الباب الثانى

إنشاء تطبيقات باستخدام

Visual Basic 2008.

٤. إنشاء أول تطبيق نوافذى Windows Application.

٥. إنشاء أول تطبيق ويب.

الفصل الرابع
إنشاء أول تطبيق نوافذى
Windows Application

سنشرح في هذا الفصل أساسيات إنشاء تطبيق نوافذى باستخدام Visual Basic 2008 وخطوات تصميم وتخطيط التطبيق، ثم نشرح مثالا تطبيقيا يستخدم عددا من أدوات التحكم الموجودة داخل بيئة التطوير وبعض مفاهيم البرمجة .الهدف من هذا الفصل أن تعتاد على بيئة تطوير Visual Studio 2008 بالإضافة إلى إعطائك فكرة عن طريقة البرمجة بلغة Visual Basic 2008

بانتهاء هذا الفصل ستتعرف على:

- *الخطوات الأساسية لإنشاء تطبيق نوافذى*
- *إنشاء واجهة المستخدم (User Interface)*
- *كيفية تنفيذ التطبيق للمهام التي أنشئ من أجلها*
- *كيفية اختبار التطبيق*

يحب الكثير من المبرمجين أن يبدءوا –كما نقول– من النهاية، أي أن يكتبوا تطبيق بأسرع وسيلة وذلك عند تعلمهم للغة برمجة جديدة أو عند تعاملهم مع إصدار حديث من لغتهم المفضلة، وفي الحقيقة فإن هذه رغبة إيجابية لأنها تعلم المبرمج أن يكتب تطبيق كامل قابل للتشغيل دون انتظار اكتمال فهمه أو تعلّمه للّغة أو لبيئة التطوير الجديدة، لأن الواقع العملي يستدعي دائماً أن يكتب المطور بلغة لا يتقنها لدرجة الإجادة التامة، كما نفعل نحن حينما نتحدث اللغات البشرية دون أن نلم بكافة مفرداتها.

ونحن أيضاً سنفعل ذلك حيث سنعرض تطبيق كامل قابل للتشغيل، ومفيد في نفس الوقت، وهو تطبيق لحساب العمولة المستحقة للبائعين.

ربما تجد صعوبة في هذا الفصل والتطبيق الذي سنشرحه هنا، رغم بساطته الشديدة، وخصوصا إذا كنت مبتدئا أو لم تكن لك خبرة بإصدارات سابقة من **Visual Basic** ، مثال ذلك الأحداث، الاستجابة للأحداث ، توقيع الأدوات على الواجهات، التعامل مع الأدوات وعبارات التطبيق البسيطة ... كل هذه المفاهيم في مجملها هي مفاهيم **Visual Basic** ولذلك فان الفصول التالية من الكتاب ستشرح لك هذه المفاهيم بشئ من التفصيل والتبسيط الذي يحقق حاجتك، ويضعك على الطريق الذي تنشده. المطلوب منك في هذا المستوى من الدراسة متابعة الشرح لتفهم عمليا المراحل التى يمر بها التطبيق والمفاهيم التي يستخدمها **Visual Basic** .

تشتمل بيئة التطوير المتكاملة **Visual Studio 2008** على مترجم للغة **(Compiler)** وأداة تنقيح **(Debugger)** ومحرر لإدخال نص التطبيق **(Editor)** وأدوات مساعدة لرسم واجهة التطبيق متكاملة ومحتواة في تطبيق واحد، خلافا لأسلوب البرمجة القديم من خلال إصدار أمر من سطر الأوامر **(Command Line)** لإجراء الترجمة ثم عرض الأخطاء ومتابعة وتنقيح الأخطاء من خلال أداة منفصلة.

تصميم التطبيق

نظرا للطبيعة الهيكلية للبرمجة فإن ما تقوم به في لحظة يؤثر على كل ما يلي ذلك عند إنشائك للتطبيق، وفي الحقيقة ومن خلال الخبرة العملية فإن التراجع عدة خطوات إلى الوراء أمر محبط وقاسٍ أثناء البرمجة. وللأسف فإن كثيراً من المبرمجين يعرّضون أنفسهم لذلك بأن يكتبوا نص التطبيق Code دون أن تكتمل رؤيتهم له ودون أن تتبلور في أذهانهم عناصره الأساسية.

ما يهمنا هنا هو التأكيد على أهمية التصميم. أمسك بورقة وقلم واجلس لترسم واجهة المستخدم واكتب ملاحظاتك على تصورك لكيفية عمل هذه الواجهة وكيفية استجابتها للمستخدم، ستساعدك هذه الخطوة رغم بساطتها لدرجة كبيرة. لا نعني بالطبع أنه لن يكون هناك أخطاء، ولكن على أية حال فإن تطبيق به خطأ في التصميم أفضل من تطبيق ليس له تصميم أساساً.

خطوات إنشاء التطبيق

١. حدد مهام التطبيق، وحدد كيف يمكن إجرائها.
٢. حدد شكل وسلوك واجهة المستخدم.
٣. أكتب نص (كود) التطبيق الذي يترجم تصميمك.
٤. اختبر ونقح التطبيق.
٥. قم بتوثيق التطبيق، ونعني بالتوثيق شقين:
   - توثيقه لنفسك بكتابة تقرير مفصل عنه ووضع ملاحظاتك عليه.
   - توثيقه لغيرك بإنشاء ملف مساعدة Help وعرض توضيحي Demo إن أمكن.

لا تحتوي هذه القائمة على كل الخطوات بالطبع، إلا أنها تعطي فكرة أولية عن أقل ما يلزم.

أهمية التصميم

نظرا لأن متعة كتابة نص التطبيق ورسم واجهته هي الأكبر في الخطوات السابقة فإن الكثير يهمل ما قبلها وما بعدها من الخطوات ، إلا أن التصميم الجيد هو أساس نجاح أي مشروع برامجي، ونعني بالتصميم الجيد ما يتفتق عن النتائج التالية:

- قائمة بالمهام المنوطة بالتطبيق.
- جدول زمني لإتمام هذه المهام.
- وضوح اعتماد جزء من التطبيق على آخر (يؤثر ذلك على ترتيب كتابة الجزأين).
- خطة اختبار للتطبيق، ومعايير الفشل والنجاح عند الاختبار.

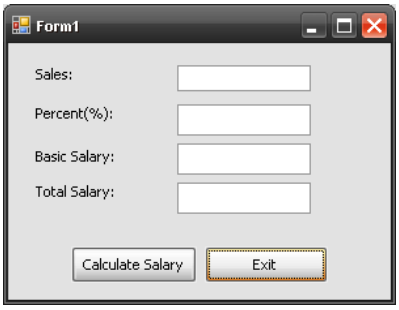
في التطبيقات التى من نوع تطبيقنا البسيط، قد تتلخص نتيجة التصميم في عبارة واحدة، أو معادلة إلا أن التطبيقات الأعقد قد يحتوي تصميمها على شروط وأسس مكتوبة، مخططات للبيانات، خرائط تدفق للبيانات **flow charts** ، ووثائق مكتوبة بدقة، وخطة للاختبار والقبول، وقد يكون ذلك في صورة عقود بين العملاء والمبرمج.

