الفصل الحادي والعشرون
أدوات التحكم الرئيسية

في هذا الفصل نناقش بالتفصيل بعض الأدوات الأساسية للتحكم في Visual Basic تشتمل على مربعات الاختيار وأزرار الاختيار ومربع السرد ومربعات السرد والتحرير وأشرطة التمرير وأخيراً المؤقت. بانتهاء هذا الفصل ستتعرف على:

- إعطاء خيارات للمستخدم كنوع من طرق الإدخال
- التعرف على مربع الاختيار Check box.
- التعرف على أزرار الاختيار Radio Buttons.
- التعرف على مربع السرد List box.
- التعرف على مربع السرد والتحرير Combo box.
- استخدام شريط التمرير Scroll Bar كأداة إدخال.
- استخدام المؤقت Timer.

تعرفنا في الفصول السابقة من الكتاب على كيفية كتابة برنامج باستخدامVisual Basic وكيفية التعامل مع أدوات التحكم بصورة إجمالية. وفي هذا الفصل والفصول التاليةسنتعرف على المزيد عن وظائف عناصر التحكم وكيفية تطويعها لأداء ما نريد. في هذا الفصل نبدأ بالفئة الأساسية من أدوات التحكموفى الفصول التالية سنشرح معظم أدوات التحكم التى تهمك وتشمل الجديد الذى لم يكن موجوداً بالإصدارات السابقة.

بعض العناصر التي شرحناها بالتفصيل في الباب الأول من الكتاب لن نعيد شرحها هنا، على أية حال إن أردت الرجوع إليها فإليك الفصول التي ذكرناها فيها:

زر الأمر **Button**	الفصل الرابع "استخدام أدوات التحكم"
مربع الصور **PictureBox**	الفصل الثامن عشر"تحسين الواجهات بالرسوم"
أداة العنوان **Label**	الفصل التاسع عشر "التعامل مع النصوص والخطوط والألوان"
مربع النص **TextBox**	

ينبغي أن تتذكر أيضاً الخصائص المشتركة بين عناصر التحكم، مثل **Name**، **Left**،**Height**، **Width**، **Enabled**، **Visible** والتي شرحناها فيما سبق،حيث يتناول هذا الفصل الأدوات الأساسية التي لم تشرح بالتفصيل بجانب بقية الخصائص الهامة للتعامل مع الأدوات.يوضح شكل ٢١-١الأدوات التى سنشرحها فى هذا الفصل وموقعها داخل مربع الأدوات،إذا لم يكن مربع الأدوات ظاهراًأمامك، انقر زر مربع الأدوات من شريط الأدوات أو اختر أمر **View**≺**ToolBox**من شريط القوائم.

شكل ٢١-١ أدوات التحكم التى سنتعرض لها بالشرح فى هذا الفصل

وقبل أن نشرح أدوات التحكم الرئيسية، سنتعرض لبعض الخصائص الهامة المشتركة بين معظم أدوات التحكم.

ترتيب أدوات التحكم

أول الخصائص الهامة المتبقية، الخاصية **TabIndex** وهي تحدد ترتيب العنصر عندما يضغط المستخدم مفتاح **Tab** للتنقل بين أدوات التحكم المختلفة على النموذج أو النافذة، هذه الخاصية يتم إعطائها قيمة افتراضية لكل عنصر تضيفه إلى النموذج، إن تم إضافة العناصر بترتيب صحيح أي من الأعلى للأسفل فستكون قيمة هذه الخاصية مضبوطة، أما إذا لم يتحقق ذلك، أو قمت بتغيير مواضع العناصر بعد ذلك على النموذج فيجب عليك أن تضبط هذه الخاصية لاحقاً.

جميع الأمثلة والتدريبات الواردة بهذا الفصل موجودة داخل المشروع **Controls**، حيث يوجد كل مثال داخل نموذج مستقل. للعمل مع أحد

النماذج، اجعل هذا النموذج هو نموذج بدء التطبيق من صفحات خصائص المشروع ثم قم بتشغيله.

يوضح شكل ٢١-٢ أحد نوافذ إدخال البيانات الموجودة بالمشروع **Controls**. قم بتشغيل التطبيق، وفى مرحلة التشغيل اضغط مفتاح **Tab** للانتقال بين عناصر النموذج لتتبين الفرق بين الضبط الجيد والرديء لهذه الخاصية.

شكل ٢١-٢نموذج إدخال بيانات يحتوى على أكثر من عنصر

تأخذ الخاصية **TabIndex** لكل عنصر قيمة بدءاً من **0**، ثم تزداد لكل عنصر، عندما يبدأ البرنامج،يستحوذ العنصر ذو القيمة "**0**" (إن كان ظاهراً، وممكناً) على التركيز**Focus**. وعند ضغط **Tab** بالترتيب ينتقل التركيز إلى العناصر ذات القيم ١، ٢ ... وهكذا. إذا أردت انتقال التركيز مباشرةًإلى العنصر السادس بعد العنصر الثاني مثلاً، اجعل قيمة الخاصية **Tabindex** له هى 2، يقوم **Visual Basic** تلقائياً بتعديل جميع العناصر ذات القيم من 2 إلى 4 ليحتفظ بالترتيب القديم مع صعود العنصر إلى المكان الثالث في الترتيب. هذه الخاصية لا توجد إلا في العناصر التي يمكن نقل التركيز إليها.

برغم أن أداة العنوان**Label** لا يمكنها الاستحواذ على التركيز إلا أن لها هذه الخاصية، فهى تقدم وسيلة للوصول إلى مربعات النصوص مباشرة عن طريق لوحة المفاتيح.

لتوضيح ذلك، تابع الخطوات التالية:

١. قم بإضافة مربع نص وليكن ترتيبه TabIndex هو 5.

٢. قم بإضافةأداة عنوان وضعها بجانب مربع النص السابق.

٣. عين لأداة العنوان مفتاح الوصول Alt+N (بوضع الحرف & قبل الحرف N بنص العنوان).

٤. قم بتشغيل التطبيق واضغط الاختصار Alt+N، يقوم Visual Basic بنقل التركيز إلى أداة العنوان، وحيث أنه لن يقبلها فسيمررها إلى العنصر الذي يليه في ترتيب TabIndex. ويؤدي ذلك إلى حصول مربع النص عليها.

