الفصل الثانى

البرمجة الموجهة بالكائنات

١-٠ تمهيد

يركز هذا الفصل على التعريف بمفهوم البرمجة الموجهة بالكائنات، من حيث بداياته، وأهم الأساليب المرتبطة به. كما يلقى الفصل الضوء على بعض المفاهيم الأساسية المرتبطة بلغة Visual Basic، وفيه نظرة تاريخية إلى تطور عائلة BASIC وإصداراته المختلفة، والتحول الجوهرى الذى طرأ عليها من خلال إطار عمل .NET ، إلى غير ذلك من الموضوعات ذات الصلة. وفى الفصل الثالث سوف نلقى الضوء على بيئة التطوير المتكاملة/ التفاعلية Integrated/Interactive Development Environment.

الأهداف التعليمية

بنهاية هذا الفصل يجب أن يكون الدارس قادرا على:

- تعريف مفهوم البرمجة الموجهة بالكائنات OOP
- تحديد مراحل تطور عائلة Visual Basic
- تمييز إصدارات Visual Basic المختلفة
- إدراك الآثار الناتجة عن التحول نحو إطار عمل .NET.

٢ـ١ المفاهيم الأساسية للبرمجة الموجهة بالكائنات

يعتبر أسلوب البرمجة الموجهة بالكائنات Object-Oriented Programming (OOP) من أساليب البرمجة الحديثة نسبيا، والتى تختلف عن أساليب البرمجة التقليدية. ونظرا إلى أهمية هذا الموضوع كاتجاه متنام، فسوف نفرد له الجزء التالى. وسوف نركز فقط على تقديم المفاهيم والمصطلحات المرتبطة بهذا الأسلوب. ولا يتوقع منك أن تفهم "بشكل دقيق" كيف تعمل البرمجة الموجهة بالكائنات؛ فحتى المبرمجون المحترفون professional programmers قد يحتاجون لأشهر عدة للوصول إلى هذا الفهم.

٢ـ١ـ١ ما الكائن؟

فكر فى الكائنات التى نتعامل معها فى حياتنا اليومية، كإطار السيارة أو القطة، على سبيل المثال. ما الحقائق facts التى نعرفها عن هذه الكائنات؟ يمكن القول – ببساطة شديدة – إن الإطار شكله دائرى ولونه أسود، أما القطة فلها أربع أرجل وفروة مميزة تغطى جسمها. وإذا أردنا أن نحدد مزيدا من التفاصيل لهذه الكائنات، يمكن أن نحدد الوظائف functions التى تقوم بها. فالإطار يدور ويتوقف أو يفقد الهواء الموجود داخله. كما أن القطة تأكل وتخرخر purr (خرير القط عبارة عن صوت يصدره القط للتعبير عن سعادته) وتعوى howl (صوت القط العالي).

فى عالم البرمجة الموجهة بالكائنات، يحتوى الكائن object على الشيء ذاته، بالإضافة إلى الحقائق facts والوظائف functions المرتبطة به. وبشكل أكثر دقة – فى مجال البرمجة – يتم تعريف الكائن بأنه وحدة ذاتية المحتوى self-contained unit

تحتوى على كل من البيانات data والحقائق والوظائف المرتبطة بها – أى التعليمات instructions التى تنفذ على هذه البيانات. وهذا يتناقض مباشرة مع البرمجة التقليدية، والتى تحدد (فى البرنامج) إجراءات procedures منفصلة عن البيانات.