تصميم تطبيقنا

سنبدأ الآن مثالنا وهو عبارة عن تطبيق لحساب العمولة المستحقة للبائعين، وإضافتها لمرتباتهم الأساسية، وفي النهاية يحسب إجمالى دخل الموظف بإضافة المرتب الأساسي للعمولة المستحقة وتفصيل ذلك ما يلي:

١. يسمح التطبيق للمستخدم بإدخال قيمة المبيعات ونسبة العمولة والمرتب الأساسي.
٢. يمد التطبيق المستخدم بطريقةٍ ما لبدء الحسابات.
٣. قبل البدء في الحسابات يقوم التطبيق باختبار صلاحية المدخلات.
٤. يقوم التطبيق بحساب إجمالى دخل الموظف من خلال إضافة العمولة ويعرض النتائج للمستخدم.

يظهر التطبيق بعد الانتهاء من تصميمه كما فى شكل ٤-١ التالى.



شكل ٤-١ التطبيق البسيط المزمع إنشاؤه.

إنشاء واجهة التطبيق

بما أننا قد وضعنا التصميم فلنا أن نشرع في إنشاء التطبيق.ونظرا لبساطة التطبيق فلن نتحدث عن تصميم الواجهة حيث سندمجه في عملية إنشائها.

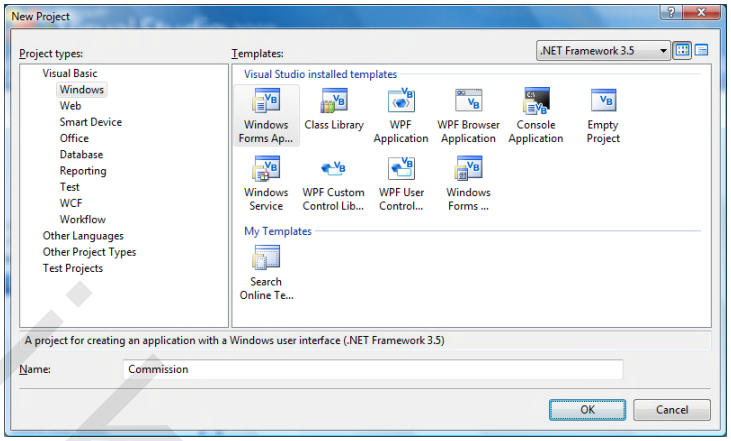
نظرا لبساطة التطبيق سنحتاج إلى ثلاثة فقط من عناصر التحكم **Controls** الموجودة في مربع الأدوات **Toolbox** وفيما بعد سنستخدم أدوات أكثر لتحسين أداء التطبيق.

ابدأ بتشغيل بيئة التطوير **Visual Studio 2008** كما تعلمت فى الفصل السابق ثم قم بإنشاء مشروع جديد باتباع الخطوات التالية:

١. من صفحة البدء **Start Page**، انقر الارتباط **Project** بالسطر **Create** أو افتح قائمة **File** من شريط القوائم واختر **New Project** من القائمة المنسدلة، يظهر المربع الحوارى **New Project** (انظر شكل ٤-٢).

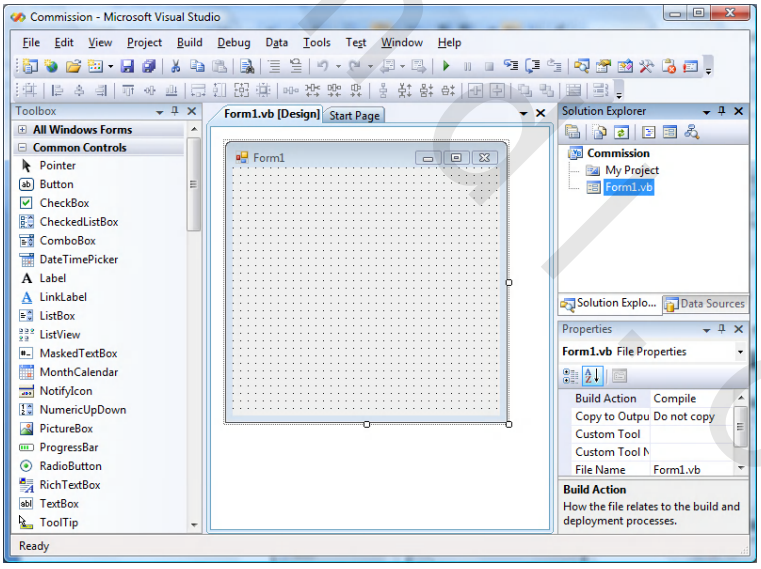
٢. من المجلد **Visual Basic** بالمربع **Project Types** بالجزء الأيسر من المربع الحوارى اختر **Windows** ثم اختر **Windows Forms Application** من المربع **Templates** بالجزء الأيمن منه.

٣. قم بتعيين اسم مميز للمشروع الجديد داخل مربع النص **Name** وليكن **Commission**.



شكل ٤–٢ المربع الحوارى **New Project** المستخدم فى إنشاء مشروع جديد.

٤. انقر زر **Ok**، تقوم بيئة التطوير بإنشاء مشروع جديد يحتوى على نموذج واحد باسم **Form1.vb** كما تقوم أيضاً بإنشاء ملفات المشروع نيابةً عنك (انظر شكل ٤–٣).

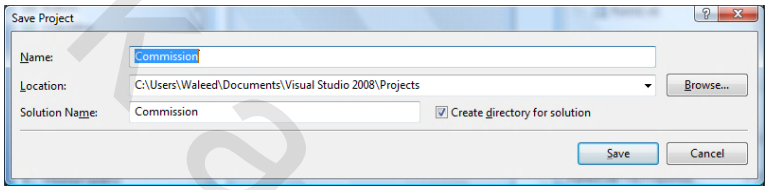


شكل ٤–٣ تقوم بيئة التطوير المتكاملة بإنشاء المشروع الجديد نيابةً عنك.

بالنظر إلى نافذة المستكشف الموجودة بالشكل السابق، تلاحظ احتوائها على قائمة بالعناصر الموجودة بالتطبيق، حيث تحتوى على نموذج واحد فقط باسم **Form1.vb**.

