند إلى المتغير ageDifference. ويستدعي الأمر الأخير الإجراء TotalDays من المتغير ageDifference للحصول على عدد الأيام المناظر.

```
' compute the difference between two dates
Dim ageDifference As TimeSpan
ageDifference = emilysBirthday.Subtract(eleanorsBirthday)
Console.WriteLine("The age difference is " & _
      ageDifference.TotalDays() & " days")
```

كما يمكن مقارنة كائن تاريخ أول بكائن تاريخ ثاني وبواسطة الإجراء Compare الذي يعيد نتيجة المقارنة والتي إما أن تكون ١- إذا كان التاريخ الأول أصغر أو أن تكون ٠ إذا كان التاريخ الأول يساوي التاريخ الثاني أو تكون ١+ إذا كان التاريخ الأول أكبر. ويوضح المثال مقارنة التاريخ emilysBirthday الذي يحتوي على April 6, 2002 مع

التاريخ eleanorsBirthday والذي يحتوي على December 15, 1998 مستخدماً الإجراء Compare الذي سوف يعيد القيمة ١+ لأن التاريخ الأول أكبر من التاريخ الثاني.

```
' compare two dates - Compare returns -1, 0, or +1 for <, =, >
If DateTime.Compare(emilysBirthday, eleanorsBirthday)< 0 Then
    Console.WriteLine("Emily is older than Eleanor")
Else
    Console.WriteLine("Emily is younger than Eleanor")
End If
```

يوضح الشكل رقم (٤.١٠) البرنامج DateDemo.vb ويوضح الشكل رقم (٤.١١) مخرجات هذا البرنامج.

```
' Chapter 4 DateDemo Example 4
Module DateDemo
  Sub Main()
      ' create a DateTime instance populated with the system date & time
      Dim today As DateTime = DateTime.Now
      Console.WriteLine("Today is " & today)
      ' add 1 to today's month
      Dim aMonthFromToday As DateTime = today.AddMonths(1)
      Console.WriteLine("One Month From Today is " & aMonthFromToday)
      ' add 1 to today's year
      Dim aYearFromToday As DateTime = today.AddYears(1)
      Console.WriteLine("One Year From Today is " & aYearFromToday)
      ' illustrate various date formats
      Console.WriteLine("MMMM dd, yyyy " & _
         today.ToString("MMMM dd, yyyy"))
      Console.WriteLine("MM/dd/yy hh:mm:ss tt " & _
         today.ToString("MM/dd/yy hh:mm:ss tt"))
      Console.WriteLine("dddd, MMMM dd, yyyy " & _
         today.ToString("dddd, MMMM dd, yyyy"))
      Console.WriteLine("MMMM yy " & today.ToString("MMMM yy"))
      ' create specific dates
      Dim eleanorsBirthday As DateTime = New DateTime(1998, 12, 15)
      Dim emilysBirthday As DateTime = New DateTime(2002, 4, 6)
      Console.WriteLine("Eleanor's Birthday is " & _
         eleanorsBirthday.ToString("MMMM dd, yyyy"))
      Console.WriteLine("Emily's Birthday is " & _
         emilysBirthday.ToString("MMMM dd, yyyy"))
      ' compute the difference between two dates
      Dim ageDifference As TimeSpan
      ageDifference = emilysBirthday.Subtract(eleanorsBirthday)
      Console.WriteLine("The age difference is " & _
         ageDifference.TotalDays() & " days")
      ' compare two dates - Compare returns -1, 0, or +1 for <, =, >
      If DateTime.Compare(emilysBirthday, eleanorsBirthday) < 0 Then
         Console.WriteLine("Emily is older than Eleanor")
      Else
         Console.WriteLine("Emily is younger than Eleanor")
      End If
   End Sub
End Module
```

الشكل رقم (٤.١٠). شفرة البرنامج DateDemo.vb.

```
Today is 2/15/2003 4:35:47 PM
One Month From Today is 3/15/2003 4:35:47 PM
One Year From Today is 2/15/2004 4:35:47 PM
MMMM dd, yyyy February 15, 2003
MM/dd/yy hh:mm:ss tt 02/15/03 04:35:47 PM
dddd, MMMM dd, yyyy Saturday, February 15, 2003
MMMM yy February 03
Eleanor's Birthday is December 15, 1998
Emily's Birthday is April 06, 2002
The age difference is 1208 days
Emily is younger than Eleanor
```

الشكل رقم (٤,١١). مخرجات البرنامج DateDemo.vb.

