**بسم الله الرحمن الرحيم**

**الحمد لله رب العالمين وأفضل الصلاة وأتم التسليم على سيدنا محمد وعلى آله وصحبه أجمعين**

**سبحانك اللهم لا علم لنا إلا ما علمتنا إنك أنت العليم الحكيم**

**الأهداف التعليمية من هذا الدرس :**

1- كيفية التصريح ( declare ) عن الأصناف واستخدامها لإنشاء الأغراض.
2- كيفية تنجيز ( implement ) سلوك الأصناف عن طريق الوسائل.
3- كيفية تنجيز واصفات الصنف كمتحولات غرض ( instance variable ) وخصائص ( properties ).
4- كيفية استدعاء وسائل الأغراض لإنجاز مهامها.
5- الفرق بين متحولات الغرض ( instance variable ) والمتحولات المحلية ( local variable ).
6- استخدام الباني ( constructor ) لإنشاء أغراض وإعطائها قيم ابتدائية ( initializing ) .
7- الفرق بين أنماط القيمة ( value type ) وأنماط المرجع ( reference type ).

## 1.4 : مقدمة :

تعلمنا في درسٍ ماضٍ المفاهيم الأساسية في البرمجة الغرضية التوجه ( OOP ) ، وبدأنا أيضاً باستخدام تلك المفاهيم لبناء برامج بسيطة تعرض رسالة للمستخدم ، وتحتفظ بمعلومات معينة ، وتنفذ عمليات حسابية ، وتتخذ قرارات منطقية.

أكثر ما يميز تلك البرامج المكتوبة في الدرس الثالث هو أن عباراتها كلها مكتوبة ضمن وسيلة واحدة هي الوسيلة `Main()`.

في البرامج الحقيقية ، أنت سوف تطور تطبيقات تتألف من أكثر من صنف وكل صنف يحتوي على أكثر من وسيلة ، وإذا أصبحت جزءاً من فريق لتطوير البرمجيات في شركة ما فربما سوف تعمل على تطبيقات تحتوي على المئات من الأصناف أو حتى الآلاف منها.

## 2.4 : التصريح عن الصنف كتاب العلامات ( GradeBook ) وإنشاء غرض منه:

سوف نبدأ هذا الدرس ببرنامج يتألف من صنفين الأول هو GradeBook والثاني هو GradeBookTest.

الصنف GradeBook سوف يُستخدم لعرض رسالة ترحيب بالمستخدم ، والصنف GradeBookTest هو صنف للتجربة حيث يحتوي على الوسيلة Main() التي فيها سيتم إنشاء واستخدام غرض من الصنف GradeBook.

بالاصطلاح : سوف نصرح عن الصنفين GradeBook و GradeBookTest في ملفين منفصلين ، وطبعاً يكون اسم كل ملف هو نفس اسم الصنف الذي يحتويه.

```
1  // Class declaration with one method.
2  using System;
3
4  public class GradeBook
5  {
6      // display a welcome message to the GradeBook user
7      public void DisplayMessage()
8      {
9          Console.WriteLine("Welcome to the Grade Book!");
10     } // end method DisplayMessage
11 } // end class GradeBook
```

الشكل 1.4 التصريح عن صنف كتاب العلامات بوسيلة واحدة

يحتوي هذا الصنف على وسيلة واحدة وهذه المرة ليست الوسيلة Main() بل هي الوسيلة DisplayMessage() والتي تقوم بعرض رسالة ترحيبية على الشاشة.

نعود فنقول إن الصنف هو تماماً كالمخطط الهندسي لصناعة السيارة ونحن بحاجة للتعامل معه لإنشاء غرض منه واستدعاء الوسيلة DisplayMessage() لتنفيذ السطر 9 لتعرض عبارته بدورها الرسالة على الشاشة ، وبالفعل ففي الصنف GradeBookTest الشكل 2.4 قد فعلنا ذلك.

التصريح عن الصنف GradeBook يبدأ بالسطر 4 والكلمة المفتاحية public ندعوها معرف وصول ( access modifier ) ، حيث يمكننا باستخدام هذه المعرفات تحديد إمكانيات الوصول من قبل الأصناف الأخرى للوسائل والخصائص وأعضاء الصنف الأخرى.

يجب أن يحتوي التصريح عن الصنف الجديد على الكلمة المفتاحية class متبوعة باسم الصنف ويكون جسم الصنف محدداً بالقوسين المعقوفين ( { } ).

التصريح عن الوسيلة DisplayMessage() يبدأ بالكلمة المفتاحية public التي تشير بأن هذه الوسيلة "متاحة للعموم" أي أنه يمكن استدعاؤها من خارج الصنف عن طريق الأغراض المنتسخة من هذا الصنف.

الكلمة المفتاحية void تعرف بما يسمى نمط إرجاع ( return type ) الوسيلة ، وتشير هذه الكلمة إلى أن هذه الوسيلة لن تقوم بإرجاع ( return ) أي معلومات للوسيلة التي قامت باستدعائها وذلك طبعاً بعد إنهاء مهمتها.

وكمثال على الوسيلة التي ترجع نتيجة ما ، عندما نذهب إلى نافذة الصراف الآلي ( ATM ) ونطلب منه المبلغ المتبقي في حسابنا ( account balance ) فنحن نتوقع من الصراف أن يعطينا القيمة التي تدل على المبلغ المتبقي في حسابنا عنده ، وأيضاً إذا كان لدينا الوسيلة Square() التي تحسب لنا مربع العدد الذي يمرر لها كمعامل فإذا كتبنا مثل هذه العبارة :

```
int result = Square(2);
```

فنحن سوف نتوقع أن تعيد لنا الوسيلة Square() القيمة 4 التي هي مربع العدد 2 ومن ثم نقوم بإسناد هذه القيمة 4 إلى المتحول result.

اسم الوسيلة DisplayMessage يتبع نمط إرجاعها ( السطر 7 ) ، وبشكل عام تكون أسماء الوسائل كأفعال أو جمل فعلية بينما تكون أسماء الأصناف ( كمفردات لغة ) هي أسماء.

بالاصطلاح : اسم الوسيلة يبدأ بحرف كبير وتكون الأحرف الأولى للكلمات المؤلفة للاسم أيضاً أحرف كبيرة ( تدوين باسكال ) مثل : SampleMethodName() والقوسين اللذان يوجدا بعد اسم الوسيلة يشيران إلى هذا التصريح على أنه تصريح عن وسيلة ، وإذا كانت فارغة كما في السطر 7 فهذا يعني أن الوسيلة لا تطلب أي معلومات إضافية للقيام بمهامها.

نسمي الأسطر مثل السطر 7 ترويسة الوسيلة ( method header ) وكما في الصنف كل جسم وسيلة يحدد بقوسين معقوفين ( { } ) ، وجسم الوسيلة DisplayMessage() يحتوي على عبارة واحدة تعرض رسالة ترحب بالمستخدم متبوعة بمحرف الهروب ( \n ) الذي يعني وضع مؤشر الكتابة في سطر جديد.

الآن نريد أن نستخدم الصنف GradeBook في برنامجنا ، وكما تعلمنا في الدرس الماضي أن الوسيلة Main() هي وسيلة خاصة تستدعى تلقائياً عند تنفيذ البرنامج وهي نقطة الدخول للبرنامج ولذلك لا يمكننا في الوضع الحالي للصنف GradeBook أن ننفذ البرنامج لأنه لا يحتوي على الوسيلة Main().