حفظ التطبيق

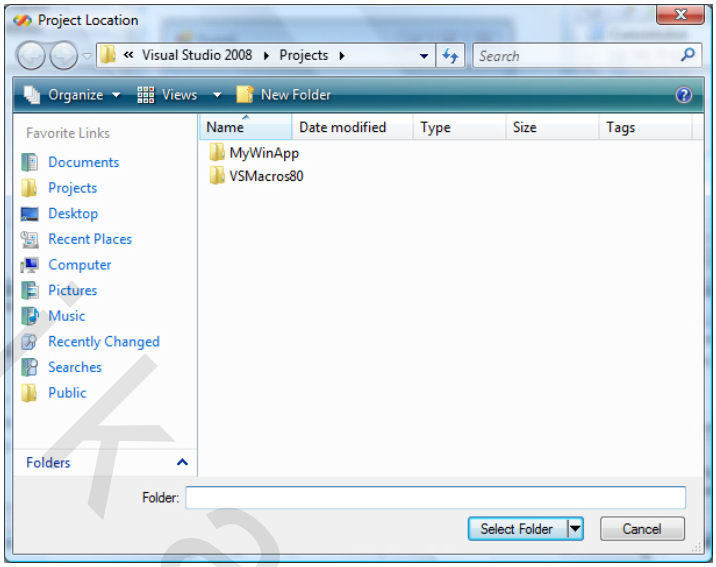
لحفظ المشروع على القرص الصلب كى تتمكن من استرجاعه فى أى وقت والتعديل فيه، افتح قائمة **File** بشريط القوائم ثم اختر **Save All** من القائمة المنسدلة الناتجة، وحينئذٍ يظهر المربع الحوارى **Save Project** (انظر شكل ٤-٤). قم بتعيين بيانات المشروع كما يلى:



شكل ٤-٤ يجب أن تقوم بحفظ المشروع قبل البدء فى إدخال التعديلات عليه

لحفظ المشروع فى أى وقت، افتح قائمة **File** ثم اختر **Save** لحفظ الكائن الحالى فقط أو **Save All** لحفظ المشروع بالكامل.

- قم بتغيير اسم المشروع إن أحببت من خلال مربع النص **Name**.
- قم إن أحببت بتغيير مكان المشروع من خلال القائمة المنسدلة **Location** التى تحتوى بدورها على المجلد الافتراضى لتخزين المشروعات وهو **C:\Users\username\Documents\Visual Studio 2008\Projects** (مع استبدال **username** باسم المستخدم الحالى) إلى جانب أى مجلدات قمت باختيارها من قبل. فإذا أردت اختيار مجلد آخر لحفظ المشروع، انقر زر **Browse** المجاور للقائمة المنسدلة ثم قم بتحديد المجلد من المربع الحوارى الناتج **Project Location** (انظر شكل ٤-٥).



شكل ٤-٥ يمكنك من خلال هذا المربع تحديد المجلد المستخدم فى تخزين المشروع الجديد

- يتم تخزين المشروع داخل حل Solution الذى قد يحتوى بدوره على أكثر من مشروع، حيث يتم افتراضياً تسمية الحل بنفس اسم المشروع. فإذا أردت تعيين اسم آخر للحل، قم إدخال الاسم الجديد إلى مربع النص Solution Name. ويمكنك التحكم فى اسم الحل فى حالة تنشيط مربع الاختيار المجاور Create directory for solution والذى يتم تنشيطه إذا ما أردت إنشاء مجلد مستقل للحل لتمييزه عن الحلول الأخرى الموجودة بنفس المجلد.

٥. بمجرد تعيين الخيارات المختلفة للمشروع والحل، انقر زر Save لإغلاق المربع الحوارى Save Project وحفظ المشروع الجديد والعودة مرةً أخرى إلى النافذة الرئيسية لبيئة التطوير.

ويمكنك فى أى وقت حفظ ملفات المشروع أثناء عملك تحسبا لانقطاع التيار أو حـــدوث أي مشكلة في Windows أو في Visual Studio 2008 قد تتسبب في اضـــطرارك لإعادة تشغيل الجهاز مما قد يضيع ساعات من العمل هدرا.

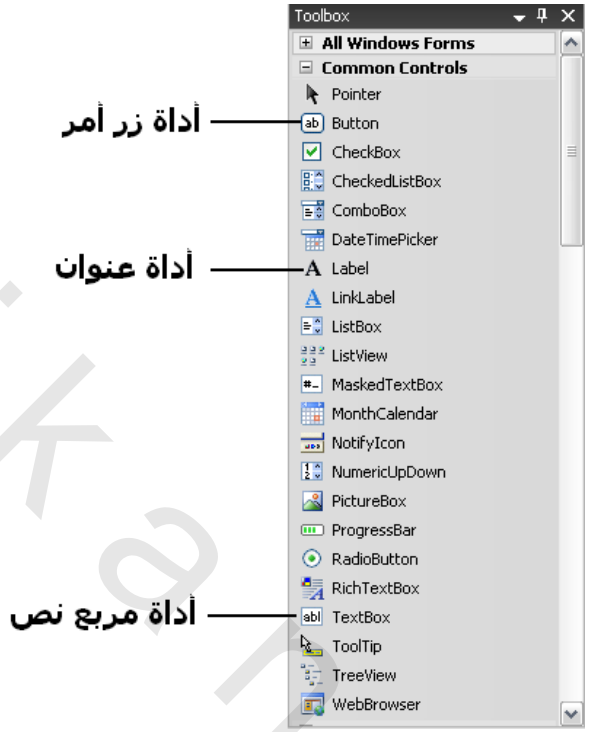
ننصح بإنشاء مجلد خاص للأمثلة أو البرامج التي تنشئها. مثلا أنشئ المجلد D:\Exercises كى تضع عليها الأمثلة التى تقوم بإنشائها من خلال هذا الكتاب.

كيف تحصل على مدخلات المستخدم

تحتاج معظم البرامج إلى طريقة يستطيع المستخدم من خلالها إدخال البيانات المطلوبة، وهناك الكثير من أدوات التحكم التى تصلح لإدخال البيانات داخل **Visual Basic** .

في البداية، انظر إلى مربع الأدوات لترى أي الأدوات تحتاج إليها، إن واجهة المستخدم لتطبيقنا مسئولة عن تلقي المدخلات، واستدعاء الدالة (الوظيفة) التي ستنفذ عملية الحساب، وأخيراً عرض النتائج، وعليه فالأدوات الثلاثة التي سنحتاجها هى مربع النص **Text Box** وأداة العنوان **Label** وزر الأمر **Command Button**.

هذه الأدوات جزء من الفئة الأساسية من أدوات التحكم التي تجدها في مربع الأدوات عند بدء تشغيل **Visual Studio 2008** كما يظهر في شكل ٤-٦.



شكل ٤-٦ نافذة مربع الأدوات ويظهر بها الفئة الأساسية من عناصر التحكم

أكثر الأدوات وأعمها استخداما هو مربع النص **Text Box** ويشبه هذا المربع المكان الذي تملأه في استمارة طلب وظيفة أو ما شابه ذلك، فربما تجد فى استمارة مربع أو خط منقط بعد عبارة الاسم الأول وتقوم أنت بكتابة الاسم داخل المربع أو فوق الخط المنقط، وهى نفس وظيفة مربع **TextBox** في **Visual Basic**.

ويمكنك ببساطه تغيير موقع المربع على نموذج النافذة أو تغير مظهره (بتغير الألوان أو الخط مثلا) كل ذلك دون أن تكتب سطرا واحداً من الكود، لك أن تتخيل كم وفر علينا وجود أداة جاهزة كمربع النص، فلو أردت محاكاة سلوك هذه الأداة باستخدام لغة أخري من اللغات القديمة لاستغرق ذلك عملا أكثر من يومين أو ثلاثة.

ولتتخيل، فلابد أن تأخذ في الاعتبار ما تقوم به هذه الأداة خلف الكواليس أثناء تشغيل التطبيق.