تنسيق البيانات الرقمية
Formatting Numeric Output

من المفضل دائماً أن تظهر مخرجات برنامجك الرقمية بشكل منسق، والتنسيق (Formatting) يعني إضافة بعض الرموز الخاصة مثل إضافة الفاصلة والعلامة العشرية وكذلك علامة العملة الخاصة ببلدك والعلامة المئوية وهكذا. ولقد رأينا في الفقرة السابقة كيف يتم تنسيق التاريخ وسوف نتعلم في هذه الفقرة كيف تنسق البيانات الرقمية.

لقد رأينا في الفصل الثالث أن المتغيرات في لغة VB .NET تقسم إلى نوعين: متغيرات أولية (Primitive Variables) ومتغيرات إشارة (Reference Variables)، حيث يتم تعريف المتغيرات الأولية مستخدماً أنواع البيانات الأولية (Integer و Single و Boolean وهكذا)، أما متغيرات الإشارة فهي متغيرات تشير إلى كائن من صنف مثل كائنات الصنف String. تمثل لغة VB .NET أنواع البيانات الأولية على شكل نوع بيان مركب (Structure) الذي يتشابه مع الصنف حيث إن كليهما يحتويان على إجراءات ومن ثم يمكن استدعاء إجراءات من المتغيرات الأولية مثل الإجراء ToString والذي يستخدم لتنسيق البيانات الرقمية. وفي الحقيقة يستخدم الإجراء ToString عادة لتحويل البيانات الرقمية إلى قيمة نصية.

فعلى سبيل المثال، إذا كان لديك رقم صحيح (Integer) وتريد أن تظهره على شكل عملة (يحتوي على علامة الدولار وفاصلة وخانتين للكسر)، يمكنك استدعاء الإجراء ToString وإرسال المعامل "C" له، وسوف يظهر البرنامج التالي القيمة ١٢٣٤ على الشكل $1,234.00، ولاحظ أن هذه الأوامر تستدعي الإجراء ToString من المتغير الأولي i:

```
' illustrate two ways to format integer as currency
Dim i As Integer = 1234
Dim s As String = i.ToString("C")
Console.WriteLine("Integer 1234 with C format is " & s)
```

كما يمكنك أيضاً استخدام قناع التنسيق (Format Mask) الذي هو عبارة عن سلسلة من الحروف تحدد شكل التنسيق المطلوب، والمثال التالي يوضح أين تريد أن تضع علامة $، ومكان الفاصلة، ومكان النقطة العشرية. بالإضافة إلى ذلك يمكنك استخدام الحرف # لإضافة أصفار في مكان هذا الحرف:

```
s = i.ToString("$#,##0.00")
Console.WriteLine("Integer 1234 with $#,##0.00 format is " & s)
```

يمكن تحويل أي رقم بأي نوع بيانات إلى شكل العملة. افترض أن لديك المتغير الأولي d (من النوع Double) يحتوي على القيمة ١٢٣٤٥.٦٧ فيمكنك أن تظهره على شكل عملة باستدعاء الإجراء ToString مستخدماً المعامل "C" كما هو واضح من المثال التالي:

```
' illustrate two ways to format Double as currency
Dim d As Double = 12345.67
s = d.ToString("C")
Console.WriteLine("Double 12345.67 with C format is " & s)
s = d.ToString("$##,##0.00")
Console.WriteLine("Double 12345.67 with $##,##0.00 format is " & s)
```

إذا كان الرقم المخزن في متغير ما يحتوي على أكثر من خانتين عشريتين فإن الإجراء ToString يقرب القيمة الكسرية إلى أقرب خانتين فقط، والمثال التالي يوضح تحويل القيمة ١٢٣٤٥.٦٧٨ إلى $12,345.68:

```
' show that rounding occurs
d = 12345.678
s = d.ToString("C")
Console.WriteLine("Double 12345.678 with C format is" & s)
s = d.ToString("$##,##0.00")
Console.WriteLine("Double 12345.678 with $##,##0.00 format is " & s)
```