إذاً لحل هذه المشكلة إما أن ننشأ ملفاً جديداً منفصلاً يحتوي على الوسيلة Main() أو أن نصرح عنها في صنفنا الحالي.

ولمساعدتك لتجهيز نفسك للتطبيقات الأكبر التي ستواجهك في هذه السلسلة أو في العمل بعد ذلك سوف نقوم بإنشاء ملف منفصل ونسميه نفس اسم الصنف الذي سيحتويه GradeBookTest ( لا ننسى هنا أن لاحقة ملف رماز السي شارب هي cs ) والذي سيحتوي على الوسيلة Main() التي ستكون مهمتها التعامل مع صنف كتاب العلامات.

```
1  // Create a GradeBook object and call its DisplayMessage method.
2  public class GradeBookTest
3  {
4      // Main method begins program execution
5      public static void Main()
6      {
7          // create a GradeBook object and assign it to myGradeBook
8          GradeBook myGradeBook = new GradeBook();
9
10         // call myGradeBook's DisplayMessage method
11         myGradeBook.DisplayMessage();
12     } // end Main
13 } // end class Addition
```

```
Welcome to the Grade Book!
```

الشكل 2.4 إنشاء غرض من الصنف كتاب العلامات واستدعاء وسيلة منه

التصريح عن الصنف GradeBookTest يبدأ بالسطر 2 وينتهي بالسطر 13 ، ويحتوي هذا الصنف على الوسيلة Main() فقط وسوف نستخدمه لتنفيذ البرنامج.

السطور 5 ... 12 نصرح فيها عن الوسيلة Main() ، والكلمة المفتاحية static هي التي تعطي لهذه الوسيلة أهميتها ( السطر 5 ) والتي تشير أن الوسيلة Main() هي وسيلة صنفية (مشتركة ) ( static method ) وهذا النوع من الوسائل خاص ويمكن استدعاؤه عن طريق الصنف مباشرة ودون الحاجة لإنشاء غرض من ذلك الصنف ( سوف نتعلم عن الوسائل الصنفية أكثر في درس قادمة إن شاء الله تعالى ).

في هذا البرنامج نريد أن نستدعي الوسيلة DisplayMessage() التي تنتمي إلى الصنف GradeBook لتعرض لنا رسالة في نافذة المترجم السطري.

بشكل عام ، أنت لن تستطيع استدعاء وسيلة تنتمي لصنف آخر ما لم تنشأ غرضاً من ذلك الصنف ( باستثناء الوسائل الصنفية ) كما في السطر 8 فقد قمنا بالتصريح ( declare ) عن المتحول myGradeBook ونمط هذا المتحول هو GradeBook وهو الصنف الذي عرفناه مسبقاً.

كل صنف جديد نقوم بإنشائه يصبح نمطاً جديداً في سي شارب بحيث يمكن استخدامه للتصريح عن متحولات ( variables ) أو لإنشاء أغراض منه.

يمكننا تعريف أصناف جديدة كلما دعت الحاجة إلى ذلك ولهذا السبب توصف السي شارب بأنها لغة قابلة للتوسع ( extensible language ).

المتحول myGradeBook في السطر 8 أخذ قيمة ابتدائية ( initialized ) من نتيجة التعبير new GradeBook ();.

العملية new ( new operator ) تستخدم لإنشاء أغراض جديدة من الصنف الموجود على يمينها مثل GradeBook والقوسان الموجودان بعد اسم الصنف مطلوبان.

وكما سوف نتعلم لاحقاً في هذا الدرس أن هذه الأقواس بالإضافة لاسم الصنف يمثلان ما يسمى بالباني ( constructor ) وهو شبيه بالوسيلة ولكنه يستخدم فقط عند إنشاء الغرض ويمكننا بواسطته تعيين قيم ابتدائية للغرض المنشأ.

نستطيع الآن بعد إنشاء المتحول الغرض myGradeBook من الصنف GradeBook أن نستخدمه لاستدعاء الوسيلة DisplayMessage().

السطر 11 يستدعي الوسيلة DisplayMessage() (الشكل 1.4 والسطور 7 ... 10 ) باستخدام المتحول الغرض myGradeBook متبوعاً بنقطة الوصول للأعضاء ( . ) ( member access operator ) واسم الوسيلة وقوسان صغيران.

إن هذا الاستدعاء للوسيلة DisplayMessage() يختلف عن استدعاء الوسائل في الدرس السابق حيث أن كل تلك الاستدعاءات كانت تزود بمعاملات ( arguments ) تحدد البيانات التي سوف تعرض ، بينما في السطر 11 الغرض myGradeBook يشير للوسيلة Main() أنها يجب أن تستخدم غرضاً من الصنف GradeBook والمنشأ في السطر 8 ، والقوسان الفارغان في نفس السطر من الشكل 1.4 يشيران إلى أن الوسيلة DisplayMessage() لا تطلب معلومات إضافية لتنفيذ مهمتها ، ولهذا السبب استدعاء الوسيلة في السطر 11 من الشكل 2.4 يكون أيضاً بقوسين فارغين.

فـي هـذه الحالـة : عنـدما تنهـي الوسـيلة ()DisplayMessage مهمتهـا يعـود الـتحكم بتنفيـذ البرنـامج للوسـيلة ()Main فـي السـطر 12 حيـث نهايـة الوسـيلة ()Main ، وبالتـالي إيقـاف تنفيذ البرنامج.

### 3.4 : التصريح عن وسيلة لها وسطاء ( parameters ) :

فـي مثـال السـيارة الـذي ناقشـناه فـي الـدرس الثـاني ، قلنـا أن حقيقـة الضـغط علـى دواسـة الوقـود هـي عمليـاً إرسـال رسـالة إلـى السـيارة بزيـادة سـرعتها ، ولكـن مـا هـو مقـدار السـرعة التي يجب على السيارة أن تزيدها ؟

كمـا تعلـم أن مقـدار السـرعة يتناسـب طـرداً مـع مقـدار الضـغط علـى دواسـة الوقـود ، لـذلك فالرسـالة الموجهـة للسـيارة يجـب أن تحتـوي بالإضـافة للمهمـة المـراد تنفيـذها علـى المعلومات الإضافية التي تساعد السيارة على تنفيذ تلك المهمة.

هـذه المعلومـات الإضـافية نسـميها الوسـطاء ( parameters ) ، وقيمـة الوسـيط هنـا يحـدد للسيارة كم يجب عليها زيادة السرعة.

بشـكل مشـابه ، الوسـيلة يمكـن أن تطلـب وسـيطاً أو أكثـر والتـي تمثـل معلومـات إضـافية لهذه الوسيلة تساعدها في إنجاز مهمتها.

نقـوم ضـمن تعبيـر اسـتدعاء الوسـيلة بتزويـد الوسـيلة بقـيم ( تـدعى هنـا معـاملات arguments ) كل وسيط من وسطاء الوسيلة المستدعاة.