- يقوم مربع النص بالتقاط واستقبال الحروف التي يدخلها المستخدم من لوحة المفاتيح.
- يضع الحرف في مكانه المناسب من الشاشة أو النافذة على وجه التحديد.
- يتعامل مع أوامر الحذف، التراجع، تحريك المؤشر.
- يمكنه إدراج النصوص أو الكتابة على نص ثابت.
- يخزن المدخلات النصية في الذاكرة (بجانب ما تعرضه الشاشة) ليستطيع التطبيق قراءتها.

إضافة أدوات التحكم إلى نموذج النافذة

قبل أن نبدأ الشرح هنا يجدر بنا التوقف قليلا أمام نموذج النافذة لنفهم معناه. نموذج النافذة هو اسم يطلق على النافذة حال تصميمها ورسمها ويشتمل على نص التطبيق الذي ينفذ العمليات المختلفة المسئولة عن العناصر المحتواة في هذه النافذة.

أي أن نموذج النافذة هو الشكل المرسوم في مساحة التصميم والذي يمثل النافذة كما ستظهر عند التشغيل بالإضافة إلى كود التطبيق المرتبط بهذه النافذة.

وينبغي أن يكون واضحاً أن الشكل المرسوم للنموذج ليس هو النافذة لأننا مثلا لو رسمنا زر وحاولنا أن نضغط عليه لن يستجيب كما نتوقع من التطبيق، وإذا حاولنا الكتابة مباشرة في مربع النصوص فلن نستطيع لأن ما يظهر أمامنا مجرد (رسم) وليس نافذة تفاعلية حقيقية ولا تظهر خصائص النافذة بالفعل إلا عند التشغيل، وفي خلال بقية الكتاب ربما استخدم كلمة نموذج فقط أو نافذة فقط عند عدم احتمال سوء الفهم.

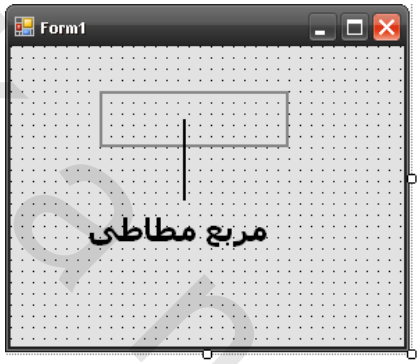
إضافة مربع نص

لكي تضيف أداة تحكم إلى تطبيقك، لابد أولا أن تضيفها إلى نموذج النافذة. أول أداة سنضيفها إلى تطبيق حساب العمولة هي مربع النص. لإتمام ذلك اتبع الخطوات التالية:

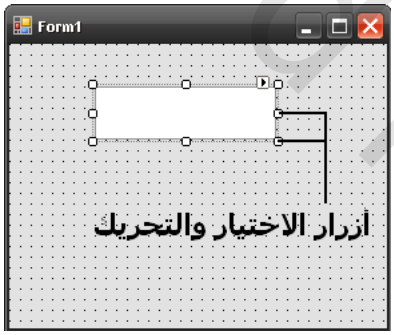
١. اختر أداة مربع النص TextBox من مربع الأدوات بالنقر عليه (يتم ضغط الأداة قليلا لأسفل دلالةً على اختيارها).

٢. حرك الفأرة إلى النموذج ثم ابدأ برسم مربع باستخدام المؤشر في الحجم والمكان الذي تريد وضع مربع النص فيه (بالسحب أثناء النقر) تلاحظ أن شكل المؤشر يتحول إلى علامة (+) أي خطين متقاطعين أثناء طور الرسم.

٣. أثناء السحب سيظهر مربع مطاطي متقطع الإطار (انظر شكل ٤-٧). قم بتحديد الحجم المطلوب للأداة ثم قم بتحرير زر الفأرة، تحصل على مربع النص مرسوماً فوق النموذج (انظر شكل ٤-٨).



شكل ٤-٧ شكل النموذج أثناء رسم مربع نص عليه بالفأرة.



شكل ٤-٨ شكل النموذج بعد إدراج مربع النص عليه.

يمكنك بسرعة إضافة أداة إلى النموذج بالنقر المزدوج على رمز الأداة في مربع الأدوات، هذه العملية تضع أداة بحجم افتراضي في وسط النموذج. يمكنك بعد ذلك تحريك وتغير حجم أداة التحكم باستخدام المقابض أو الأذرع المحيطة

بالمربع كما سيلي فيما بعد.

ضبط خصائص عنصر التحكم

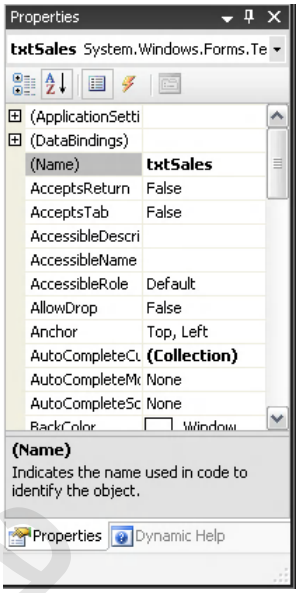
بعد إضافة مربع النص إلى نموذجك، تحتاج إلى ضبط بعض خصائصه. تذكر أن الخصائص تتحكم في مظهر عنصر التحكم أو في مظهر الكائن بوجه أعم وفي سلوكه. بالنسبة لمربع النص السابق نحتاج ضبط خاصيتين فقط من خصائصه وهما **Name** و**Text** .

(الخاصية **Name** ومعناها الاسم) خاصية هامة جدا وهي موجودة لكل عناصر التحكم على الإطلاق كما هي للنماذج وللكائنات عموما وهي تستخدم في التطبيق لتمييز العنصر حتى يسهل التعامل معه، فإن لم تعلم اسم العنصر فكيف يمكنك كتابة تطبيق يتعامل معه ؟.

ولكن ماذا لو أنك نسيت إعطاء اسم لعنصر ما ؟ يقوم **Visual Basic** بإعطائه اسم مميز، فمثلا لو وضعت مربع نص ؟ سيطلق عليه **TextBox1** وإذا أضفت مربع نص آخر سيسميه **TextBox2** وهكذا، ويمكنك استخدام هذه الأسماء الافتراضية ولكنها عادة سيئة في البرمجة، حيث أن الاسم المعبر سهل الحفظ والمتابعة عند حدوث الأخطاء، فمثلا لو أن لديك ثلاثة مربعات نصوص لإدخال **First name** و **Last name** و **Address** في أحد البرامج فإن أسماء مثل **txtFname** و **txtLname** و **txtAddress** أفضل من **TextBox1**و **TextBox2** و **TextBox3** .

الأسماء المقترحة تبدأ كما ترى بالبادئة **txt** والبادئة المكونة من ثلاثة حروف شائعة ومفيدة عند تسمية أدوات التحكم ومن الأسماء التي سنقابلها قريبا **lbl** الخاصة بمربعات التسمية (العنوان) و**cmd** الخاصة بأزرار الأوامر وهي كما ترى مشتقة من أسمائها الإنجليزية.

لتغيير اسم مربع النص، اذهب إلى خاصية **Name** في نافذة الخصائص وفي المساحة المقابلة لها اكتب **txtSales** وهو المربع الذي سيحوي المبيعات (انظر شكل ٤-٩).