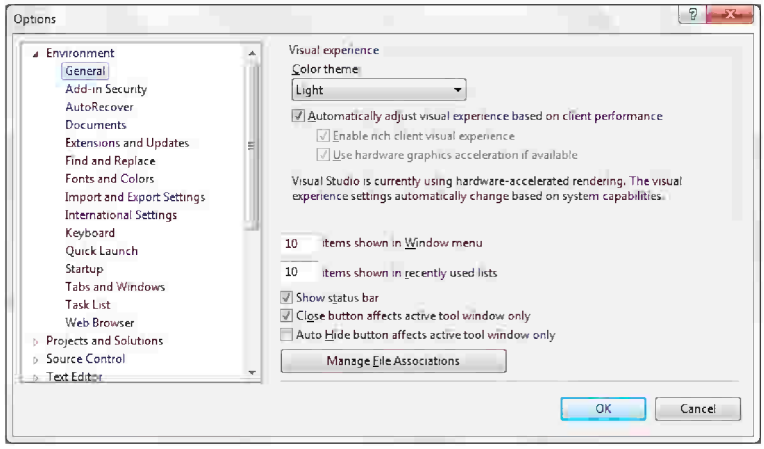
هذا الأسلوب شهير، ولا يمكن بغيره عمل مفاتيح وصول لمربع النص. من الخصائص المرتبطة بـTabIndexأيضاً الخاصية TabStop.وهذه الخاصية تحدد هل يتحرك التركيز إلى العنصر في ترتيبه عند استخدام Tab أم يتم تجاوزه. هذا الأمر لا يعني عدم تمكين العنصر، حيث يظل المستخدم قادراً على الوصول إليه باستخدام الفأرة أو مفاتيح الوصول.

استخدام مربع الاختيار Check Box

وظيفة مربع الاختيار Check Box أخذ إجابة محددة من المستخدم (نعم/لا) أو(Yes/No).ومربع الاختيار عبارة عن مربع صغير تظهر به العلامة (✓) عندما تكون الإجابة بنعم ولا تظهر به إذا كانت الإجابة بـ(لا) (انظر شكل ٢١-٣).

شكل ٢١-٣ مربع اختيار نشط (نعم) وآخر معطَّل (لا)

يمكنك استخدام مربع الاختيار لتحديد خصائص معينة للبرامج، مثال على ذلك خصائص بيئة التطوير المتكاملة والتي تظهر في مربع الخيارات Options في خيارات Visual Studio 2012(انظر شكل ٢١-٤).



شكل ٢١-٤ صفحة خصائص بيئة التطوير المتكاملة داخل مربع الخيارات Options.

خصائص مربع الاختيار

يمكنك إضافة مربع الاختيار إلى نموذجك باتباع أيٍ من الطرق المعتادة لإضافة جميع أدوات التحكم إلى النماذج. وما يعنينا هنا هو إلقاء الضوء على أهم الخصائص المصاحبة لمربع الاختيار. يوضح جدول ٢١-١ التالى أهم هذه الخصائص.

جدول ٢١-١ الخصائص المصاحبة لمربع الاختيار

الخاصية	الاستخدام
Appearance	إذا احتوت هذه الخاصية على القيمة Button، يظهر مربع الاختيار كما لو كان زراً مفصلياً له حالتان فقط، مضغوط وغير مضغوط. والقيمة الافتراضية لهذه الخاصية هى Normal.
AutoChek	إذا لم ترغب فى أن يقوم المستخدم بتنشيط مربع الاختيار أو تعطيله

الخاصية	الاستخدام
	وإنما يكون التنشيط أو التعطيل مبنياً على عوامل أخرى تتم من داخل الكود، قم بتخصيص القيمة **False** لهذه الخاصية. والقيمة الافتراضية لهذه الخاصية هى **True** بمعنى قدرة المستخدم على تنشيط وتعطيل مربع الاختيار.
CheckAlign	تحدد هذه الخاصية موقع مربع الاختيار نفسه داخل حدود الأداة، حيث يوجد تسعة أماكن مختلفة بعدد الاتجاهات والمواضع المتاحة داخل حدود الأداة. القيمة الافتراضية هى **MiddleLeft** بمعنى وضع المربع يسار منتصف الأداة.
Checked	تحتوى هذه الخاصية على القيمة **True** أو **False** تبعاً لحالة مربع الاختيار من التنشيط أو عدم التنشيط. والقيمة الافتراضية لهذه الخاصية هى **False**.
CheckState	توضح هذه الخاصية حالة مربع الاختيار، وتحتوى على قيمة من ثلاث قيم هى (نشط) **Checked** أو (معطَّل) **Unchecked** أو (غير محدد) **Indeterminate**.
FlatStyle	تحدد هذه الخاصية مظهر مربع الاختيار وخاصةً عند مرور مؤشر الفأرة عليه. القيمة الافتراضية لهذه الخاصية هى **Standard** وفيها يظهر مربع الاختيار بمظهره المعتاد. يمكنك أيضاً استخدام القيمة **Flat** كى يظهر مسطح أو **Popup** كى يقوم بأداء حركة منسدلة بسيطة عند مرور مؤشرالفأرة.
Image	تحدد هذه الخاصية الصورة التى تظهر على الأداة أو داخلها (تبعاً لقيمة الخاصية **Appearance**).
ImageAlign	تحدد هذه الخاصية موضع الصورة التى تم تعيينها داخل الخاصية

الخاصية	الاستخدام
	Image (إن وجدت) داخل الأداة، حيث يوجد تسعة مواضع ممكنة أيضاً.
Text	تستخدم هذه الخاصية كالمعتاد فى إظهار عنوان مربع الاختيار.
TextAlign	تحدد هذه الخاصية موضع عنوان مربع الاختيار الذى تم تعيينه داخل الخاصية Text داخل الأداة، حيث يوجد تسعة مواضع ممكنة أيضاً.
ThreeState	إذا احتوت هذه الخاصية على القيمة True، يستطيع المستخدم ثلاث حالات ممكنة لمربع الاختيار وهى (نشط) Checked أو (معطَّل) Unchecked أو (غير محدد) Indeterminate. القيمة الافتراضية لهذه الخاصية هى False وهذا يعنى تعيين قيمتين فقط (نشط) Checked أو (معطَّل) Unchecked.

والآن دعنا نقوم بإلقاء نظرة خاطفة على خاصيتين هامتين من الخصائص الموجودة فى الجدول السابق.

الخاصية Appearance

الوضع الطبيعى لمربع الاختيار هو يملك سيارة ، حيث يظهر مربع صغير وبجواره عنوان الأداة، ويحتوى هذا المربع على العلامة ✓ فى حالة تنشيطه بينما يظهر خالياً فى حالة تعطيله. يمكنك بدلاً من ذلك إظهار مربع الاختيار فى صورة زر مفصلى، يكون هذا الزر مضغوط هكذا يملك سيارة فى حالة تنشيط الأداة أو بارز هكذا يملك سيارة فى حالة تعطيلها وذلك من خلال الخاصية Appearance التى تحتوى على قيمتين هما Normal (الحالة الأولى) وButton (الحالة الثانية).