علـى سـبيل المثـال : الوسـيلة ()Console.WriteLine تطلـب معـاملاً يحـدد لهـا البيانـات التـي سـوف تظهرهـا علـى نافـذة المتـرجم السـطري ، وأيضـاً إذا أردت أن تقـوم بعمليـة إيـداع ( deposit ) فـي حسـاب مصـرفي ، فالوسـيلة ( deposit ) يجـب أن يكـون لهـا وسـيط يمثـل مبلـغ الإيـداع ( amount ) وعنـدما تسـتدعى الوسـيلة ( deposit ) تكـون قيمـة المعامـل الممـرر لهـا قـد أسـندت لوسـيط الوسـيلة ( أي أخـذ الوسـيط قيمـة المعامـل الممرر ).

مثالنـا التـالي يصـرح عـن الصـنف GradeBook فـي الشـكل 3.4 والوسـيلة ()DisplayMessage التـي سـوف تعـرض هـذه المـرة اسـم المـادة كجـزء مـن الرسـالة الترحيبية ( يمكنك أن ترى مثالاً عن خرج هذا البرنامج في الشكل 4.4 ).

الوسـيلة ()DisplayMessage الجديـدة تطلـب وسـيطاً يمثـل اسـم المـادة وذلـك لإظهارهـا ضمن رسالة الترحيب.

```
1  // Class declaration with a method that has a parameter.
2  using System;
3
4  public class GradeBook
5  {
6      // display a welcome message to the GradeBook user
7      public void DisplayMessage(string courseName)
8      {
9          Console.WriteLine("Welcome to the grade book for\n{0}!",
10             courseName);
11     } // end method DisplayMessage
12 } // end class GradeBook
```

الشكل 3.4 التصريح عن صنف بوسيلة لها وسيط

قبل أن نناقش الميزات الجديدة في الصنف GradeBook ، دعنا نلقي نظرة على الوسيلة Main() في الصنف GradeBookTest في الشكل 4.4

```
1  /* Create a GradeBook object and pass a string to
2     its DisplayMessage method. */
3  using System;
4
5  public class GradeBookTest
6  {
7      // Main method begins program execution
8      public static void Main()
9      {
10         // create a GradeBook object and assign it to myGradeBook
11         GradeBook myGradeBook = new GradeBook();
12
13         // prompt for and input course name
14         Console.WriteLine("Please enter the course name:");
15         string nameOfCourse = Console.ReadLine(); // read a line of text
16         Console.WriteLine(); // output a blank line
17
18         // call myGradeBook's DisplayMessage method
19         // and pass nameOfCourse as an argument
20         myGradeBook.DisplayMessage(nameOfCourse);
21     } // end Main
22 } // end class GradeBookTest
```

```
Please enter the course name:
```

```
CS101 Introduction to C# Programming

Welcome to the grade book for
CS101 Introduction to C# Programming
```

الشكل 4.4 إنشاء غرض من الصنف كتاب العلامات وتمرير سلسلة محارف كمعامل للوسيلة العرض على الشاشة

نقوم في السطر 11 بإنشاء غرض من الصنف GradeBook ونسنده إلى المتحول myGradeBook ، وفي السطر 14 نوجه ( prompt ) المستخدم لإدخال اسم المادة ، وفي السطر 15 نقرأ اسم المادة بواسطة الوسيلة Console.ReadLine() ومن ثم نخزن القيمة في المتحول nameOfCourse ، وبعد أن يكتب المستخدم اسم المادة يقوم بالنهاية بالضغط على زر الإدخال لإرسال النص ( اسم المادة ) إلى البرنامج.

إن ضغط مفتاح الإدخال يضيف محرف السطر الجديد في نهاية سلسلة المحارف المدخلة من قبل المستخدم ، تقرأ الوسيلة Console.ReadLine() المحارف المدخلة حتى تصل إلى محرف السطر الجديد فإذا كان وحيداً فترجع ( return ) سلسلة المحارف ( string ) كاملة باستثناء محرف السطر الجديد فإنها تتجاهله.

في السطر 20 يتم استدعاء الوسيلة DisplayMessage() الجديدة عن طريق الغرض myGradeBook ، والمتحول nameOfCourse ( نمطه string ) الممرر كمعامل للوسيلة DisplayMessage() يساعد هذه الأخيرة على تنفيذ مهمتها.

يأخذ الوسيط courseName ( السطر 7 من الشكل 3.4 ) قيمته من المعامل nameOfCourse الممرر لاستدعاء الوسيلة DisplayMessage() ( السطر 20 من الشكل 4.4 ).

**وجهة نظر هندسة البرمجيات :**

تُنشأ الأغراض بواسطة الكلمة المفتاحية new باستثناء الثابت النصي ( string literals ) المحتوى ضمن علامتي الاقتباس ( " " ) مثل الثابت المحرفي التالي ( "The_First_ISE" ) ، ومثل هذه الثوابت تشير إلى أغراض تُنشأ ضمنياً ( implicitly ) في سي شارب.

**المزيد حول المعاملات ( arguments ) والوسطاء ( parameters ) :**

عندما نصرح عن وسيلة ما يجب علينا تحديد فيما إذا كانت بحاجة لمعلومات إضافية لإنجاز مهمتها أم لا ، ولفعل ذلك يكفي وضع المعلومات الإضافية ضمن قائمة الوسطاء ( parameter list ) التي توجد ضمن القوسين الصغيرين اللذين يتبعان اسم الوسيلة.

قائمة الوسطاء يمكن أن تحتوي على أي عدد من الوسطاء ، بما في ذلك العدد صفر.
كل وسيط يصرح عنه كمتحول أي أن له نمط ومعرف ( اسم ).
في مثالنا النمط `string` والمعرف `courseName` يشيران بأن الوسيلة `DisplayMessage()` ( السطر 7 من الشكل 3.4 ) تطلب وسيطاً نصاً لتنفيذ مهمتها وقائمة الوسطاء هنا هي عبارة عن وسيط واحد.
في الوقت الذي تستدعى فيه الوسيلة فإن قيمة المعامل في الاستدعاء تسند إلى الوسيط الموافق المعرف في ترويسة الوسيلة.
في السطر 9 – 10 من الشكل 3.4 يعرض قيمة الوسيط باستخدام التنسيق النصي ( string format ) واستبدال عنصر التنسيق ( `{0}` ) بقيمة الوسيط.
اسم المتحول الوسيط يمكن أن يكون نفس اسم المعامل ، ويتم فصل الوسطاء في قائمة الوسطاء المتعددة بفاصلة بين كل وسيطين.
عدد المعاملات الممررة عند استدعاء وسيلة ما يجب أن يوافق عدد الوسطاء المطلوبة في قائمة وسطاء الوسيلة المستدعاة ، ويجب أيضاً أن يتوافق نمط المعاملات مع الوسطاء الموافقة.
في مثالنا ، نمرر في استدعائنا للوسيلة معاملاً من النمط `string` ( في الشكل 4.4 السطر 20 ) والتصريح عن الوسيلة المستدعاة ( الشكل 3.4 السطر 7 ) يطلب وسيطاً من النمط `string` أيضاً.

**أخطاء برمجية شائعة :**
يحدث خطأ ترجمة إذا كان عدد المعاملات الممررة في استدعاء الوسيلة لا يوافق عدد الوسطاء المطلوبة في ترويسة التصريح عنها.