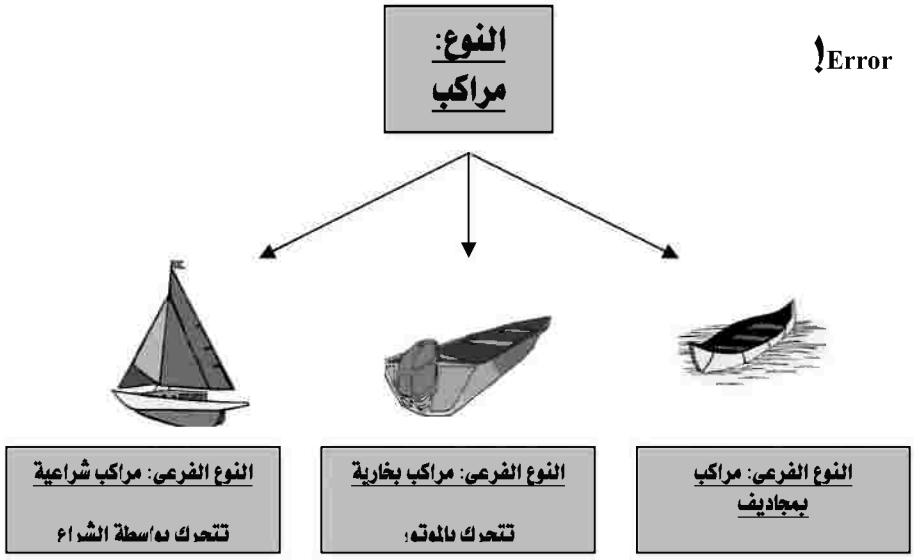
تتم الإشارة إلى خاصية ذاتية محتوى الكائن بمصطلح عدم الكبسلة (الحفظ والتخزين) encapsulation. بمعنى أن الكائن يقوم بحفظ وتخزين البيانات والتعليمات المرتبطة بها. كما يطلق على الحقائق المرتبطة بالكائن: خصائص attributes ، والتعليمات التى تحدد للكائن ما الذى يفعله: طرق methods أو عمليات operations. كما قد توجد أكثر من حالة instance للكائن؛ فالقط نيمو حالة من حالات الكائن قط.

٢-١-٢ البدايات: المراكب ككائنات

ترجع بدايات البرمجة الموجهة بالكائنات إلى عام ١٩٦٩، وذلك عندما كان الدكتور كريستن نيجارد Kristin Nygaard يحاول تطوير نموذج كمبيوتر للمراكب التى تمر فى الخلجان النرويجية الصغيرة Norwegian fjords .وعندما كان نيجارد يستجمع المكونات المعقدة من الأمواج ومصبات الأنهار والسواحل غير الممهدة والمراكب المتحركة، تفتق ذهنه عن فكرة عزل كل مكون من هذه المكونات فى عناصر مستقلة – أو كائنات – ثم تحديد العلاقات فيما بينهم جميعا. افحص جديدا شكل (٢-١). يتكون الكائن مركب من الكائن ذاته، وخصائصه، وطرقه- توصيف الأشياء التى يفعلها، كالعوم أو الغرق. ومن الجدير بالذكر فى هذا السياق أن نشير إلى إمكان حدوث تغيير فى وضع الكائن بواسطة طرق من خارجه؛ أى أن الكائن قد يتأثر بطرق الكائنات المحيطة به.

وباستخدام البرمجة الموجهة بالكائنات، يقوم المبرمجون بتحديد أنواع classes الكائنات. يحتوى كل نوع على مجموعة من السمات characteristics التى تميز النوع عن غيره من الأنواع الأخرى. فى شكل (٢-١) على سبيل المثال، يعتبر كائن مركب حالة من النوع مراكب. وبالإضافة إلى الأنواع، قد تنتمى الكائنات إلى أنواع فرعية

subclasses. حيث ترتب الكائنات – طبقا لسماتها الفريدة – فى شكل هرمى يتكون من الأنواع والأنواع الفرعية. ففى شكل (٢-١) يعتبر كل من المركب الشراعى واللنش والقارب (ذى المجدافين) أنواعا فرعية للكائن مركب.



شكل (٢-١) كائنات النوع والنوع الفرعى.
تتوارث الأنواع الفرعية، كالمراكب الشراعية والبخارية وذات المجاديف، سمات أو خصائص العموم والفرق من النوع الأعلى (مراكب). هذا بالإضافة إلى أن كل نوع فرعى – وفقا لمبدأ تعدد الأشكال polymorphism – يمكنه الاستجابة للرسالة (تحرك) مستخدما طرقه الخاصة.

كل كائن يندرج تحت نوع فرعى يمتلك تلقائيا جميع سمات النوع المتفرع منه؛ ويطلق على هذه الخاصية مصطلح التوارث أو الوراثة inheritance. فكائن النوع الفرعى: القارب (ذى المجدافين) – على سبيل المثال – يمتلك سمات القدرة على العوم والقابلية للغرق، والمتوارثة من نوع الكائن الأعلى: مراكب، بالإضافة إلى سماته الخاصة به، كالحاجة إلى تحريك المجاديف. ومن ثم فلا داعى لتكرار سمات النوع ضمن الأنواع الفرعية. وذلك يعنى عدم قيام المبرمج بتكرار استخدام السمات المتوارثة؛ الأمر الذى يوفر الوقت والجهد.