الخاصية ThreeState

كما ذكرنا فإن الوضع الطبيعى لمربع الاختيار هو احتوائه دائماً على قيمتين إما نشط أو غير نشط (معطَّل). لكن هناك بعض الحالات التى لا تصلح فيها أى القيمتين. فإذا كنا مثلاً نتكلم عن مدرسة من المدارس التى تحتوى بداخلها بالطبع على فصول متعددة، وأردنا استخدام مربع اختيار لتوضيح ديانة الفصل بالكامل ولتكن هكذا 

. فبالتأكيد هناك العديد من الفصول التى يوجد بها مسلمون ومسيحيون وفى هذه الحالة لا يمكنك تنشيط مربع الاختيار أو تعطيله وإنما يتم استخدام قيمة وسط هى القيمة **Indeterminate** وفيها تظهر العلامة ✓ باهتة داخل مربع الاختيار هكذا 

. وكى تتمكن من تعيين هذه القيمة، يجب تخصيص القيمة **True** للخاصية **ThreeState** وإلا لن تتمكن من استخدامها.

إذا كانت قيمة الخاصية **ThreeSate** هى **True**، تقوم الخاصية **Checked** بإرجاع القيمة **True** سواءً كانت قيمة الخاصية **CheckState** هى **Checked** أو **Indeterminate**.

تحديد وقراءة حالة مربع الاختيار

أياً كان مظهر مربع الاختيار (طبيعى **Normal** أو زر **Button**) فإن الهدف في النهاية معرفة القيمة التي اختارها المستخدم، يحدد ذلك الخاصية **CheckState** وهي تأخذ إحدى القيم الثلاثة الآتية:

- القيمة **Checked** إذا قام المستخدم باختيار الأداة (حالة التنشيط).
- القيمة **Unchecked** إذا لم يقم المستخدم باختيار الأداة (حالة التعطيل).
- القيمة **Indeterminate** وهى حالة التنشيط غير الكامل.

يمكنك قراءة هذه القيم وتحديدها وضبطها أثناء التصميم، كما يمكنك أيضاً التحكم فيها أثناء التشغيل من خلال الكود.

أثناء تشغيل البرنامج يمكن للمستخدم تغيير اختياره، إما بنقر المربع أو نقل التركيز **Focus** إليه ثم ضغط مفتاح **Space**.

دعنا نرى مثالاً يقوم فيه مربع اختيار بتغيير خصائص الخط في مربع نص موجود داخل النموذج. قم بإنشاء نموذج جديد وليكن باسم frmCheckBoxثم قم بإضافة زر أمر باسم btnApply وعنوان "تطبيق التنسيق"، ومربع نص باسم txtMainمع تخصيص القيمة True للخاصية Multiline المصاحبة له ومربع اختيار باسم chkBold وعنوان Bold (انظر شكل ٢١-٥).انقر زر "تطبيق التنسيق" نقراً مزدوجاً ثم قم بإدخال الكود التالى داخل حدث نقر الزر:

```
Private Sub btnApply_Click(sender As Object, e As EventArgs)
Handles btnApply.Click
    If chkBold.CheckState = CheckState.Checked Then
txtMain.Font = New System.Drawing.Font(txtMain.Font,
FontStyle.Bold)
    Else
txtMain.Font = New System.Drawing.Font(txtMain.Font,
FontStyle.Regular)
    End If
  End Sub
```

وكما ترى يتم قراءة قيمة الاختيار بالخاصية CheckState.

شكل ٢١-٥ تحديد وقراءة حالة مربع الاختيار أثناء التشغيل

يحتوى شكل ٢١-٥السابق على مربع اختيار واحد لكن كما سبق وأوضحنا نستطيع إضافة أى عدد من المربعات كما يظهر فى شكل ٢١-٤ السابق الذى يشتمل على مربعات كثيرة داخل المربع الحوارىOptions.

استخدام أزرار الاختيار Radio Buttons

إذا تعددت الاختيارات، فلا يصلح مربع الاختيار، وإنما نلجأ إلى أزرار الاختيار والتي تستعمل غالباً في مجموعات Groups وعند اختيار أحدها، يتم تعطيل الاختيار عن بقيتهاوحتى إذا لم تستخدم فى مجموعة فتكون ضمن نطاق النموذج نفسه ويتم تنشيط أحدها فقط أيضاً. يظهر في شكل ٢١-٦ نموذج لشكل نافذة بها عدد من أزرار الاختيار.

شكل ٢١-٦ مجموعة من أزرار الاختيار.

إنشاء مجموعة من أزرار الاختيار

لإنشاء مجموعة من أزرار الاختيار، اختر أداة زر الاختيار من مربع الأدوات ثم ارسم كل زر على حده على النموذج بعد اختياره من مربع الأدوات كما فعلنا مع مربع الاختيار ثم اضبط خصائص كل زر على حده (غير العنوان والاسم وبقية الخصائص). مثلاً لإنشاء النافذة في الشكل ٢١-٦ السابق،قم إضافة أربعة أزرار اختيار وتغيير عناوينهم وأسمائهم.يمكنك إنشاء أى عدد من أزرار الاختيار للنموذج حسب حاجتك وحاجة برنامجك.

تغيير شكل أزرار الاختيار

كافة إمكانيات تغيير الشكل في مربع الاختيار موجودة في أزرار الاختيار مثل **Font** لتغيير الخط و **TextAlign** لتغيير محاذاة النص مع زر الاختيار و**Backcolor**و**ForeColor** لتغيير اللون و**RightToLeft** عند استخدام عنوان باللغة العربية.

هناك أيضاً طريقتان لترتيب أزرار الاختيار، حيث يمكن ترتيبها أفقياً أو رأسياً كما في شكل ٢١-٧ وإن كان الشكل الرأسي مفضل لأنه الشكل القياسي.

شكل ٢١-٧ يمكن ترتيب مجموعة أزرار الاختيار أفقياً أو رأسياً

تحديد الزر المختار عند بدء البرنامج

عند بداية البرنامج ستجد جميع الأزرار غير مختارة، ويستحسن دائماً البدء بأحد الأزرار مختاراً، يمكن ذلك أثناء التصميم أو عند بدء تشغيل البرنامج بضبط الخاصية **Checked** للزر المراد تنشيطه بالقيمة **True**.