**أخطاء برمجية شائعة :**
يحدث خطأ ترجمة إذا كان نمط المعاملات الممررة في استدعاء الوسيلة لا يوافق نمط الوسطاء المطلوبة الموافقة في ترويسة التصريح عنها.

**ملاحظات حول التوجيه باستخدام ( using ) :**

لاحظ أن التوجيه ( using ) في الشكل 4.4 السطر 3 يشير للمترجم أن البرنامج يستخدم أصنافاً من فضاء الأسماء ( System ) مثل الصنف ( Console ).

لكن لماذا نحتاج التوجيه لاستخدام الصنف ( Console ) ولا نحتاجه للصنف ( GradeBook ) مع أننا سنستخدمه أيضاً ؟

الجواب : هو أنه هناك علاقة بين الأصناف التي تترجم في نفس المشروع مثل ( GradeBook و GradeBookTest ) ، فافتراضياً : مثل تلك الأصناف تعتبر بنفس فضاء الأسماء واستخدام التوجيه غير مطلوب عندما يريد صنفاً ما استخدام صنفاً آخر موجود معه بنفس فضاء الأسماء.

ولكن إذا راجعنا الرماز الذي كتبناه فسوف نلاحظ أننا لم نصرح عن فضاء أسماء جديد لبرنامجنا ، ولكن كل الأصناف التي لا تنتمي صراحة ( explicitly ) لفضاء أسماء معين فسوف تعتبر منتمية لفضاء الأسماء العام ( global namespace ).

عملياً : استخدام التوجيه في السطر 3 ليس مطلوباً ( يعمل البرنامج بدونه ) إذا أشرنا دائماً ( كل مرة ) للمترجم أن الصنف ( Console ) هو من فضاء الأسماء ( System ) أي أن نكتب : ( System.Console ).

معظم مبرمجي سي شارب يعتبرون هذه الطريقة مرهقة ويستخدمون عوضاً عن ذلك التوجيه باستخدام ( using ).

## 4.4 : متحولات الغرض( instance variable ) والخصائص ( properties ) :

في الدرس الثالث كنا نصرح عن المتحولات جميعها ضمن الوسيلة ()Main.

والمتحولات التي يصرح عنها ضمن جسم الوسيلة تدعى المتحولات المحلية ( local variable ) ويمكن استخدامها فقط ضمن هذه الوسيلة ، عندما تنهي الوسيلة مهمتها تضيع قيم متحولاتها المحلية.

نعود فنقول أن كل غرض له واصفات وهذه الواصفات تُمثل كمتحولات عند التصريح عن الصنف ، وندعوها حينئذ حقولاً ( fields ) ويصرح عنها ضمن جسم الصنف وخارج أجسام الوسائل التي تنتمي لهذا الصنف.

وعندما يصبح لكل غرض نسخته الخاصة من تلك الواصفات فالحقل يُمثِّل حينئذ متحولات الغرض ( instance variable ) بحيث يكون لكل غرض ( instance ) من هذا الصنف متحولاته ( واصفاته ) الخاصة به والتي تميزه عن غيره.

سوف نتكلم في درس قادم عن نوع آخر من الحقول يدعى الحقل المشترك ( static variable ) وسمي بهذا الاسم لأن كل الأغراض المنشأة من ذلك الصنف تتشارك في هذا النوع من الحقول.

**الصنف GradeBook بمتحول غرض وحيد ( instance variable ) وخاصية ( property ) :**

في مثالنا التالي الشكل 5.4 والشكل 6.4 الصنف ( GradeBook ) يحتفظ باسم المادة courseName كمتحول غرض لذلك يمكننا استخدامه وتعديله في أي وقت خلال وقت تنفيذ البرنامج.

الصنف أيضاً يحتوي على وسيلة واحدة DisplayMessage() ( السطر 23 ... 29 من الشكل 5.4 ) وخاصية واحدة وهي CourseName ( السطور 10 ... 20 )

```
1  // GradeBook class that contains a private instance variable, courseName,
2  // and a public property to get and set its value.
3  using System;
4
5  public class GradeBook
6  {
7      private string courseName; // course name for this GradeBook
8
9      // property to get and set the course name
10     public string CourseName
11     {
12         get
13         {
14             return courseName;
15         } // end get
16         set
17         {
18             courseName = value;
19         } // end set
20     } // end property CourseName
21
22     // display a welcome message to the GradeBook user
23     public void DisplayMessage()
24     {
25         // use property CourseName to get the
26         // name of the course that this GradeBook represents
27         Console.WriteLine("Welcome to the grade book for\n{0}!",
```

```
28              CourseName); // display property CourseName
29          } // end method DisplayMessage
30      } // end class GradeBook
```

الشكل 5.4 الصنف كتاب العلامات بمتحول غرض خاص وخاصية عامة

قلنا في الدرس الثاني أن الخاصية تستخدم كواجهة للتعامل مع واصفات الغرض ، فعلى سبيل المثال : استخدمنا الخاصية Text لعنصر التحكم عنوان Lable لتغيير النص الذي يظهر ضمن هذا العنصر ، وفي مثالنا هذا سوف نستخدم الخاصية ضمن الرماز وليس ضمن نافذة الخصائص في بيئة العمل المتكاملة.

ولفعل ذلك نصرح عن الخاصية CourseName كعضو جديد في الصنف GradeBook ، وكما سوف ترى من خلال البرنامج كيف سنستخدم هذه الخاصية لحفظ اسم المادة في متحول الغرض courseName ونستخدمها أيضاً لاسترجاع اسم المادة من المتحول ذاته.

الوسيلة DisplayMessage() التي لا تطلب هنا أي وسيط لا تزال تعرض رسالة ترحيبية تحتوي على اسم المادة المدخلة من قبل المستخدم.

على كل حال ، الوسيلة الآن تستخدم الخاصية CourseName للحصول على اسم المادة من متحول الغرض courseName ( لأنه لا يمكن الوصول إليه مباشرة ).

بشكل عام المدرس يدرس أكثر من مادة وكل مادة لها اسم مختلف.

نصرح في السطر 7 عن المتحول courseName كمتحول من النمط string ( سلسلة محارف نصية ) وندعوه كما قلنا متحول غرض ( instance variable ) لأنه عُرف داخل جسم الصنف وخارج أجسام الوسائل في نفس الصنف.

طبعاً كل غرض ( instance or object ) منتسخ من الصنف GradeBook يحتوي على نسخته الخاصة من المتحول courseName.

كل الوسائل والخصائص في الصنف GradeBook تستطيع التعامل مع المتحول courseName ولكن تعتبر عادة برمجية جيدة أن تستخدم الوسائل في الصنف الخصائص للتعامل مع متحولات الغرض فيه كما فعلنا في السطر 28 من الوسيلة DisplayMessage() ، وسيتوضح لك قريباً أسباب هندسة البرمجيات المتطلبة لذلك.

**معرف الوصول ( access modifier ) public و private :**

معظم متحولات الغرض يسبق عبارة التصريح عنه الكلمة المفتاحية private كما في السطر 7.

المتحـــولات ( variables ) والخصـــائص ( properties ) والوســـائل ( methods ) التـي يصـرح عنهـا مـع معـرف الوصـول private تكـون مرئيـة وتقبـل التعامـل معهـا فقـط مـن قبـل الخصـائص والوسـائل ( الأعضـاء ) التـي تنتمـي لـنفس صـنفها ، لـذلك المتحـول courseName يمكـن التعامـل معـه والوصـول إليـه فقـط مـن قبـل الخاصـية CourseName والوسيلة DisplayMessage() في الصنف GradeBook.