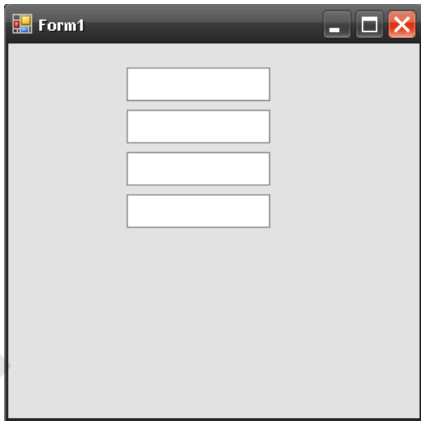
شكل ٤-٩ نافذة الخصائص حيث يمكن تغيير خصائص أي أداة تحكم

الخاصية الأخرى التي نريد تغييرها هي **Text** ومعناها النص وأثناء تشغيل التطبيق ستحتوى هذه الخاصية على النص أو الحروف المكتوبة فعليا في مربع النص والقيمة الافتراضية التي يعطيها **Visual Basic** لمربع النص هى القيمة **Null** والتى تعنى عدم وجود أى بيانات بالمربع، لتغيير هذه القيمة اذهب إلى الخاصية **Text** في نافذة الخصائص واكتب ما تشاء من الحروف لتظهر للمستخدم عند بدء التشغيل.

تذكر أننا ما نزال حتى الآن في طور التصميم وما أن نبدأ تشغيل التطبيق لن نستطيع تغيير هذه الخصائص إلا من خلال الكود.

إضافة مربعات النصوص الباقية

باستخدام الخطوات السابقة أضف إلى نموذج النافذة ثلاثة مربعات نصوص أخرى بأسماء **txtPercent**و **txtSalary** و**txtTotal**. بعد قيامك بالخطوات السابقة سيبدو النموذج كما ترى في شكل ٤-١٠.



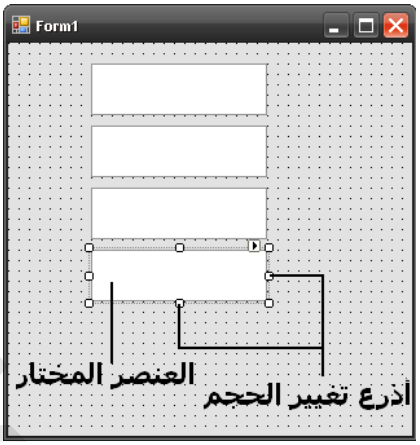
شكل ٤-١٠ نموذج النافذة بعد إضافة مربعات النصوص الأربعة.

لرسم عدة عناصر من نفس الأداة قم بالضغط على مفتاح **Ctrl** عند اختيارك للأداة من مربع الأدوات، سيمنع هذا عودة الوضع إلى أداة المؤشر **(pointer tool)** وهي الأداة الافتراضية التي يعود الوضع إليها بعد كل عملية رسم لأداة تحكم.

تحريك وتغيير حجم مربعات النصوص

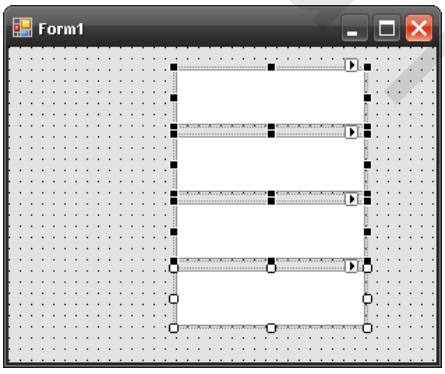
لتحريك أي من مربعات النصوص السابقة قم بالنقر عليها ثم اسحبها (مستمرا في ضغط زر الفأرة) إلى المكان الذي تريده.

تغيير حجم مربع النصوص مماثل لتغيير حجم أي نافذة في **Windows** وذلك بنقره أولا لاختياره ثم سحب أحد المقابض الثمانية (المربعات الصغيرة) الموجودة على أضلاع وأركان مربع النصوص كما يبدو في الشكل ٤-١١. المقابض الموجودة فى أركان المربع لتغيير حجمه أفقياً ورأسياً فى نفس الوقت والمربعات الموجودة على أضلاع المربع لتغيير الحجم أفقياً أو رأسياً حسب اتجاه السهم.



شكل ٤-١١ مقابض الأداة تتيح لك تغيير الحجم والموقع بالفأرة.

الآن، وبعد رسمنا مربعات النصوص ينبغي أن نضع عنوان لكل مربع حتى يدرك المستخدم المحتويات المطلوبة في كل منهم، ولنفسح مكانا للعناوين فلنقم بتحريك المربعات إلى اليمين قليلا، بالطبع تستطيع أن تحركها واحدا واحداً ولكننا سنقوم بطريقة أفضل بتحريكهم جميعا في مرة واحدة، للقيام بذلك قم بالضغط على مفتاح **Ctrl** واستمر في الضغط عليه أثناء قيامك بالنقر على المربعات واحدا تلو الآخر حتى يتم اختيارهم جميعا، ثم قم بسحبهم كمجموعة إلى اليمين قليلا حتى يبدو النموذج قريباً من الشكل ٤-١٢.



شكل ٤-١٢ تحريك مجموعة من عناصر التحكم مرة واحدة بعد تحديدهم

تحتوى بعض أدوات التحكم على مقبضين فقط نشطين بينما تظهر المقابض الستة الأخرى باهتة وهذا يرجع فى الواقع إلى تخصيص القيمة **True** افتراضياً للخاصية **AutoSize** وهذا يعنى تغير حجم مربع النص تبعاً للنص المدخل، لذا فلا حاجة لاستخدام بقية المقابض وبالتالى تظهر باهتة.

عنونة المدخلات

كما أشرنا سابقا، فإننا بحاجة إلى إشارة ظاهرة للمستخدم تخبره بنوعية المعلومات المرادة في كل مربع نصي (لا تنس أن الأسماء المعطاة سابقا للمربعات تستخدم فقط في كود التطبيق ولا تظهر على الشاشة ولا يعلم عنها المستخدم شيئا)، أسهل الطرق أن نضع أداة عنوان (يطلق عليها أيضاً "مربع تسمية") **Label** ملاصقة للمربع النصي بها الكلمات المراد الإشارة بها إلى المحتوى.

الفرق الأساسي بين مربع النص والعنوان أن العنوان لا يستطيع المستخدم تعديله أثناء تشغيل التطبيق كما يفعل مع محتويات مربع النص.

لكي تضيف أحد العناوين إلى نموذجك، اتبع الخطوات التالية:

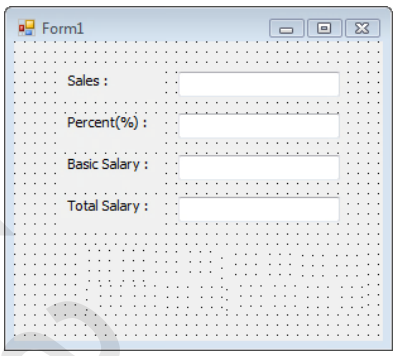
١. اختر أداة مربع التسمية أو العنوان Label A من مربع الأدوات.
٢. ارسم مربع لاحتواء العنوان مجاور لأعلى مربع نصي في النموذج.
٣. غيّر اسم العنوان (أي غير خاصية **Name**) إلى **lblSales**.
٤. غيّر التعليق الخاص بالعنوان (خاصية **Text**) إلى**"Sales:"** ، لا تنس النقطتان في نهاية الكلمة.