قراءة اختيار المستخدم

لمعرفة اختيار المستخدم من الأزرار، نقوم باختبار قيمة الخاصية **Checked** لكل الأزرار في المجموعة، واحد فقط منها هو الذي تكون قيمته **True** وهو الزر المختار. يوضح الكود التالي كيفية التعرف على الزر المختار في النموذج السابق عند النقر على زر الأمر الذي يظهر على النافذة:

```
Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
```

```
        If optSingle.Checked Then
MessageBox.Show("الحالة الاجتماعية: أعزب"، "الحالة الاجتماعية",
MessageBoxButtons.OK)
ElseIfoptMarried.CheckedThen
MessageBox.Show("الحالة الاجتماعية: متزوج"، "الحالة الاجتماعية",
MessageBoxButtons.OK)
ElseIfoptDivorced.CheckedThen
MessageBox.Show("الحالة الاجتماعية: مطلق"، "الحالة الاجتماعية",
MessageBoxButtons.OK)
ElseIfoptWidower.CheckedThen
MessageBox.Show("الحالة الاجتماعية: أرمل"، "الحالة الاجتماعية",
MessageBoxButtons.OK)
      End If
   End Sub
```

إنشاء عدة مجموعات من أزرار الاختيار

في شكل ٢١-٦ السابق قمنا بوضع أزرار الاختيار مباشرةً على النموذج، هذه الطريقة كانت للتوضيح فقط، وهي طريقة غير جيدة، حيث أنها لا تتيح تكوين مجموعات منفصلة من أزرار الاختيار حيث تشمل المجموعة فى هذه الحالة جميع أزرار الاختيار الموجودة فى أى مكان على النموذج. لتكوين مجموعات منفصلة نحتاج حاويات **Containers** مختلفة، أكثر الحاويات استخداماً مع أزرار الاختيار هو مربع المجموعة **GroupBox**.

يوضح شكل ٢١-٨ مثالاً لاستخدام أكثر من مجموعة منفصلة على نفس النموذج.

شكل ٢١-٨ استخدام مربع المجموعة لإنشاء مجموعات منفصلة من أزرار الاختيار.

يعتبر مربع المجموعة GroupBox بديل لأداة الإطار Frame Control الموجودة بالإصدارات القديمة، كما تم تغيير اسم زر الاختيار من OptionButton فى الإصدارات القديمة إلى RadioButton بدءاً من Visual Basic.Net.

استخدام مربع السرد ListBox

تعتبر أداة مربع السرد أبسط الأدوات التي تمكنك من عرض قائمة من الخيارات. يوضح شكل ٢١-٩ مثالاً لأحد مربعات السرد الذى يحتوى على مجموعة من محافظاتجمهورية مصر العربية.

شكل ٢١-٩ مثال لمربع سرد في أحد التطبيقات

لمربع السرد سمات رئيسية وهي:

- **Items** (قائمة العناصر): وهي قائمة بالخيارات المتاحة، يمكن إنشائها أثناء التصميم أو أثناء تشغيل البرنامج.
- **SelectedItem** (العنصر المختار): وهو العنصر الذي قام المستخدم باختياره وتعليمه، ويعرف بظهوره تحت الشريط المضاء أو بوجود مربع اختيار بجانبه معلم بـ(✓) تبعاً لنمط مربع السرد نفسه.
- **ScrollAlwaysVisible** (شريط التمرير): يدل هذا الشريط على وجود عناصر في القائمة أكثر مما يظهر في مساحة مربع السرد، يمكن الوصول إليها باستخدام

شريط التمرير، فإذا قمت بتخصيص القيمة **True** لهذه الخاصية يتم إظهار شريط التمرير باستمرار، أما إذا قمت بتخصيصها بالقيمة **False**، فلا يتم إظهاره إلا وقت الحاجة فقط.

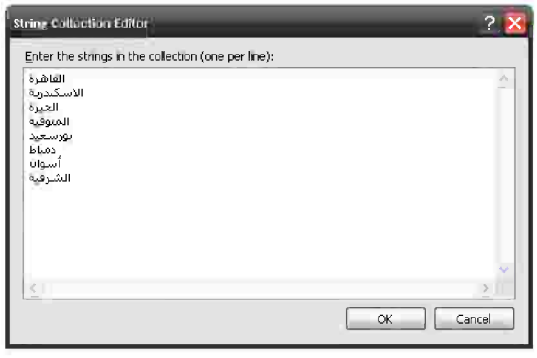
إعداد مربع السرد

عند إضافتك لمربع سرد ListBox إلى النموذج، يظهر مستطيل خالي عبارة عن الحد الخارجي للمربع وتظهر الكلمة **ListBox1** فقط في أعلى المربع، كما يظهر بلا أشرطة تمرير، وبلا عناصر في القائمة (عدا **ListBox1** بالطبع).يمكنك فى هذه الحالة ملء مربع السرد بقائمة العناصر التى ترغب فى إظهارها إذا أردت تعيينها أثناء التصميم (وهذا هو المتبع فى كثيرٍ من الأحوال) وذلك من خلال الخاصية **Items** وهي عبارة عن مصفوفة **Array**أو تجمع **Collection**من المتغيرات الحرفية **Strings**. وكل عنصر في القائمة هو أحد عناصر هذه المصفوفة.

لإضافة العناصر إلى مربع السرد أثناء التصميم، تابع معنا الخطوات الآتية:

١. تأكد من اختيار مربع السرد داخل النموذج ثم قم باختيار الخاصية **Items** من نافذة خصائص مربع السرد، ثم انقر الزر ... المجاور للخاصية، تظهر نافذة **String Collection Editor** التى يمكنك من خلالها تعيين عناصر قائمة مربع السرد (انظر شكل ٢١-١٠).

٢. قم بإدخال كل عنصر من عناصر القائمة فى سطر مستقل إلى أن تنتهى من إدخال جميع العناصر ثم انقر زر **Ok**، تلاحظ ظهور العناصر التى قمت بتعيينها داخل مربع السرد سواءً أثناء التصميم أو وقت التشغيل (انظر شكل ٢١-١١).



شكل ٢١-١٠ إضافة العناصر إلى مربع السرد.

شكل ٢١-١١ تظهر العناصر داخل مربع السرد أثناء التصميم ووقت التشغيل.

تغيير محتويات القائمة من خلال كود البرنامج

لا تقتصر إمكانية تحديد عناصر القائمة على مرحلة التصميم، بل يمكن الإضافة والحذف والتعديل فيها أثناء تشغيل البرنامج.

تستخدم الوظيفة **Add()** لإضافة عنصر إلى مربع السرد، واستخدام هذه الوظيفة سهل للغاية ويتمثل في استخدام السطر التالى:

```
lstCity.Items.Add("سيناء")
```

كما ترى يتم استخدام التسمية المنقوطة لتحديد اسم مربع السرد (lstCity فى هذه الحالة) كما يتم استخدام التجمعItems الذى يحتوى على عناصر مربع السرد. وأخيراً يتم ذكر العنصر المراد إضافته.

إذا أردت تعيين عناصر مربع السرد بمجرد فتح النموذج، قم بتعريف العناصر المختلفة باستخدام الأمر السابق داخل الإجراء New الخاص بالنموذج أو الإجراء Load.