**وجهة نظر هندسة البرمجيات :**

ضـع قبـل كـل تصـريح عـن حقـل أو وسـيلة معـرف وصـول ( access modifier ). بشـكل عـام متحـولات الغـرض يجـب أن يسـبق التصـريح عنهـا بالكلمـة المفتاحيـة private ويجب أن يسبق التصريح عن الوسائل والخصائص الكلمة المفتاحية public.

إذا لـم يُسـبق التصـريح عـن عضـو الصـنف معـرف وصـول فـإن ذلـك العضـو سـوف يعتبـر ضـمنياً علـى أنـه خـاص private ، وسـوف نـرى أنـه مـن المفيـد أحيانـاً التصـريح عـن وسـائل خاصـة private إذا لـم يكـن مـن المطلـوب الوصـول إليهـا إلا مـن الوسـائل التـي تنتمي إلى صنفها ذاته.

**وجهة نظر هندسة البرمجيات :**

التصـريح عـن متحـولات الغـرض فـي الصـنف علـى أنهـا خاصـة private والوسـائل علـى أنهـا عامـة public يسـهل علينـا عمليـة التفلية ( debugging ) أو مـا يسـمى بتشـخيص الأخطـاء ، لأن مشـاكل التعامـل مـع البيانـات سـوف تكـون محصـورة فـي الوسـائل والخصـائص العامـة فـي الصـنف طالمـا أن المتحـولات الخاصـة لا يمكـن الوصـول إليهـا إلا عن طريق تلك الوسائل والخصائص الأعضاء.

التصـريح عـن متحـولات الغـرض بمعـرف وصـول private هـو تطبيـق لمفهـوم أساسـي فـي البرمجـة الغرضـية التوجـه ( OOP ) يـدعى إخفـاء المعلومـات ( information hiding ) ، فعنـدما نقـوم فـي البرنـامج بإنشـاء ( instantiate ) غرضـاً مـن الصـنف GradeBook فـإن المتحـول courseName يُغلـف ( encapsulate ) أو يُخفـى ( hidden ) ضـمن هـذا الغـرض ولـن نسـتطيع الوصـول إليـه إلا عـن طريـق الوسائل أو الخصائص العامة المُغلفة ضمن غرض الصنف نفسه.

**تعديل ( set ) واسترجاع ( get ) قيم متحولات الغرض الخاصة :**

إذا تساءلنا كيف يمكننا أن نسمح للبرنامج أن يتعامل مع ( يُعدل أو يسترجع ) قيم متحولات الغرض الخاصة ( private instance variable ) التي لا يمكن الوصول إليها من خارج جسم الصنف مع ضمان إبقائها بحالة منطقية صحيحة ؟

الجواب عن هذا التساؤل يضطرنا للتفكير بآلية تسمح لنا بالتحكم بطريقة استرجاع ( get ) وتعديل ( set ) قيمة متحول الغرض المطلوب وذلك عن طريق وسائل نعرفها نحن لفرض منطق ( logic ) معين للاسترجاع مثلاً ( GetCourseName() ) و للتعديل مثلاً ( SetCourseName() ).

بصراحة قامت سي شارب بتحمل ذلك العناء عنا عن طريق حل أذكى وأكثر منطقية ، ألا وهو الخصائص ( properties ).

دعونا الآن نرى كيف يكون شكل الصنف GradeBook مع الخاصية CourseName.

يبدأ التصريح عن الخاصية CourseName ( السطر 10 من الشكل 5.4 ) بمعرف الوصول public متبوعاً بنمط هذه الخاصية وهو string واسمها CourseName.

يُتبع في تسمية الخصائص طريقة تدوين أسماء الوسائل والأصناف ( تدوين باسكال ).

تحتوي الخصائص بشكل عام على متحكمين لإدارة تفاصيل استرجاع ( return ) وتعديل ( modify ) البيانات ، حيث يمكن أن تحتوي على متحكم الإرجاع get أو متحكم التعديل set أو كلاهما معاً.

متحكم الإرجاع get ( السطور 12 ... 15 ) يُمكن الزبون ( العميل ) من استرجاع قيمة متحول الغرض الخاص courseName ، و متحكم التعديل set ( السطور 16 ... 20 ) يُمكن العميل من تعديل قيمة متحول الغرض الخاص courseName.

بعد تعريف الخاصية يمكننا استخدامها كمتحول ضمن الرماز ( أكثر منطقية من استخدام وسائل لهذا الغرض ) ، فعلى سبيل المثال : يمكن إسناد قيمة للخاصية باستخدام عملية الإسناد ( = ) وهذه العملية تقوم بدورها بتنفيذ المتحكم set لتعديل قيمة المتحول الموافق للخاصية courseName في برنامجنا.

وبشكل مشابه ، استخدام الخاصية لاسترجاع البيانات ( لعرضها على الشاشة مثلاً ) ينفذ متحكم الاسترجاع get في الخاصية للحصول على قيمة المتحول الموافق للخاصية courseName في برنامجنا.

بالاصطلاح : نسمي الخاصية بنفس اسم ( معرف ) متحول الغرض الخاص الذي سوف تستخدم كواجهة له ولكن بالأحرف الكبيرة ( تدوين باسكال ).

ولا ننسى أن سي شارب حساسة لحالة الأحرف ( case sensitive ) ، لذلك فأسماء المعرفات صارمة في سي شارب.

**المتحكمين ( get و set ) :**

دعنا ننظر عن قرب لمتحكمي الخاصية CourseName في الشكل 5.4 المتحكم get ( السطور 12 ... 15 ) يبدأ بالكلمة المفتاحية get وله جسم محدد بالقوسين المعقوفين ( { } ) ويحتوي داخل هذا الجسم على عبارة ( return statement ) التي تتألف من الكلمة المفتاحية return متبوعة بتعبير ( expression ) ، قيمة هذا التعبير تُرجَع للعميل الذي يستخدم هذه الخاصية ، في مثالنا : قيمة المتحول courseName ترجع عندما يشار إلى الخاصية CourseName.

على سبيل المثال : في العبارة التالية :

```
string theCourseName = gradeBook.CourseName;
```

التعبير gradeBook.CourseName ( gradeBook هو غرض من الصنف GradeBook ) ينفذ متحكم الاسترجاع في الخاصية CourseName والذي يرجع قيمة متحول الغرض courseName ثم تُسنَد القيمة المرجعة في المتحول النصي theCourseName ( string ).

كما رأينا ، استخدمنا الخاصية كما لو أنها متحول غرض ( بكل بساطة ).

نقول مجدداً : لا يستطيع العميل التعامل ( الاسترجاع والتعديل ) مباشرة مع متحول الغرض courseName لأنه خاص private.

المتحكم set ( السطور 16 ... 19 ) يبدأ بالكلمة المفتاحية set وله جسم محدد بالقوسين المعقوفين ( { } ) وعندما تستخدم الخاصية في عبارة الإسناد ( assignment statement ) مثل :

```
gradeBook.CourseName = "CS100 Introduction to Computers";
```

فإن : اسم المادة "CS100 Introduction to Computers" تسند إلى الكلمة المفتاحية السياقية ( contextual keyword ) value الموجودة ضمن سياق جسم متحكم التعديل set ومن ثم تنفذ عبارات جسم المتحكم.