بإتباع نفس الخطوات يمكننا أن نضيف العناوين الثلاثة الباقية بجوار مربعات النصوص المناظرة مع اختلاف الاسم والتعليق الخاص بكل منها، يحتوى الجدول ٤-١على خصائص العناوين الأربعة التي سننشئها. بعد الانتهاء ستبدو النافذة كما ترى في شكل ٤-١٣.

جدول ٤-١خصائص العناوين التي توضح مدخلات التطبيق

الاسم Name	العنوان Text
lblSales	Sales :

Text العنوان	Name الاسم
Percent(%) :	lblPercent
Basic salary :	lblBasic
Total Salary :	lblTotal



شكل ٤-١٣ نموذج النافذة بعد إضافة العناوين لتعريف مربعات النصوص.

لغير المتخصصين نقول فضلنا كتابة تعليقات مربعات النصوص باللغة الإنجليزية من بـــاب التسهيل باعتبار أن هذا التطبيق هو أول تطبيق تقوم بتجربته. وذلك لأن الكتابة الإنجليزية هي الوضع الافتراضي في Visual Basic رغم أنه بإمكانك كما ســـترى في الفصـــول القادمة كتابة التعليقات والنصوص التي تظهر على الواجهة أو النموذج باللغة العربية. في هذه الحالة يلزمك تغيير لغة لوحة المفاتيح عن طريق ضغط مفتـــاحي Alt+Right shift للتبديل إلى اللغة العربية أو Alt+Left shift للتبديل إلى اللغة الإنجليزية. ولذلك فإننا نرى إتماما للفائدة أن نبين فيما يلي معاني الكلمات الإنجليزية التي تظهر على النموذج.

Sales معناها قيمة المبيعات، Percent معناها نسبة العمولة ، Basic Salary معناها أساس المرتب، و Total Salary هو إجمالي المرتب بعد إضافة العمولة.

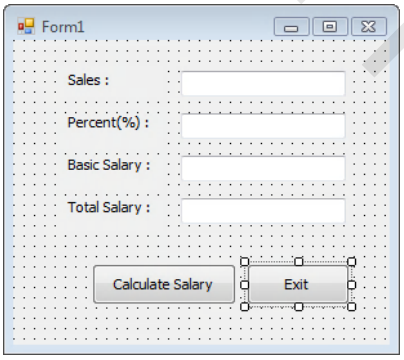
لكي تحاذي العناوين إلى اليسار لتجاور مربعات النصوص كما يبـــدو في شـــكل ٤-١٣ السابق، اختر العناوين كلها ثم انقر زر المحاذاة إلى اليسار من شريط التخطيط الموجود أسفل شريط الأدوات (إذا لم يظهر الشريط أمامك، انقر شريط الأدوات بزر الفأرة الأيمن ثم اختر Layout من القائمة الموضعية الناتجة).

إضافة زر أمر Command Button

آخر ما نحتاج من العناصر هو زر الأمر، وهو مسئول عن بدء عملية ما. تماما كوظيفة الزر في حياتنا العادية (زر جرس الباب، زر تشغيل الآلات..إلخ.). وهو يوضع كما توضع العناصر الأخرى بالرسم على النموذج باستخدام الفأرة.

ابدأ بالنقر على الأداة Button في مربع الأدوات لاختيارها، ارسم الزر الأول على النافذة وامنحه اسما مميزا وليكن **cmdCalculate** كما فعلت بالعناصر السابقة ، أيضا للزر خاصية **Text** مثل العنوان وتحوي الكلمات التي ستظهر على سطح الزر وغالبا ما تعبر عما يقوم به التطبيق عند الضغط على الزر، ضع التعليق **Calculate Salary** في الخاصية **Text** الخاصة بالزر. ارسم زرا آخر، سمّه **cmdExit** وأعطه التعليق **Exit** .سيظهر نموذج النافذة حينئذٍ كما بالشكل ٤-١٤.

من الأفضل الآن فى هذه المرحلة أن تقوم بحفظ عملك تحسباً لانقطاع التيار أو حدوث خطأ ما. لأداء ذلك، افتح قائمة **File** من شريط القوائم ثم اختر **Save All** من القائمة المنسدلة أو اضغط الزر  مباشرةً من شريط الأدوات.



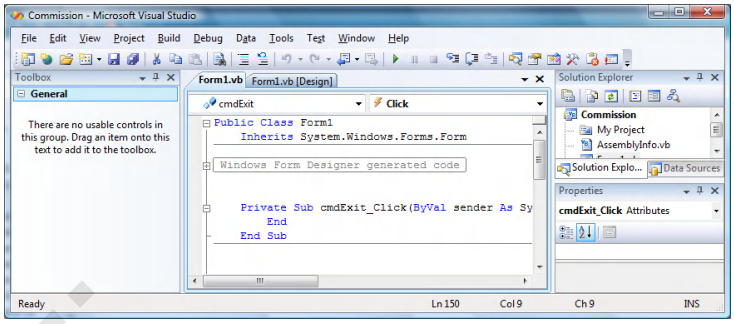
شكل ٤-١٤نموذج النافذة بعد إضافة أزرار الأوامر.

تنفيذ المهام

عند هذه النقطة اكتملت واجهة التطبيق، إلا أنك لو شغّلته لن يقوم بأي وظيفة، صحيح أنك تستطيع كتابة ما تريد في مربعات النصوص من أرقام ولكن لن يقوم التطبيق بأية حسابات لإيجاد النواتج المرغوبة وهنا تكمن وظيفة كود التطبيق. في الحقيقة لن يقوم التطبيق بأية وظائف بطريقة أوتوماتيكية أو يقوم بإجراء الحسابات في الوقت الذي يحدده ، إن البدء في مهمة ما لابد إن يرتبط بحدث Event (مثلاً النقر بالفأرة) هذا الحدث يخص كائناً ما على النافذة (مثل الزر المسمي CmdCalculate). يتيح لك Visual Basic كتابة إجراء ما Procedure يتم تنفيذه عند النقر على زر معين أو أي حدث لأي كائن آخر (الكائن مفهومه أوسع من أداة التحكم حيث أن الشاشة والطابعة يعتبران كائنات من وجهة نظر Visual Basic بينما ليسا بأدوات تحكم). يسمى هذا النوع من الإجراءات بالإجراءات الحدثية Event Procedures .