كما يمكن تنسيق الأرقام لتكون على شكل نسبة مئوية، أو تحتوي على فواصل، أو لا تحتوي على فواصل باستخدام المعاملات "P" و "N" و "F" على التوالي كما هو واضح من المثال التالي:

```
' illusrate percent, numeric with comma, numeric without comma
Console.WriteLine("Double 12345.67 as percentage is " & _
      d.ToString("P"))
Console.WriteLine("Double 12345.67 as numeric with commas is" & _
      d.ToString("N"))
Console.WriteLine("Double 12345.67 as numeric no commas is " & _
      d.ToString("F"))
```

في بعض الأحيان نريد أن نطبع الأرقام باستخدام تنسيق خاص مثل رقم الهاتف ورقم الضمان الاجتماعي، ولعمل ذلك نستخدم قناع التنسيق كما هو واضح في المثال التالي:

```
' format ss no & phone no
Dim phoneNo As Double = 1234567890
s = phoneNo.ToString("(###) ###-####")
Console.WriteLine("phone number format example " & s)
Dim ssNo As Double = 123456789
s = ssNo.ToString("###-##-####")
Console.WriteLine("Social Security number format example" & s)
```

يوضح الشكل رقم (٤,١٢) البرنامج FormatDemo.vb، كما يوضح الشكل رقم (٤,١٣) مخرجات هذا البرنامج.

```
' Chapter 4 FormatDemo Example 5
Module FormatDemo
   Sub Main()
      ' illustrate two ways to format integer as currency
      Dim i As Integer = 1234
      Dim s As String = i.ToString("C")
      Console.WriteLine("Integer 1234 with C format is " & s)
      s = i.ToString("$#,##0.00")
      Console.WriteLine("Integer 1234 with $#,##0.00 format is " & s)

      ' illustrate two ways to format Double as currency
      Dim d As Double = 12345.67
      s = d.ToString("C")
      Console.WriteLine("Double 12345.67 with C format is " & s)
      s = d.ToString("$#,##0.00")
      Console.WriteLine("Double 12345.67 with $#,##0.00 format " & s)
      ' show that rounding occurs
      d = 12345.678
      s = d.ToString("C")
      Console.WriteLine("Double 12345.678 with C format is " & s)
      s = d.ToString("$#,##0.00")
      Console.WriteLine("Double 12345.678 with $#,##0.00 format " & s)

      ' illustrate percent, numeric with comma, numeric without comma
      Console.WriteLine("Double 12345.67 as percentage is " & _
         d.ToString("P"))
      Console.WriteLine("Double 12345.67 as numeric with commas " & _
         d.ToString("N"))
      Console.WriteLine("Double 12345.67 as numeric no commas " & _
         d.ToString("F"))

      ' format ss no & phone no
      Dim phoneNo As Double = 1234567890
      s = phoneNo.ToString("(###) ###-####")
      Console.WriteLine("phone number format example " & s)
      Dim ssNo As Double = 123456789
      s = ssNo.ToString("###-##-####")
      Console.WriteLine("Social Security number format example " & s)
   End Sub
End Module
```

الشكل رقم (٤,١٢). شفرة البرنامج FormatDemo.vb.

```
Integer 1234 with C format is $1,234.00
Integer 1234 with $#,##0.00 format is $1,234.00
Double 12345.67 with C format is $12,345.67
Double 12345.67 with $##,##0.00 format is $12,345.67
Double 12345.678 with C format is $12,345.68
Double 12345.678 with $##,##0.00 format is $12,345.68
Double 12345.67 as percentage is 1,234,567.80 %
Double 12345.67 as numeric with commas is 12,345.68
Double 12345.67 as numeric no commas is 12345.68
phone number format example (123) 456-7890
Social Security number format example 123-45-6789
```

الشكل رقم (٤,١٣). مخرجات البرنامج FormatDemo.vb.

استخدام الصنف MessageBox
Using the MessageBox Class

في العديد من الأحيان يريد المبرمج إظهار رسائل للمستخدم في صناديق حوار ثم استقبال استجابة المستخدم والتعامل معها. ولإظهار رسالة في صندوق حواري، يستخدم الصنف MessageBox وهو أحد أعضاء المجموعة المعرفة System.Windows.Forms التي تشتمل على العديد من الأصناف الأخرى الخاصة بإظهار واجهة رسومية (GUI) مثل الصنف Form والصنف Button والصنف Label، إذ لابد من استخدام الأمر Imports لجلب هذه المجموعة المعرفة والتعامل مع الصنف MessageBox. وسوف تتعلم الكثير عن أصناف GUI خلال الفصل العاشر والحادي عشر.