لاحظ أن المتحول value قد صرح عنه ضمنياً ( implicitly ) وأخذ قيمته الابتدائية ( initialized ) في متحكم التعديل set ، وإنه ليعتبر خطأ ترجمة أن نصرح عن متحول محلي باسم value في جسم متحكم التعديل.

السطر 18 يخزن قيمة value في متحول الغرض courseName.

متحكم التعديل set لا يرجع أي قيمة عندما ينهي مهمته ، ونلاحظ أنه باستطاعتنا استخدام متحول الغرض courseName في الوسائل والخصائص ضمن الصنف GradeBook لأن courseName ينتمي لنفس الغرض.

**استخدام الخاصية CourseName ضمن الوسيلة DisplayMessage() :**

الوسيلة DisplayMessage() ( السطور 23 ... 29 من الشكل 5.4 ) لا تطلب أي وسطاء ، السطران 27 – 28 يعرضان رسالة ترحيبية تحتوي على قيمة متحول الغرض courseName ، هنا لم نقم بالإشارة إلى متحول الغرض courseName مباشرة وعوضاً عن ذلك تعاملنا مع الخاصية CourseName ( السطر 28 ) والتي نفذت بدورها متحكم الإرجاع get الذي يرجع قيمة المتحول courseName.

الصنف GradeBookTest ( الشكل 6.4 ) يشرح لنا كيفية التعامل مع الصنف GradeBook ننشأ ضمن هذا الصنف غرضاً من الصنف GradeBook ( السطر 10 ) ونسنده إلى المتحول المحلي myGradeBook.

```
1  // Create and manipulate a GradeBook object.
2  using System;
3
4  public class GradeBookTest
5  {
6      // Main method begins program execution
7      public static void Main()
8      {
9          // create a GradeBook object and assign it to myGradeBook
10         GradeBook myGradeBook = new GradeBook();
11
12         // display initial value of CourseName
13         Console.WriteLine("Initial course name is: '{0}'\n",
14            myGradeBook.CourseName);
15
16         // prompt for and read course name
17         Console.WriteLine("Please enter the course name:");
18         myGradeBook.CourseName = Console.ReadLine(); // set CourseName
19         Console.WriteLine(); // output a blank line
20
21         // display welcome message after specifying course name
22         myGradeBook.DisplayMessage();
23     } // end Main
24 } // end class GradeBookTest
```

```
Initial course name is: ''

Please enter the course name:
CS101 Introduction to C# Programming!

Welcome to the grade book for
CS101 Introduction to C# Programming!
```

الشكل 6.4 إنشاء غرض من الصنف كتاب العلامات والتعامل معه

السطران 13 – 14 تعرضان لنا كيفية الوصول لقيمة المتحول courseName باستخدام خاصية الغرض CourseName.

السطر الأول من خرج البرنامج يظهر رسالة الترحيب باسم مادة فارغ محدد بعلامتي اقتباس مفردة (''').

بخلاف المتحولات المحلية ( التي لا تأخذ قيمة ابتدائية ابتدائية افتراضية ) ، كل حقل له قيمة ابتدائية افتراضية تزوده بها سي شارب عندما نتركه بدون قيمة بعد التصريح ، لذلك فالحقول ليس من الضروري تعيين قيمتها الابتدائية قبل استخدامها في البرنامج ما لم نكن بحاجة لذلك.

القيمة الافتراضية لمتحول غرض من نمط `string` مثل courseName هي القيمة `null` ، وعندما نريد عرض قيمة متحول نصي ( string ) يحتوي على القيمة `null` لن يعرض شيء على الشاشة.

السطر 18 يُسند اسم المادة المدخل من قبل المستخدم إلى الخاصية CourseName للغرض myGradeBook ، وعندما تسند القيمة إلى الخاصية فإن القيمة المدخلة ( التي تعيدها الوسيلة `Console.ReadLine()` ) تسند ضمنياً إلى المتحول `value` لمتحكم الخاصية `set` ( السطور 16 ... 19 من الشكل 5.4 ).

والسطر 19 يعرض سطراً فارغاً ، ثم يقوم السطر 22 باستدعاء الوسيلة DisplayMessage() عن طريق الغرض myGradeBook ليعرض الرسالة الترحيبية التي تحتوي على اسم المادة.

## 5.4 : الفرق بين أنماط القيمة ( value type ) وأنماط المرجع ( reference type ) :

تقسم الأنماط ( types ) في سي شارب إلى نوعين : أنماط قيمة وأنماط مرجع.

أنماط سي شارب البسيطة ( مثل `double` و `int` ) كلها أنماط قيمة.

المتحولات من نوع أنماط القيمة ببساطة تحتوي على قيمة من ذلك النمط ، على سبيل المثال : الشكل 7.4 يرينا أن المتحول من النمط `int` المسمى count يحتوي على

القيمة 7 ، أنماط القيمة تُنَجز ( implement ) في بنى تشبه الأصناف تسمى ( structs ) والتي سوف نناقشها في درس قادم إن شاء الله.

```
int count = 7;
```

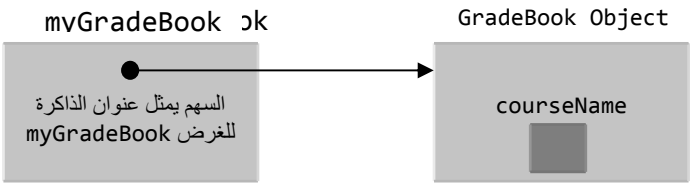
المتحول count من النمط int يحتوي على القيمة 7 من ذلك النمط.


الشكل 7.4 أنماط القيمة

وبخلاف النوع السابق أنماط المرجع تحتوي على عنوان ( address ) لموقع في الذاكرة حيث توجد بيانات ذلك المتحول ( مثل المتحولات التي نقول أنها تشير إلى غرض في البرنامج ).

في الشكل 6.4 السطر 10 ننشأ غرضاً من الصنف GradeBook ونضعه في الذاكرة ونخزن عنوانه في المتحول myGradeBook الذي هو من النمط GradeBook كما هو موضح في الشكل 8.4 :

```
GradeBook myGradeBook = new GradeBook();
```




المتحول myGradeBook من نمط المرجع GradeBook ويحتوي على عنوان ذاكرة لغرض من ذلك الصنف.

الشكل 8.4 أنماط المرجع

تأخذ متحولات الغرض من نوع نمط المرجع القيمة الافتراضية null كما في الشكل السابق ، والنمط النصي string هو من نوع نمط المرجع أيضاً ولهذا السبب المتحول النصي courseName المعروض في الشكل 8.4 يمثل بصندوق فارغ.

المتحول النصي ذو القيمة null لا يعني نصاً فارغاً ( "" ) أو string.Empty بل يمثل مرجعاً ( عنواناً ) لا يشير إلى أي غرض ، بينما النص الفارغ هو غرض نصي ولكن بدون محارف بداخله.