هذا ومن الممكن تحقيق مزايا عدة من جراء إعادة استخدام الكائنات. فالمؤسسة التى تتبنى أسلوب البرمجة الموجهة بالكائنات تقوم تدريجيا ببناء مكتبة من الأنواع library of classes. وبمجرد إنشاء نوع ما واختباره والتأكد من صلاحيته، يمكن إعادة استخدامه مرات كثيرة فى مشاريع البرامج المستقبلية. وذلك راجع إلى أن هذه الكائنات عبارة عن وحدات ذاتية المحتوى self-contained units (أى تتمتع بخاصية الحفظ والتخزين / عدم الكبسلة encapsulation)، فلن تحتاج المؤسسة إلى إجراء أية تعديلات عليها لاستخدامها فى التطبيقات المستقبلية. وهذا من شأنه أن يقلل من الأخطاء بدرجة كبيرة؛ حيث يمكن بناء البرامج الجديدة من الأنواع الخالية من الأخطاء والتى سبق اختبارها فى برامج سابقة. وبالطبع لن تتمكن المؤسسات من جنى ثمار إعادة استخدام الكائنات إلا بعد تنفيذ عدد من المشاريع البرمجية.

٢-١-٣ تنشيط الكائن

إذا كان الكائن وحدة ذاتية المحتوى، كيف، إذن، تستحثه على القيام بشىء ما؟ يتم ذلك بإرسال أمر يطلق عليه رسالة message للكائن وذلك من خارج الكائن ذاته. توضح الرسالة ما المطلوب عمله، أما كيفية عمل ذلك فقد يكون موجودا ضمن طرق methods الكائن. على سبيل المثال: يمكن إطلاق الرسالة "تحرك move" إلى الكائنات التى تنتمى إلى النوع الفرعى مراكب: مراكب شراعية – لنشات – قوارب (ذات مجاديف). وهذا يدفعنا إلى تقديم مصطلح آخر من المصطلحات المرتبطة بتكنولوجيا الكائنات الموجهة، وهو تعدد الأشكال polymorphism. عند إطلاق رسالة ما، فإن خاصية تعدد الأشكال تسمح لكائن محدد يستقبل الرسالة بأن يعرف كيف يقوم بالاستجابة لها بطريقة مناسبة، وفقا للطرق الخاصة به. على سبيل المثال: عند استقبال الرسالة "تحرك move" ، يعرف الكائن: "مركب شراعي" أنه يفترض أن يتحرك بمساعدة الشراع. أما الكائن: "لنش" فيعرف أنه يتحرك بواسطة المحرك. فى حين يعرف الكائن: "قارب" (ذو

المجدافين) أنه يتحرك عن طريق المجاديف. فى كل حالة يحتاج الكائن فقط لاستلام أمر التحرك، فيستجيب إلى طرقه المبنية داخله.

٢-٢ تطور عائلة Visual Basic كنموذج للغة برمجة كائنية

يبلغ عمر Visual Basic for Windows أكبر قليلا من عشر سنوات. فقد دشن للمرة الأولى فى ٢٠ مارس ١٩٩١ فى عرض أطلق عليه "عالم النوافذ Windows World".

وقد كان ذلك بمثابة ضجة فى أوساط المهتمين. فقد وصفه Steve Gibson بأنه أحد "العجائب الجديدة الرائعة"، التى سوف "تغير بشكل جذرى الطريقة التى يتعامل بها الناس مع برامج ميكروسوفت للنوافذ". كما اعتبره Stewart Alsop أنه "أفضل بيئات البرمجة فى التسعينيات".

وكما بدأ عقد التسعينيات من القرن المنصرم بهذه الضجة، كذلك بدأت الألفية الجديدة مع ظهور Visual Basic .NET الذى يختلف تماما عن أسلافه، كما اختلفت الإصدارة الأولى من VB for Windows عن سلفه QuickBasic. وبينما لا نستطيع أن ننكر أن الخبرة التى اكتسبها المبرمجون فى بيئة VB for Windows يمكن أن تفيدهم بشكل أو بآخر فى منظومة عمل .NET الجديدة، فإن هناك الكثير من التغيرات التى طرأت على Visual Basic من خلال هذه المنظومة.