يحتوي الصنف MessageBox على الإجراء Show الذي يستخدم لإنشاء كائن من الصنف MessageBox وإظهاره حيث يستقبل الإجراء MessageBox أربع معاملات (Arguments). يمثل المعامل الأول الرسالة المراد عرضها للمستخدم (كائن من النوع String أو نص ثابت)، ويمثل المعامل الثاني عنوان صندوق الحوار. في حين يحدد المعامل الثالث أنواع الأزرار (Buttons) المراد عرضها، بينما يحدد المعامل الرابع نوع الرمز (الصورة) المراد عرضه بجوار الرسالة.

إن أبسط شكل للإجراء Show هو إرسال الرسالة المراد عرضها للمستخدم فقط. فعلى سبيل المثال، إذا أردنا إظهار الرسالة "Hello Again" للمستخدم، فعند ذلك نكتب الأمر التالي. ويجب أن تلاحظ أن الزر OK سيظهر فقط بشكل تلقائي ولا يظهر عنوان للرسالة ولا يظهر رمز بجوار الرسالة. يوضح الشكل رقم (٤,١٤) نتيجة تنفيذ هذا الأمر.

```
' display a message
MessageBox.Show("Hello Again")
```

الشكل رقم (٤,١٤). MessageBox مع رسالة.

يمكنك أيضاً إظهار عنوان للرسالة وذلك بإرسال معامل ثانٍ كما هو واضح في الأمر التالي. مخرجات هذا الأمر واضحة في الشكل رقم (٤,١٥).

```
'display a message and a caption
MessageBox.Show("Hello Again", "MessageBox Demo")
```

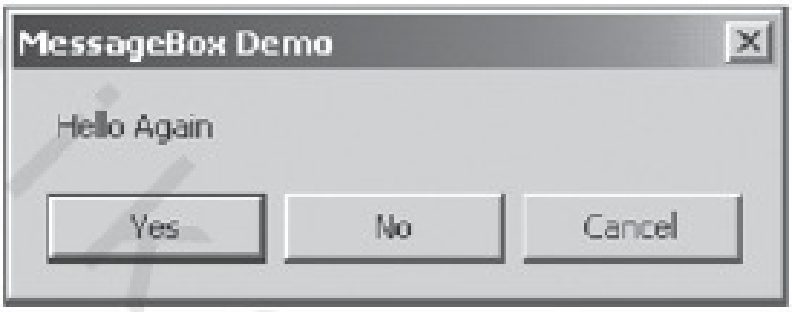
الشكل رقم (٤,١٥). MessageBox مع رسالة وعنوان.

كما يمكنك إضافة أي توليفة من الأزرار وذلك بناء على احتياجات البرنامج، فعند كتابة فاصلة لإضافة المعامل الثالث تظهر نافذة تلقائية لاختيار ما يناسبك من أزرار (كما هو واضح في الشكل رقم ٤,١٦).

الشكل رقم (٤,١٦). اختيارات أزرار MessageBox.

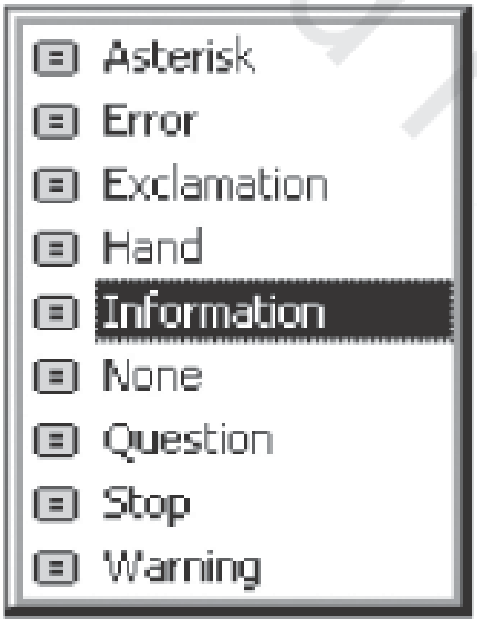
إذا اخترت الاختيار MessageBoxButtons.YesNoCancel للمعامل الثالث (كما هو واضح في المثال التالي) سيظهر الشكل رقم (٤.١٧).

```
'display a message, caption, Yes/No/Cancel buttons
MessageBox.Show("Hello Again", "MessageBox Demo", _
    MessageBoxButtons.YesNoCancel)
```



الشكل رقم (٤.١٧). MessageBox مع الأزرار Yes و No و Cancel.