زبون الغرض يجب أن يستخدم المتحول الذي يشير إلى الغرض في الذاكرة ليستدعي ( call or invoke ) وسائل الغرض ويستطيع الوصول لخصائصه.

## 6.4 : تزويد الغرض بقيم ابتدائية عن طريق الباني ( constructor ) :

عند إنشاء غرض من الصنف GradeBook ( الشكل 5.4 ) فإن متحول الغرض courseName يأخذ قيمة ابتدائية null بشكل افتراضي.

الآن ماذا لو أردنا أن نعطي اسماً للمادة عند إنشاء الغرض ؟

كل صنف له باني وباستخدامه نستطيع تعيين قيم ابتدائية للغرض عند إنشاءه.

في الحقيقة : سي شارب تستدعي باني الصنف الافتراضي لكل غرض يتم إنشاؤه ، والعملية new تستدعي باني الصنف المطلوب أثناء التصريح.

استدعاء الباني يشار إليه في الرماز باسم الصنف نفسه متبوعاً بقوسين صغيرين ، فعلى سبيل المثال : السطر 10 من الشكل 6.4 نستخدم العملية new لننشأ غرضاً من الصنف GradeBook والقوسين الفارغين بعد اسم الصنف تشير إلى أن هذا الاستدعاء يتم بدون معاملات يطلبها هذا الباني.

المترجم بدوره يقوم بتزويد بان افتراضي عام بدون وسطاء لكل صنف لم نصرح به عن باني ، لذلك كل صنف له باني والباني الافتراضي لا يعدل قيم متحولات الغرض في الصنف.

عندما نصرح عن صنف جديد يمكننا تزويده ببانٍ غير الباني الافتراضي أو حتى عدة باني.

```
1  // GradeBook class with a constructor to initialize the course name.
2  using System;
3
4  public class GradeBook
5  {
6      // auto-implemented property CourseName implicitly creates an
7      // instance variable for this GradeBook's course name
8      public string CourseName { get; set; }
9
10     // constructor initializes auto-implemented property
11     // CourseName with string supplied as argument
12     public GradeBook(string name)
13     {
14         CourseName = name; // set CourseName to name
15     } // end constructor
16
17     // display a welcome message to the GradeBook user
18     public void DisplayMessage()
19     {
20         // use auto-implemented property CourseName to get the
```

```
21          // name of the course that this GradeBook represents
22          Console.WriteLine("Welcome to the grade book for\n{0}!",
23             CourseName);
24       } // end method DisplayMessage
25    } // end class GradeBook
```

الشكل 9.4 الصنف كتاب العلامات المعدل.

السطور 12 ... 15 نصرح عن باني الصنف GradeBook ، ويجب أن يكون اسم الباني نفس اسم الصنف ، وكما في الوسيلة الباني يطلب وسطاء إن احتاج لذلك ، وبخلاف الوسيلة لا يتم تحديد نمط إرجاع للباني ( ولا حتى void ).

السطر 12 يشير إلى أن باني الصنف GradeBook لديه وسيط من النمط string أسميناه name ، وفي السطر 14 استخدمنا الاسم الممرر للباني لتعيين قيمة ابتدائية للخاصية CourseName عن طريق متحكم التعديل set للخاصية.

```
1  // GradeBook constructor used to specify the course name at the
2  // time each GradeBook object is created.
3  using System;
4
5  public class GradeBookTest
6  {
7      // Main method begins program execution
8      public static void Main()
9      {
10         // create GradeBook object
11         GradeBook gradeBook1 = new GradeBook( // invokes constructor
12            "CS101 Introduction to C# Programming");
13         GradeBook gradeBook2 = new GradeBook( // invokes constructor
14            "CS102 Data Structures in C#");
15
16         // display initial value of courseName for each GradeBook
17         Console.WriteLine("gradeBook1 course name is: {0}",
18            gradeBook1.CourseName);
19         Console.WriteLine("gradeBook2 course name is: {0}",
20            gradeBook2.CourseName);
21     } // end Main
22 } // end class GradeBookTest
```

```
gradeBook1 course name is: CS101 Introduction to C# Programming
gradeBook2 course name is: CS102 Data Structure in C#
```

الشكل 10.4 تعيين قيمة ابتدائية للأغراض باستخدام الباني

السطران 11 – 12 ننشأ غرض جديد من الصنف GradeBook ونحدد له قيمة ابتدائية.
يتم استدعاء الباني للصنف GradeBook مع المعامل "CS101 Introduction to C# Programming" لتعيين قيمة ابتدائية لاسم المادة.
تعبير إنشاء الغرض على يمين عملية الإسناد ( = ) يرجع عنواناً ( مرجعاً ) لغرض جديد ويتم تخزين هذا العنوان في المتحول gradeBook1.
السطور 17 ... 20 نستخدم خاصية اسم المادة CourseName لكل من الغرضين لنحصل على اسم المادة ، ويرينا أنه قد تم بالفعل إعطاء قيمة ابتدائية أثناء عملية إنشاء الغرض.
تعلمنا سابقاً أن لكل غرض ( object ) من صنف ما يكون له نسخته الخاصة من متحولات الغرض الموجودة ضمن ذلك الصنف الذي أنشأ من خلاله ، وخرج برنامجنا هنا يؤكد لنا ذلك الكلام ( الشكل 10.4 ).
بشكل عام يصرح عن الباني بمعرف وصول public ( عام ) ، وإذا لم نقم بالتصريح عن باني في الصنف فإن متحولات الغرض تأخذ قيم ابتدائية افتراضية بحسب نمط المتحول ، ( 0 ) للأنماط العددية و ( false ) للنمط المنطقي bool و ( null ) للأنماط المرجعية.
إذا صرحنا عن باني جديد في الصنف فعلينا التصريح عن الباني الافتراضي للصنف صراحة لأن سي شارب لن تصرح تلقائياً عن الباني الافتراضي.

## 7.4 : الأعداد الكسرية ( Floating point ) والنمط ( decimal ) :

في برنامجنا التالي سوف نترك مؤقتاً صنف كتاب العلامات لنصرح عن صنف جديد يدعى Account والذي يمثل حساباً مصرفياً.
معظم أرصدة الحسابات المصرفية ليست أعداداً صحيحة مثل ( 0 أو 22- أو 102.4 ) وإنما هي أعداد حقيقية ( تحتوي أحياناً على فاصلة عشرية مثل 7.33 أو 0.0975 ).
يوجد في سي شارب ثلاثة أنماط بسيطة لتخزين الأعداد الحقيقية وهي ( float , double , decimal ).
النمطان ( float , double ) يسميان بأنماط الفاصلة العشرية ( floating point ) والاختلاف الرئيسي بينهما وبين النمط ( decimal ) هو أن متحولات النمط الأخير تخزن أعداداً حقيقية بشكل دقيق وضمن مجال محدد ، بينما نمط أعداد الفاصلة العشرية تخزن فقط قيم تقريبية لأعداد حقيقية ولكن ضمن مجال واسع من القيم.

وأيضاً متحولات النمط ( double ) يمكن أن تخزن أعداداً أكثر ضخامة وتفاصيل أكبر ( أي عدد خانات أكبر على يمين الفاصلة العشرية وهو ما يسمى دقة العدد ) من متحولات النمط ( float ).

التطبيق الرئيسي للنمط ( decimal ) هو استخدامه في التطبيقات المالية والرياضية لأنه أكثر دقة من النمطين السابقين.