يقوم الأمر السابق بإضافة العنصر المحدد إلى نهاية قائمة عناصر المربع (ما لم تكن خاصية الترتيب فعالة). إن أردت وضع هذا العنصر في مكان معين من القائمة، يمكنك استخدام الوظيفة Insert() كما يلى:

```
lstCity.Items.Insert(3,"سيناء")
```

حيث يتم تحديد الموضع المراد إضافة العنصر فيه إلى القائمة.ويراعى هنا أن أول عنصر من القائمة يخصص له الرقم (0) والثاني (1) وهكذا، أي أن العنصر السابق في المثال سيتم وضعه في رابع سطر بالقائمة.

مما ينبغي أخذه في الاعتبار أيضاً أنك لو حددت دليلاً يخرج عن حدود العدد الموجود حالياً في القائمة أو قمت بتحديد رقم أقل من الصفر، فسيصدر البرنامج خطأ ويتوقف عن العمل. لتجنب حدوث ذلك، يمكنك اختبار العدد الموجود في القائمة حالياً بالخاصية Count.

حذف العناصر من مربع السرد

يمكنك حذف العناصر داخل مربع السرد من خلال الوظيفة Remove(). ولكى تقوم بعملية الحذف، يجب أن تقوم بتحديد دليل العنصر Index (أي ترتيبه بالقائمة) أو تحديد نص العنصر أو حذف العنصر المختار.

- لحذف العنصر عن طريق ترتيبه داخل قائمة مربع السرد (وليكن العنصر الأول)، قم باستخدام السطر التالى:

```
lstCity.Items.RemoveAt(0)
```

- لحذف العنصر من خلال تعيين نص هذا العنصر، قم باستخدام السطر التالى:

```
lstCity.Items.Remove("سيناء")
```

- لحذف العنصر المختار حالياً داخل مربع السرد، قم باستخدام السطر التالى:

```
lstCity.Items.Remove(lstCity.SelectedItem)
```

- لحذف جميع عناصر مربع السرد مرةً واحدة، استخدام الوظيفة **Clear** كما فى السطر التالى:

```
lstCity.Items.Clear()
```

- يمكنك حذف عناصر مربع السرد عن طريق ضغط مفتاح **Delete** من لوحة المفاتيح. لأداء ذلك، قم بإدخال الكود التالى داخل الحدث **KeyDown** الخاص بمربع السرد:

```
Private Sub lstCity_KeyDown(sender As Object, e As
System.Windows.Forms.KeyEventArgs) Handles
lstCity.KeyDown
    If e.KeyCode = Keys.Delete Then
lstCity.Items.Remove(lstCity.SelectedItem)
    End If
End Sub
```

ترتيب عناصر القائمة أبجديا

في كثير من الأحيان نحتاج إلى عرض عناصر القائمة مرتبة أبجدياً، هذا الأمر لن يحتاج إلى كود كما تتوقع، كل ما تحتاجه هو جعل الخاصية **Sorted** بالقيمة **True**. لذا فعند إضافة أو حذف أو تعديل عناصر، سيحافظ المربع على ترتيب العناصر أبجدياً بداخله.

معرفة اختيار المستخدم من القائمة

الهدف الأساسي من مربع السرد هو تمكين المستخدم من الاختيار، بالطبع سنريد بعد ذلك معرفة اختيار المستخدم لاستخدامه في عمليةٍ ما. سنحتاج في ذلك إلى التعامل مع الخاصيتين **SelectedIndex** و **SelectedItem** كما يلي:

- تحمل الخاصية **SelectedIndex** دليل العنصر المختار حالياً، يمكن استخدام هذا الدليل لاحقا في معرفة العنصر (نصه) كما في المثال التالي:

```
i = lstCity. SelectedIndex
Strchosen = lstCity (i)
```

- تحمل الخاصية **SelectedItem** نص العنصر المختار كما يلى:

```
strCity = lstCity.SelectedItem
```

وعلى فرض أن العنصر المختار حالياً هو العنصر "القاهرة" وترتيبه الثانى داخل مربع السرد فإن العبارة التالية:

```
MessageBox.Show(lstCity.SelectedIndex)
```

تقوم بإظهار مربع رسالة يحتوى على القيمة **1**. بينما تقوم العبارة التالية:

```
MessageBox.Show(lstCity.SelectedItem)
```

بإظهار رسالة تحتوى على نص العنصر أى "القاهرة".

- يمكنك أيضاً معرفة العنصر المختار من خلال الخاصية **Text** الخاصة بمربع السرد نفسه التى تحتوى دائماً على نص العنصر المختار داخل المربع.

تقوم الخاصية **SelectedIndex** بإرجاع القيمة **1-** فى حالة عدم اختيار أية عناصر. كما تقوم الخاصية **Text** الخاصة بمربع السرد بإرجاع القيمة " " فى حالة عدم اختيار أية عناصر أيضاً.

تعيين الاختيار الافتراضى

تبعاً لمتطلبات وطبيعة تطبيقك، يمكنك تحديد العنصر المختار داخل مربع السرد والذى يتم اختياره دائماً بمجرد بدأ التطبيق وذلك بطريقتين، الأولى من خلال الخاصية **SelectedIndex** كما يلى:

```
lstCity.SelectedIndex = 3
```

أما الطريقة الثانية فمن خلال الخاصية **Text** المصاحبة لمربع السرد كما يلى:

```
lstCity.Text = "أسيوط"
```

التعامل مع الاختيارات المتعددة

قد تحتاج لأغراض معينة إتاحة الفرصة للمستخدم لاختيار عدة عناصر بدلاً من عنصر واحد وعندئذ نقوم بضبط الخاصية **SelectionMode**، حيث تأخذ هذه الخاصية أربع قيموهى:

- **One** وهو الخيار الافتراضى ويتسبب فى منع تعدد الاختيار وبالتالى لا يمكنك اختيار أكثر من عنصر.
- **None** وهذا يعنى عدم قدرة المستخدم على اختيار العناصر داخل مربع السرد.
- **MultiSimple** (اختيار متعدد بسيط): يسمح باختيارات متعددة بمجرد نقر زر الفأرة أو ضغط مفتاح المسافةSpace ولكن لا يمكِّن هذا الخيار من اختيار مجموعة متجاورة من العناصر.
- **MultiExtended** (اختيار ممتد): يسمح باختيارات متعددة بنقر زر الفأرة +مفتاح **Ctrl** ولكنه يسمح أيضا باختيار مجموعة متجاورة من العناصر باستخدام **Shift**+ نقر زر الفأرة.

ولا يبدو بين نوعي الاختيار (بسيط، ممتد) اختلاف في الشكل أثناء تشغيل البرنامج، فقط تتبين الاختلاف عند الاستخدام (انظر شكلى ١٠- ١٢و١٠-١٣).