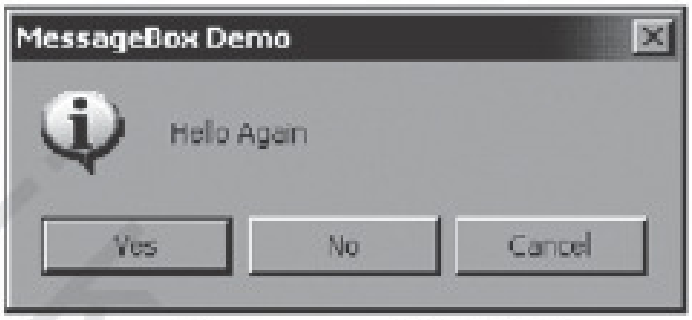
أما المعامل الرابع فيضيف رمزاً للرسالة حيث تكتب فاصلة ثم MessageBoxIcon ثم نقطة، عندئذ تظهر أنواع الرموز المتوفرة في النافذة التلقائية كما هو واضح في الشكل رقم (٤.١٨).



الشكل رقم (٤.١٨). اختيارات رموز MessageBox.

لو اخترت الرمز Information كما هو واضح في المثال التالي ستظهر الرسالة الموضحة في الشكل رقم (٤,١٩).

```
'display message, caption, Yes/No/Cancel buttons, and Icon
MessageBox.Show("Hello Again", "MessageBox Demo", _
      MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information)
```



الشكل رقم (٤,١٩). MessageBox مع أزرار ورمز.

وفي النهاية يكفي استقبال القيمة العائدة من الإجراء Show لمعرفة الزر الذي ضغط عليه المستخدم. تنتمي القيمة العائدة من الإجراء Show إلى النوع DialogResult حيث تستطيع مقارنتها بقيم محددة مستخدماً الأمر If الشرطي.

وعندما تكتب DialogResult ثم نقطة، ستظهر نافذة تلقائية بالاختيارات الخاصة كما هو واضح في الشكل رقم (٤,٢٠).

الشكل رقم (٤,٢٠). قيم DialogResult.

يوضح المثال التالي عرض رسالة تحتوي الأزرار Yes و No و Cancel ثم تحدد أي زر تم الضغط عليه حيث تظهر رسالة تفيد ذلك.

```
'display message, caption, Yes/No/Cancel buttons, Icon and get result
Dim result As DialogResult
result = MessageBox.Show("Hello Again", "MessageBox Demo", _
    MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information)
If result = DialogResult.Yes Then
    Console.WriteLine("The Yes button was clicked")
End If
If result = DialogResult.No Then
    Console.WriteLine("The No button was clicked")
End If
If result = DialogResult.Cancel Then
    Console.WriteLine("The Cancel button was clicked")
End If
```

ويوضح الشكل رقم (٤,٢١) البرنامج MessageBoxDemo.vb كاملاً.

```
' Chapter 4 MessageBoxDemo Example 6
Option Strict On
Imports System.Windows.Forms
Module MessageBoxDemo
    Sub Main()
        ' display a message
        MessageBox.Show("Hello Again")
        'display a message and a caption
        MessageBox.Show("Hello Again", "MessageBox Demo")
        'display a message, caption, Yes/No/Cancel buttons
        MessageBox.Show("Hello Again", "MessageBox Demo", _
            MessageBoxButtons.YesNoCancel)
        'display message, caption, Yes/No/Cancel buttons, and Icon
        MessageBox.Show("Hello Again", "MessageBox Demo", _
            MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information)
        'display message, caption, Yes/No/Cancel buttons, Icon, and get
result
        Dim result As DialogResult
        result = MessageBox.Show("Hello Again", "MessageBox Demo", _
            MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information)
        If result = DialogResult.Yes Then
            Console.WriteLine("The Yes button was clicked")
        End If
        If result = DialogResult.No Then
            Console.WriteLine("The No button was clicked")
        End If
        If result = DialogResult.Cancel Then
            Console.WriteLine("The Cancel button was clicked")
        End If
    End Sub
End Module
```

الشكل رقم (٤,٢١). شفرة البرنامج MessageBoxDemo.vb.