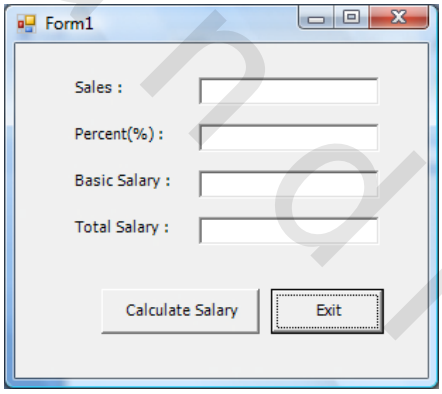
لنبدأ بالإجراء الأسهل وهو الخاص بالخروج من التطبيق، والمرتبط بالنقر على الزر cmdExit. لكي تكتب هذا الإجراء انقر نقرا مزدوجا على زر cmdExit في شاشة التصميم، ستظهر نافذة الكود محتوية على هيكل جاهز لإجراء النقر Clicking. يتكون الهيكل من سطر بداية ونهاية للإجراء كما في الشكل ٤-١٥. يفترض Visual Basic عند النقر المزدوج على زر أمر أنك ستكتب إجراءا ينفذ في حالة النقر عليه أثناء التشغيل ، لذا يقوم بإظهار الهيكل السابق والمسمى cmdExit_Click. بين السطرين الظاهرين من الهيكل ، اكتب الآتى كما يبدو من شكل ٤-١٥ (يمكنك استخدام Tab قبلها لتنسيق الشكل نوعا ما):

```
Private Sub cmdExit_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdExit.Click
    End
End Sub
```



شكل ٤-١٥ نافذة الكود وبها هيكل جاهز لكتابة الإجراء

يمكننا الآن أن نشغل التطبيق لأول مرة. اضغط الزر **Start** من شريط الأدوات، أو افتح قائمة **Debug** من شريط القوائم ثم اختر **Start Debugging** من القائمة المنسدلة أو اضغط المفتاح **F5** ، وفى جميع الحالات يظهر نموذج التطبيق (انظر شكل ٤-١٦). بعد التشغيل، انقر زر **Exit** لكي تخرج من التطبيق.



شكل ٤-١٦ النموذج أثناء تشغيل التطبيق

استثارة حدثٍ ما

في **Visual Basic** يقوم التطبيق بالأفعال عادةً استجابة لحدث ما. فيما مضى قمت بتعريف الإجراء المراد استجابةً للحدث **Click** المرتبط بالزر **cmdExit** . الحدث هو ما يطرأ على كائن نتيجةً لفعل المستخدم أو لفعل من قِبَل جزء آخر من التطبيق أو لفعل من قبل نظام التشغيل. من أمثلة الأحداث **Events** أن يضغط المستخدم مفتاحاً أو يحرك

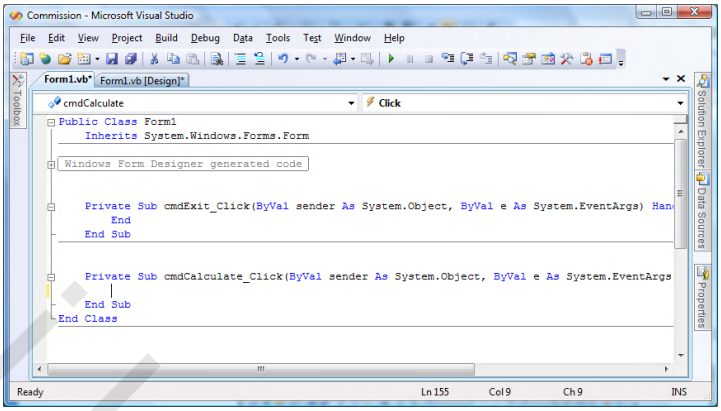
الفأرة، أو يغير قيمة في أحد أدوات التحكم، أو مرور فترة زمنية محددة. عندما يستحث المستخدم حدث النقر Click للزر cmdExit يقوم التطبيق بتنفيذ الإجراء المرتبط بهذا الحدث والذي يتكون من كلمة End فقط التي تنهي التطبيق.

كل أداة تحكم تستخدمها في تطبيقك أعدت لاستقبال أو تحسس أنواع محددة من الأحداث. بعض الأدوات تستجيب لنقر الفأرة بينما تستجيب بعضها لتغيرات في قيم بعض خصائصها. لو أردت لأداة أن تستجيب لحدث معين لا بد أن تضع كود في الإجراء الحدثي الخاص بهذه الأداة وإلا سيتجاهل تطبيقك هذا الحدث تماماً.

لكل أداة من أدوات التحكم الموجودة داخل بيئة التطوير Visual Studio 2008 حدثها الافتراضي الذى يظهر عند النقر المزدوج للأداة داخل النموذج. فعند نقر زر الأمر على سبيل المثال نقراً مزدوجاً، يتم فتح نافذة الكود على حدث نقر هذا الزر كما رأينا منذ قليل.

إذا نظرت أعلى نافذة الكود ستلاحظ مربعان منسدلان. الأيسر يستعرض كل الكائنات الموضوعة علي نموذج النافذة بما فيهم النافذة Form نفسها، بينما الأيمن يستعرض الأحداث المتاحة للكائن المختار حالياً في المربع الأيسر. لو اخترت cmdExit من المربع الأيسر ثم استعرضت محتويات المربع الأيمن ستجد أن زر الأمر ممكن أن يستجيب لأحداث أخرى كثيرة غير حدث النقر Click. في الحقيقة يمكنك أن تفكر في تطبيقك كصفحة طويلة من الكود تقوم نافذة الكود بمساعدتك في التنقل عبر إجراءاتها المختلفة.

أسهل الطرق لإدخال الكود أن تنقر نقراً مزدوجاً على أداة التحكم التي تريدها أن تستجيب لأحد الأحداث. ستفتح لك نافذة الكود حيث تقوم باختيار الأداة أوتوماتيكياً من المربع الأيسر كما يبدو من شكل ٤-١٧.



شكل ٤-١٧ فتح نافذة الكود بمجرد النقر المزدوج على الزر **cmdCalculate** .

يمكنك أيضا فتح نافذة الكود بضغط مفتاح F7 أو بنقر زر إظهار الكود من نافذة المستكشف Solution Explorer أو بفتح قائمة View واختيار Code من القائمة المنسدلة.

الإجراء الحدثي التالي الذي سنكتبه هو الخاص بإجراء الحسابات وهو الخاص بالنقر على زر Calculate وهو الإجراء cmdCalculate_Click وتلاحظ أنه مكون من اسم أداة التحكم والحدث المراد الاستجابة له. هذا الكود هو لب التطبيق.

كتابة تعليمات (Code) التطبيق

قبل أن نجري حساباتنا علينا أن نخبر Visual Basic عن ماهية المتغيرات التي سنجري عليها الحسابات، إخبار مترجم اللغة بأنماط المتغيرات Variable types تسمى عملية "الإعلان عن المتغيرات Variable declaration "ولابد أن تسبق كتابة جمل التطبيق التنفيذية.