شكل ٢١-١٢ SelectionMode = MultiSimple

شكل ٢١-١٣ SelectionMode = MultiExtended

الحصول على الاختيارات المتعددة

تختلف طريقة معرفة الاختيارات المتعددة عن الاختيار المنفرد نوعاً ما، حيث أن الخاصية **SelectedIndex** لا تحمل إلا رقم واحد فقط ، وهو دليل العنصر الذي عليه التركيز بينما لا يحمل أرقام العناصر الأخرى المختارة، كما أن الخاصية **SelectedItem** تحتوى على نص عنصر واحد أيضاً . وإنما يمكنك استخدام تجمعين بديلين وهما التجمع **SelectedIndices** والتجمع **SelectedItems**، حيث يحتوى الأول على مصفوفة بأدلة العناصر المختارة بينما يحتوى الثانى على قيم هذه العناصر. وكل منهما يدعم الخاصية **Count** التى يمكنك عن طريقها معرفة عدد العناصر المختارة. يوضح الكود التالى كيفية استخدام التجمع **SelectedItems** لإظهار جميع العناصر المختارة فى مربعات رسائل:

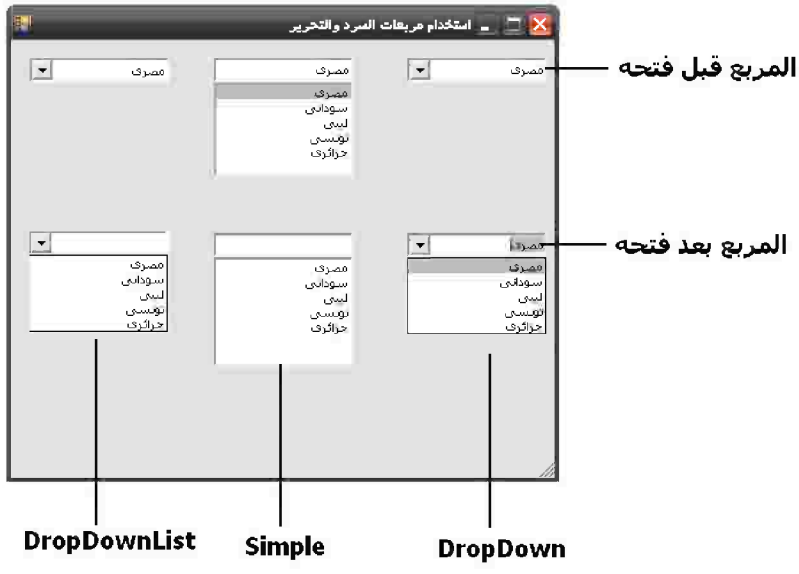
```
Dim i As Integer
For i = 0 To lstCity.SelectedItems.Count-1
MessageBox.Show(lstCity.SelectedItems(i))
Next i
```

استخدام مربع السرد والتحرير ComboBox

من العناصر الأخرى التي تمكنك من عرض القوائم التي تحمل خيارات للمستخدم أداة مربع السرد والتحرير **ComboBox** وسنرمز إليها أحياناً بمربع السرد والتحرير اختصاراً. وهذا العنصر له ثلاثة أشكال يمكنك اختيار أيٍ منها من خلال الخاصية **DropDownStyle** (انظر شكل ٢١-١٤).

- مربع السرد والتحرير Drop-Down ComboBox: يعرض مربع نص به سهم عند ضغطه تنسدل قائمة أسفل المربع حيث يستطيع المستخدم اختيار أحد عناصرها، ويمكن للمستخدم أن يكتب في مربع النص مباشرة إذا أراد.
- مربع التحرير والسرد البسيط Simple Combo box: يعرض مربع نص شبيهبمربع النص Textbox تماماً. وأسفل منه يعرض قائمة ظاهرة دائماً (غير منسدلة) تشبه ListBox تماماً كأن هذا الشكل دمج لمربع النص ومربع السرد.
- قائمة منسدلة Drop Down List: يعرض مربع نص ولكن لا يمكن الكتابة فيه مباشرة، فقط يمكننا استخدام القائمة المنسدلة لاختيار أحد عناصرها ليظهر داخل مربع النص. (هذا يعني أن المستخدم لابد أن يختار أحد العناصر الموجودة تماماً مثل مربع السرد ولكن مع اختزال المساحة المطلوبة من النافذة).

ولمربع السرد والتحرير صفات كثيرة مشتركة مع مربع السرد، فهو يحتوي أيضاً على الوظائف Add() و Remove()وClear()بالإضافة إلى ترتيب العناصر من خلال الخاصية Sorted. الفرق بينهما أن مربع السرد والتحرير ليس به اختيار متعدد أي لا يملك الخاصية SelectionMode.



شكل ٢١-١٤ الأنماط المختلفة من مربع السرد والتحرير

إنشاء مربع السرد والتحرير

يتم إنشاء مربع السرد والتحرير كالأدوات الأخرى باختيار الأداة 

من مربع الأدوات ثم رسم المربع على النموذج ثم ضبط اسمه **Name** ونمطه **DropDownStyle**. لو أردنا مثلاً للمربع أن يكون للسرد فقط أي لا يستطيع المستخدم إدخال شئ غير العناصر فإننا نجعل هذا النمط بالقيمة **DropDownList**. بعد ذلك نضيف ما نشاء من العناصر من خلال الخاصية **Items** سواءً في مرحلة التصميم أو أثناء تشغيل البرنامج.

العمل مع الاختيارات غير الموجودة في القائمة

ترجع فائدة مربع السرد والتحرير الحقيقية في أنه يتيح للمستخدم إدخال اختيار لا يوجد في القائمة. هذه الإمكانية متاحة في نوعي مربع السرد والتحرير الآخرين وهما **Simple**و**DropDown**وذلك باختيار إحداهما من الخاصية **DropDownStyle**.

اختيار عنصرمن خلال الكود

قد تحتاج لاختيار أحد العناصر من مربع السرد والتحرير خاصة عند بدء البرنامج كأن تضع اختيار ما ليظهر تلقائياً لاحتمال أن يختاره المستخدم. يتم ذلك بطريقتين:

- استخدام الخاصية **SelectedIndex** مع تحديد دليل العنصر المراد كالتالي:

```
cboCity.SelectedIndex = 3
```

- استخدام الخاصية **SelectedItem** مع تحديد العنصر المراد كالتالى:

```
cboCity.SelectedItem = "مصرى"
```

- استخدام الخاصية **Text** بكتابة نص العنصر المراد مباشرة كالتالي:

```
cboCity.Text = "مصرى"
```

مع العلم أن هذه الطريقة لا تستخدم في حالة كون مربع السرد والتحرير من النوع **DropDwonList** أي نوع القائمة المنسدلة.

قراءة اختيار المستخدم من مربع السرد والتحرير

بما أن المستخدم قد يدخل قيم غير موجودة في القائمة، فغالباً ما تسترجع اختيار المستخدم من خلال الخاصية **Text**أو من خلال الخاصيتين **SelectedItem** و**SelectedIndex** تماماً كما كنا نفعل مع مربع السرد.