عرض نموذج من برنامج Console
Displaying a Form

سوف نعرض في هذا المثال كيفية إظهار نموذج (Form) من برنامج Console (يعمل تحت بيئة DOS). يحتوي البرنامج على ملف برمجي (Module) اسمه GuiModule.vb وملف نموذج GuiForm.vb، ولإظهار هذا النموذج لابد أن يحتوي الملف البرمجي GuiModule.vb على الأوامر اللازمة لإنشاء كائن من النموذج وإسناد عنوان (Title) له، ثم إظهاره وإخفاؤه، حيث تم استدعاء الإجراء Dispose لإزالته من الذاكرة. ويعتبر النموذج GuiForm.vb صنفاً فرعياً (Subclass) من الصنف Form ولذلك يرث جميع صفاته وإجراءاته.

يحتوي الملف البرمجي GuiModule.vb على الأمر Imports System.Windows.Forms لتمكين مترجم اللغة من التعامل مع الصنف Form. يحتوي البرنامج على أمر تعريف متغير myForm ثم يتم إنشاء كائن من الصنف GuiForm وإسناده لهذا المتغير.

```
Dim myForm As GuiForm
myForm = New GuiForm()
```

ثم يحتوي البرنامج على أمر إسناد العنوان "My GUI Form Demo" ثم يتم إسناد خلفية للنموذج تساوي System.Drawing.Color.Brown :

```
myForm.Text = "My GUI Form Demo"
myForm.BackColor = System.Drawing.Color.Brown
```

سوف نظهر رسالة صندوق حوار للمستخدم لمعرفة إذا ما كان المستخدم يريد أن يظهر النموذج أو يخفيه أو ينهي البرنامج ووضع هذه الأوامر في تكرار Do While. سوف تظهر نتيجة الزر التي تم حفظه في المتغير Result، وسوف ينتهي تنفيذ التكرار عند الضغط على زر Cancel. ويوضح الشكل رقم (٤,٢٢) الرسالة التي سوف تظهر للمستخدم، ويعرض الشكل رقم (٤,٢٣) النموذج المراد ظهوره، وأخيراً يوضح الشكل رقم (٤,٢٤) شفرة برنامج الملف البرمجي GuiModule.vb.

الشكل رقم (٤,٢٢). MessageBox تظهر بواسطة FormDemo.

الشكل رقم (٤,٢٣). النموذج الذي سيظهر بواسطة FormDemo.

```
' Chapter 4 GuiModule Example 7
Option Strict On
Imports System.Windows.Forms
Module GuiModule
   Sub Main()
      Dim myForm As GuiForm = New GuiForm()
      myForm.Text = "My GUI Form Demo"
      myForm.BackColor = System.Drawing.Color.Brown
      ' use MessageBox to determine what to do
      Dim result As DialogResult
      Do Until result = DialogResult.Cancel
         result = MessageBox.Show("Press Yes to Show, No to Hide, _
           Cancel to stop", "Form Demo", MessageBoxButtons.YesNoCancel)
         If result = DialogResult.Yes Then myForm.Visible = True
         If result = DialogResult.No Then myForm.Visible = False
      Loop
      myForm.Dispose()
   End Sub
End Module
```

الشكل رقم (٤,٢٤). شفرة البرنامج GuiModule.vb.

ملخص الفصل

Chapter Summary

- تعتبر لغة VB .NET جزءاً من فيجوال ستوديو .نت لمايكروسوفت التي تقدم مكتبة كبيرة من الأصناف المصنفة إلى مجموعات يطلق عليها مجموعات معرفة (Namespaces) حيث يمكنك استخدام الكلمة المحجوزة

Imports لكي تجعل المترجم يقوم بتحميل أصناف موجودة في مجموعة معرفة معينة. يحمل مترجم اللغة المجموعة System بشكل تلقائي، وكما ذكرنا سابقاً إذا أردت أن تتعامل مع أصناف أي مجموعة معرفة أخرى فلابد من استخدام الكلمة المحجوزة Imports متبوعة باسم المجموعة المعرفة المراد التعامل معها.