٢-٣ إصدارات Visual Basic

تميزت الإصدارتان الأوليان من عائلة VB for Windows بقدرتهما على بناء نماذج تجريبية من التطبيقات prototypes and demo applications لكن لا شيء سوى ذلك. احتوت هاتان الإصدارتان على بيئة تطوير متكاملة/ تفاعلية IDE ممتازة، إضافة إلى لغة برمجة سهلة التعلم. لكن هذه اللغة كانت ذات إمكانات محدودة.

ومع الإصدارة الثالثة VB3 ظهر لأول مرة إمكان الوصول إلى قواعد

البيانات. فى حين تميزت الإصدارة الرابعة VB4 بقدرة محدودة على تكوين أهداف objects وكانت هذه بداية البرمجة الموجهة بالأهداف فى عائلة Visual Basic. وهو الاتجاه ذاته الذى استمر خلال الإصدارتين التاليتين؛ الخامسة VB5 والسادسة VB6.

ومع بداية الألفية الثالثة، أدرك القائمون على VB الحاجة إلى إدخال المزيد من الإضافات إلى هذه اللغة حتى تصبح متوافقة تماما مع البرمجة الموجهة بالأهداف. ومن هنا كان إطار عمل .NET Framework

٢ـ٤ التحول نحو إطار عمل .NET

لقد أحدث استخدام إطار عمل .NET تغيرا كبيرا فى طريقة تصميم البرامج، تعادل –بل تفوق – التغير الذى حدث عندما أتيحت الفصائل classes فى الإصدارتين VB5&6. تعال معى ـ عزيزى القارئ ـ لنرى بعض التغيرات التى يجب الانتباه إليها – أو الإفادة منها – عند التحول من VB6 إلى VB.NET:

لغة وقت التنفيذ العامة The Common Language Runtime

أحد أكبر وأهم التغيرات التى طرأت على VB من خلال منظومة .NET. هى CLR التى يمكن استخدامها مع أى من لغات .NET. وهو الأمر الذى يعنى أن الكود يمكن استخدامه فى أى لغة من هذه اللغات. على سبيل المثال، يمكن استخدام كود برنامج كتب بواسطة لغة VB بواسطة مبرمجين يعملون بلغة C#، والعكس صحيح.

إضافة إلى ذلك، هناك لغة وسيطة لتنفيذ كود لغات .NET. يطلق عليها لغة ميكروسوفت الوسيطة Microsoft Intermediate Language أو اختصارا MSIL أو فقط IL. وهى لغة وسيطة تتولى تحويل الكود الذى تم تجميعه بواسطة CLR إلى الشكل الذى يستطيع الكمبيوتر التعامل معه، وذلك وقت التنفيذ.

البرمجة الكاملة للأهداف Completely Object Oriented

إن التوجه نحو البرمجة الموجهة بالأهداف فى VB5&6 كان محدودا بدرجة كبيرة. على سبيل المثال، لم تكن هاتان الإصدارتان قادرتين على التعامل مع البيانات داخل الفصيلة class من دون إنشاء حالة instance من الفصيلة أولا.

أضف إلى ذلك أن هاتين الإصدارتين كانتا تفتقدان التوارث بمعناه الفعلى inheritance. والتوارث هذا عبارة عن شكل من أشكال إعادة استخدام الكود، يتيح استخدام أهداف objects تمثل حالات مخصصة من أهداف أخرى موجودة بالفعل.

التنفيذ المتزامن للتعليمات Multithreading

والمقصود بذلك قدرة البرنامج الواحد على تنفيذ عمليات مختلفة فى وقت واحد. وهى طريقة تهدف إلى الإسراع بتنفيذ تعليمات الكود فى حالة استنفاد أحد الأجزاء لوقت طويل فى التنفيذ، بينما توجد هناك حاجة إلى تنفيذ مهام أخرى فى ذات الوقت. وهى أحد الإمكانات التى أتاحتها أداة التجميع الجديدة CLR.