الإعلان عن المتغيرات

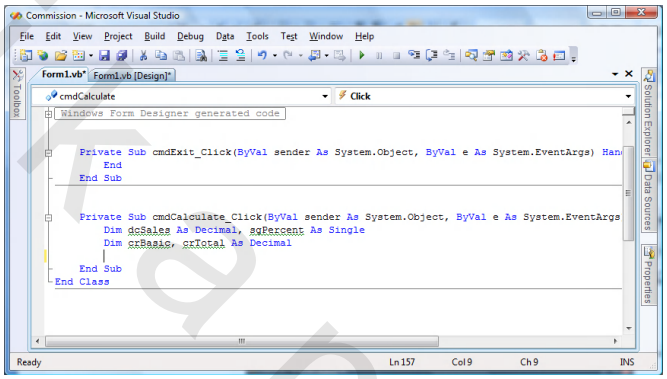
يعني الإعلان عن المتغيرات إخبار Visual Basic بأسماء المتغيرات التي ستستخدمها وأنماط البيانات التي ستحتويها هذه المتغيرات مثل integer, Date, string... ورغم

عدم احتياج لغة Visual Basic لهذه الخطوة بالضرورة إلا أنها هامة للغاية ولذلك فهي إجبارية في كافة لغات الكمبيوتر الهامة وسيتبين فائدتها فيما يلي.

سيتكون مقطع الإعلان من الأسطر التالية:

```
Dim dcSales As Decimal, sgPercent As Single
Dim crBasic , crTotal As Decimal
```

ومن المفترض أن تبدو على نافذة البرمجة كما في الشكل ٤-١٨.



شكل ٤-١٨ نافذة الكود عند إضافة مقطع الإعلان عن المتغيرات.

كما تلاحظ تتكون صيغة الإعلان عن متغير من الكلمة Dim يليها اسم اختياري للمتغير يليها As ثم نمط المتغير ، ويمكن الإعلان عن المتغيرات في أسطر منفصلة أو في نفس السطر كما ترى.

كما في تسمية عناصر التحكم يستحسن أن نتبع الطريقة القياسية في التسمية، هذه الطريقة القياسية تتكون من حرفان كبادئة تعبر عن نمط المتغير ثم اسم المتغير، في مثالنا السابق يتكون dcSales من dc وهي اختصار لـ Decimal ثم Sales ومعنى dcSales هنا أنه متغير من نمط القيم العشرية (المالية) يحمل قيمة المبيعات.

في الإصدارات التي تسبق Visual Basic.NET يتم تعريف كل متغير في سطر مستقل وهو ما يعني مصاحبة كل متغير لنوع بياناته وإلا يكون نوع البيانات من النوع العام Variant. ففي الإصدارات السابقة وفي العبارة التالية على سبيل المثال:

Dim x, y, z As integer

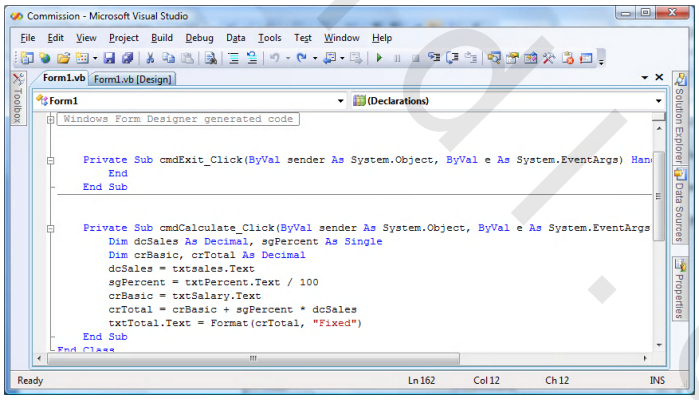
يتم اعتبار z فقط من النوع integer أما x,y فسيعتبران من النوع Variant.

أما فى Visual Basic 2008 فيتم اعتبار المتغيرات الثلاثة x, y, z من النوع integer.

كود الإجراء

كود الإجراء هو العبارات التي تقوم بالحسابات الفعلية، سوف يشتمل هذا الإجراء على قراءة المدخلات من مربعات النص الثلاثة الأولى ثم حساب المرتب الإجمالي بعد إضافة العمولة وعرضه في المربع المحدد، انظر نص الإجراء فيما يلي وشكله عند كتابته في نافذة الكود في شكل ٤-١٩.

```
dcSales = txtSales.Text
sgPercent = txtPercent.Text /100
crBasic = txtSalary.Text
crTotal = crBasic + sgPercent * dcSales
txtTotal.Text = Format(crTotal,"Fixed")
```



شكل ٤-١٩شكل نافذة البرمجة عند كتابة إجراء الحسابات.

والآن تعال نوضح باختصار ماذا يفعل هذا الإجراء:

- تم جلب قيم المدخلات وتخزينها في متغيرات
- العمليات الحسابية تمت باستخدام المعاملات الحسابية

- قيمة المتغير **crTotal** وهي الناتج تم تشكيلها (بشكل النقطة العشرية الثابتة) وإظهارها في المربع النصي الأخير.

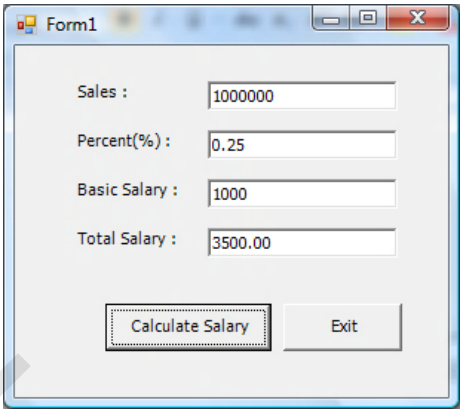
تلاحظ أن قراءة قيمة خاصية لأداة تحكم أو تعديلها يتم بالإشارة إليها باسم مثل **txtPercent.Text** ويحتوي هذا الاسم على اسم العنصر أو الكائن ثم نقطة ثم اسم الخاصية ويسمى ذلك بالتسمية المنقوطة (**Dot notation**)، باستخدام هذه الطريقة في التسمية يمكن القراءة من أو الكتابة في أي خاصية لأي كائن على نافذة التطبيق ، للتعامل مع كائن في نموذج نافذة آخر أضف اسم النموذج مع نقطة قبل اسم الكائن، مثل **Form1.txtPercent.Text**.

تشغيل التطبيق

لتشغيل التطبيق اضغط زر ▶ من شريط الأدوات، أو افتح قائمة **Debug** من شريط القوائم ثم اختر **Start Debugging** من القائمة المنسدلة أو اضغط المفتاح **F5** مباشرةً من لوحة المفاتيح، تقوم بيئة التطوير في البداية بترجمة التطبيق لاكتشاف الأخطاء وإنشاء النسخة التنفيذية منه داخل مجلد التطبيق، ومن ثمَّ يتم تشغيل التطبيق (راجع شكل ٤-١٦). للاختبار أدخل القيم الآتية في المربعات:

Sales **1000000**
Percent **0.25**
Basic Salary **1000**

ثم انقر زر **Calculate Salary** ينبغي أن تكون قيمة **Total Salary** المعروضة مساوية **٣٥٠٠** (انظر شكل ٤-٢٠). بعد اختبار التطبيق بعدة قيم يمكنك العودة لبيئة التطوير بنقر زر **Exit** داخل النموذج أو زر **Stop** من شريط الأدوات.



شكل ٤-٢٠ نموذج التطبيق فى طور التشغيل.