- تستخدم لغة VB .NET الصنف String لإنشاء متغير من هذا النوع حيث يحتوي الصنف String على العديد من الإجراءات (Methods) والخصائص (Properties)، مع العلم أن قيم كائنات الصنف String غير قابلة للتغيير (Immutable).
- تستخدم الصفة Length لكائن المصفوفة للحصول على عدد عناصر المصفوفة، وتستخدم مع كائن الصنف String للحصول على عدد حروف النص.
- يُمكّن الصنف ArrayList المبرمج من إنشاء مصفوفة متغيرة الحجم تلقائياً، مع العلم أن هذا الصنف هو أحد عناصر المجموعة المعرفة System.Collections. يحتوي هذا الصنف على إجراءات عديدة تمكن المبرمج من معالجة عناصر المصفوفة.
- يحتوي كائن الصنف DateTime على تاريخ بينما يحتوي كائن الصنف TimeSpan على حاصل طرح تاريخين، ويتواجد كل من الصنفين داخل المجموعة المعرفة System والتي يتم جلبها بشكل آلي بواسطة مترجم لغة VB .NET، ولذلك لا يستوجب استخدام الأمر Imports في بداية البرنامج.
- نعني بتنسيق المخرجات (Formatting) إضافة بعض الرموز الخاصة مثل إضافة الفاصلة والعلامة العشرية وكذلك علامة العملة الخاصة ببلدك والعلامة المئوية وهكذا. تمثل لغة VB .NET أنواع البيانات الأولية على شكل نوع بيان مركب (Structure) الذي يتشابه مع الصنف حيث إن كليهما يحتويان على إجراءات ومن ثم يمكن استدعاء إجراءات من المتغيرات الأولية مثل الإجراء ToString والذي يستخدم لتنسيق البيانات الرقمية. كما يمكنك أيضاً استخدام قناع التنسيق (Format Mask) الذي هو عبارة عن سلسلة من الحروف التي تحدد شكل التنسيق المطلوب.
- في العديد من الأحيان يريد المبرمج إظهار رسائل للمستخدم في صناديق حوار ثم استقبال استجابة المستخدم والتعامل معها. ولإظهار رسالة في صندوق حواري، نستخدم الصنف MessageBox وهو أحد أعضاء المجموعة المعرفة System.Windows.Forms، إذ لابد من استخدام الأمر Imports لجلب هذه المجموعة المعرفة والتعامل مع الصنف MessageBox. يحتوي الصنف MessageBox على الإجراء Show الذي يستخدم لإنشاء كائن من الصنف MessageBox وإظهاره بالشكل المطلوب حيث يستقبل الإجراء MessageBox أربع معاملات (Arguments) لذلك.

- ينتمي الصنف Form إلى المجموعة System.Windows.Forms والذي يستخدم لإظهار نموذج (Form) حيث يمكن كتابة الأوامر اللازمة لإنشاء كائن من هذا الصنف وإسناد عنوان (Title) له، ثم إظهاره وإخفاؤه (استدعاء الإجراء Dispose لإزالته من الذاكرة).

المصطلحات الأساسية

Key Terms

محرك اللغة المشترك (Common Language Runtime)	قناع التنسيق (Format Mask)
إجراء الإنشاء (Constructor)	نوع بيان مركب (Structure)
غير قابلة للتغيير (Immutable)	مجموعة معرفة (Namespace)

أسئلة المراجعة

Review Questions

١- تعيير المجموعات المعرفة يرجع إلى:

(أ) تخصيص اسم إلى وحدة برمجية.

(ب) تخصيص اسم إلى صنف.

(ج) مكتبة أصناف.

(د) طريقة حالة صنف.

٢- لإنشاء كائن من الصنف:

(أ) حذف المتغيرات داخل الصنف.

(ب) إنشاء متغيرات داخل الصنف.

(ج) إنشاء نسخة من الصنف.

(د) استدعاء إجراء من الصنف

٣- الخطوة الصحيحة لاستدعاء الإجراء To Upper للسلسلة النصية aString:

(أ) a String.ToUpper().

(ب) String.ToUpper(aString).

(ج) ToUpper.String().

(د) aString.String(ToUpper).

٤- فهرس الحرف الأول لسلسلة نصية هو:

(أ) (١).

(ب) (صفر).

(ج) القيمة المحددة عند تعريف السلسلة النصية.

(د) (٢).

٥- إجراء السلاسل النصية SubString:

(أ) يعيد الفهرس.