مراجعة شاملة لأنواع البيانات Completely Revised Types

أولا تم صرف النظر عن دعم البيانات من نوع Variant والذى تم تقديمه فى Visual Basic. بمعنى أنه لا يستخدم إطلاقا فى بيئة NET. الجديدة. هذا فضلا عن أن NET. يدعم بشدة عملية التحقق checking من نوع البيانات. كما تم إضافة مدى أكبر للبيانات من نوع Integer و Long. أما المصفوفات Arrays فأساسها الصفر (بدلا من ١).

إدارة الذاكرة Memory Management

يقصد بهذا المفهوم قدرة لغة البرمجة على التحقق مما إذا كانت هناك حاجة إلى كائنات البرنامج، مثل المتغيرات variables، وذلك بهدف استعادة مساحات

التخزين التى تشغلها، عندما تصبح عديمة الاستخدام. وعلى الرغم من أنه يمكن القيام بذلك من خلال الكود، إلا أن NET.، يقوم بذلك بطريقة مختلفة، دون الاستعانة بالكود، وهو ما لم يكن ممكنا فى VB 6.

تحميل البرامج Program Installation

يمكن وضع عنوان مختلف لهذه الفقرة، وهو: استبعاد البرامج من نوع DDL فى بيئة NET. فنظرا إلى الطريقة التى يستخدمها NET. ولتتبع أجزاء البرنامج program modules ، يمكنك تحميل البرنامج الجديد فقط عن طريق نسخ copying الملفات فى مجلد folder.

كائنات الجرافيكس الجديدة GDI+

لن نستطيع أن نغطى هذا الموضوع بعجالة فى فقرة مختصرة. إلا أنه يمكننا أن نقول إن بيئة NET. قد قدمت عالما جديدا للجرافيكس. أحد الأمثلة التى توضح ذلك هو إمكان إنشاء نماذج شفافة transparent forms.

أداة قواعد البيانات ADO.NET

يصف مبرمجو قواعد البيانات بيئة NET. بأنها أفضل البيئات التى قدمتها مايكروسوفت، من حيث دعم تقنية ADO الخاصة بإدارة البيانات.

المعالجة الهيكلية للاستثناءات Structured Exception Handling

استخدمت جميع إصدارات Visual Basic طرقا معينة للتعامل مع الأخطاء، إلى درجة أن ذلك يعود إلى الإصدارة الأولى مذ أربعين عاما تقريبا. إلا أن الطريقة التقليدية التى اعتمدتها الإصدارات الأخير – وهى طريقة OnError GoTo – لم تعد مقبولة فى لغات البرمجة الحديثة (الموجهة بالأهداف). وقد عالجت منظومة NET هذه المشكلة بطريقة هيكلية سوف نتعرف عليها بالتفصيل فى الفصول التالية من هذا الكتاب.

ملخص

تعرفنا فى هذا الفصل على مفهوم البرمجة الموجهة بالكائنات، وهو الأسلوب المتبع فى جميع لغات البرمجة الحديثة. وقد حاولنا إلقاء الضوء على بداياته، وأهم المفاهيم المرتبطة به، مثل مفهوم التوارث، والكبسلة، وتعدد الأشكال. كما ألقينا الضوء عبر نظرة تاريخية إلى تطور عائلة BASIC وإصداراته المختلفة. ثم ختمنا باستعراض لأهم ملامح التحول الجوهرى الذى طرأ على هذه العائلة من خلال إطار عمل NET.

وبنهاية هذا الفصل، نكون قد انتهينا من الأسس النظرية، على أن نبدأ من الفصل التالى التعرف على بيئة البرمجة المتكاملة فى لغة VB 2005 Express Edition.

تمارين

١. ما المقصود بالبرمجة الموجهة بالأهداف؟.

٢. تتم الإشارة إلى خاصية ذاتية محتوى الكائن بمصطلح .....

٣. مع الإصدارة الثالثة VB3 ظهر لأول مرة إمكان (اختر واحدة):

a. تنفيذ تطبيقات الويب.

b. الوصول إلى قواعد البيانات.

c. تصميم الرسوم المتحركة.

٤. لا يستخدم نوع البيانات Variant فى لغة VB.NET (صح أم خطأ).

٥. ناقش أهمية لغة وقت التنفيذ العامة The Common Language Runtime فى منظومة .NET.