إضافة ما يكتبه المستخدم إلى القائمة

قلنا أن المستخدم يستطيع كتابة قيمة غير موجودة داخل قائمة مربع السرد والتحرير، ولكن لو كان مربع السرد والتحرير يوجد في نافذة لإدخال بيانات مثلاً، فقد نريد إضافة ما كتبه المستخدم إلى القائمة، كي يكون متاحاً كاختيار فيما بعد ولا يحتاج المستخدم إلى كتابته مرةً ثانية. مثال لذلك في النافذة الموجودة في شكل ٢١-١٥حيث وضعنا مربع سرد وتحرير لإدخال الجنسية، وعند إضافة المستخدم لجنسية غير موجودة بالقائمة يتم إضافة هذه الجنسية إلى القائمة الموجودة من قبل.

هذا الأمر لا يتم تلقائياً. وللأسف لا يوجد حدث ينطلق إذا أدخل المستخدم عنصراً لا يوجد في القائمة. أي لإجراء هذه العملية يجب تنفيذها بالكامل بالكود.

شكل ٢١-١٥ مربع "الجنسية" المنسدل يسمح بكتابة عنصر يضاف إلى القائمة لاحقاً

لإتمام عملية الإضافة هذه نستخدم حدث "فقدان التركيز Leave" الذي يحدث لمربع السرد عند فقدانه للتركيز(وهذا ما يحدث عادةً في نوافذ إدخال البيانات)، عند فقدان التركيز سنقوم بالآتي:

- قراءة قيمة الخاصية Text لمربع السرد والتحرير.
- مقارنة هذه القيمة بكافة القيم الموجودة في القائمة للتأكد من عدم وجودها في القائمة من قبل.
- إن كانت هذه القيمة جديدة، يتم إضافتها إلى القائمة.

يوضح ذلك الكود التالي بالتفصيل:

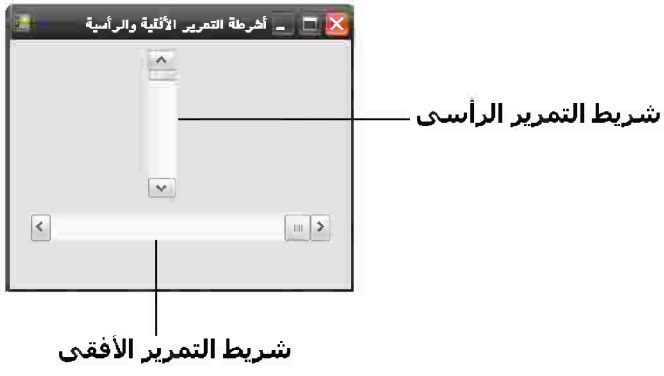
```
Private Sub CmboNation_Leave(sender As Object, e As
EventArgs) Handles cmboNation.Leave
    Dim NewItmAs String, I As Integer, AddNewItm As Boolean
AddNewItm = True
    With cmboNation
NewItm = cmboNation.Text
        For I = 0 To .Items.Count - 1
          If .Items(I) = NewItm Then
AddNewItm = False
            Exit For
          End If
        Next I
        If AddNewItm Then
          .Items.Add(NewItm)
        End If
    End With
End Sub
```

استخدام أشرطة التمرير Scroll Bars كأداة للإدخال

لقد شاهدنا أشرطة التمرير من قبل في مربع النص متعدد الأسطر، وفي مربعات السرد ومربعات السرد والتحرير كأداة مساعدة للتنقل بين محتوياتها عند زيادةحجم المحتويات عن حجم الأداة.وهناك أداتان داخل Visual Basic هما شريط الأدوات الأفقي

HScrollBar وشريط الأدوات الرأسي VScrollBarوهي تشبه أشرطة التمرير المساعدة للعناصر الأخرى، إلا أنها تختلف عنها قليلاً (انظر شكل ٢١-١٦).

أشرطة التمرير تعمل كزر الصوت في المسجل الصوتي والذى يعمل على رفع وخفض الصوت، كذلك أشرطة التمرير تعمل على رفع وخفض قيمة أحد خصائصها عند تحريكها. هذه القيمة هي الخاصية Value.



شكل ٢١-١٦شريط التمرير الأفقى وشريط التمرير الرأسى

وفي الحقيقة فإن الفرق بينهما في الشكل فقط. بل كان من الأفضل عمل أداة واحدة، بها خاصية تحدد هل هو أفقي أم رأسي. على أية حال لدينا أداتين منفصلتين ولكنهما يخضعان لنفس الاعتبارات ولهما نفس الخصائص والوظائف.

فيما يلي مثلاً سنتحدث عن الشريط الأفقي، وهذا الكلام ينطبق تماماً على الشريط الرأسي.

إعداد شريط التمرير

يستخدم شريط التمرير كبديل رسومي لإدخال القيم الرقمية، يتم ذلك من خلال تحديد الحد الأدنى والحد الأقصى للشريط، ومن ثمَّفعند تحريك المؤشر في الشريط، تتغير قيمة الخاصية Valueبحيث يمكن قراءتها بعد ذلك في البرنامج.

من الاستخدامات النمطية لشريط التمرير استخدامه في إدخال النسب اللونية وكذلك تحديد درجة الصوت في تطبيقات الوسائط المتعددة.

تحديد الحد الأدنى والأقصى لشريط التمرير

يتم تحديد حدود قيم الشريط بالخاصيتين Minimumو Maximumوالقيمة الافتراضية للخاصية Minimum هي 0، بينما القيمة الافتراضية للخاصية Maximum هى100، يمكن تغيير هذه القيم أثناء التصميم أو التشغيل، فمثلاً يمكننا جعل مدى الشريط من 0 إلى 100 باستخدام الكود الآتي:

```
hScroll1.Minimum =0
hScroll1.Maximum =100
```

على اعتبار أن الشريط يعبر عن نسبة مئوية.

التحكم في مقدار التغير في القيمة Value

بالتأكيد سبق لك استخدام شريط التمرير ولاحظت وجود طريقتين لتغيير القيمة بالنقر، الأولى النقر على أي مساحة خالية من شريط التمرير وهذا يحرك المؤشر بمقدار كبير، الثانية النقر على الأسهم في أول وآخر الشريط وهذا يحرك المؤشر بمقدار صغير. هذه المناطق مبينة في شكل ٢١-١٧.

شكل ٢١-١٧ المناطق المختلفة للنقر واختلاف مقدار التغير تبعاً لها

يمكننا تحديد مقدار التغير الصغير من خلال الخاصية SmallChange والكبير من خلال الخاصية LargeChange، القيمة الافتراضية للخاصية الأولى تكون 1بينما تكون ١٠ للخاصية الثانية ولكن قد تحتاج لتغييرها في برنامجك لتناسب الاستخدام.