(ب) يعيد السلسلة النصية.

(ج) يعيد القيمة الأولية.

(د) يعيد الحرف.

٦- فهرس العنصر الأول للمصفوفة هو:

(أ) (١).

(ب) (صفر).

(ج) القيمة المحددة عند تعريف المصفوفة.

(د) (٢).

٧- كل عنصر في المصفوفة النصية هو:

(أ) متغير أولي.

(ب) عديم القيمة.

(ج) متغير إشارة.

(د) هذا يعتمد على نوع البيانات المحدد.

٨- للحصول على عدد عناصر المصفوفة النصية المسمي sArray، تحتاج لكتابة:

(أ) sArray.Length.

(ب) sArray.Lengthof.

(ج) StringArray.Length.

(د) String.Length.

٩- قيم السلاسل النصية في الفيجوال بيسك .نت:

(أ) لا يمكن أن تتغير.

(ب) هي من نوع البيانات الأولية.

(ج) إما أن تكون متغيرات أولية أو متغيرات إشارة.

(د) لا يمكن وضعها في مصفوفة.

١٠- العناصر في ArrayList:

(أ) تبدأ بقيمة فهرس ٠.

(ب) من الممكن أن تحتوي فقط على قيم أولية.

(ج) تبدأ بقيمة فهرس ١.

(د) يجب أن تكون من نوع البيانات مصفوفة.

١١- إجراء ArrayList الذي يعيد عدد العناصر هو:

(أ) الإجراء Size.

(ب) الإجراء numberOfElements.

(ج) الإجراء Capacity.

(د) لا يوجد مثل هذا الإجراء.

١٢- كائن الصنف DateTime يحتوي على:

(أ) كائن التقويم.

(ب) قيمة الوقت والتاريخ.

(ج) قيمة التاريخ فقط.

(د) قيمة الوقت فقط.

١٣- كائن الصنف TimeSpan يحتوي على:

(أ) الاختلاف بين التاريخين.

(ب) قيمة الوقت والتاريخ.

(ج) قيمة التاريخ فقط.

(د) قيمة الوقت فقط.

١٤- ما هو Color.red؟

(أ) إجراء.

(ب) صفة.

(ج) ثابت.

(د) خاصية.

١٥- صنف MessageBox:

(أ) له إجراء فردي يسمى View.

(ب) له إجراء فردي يسمى Show.

(ج) ليس له إجراء.

(د) جزء من النموذج.

أسئلة المناقشة

Discussion Questions

١- ما هي مميزات استخدام الإجراءات بدلاً من الاعتماد على الإجراءات المعرفة داخل اللغة؟

٢- كائنات الصنف String غير قابلة للتغيير (Immutable). ما هو تأثير ذلك على استخدام بعض الإجراءات مثل Substring و ToUpper التي تعيد كائناً من الصنف String؟

٣- يبدو أن استخدام الصنف Arraylist أفضل من استخدام الصنف Array. علل ذلك.

٤- لا يحتوي الصنف Datetime على إجراء Equals. كيف يمكن استخدام الإجراء Compare لمعرفة أن قيم تاريخين متساويين؟

مشاريع الفصل

Projects

١- أعد كتابة برنامج ArrayDemo.vb المعرف في الفصل الثالث مستخدماً الصنف Arraylist.

٢- قم بإنشاء مصفوفة ذات بعدين (٤ صفوف، ٣ أعمدة) من النوع String، ثم أسند إليه القيم الموضحة في الجدول التالي. ثم اكتب إجراءً لعرض محتويات هذه المصفوفة مستخدماً عناوين مناسبة.

Flight	Gate	Destination
AA 7401	C33	St. Louis
AA 431	D8	Dallas
Delta 94	A12	Atlanta
United 155	B4	Chicago

٣- قم بإعادة تصميم المشروع رقم ٢ مستخدماً الصنف Arraylist الذي سيحتوى على أربعة عناصر (واحد لكل صف) حيث يشير كل عنصر إلى مصفوفة ذات بعد واحد تحتوي على ثلاثة عناصر (Flight و Gate و Destination)؟

٤- يمثل الشكل رقم (٤.٥) رسماً توضيحياً لمصفوفة ذات بعد واحد من النوع String. وضح بالرسم التوضيحي كيف تكون المصفوفة ذات البعدين من النوع String.