الحصول على قيمة شريط التمرير

يمكنك الحصول على قيمة شريط التمرير باستمرار أثناء التشغيل من خلال الحدث **Scroll** الذى يحدث عندما يقوم المستخدم بالنقر على الأسهم أو الشريط لتمرير المؤشر.

يوضح النموذج الموجود فى شكل ٢١-١٨استخدام الحدث **Scroll** لإظهار قيمة شريط التمرير داخل مربع النص باستمرار، حيث يحتوى الحدث **Scroll** الخاص بشريط التمرير على الكود التالى:

```
Private Sub HScrollBar1_Scroll(sender As Object, e As
System.Windows.Forms.ScrollEventArgs) Handles
HScrollBar1.Scroll
txtNum.Text = HScrollBar1.Value
End Sub
```

شكل ٢١-١٨الحصول على قيمة شريط التمرير من خلال الحدث **Scroll**

استخدام أداة المؤقت Timer

أداة المؤقت **Timer**إحدى الأدوات المفيدة جداً في **Visual Basic** وهي تعمل كالمنبه، إذ تقوم بإطلاق حدث معين كل فترة زمنية محددة. لنفرض مثلاً أنك تريد إظهار الوقت الحالي ثانية بثانية على الشاشة كالساعة. لفعل ذلك ستحتاج إلى كتابة الوقت كل ثانية، هذا الأمر يتضمن شيئين:

أولاً : أن تحدد للمؤقت أن يفعل مهمة ما كل ثانية.

ثانياً : أن تحدد له المهمة التي سيؤديها.

تحديد الزمن الدوري الذي سيتم تنفيذ المهمة عنده يتم بالخاصية **Interval**، وهو لا يحدد بالثواني وإنما بالملي ثانية (واحد على ألف من الثانية)، أي لتحديد الزمن بثانية واحدة

سنضبط **Interval** على القيمة **1000**. تحديد المهمة يتم من خلال كتابة كود في الإجراء الحدثي **Tick**، وهذا هو الحدث الوحيد الذي يملكه المؤقت.

إنشاء حافظ للشاشة ScreenSaver

سنوضح في المثال التالي كيفية استخدام المؤقت من خلال برنامج بسيط لحفظ الشاشة **ScreenSaver** يقوم بإخفاء محتويات الشاشة بعد مرور فترة زمنية محددة لصيانة الشاشة من جانب، وتأمين المحتويات التي قد تكون سرية. تابع معنا الخطوات الآتية:

١. قم بإضافة نموذج جديد إلى المشروع **Controls**الذى بدأناه فى هذا الفصل وليكن باسم **frmScreenSaver**.

٢. قم بإضافة أداة المؤقت **Timer** إلى النموذج الجديد وأعد تسميتها إلى **tmrBlank** مثلاً. يظهر المؤقت على هيئة رمز أسفل النموذج، بينما لن يظهر تماماً أثناء تشغيل البرنامج (انظر شكل ٢١-١٩).

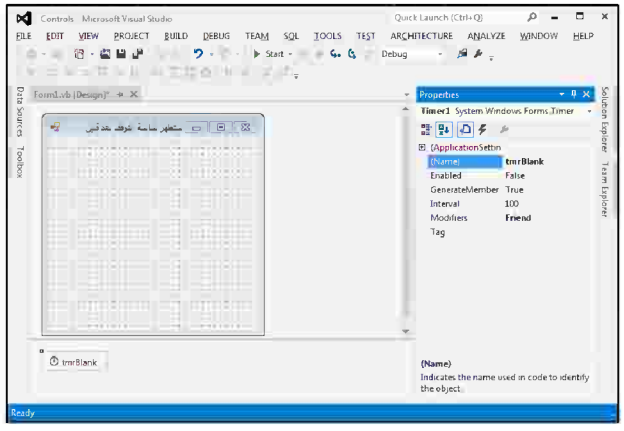
٣. قم بضبط الخاصية **Interval**بوضع القيمة **10000** مثلاً وهي تساوي عشر ثواني. قم أيضاً بتمكين المؤقت عن طريق تخصيص القيمة **True** للخاصية **Enabled**.

٤. قم بإضافةنموذج آخر إلى المشروع وليكن باسم **frmBlank** ثم قم بإعداد خصائصه كما يلى:

```
BackColor = Black
FormBorderStyle = None
WindowState = Maximized
```

وهذا يعني أن النافذة الثانية ستكون سوداء تشمل الشاشة كلها وبلا عنوان، أي أنها ستكون كستار يخفي الشاشة تماماً.

٥. قم بإدخال الكود التالى داخل الإجراء الحدثي **Tick** الخاص بالمؤقت:



شكل ٢١-١٩المؤقت بعد إضافته إلى النافذة.

```
Private Sub tmrBlank_Tick(sender As Object, e As EventArgs)
Handles tmrBlank.Tick
frmBlank.ShowDialog()
End Sub
```

٦. لاسترجاع محتويات الشاشة مرةً أخرى، ضع الكود التالى في الإجراء الحدثي **Click** والإجراء الحدثي **KeyPress** في النافذة الثانية **frmblank**:

```
Me.Close()
```

قم بتشغيل البرنامج، ولاحظ اختفاء محتويات الشاشة بعد عشر ثواني تماماً. فإذا قمت بنقر الشاشة أو ضغط أى مفتاح من لوحة المفاتيح، يتم الرجوع مرةً أخرى إلى النموذج الأول.

الجدير بالذكر أن الخاصية **Interval** يمكنها أن تأخذ القيم من **1** إلى **65,535**، وهي تكافئ ما يقرب من دقيقة واحدة، كيف إذاً يمكننا تنفيذ أحداث إذا أردنا الانتظار فترة زمنية أطول؟ يتم ذلك بطريقة شهيرة وهو استخدام متغير ساكن **Static** يعمل كعداد لعدد المرات التي تم فيها استدعاء الإجراء، مع ضبط الخاصية **Interval** على قيمة أساسية، مثلاً في المثال السابق لو أردنا لعملية حفظ الشاشة أن تتم بعد **10** دقائق بدلاً من **10** ثواني فعلينا أن نضبط **Interval** على القيمة **60000** أي دقيقة، ثم نعد **10** مرات لتنفيذ الإجراء كما في الكود التالي:

```
Private Sub tmrBlank_Tick(sender As Object, e As EventArgs)
Handles tmrBlank.Tick
```

```
Static counter As Integer
counter = counter +1
  If counter = 10 Then
frmBlank.ShowDialog()
counter = 0
    End If
End Sub
```

بالطريقة السابقة يمكن تنفيذ أحداث برغم طول الفترة الزمنية.