**دقة الأعداد الحقيقية ومتطلبات التخزين :**

متحولات النمط ( float ) تقدم أعداداً عشرية أحادية الدقة ( single precision ) أي سبع أرقام بعد الفاصلة ، ومتحولات النمط ( double ) تقدم أعداداً عشرية مضاعفة الدقة ( double precision ) أي 15 – 16 رقماً بعد الفاصلة.

بالإضافة إلى ذلك متحولات النمط ( decimal ) تمتلك دقة تصل حتى 28 – 29 رقماً بعد الفاصلة.

في بعض التطبيقات لن تكون الأنماط الثلاث السابقة كافية ، ومثل تلك التطبيقات خارج إطار هذه السلسلة.

معظم المبرمجين يستخدمون النمط ( double ) لتمثيل الأعداد الكسرية ، وفي الحقيقة تعالج سي شارب كل الأعداد الحقيقية التي تكتبها في رمازك ( مثل 7.33 أو 0.05 ) على أنها قيمة من النمط ( double ) ، ومثل هذه القيم تسمى الثوابت العددية الكسرية ( floating-point literals ) ، وبالنسبة للثابت العددي من النمط ( decimal ) يجب إلاحقه بالحرف ( M أو m ) وكأننا نقول " money ".

مثال : ( 7.33 m يمثل ثابتاً عددياً من النمط ( decimal ) ).

الثوابت الصحيحة ( integer literals ) تحول ضمنياً إلى النمط ( float ) أو ( double ) أو ( decimal ) عندما تسند لمتحول من تلك الأنماط.

بالرغم من أن أنماط أعداد الفاصلة العشرية في سي شارب ليست دقيقة 100% فإن لها تطبيقات متعددة ، فعلى سبيل المثال : عندما نتحدث عن الحرارة الطبيعية لجسم الإنسان ( 98.6 فهرنهايت ) فنحن هنا لسنا بحاجة لدقة كبيرة بعد الفاصلة ، وعندما نقرأ قيمة مقياس الحرارة كـ 98.6 فإنها فعلياً قد تكون 98.599947 والقول بأن هذا العدد هو 98.6 هو جيد للكثير من التطبيقات المهتمة بدرجة حرارة الإنسان الطبيعية.

بينما يفضل استخدام النمط ( decimal ) أكثر من نمط الأعداد العشرية عندما نريد الدقة الفائقة في حساباتنا مثل الحسابات المالية.

فـــي الحـــالات التـــي يكـــون فيهـــا التقريـــب كافيـــاً فاســـتخدام الـــنمط ( `double` ) مقـــدم علـــى استخدام النمط ( `float` ).

**أخطاء برمجية شائعة :**

استخدام نمط أعداد الفاصلة العشرية في الحالات التي تتطلب الدقة في الحساب غالباً ما يؤدي إلى أخطاء منطقية ( logic error ).

**الصنف Account ومتحولات غرض من النمط decimal :**

برنامجنا التالي ( الشكل 11.4 و الشكل 12.4 ) يحتوي على صنف بسيط سميناه Account ( الشكل 11.4 ) ويمثل حساباً مصرفياً .

طبعاً المصارف تخدم عملياً أكثر من حساب وكل من هذه الحسابات له قيمة رصيد خاصة به ، لذلك السطر 6 يصرح عن متحول غرض يسمى balance ونمطه decimal.

```
1  // Account class with a constructor to
2  // initialize instance variable balance.
3
4  public class Account
5  {
6      private decimal balance; // instance variable that stores the balance
7
8      // constructor
9      public Account(decimal initialBalance)
10     {
11         Balance = initialBalance; // set balance using property
12     } // end Account constructor
13
14     // credit (add) an amount to the account
15     public void Credit(decimal amount)
16     {
17         Balance = Balance + amount; // add amount to balance
18     } // end method Credit
19
20     // a property to get and set the account balance
21     public decimal Balance
22     {
23         Get
24         {
25             return balance;
26         } // end get
27         Set
28         {
29             // validate that value is greater than or equal to 0;
30             // if it is not, balance is left unchanged
31             if (value > 0)
32                 balance = value;
33         } // end set
```

```
34      } // end property Balance
35  } // end class Account
```

الشكل 11.4 الصنف حساب مصرفي يبان لتعيين قيمة ابتدائية للحساب.

من الطبيعي أنه إذا أراد شخص ما فتح حساب جديد في المصرف أن يُودع مبلغاً من المال في حسابه مباشرة ، الباني ( السطور 9 – 12 ) يطلب الوسيط initialbalance من النمط decimal لتعيين قيمة ابتدائية للحساب ( السطر 11 ) حيث يتم إسناد المعامل الممرر للباني initialbalance إلى الخاصية Balance التي تستدعي متحكم التعديل set فيها وتعديل قيمة متحول الغرض balance.

الوسيلة credit() ( السطور 15 ... 18 ) لا ترجع أي بيانات عند إنهاء مهمتها ، لذلك نمط إرجاعها هو void ، والوسيلة تطلب وسيطاً واحداً اسمه amount من النمط decimal والتي تمثل المبلغ المراد إيداعه بعد فتح الحساب والتي ستضاف للخاصية Balance.

السطر 17 يستخدم كلاً من متحكمي الخاصية Balance ، والتعبير Balance + amount يستدعي متحكم الإرجاع للخاصية الذي يرجع قيمة المتحول balance ثم يضيف إليها المبلغ المراد إيداعه amount ، ثم نقوم بإسناد ناتج عملية الجمع إلى المتحول balance باستدعاء متحكم التعديل للخاصية Balance ( وبهذا يتم استبدال القيمة القديمة للخاصية Balance ).

في السطور 21 ... 34 نصرح عن الخاصية Balance و فيها متحكم الإرجاع get ( السطور 23 ... 26 ) الذي يسمح لعميل الصنف أن يحصل على القيمة الفعلية لمتحول الغرض balance من الصنف Account.

قلنا أن متحكم التعديل get في الخاصية يسمح لعميل الصنف أن يعدل قيمة متحول غرض خاص فيه ، وفي القسم 5.4 الصنف GradeBook يصرح عن الخاصية CourseName ومتحكم التعديل set فيها لا يتحقق من اسم المادة الممرر إن كان ذو قيمة صالحة.

بينما هنا قد طورنا برنامجنا هذا ( الشكل 11.4 ) ليتحقق من صلاحية القيمة المدخلة كمبلغ مالي مراد إيداعه ( السطر 31 ) فيتأكد متحكم التعديل set في الخاصية من أن قيمة المبلغ أكبر من الصفر وإلا فلن يقوم بتخزينها في المتحول balance.

```
1  // Create and manipulate Account objects.
2  using System;
3
4  public class AccountTest
5  {
6      // Main method begins execution of C# application
7      public static void Main()
8      {
9          Account account1 = new Account(50.00M); // create Account object
10         Account account2 = new Account(-7.53M); // create Account object
11
12         // display initial balance of each object using a property
13         Console.WriteLine("account1 balance: {0:C}",
14            account1.Balance); // display Balance property
15         Console.WriteLine("account2 balance: {0:C}\n",
16            account2.Balance); // display Balance property
17
18         decimal depositAmount; // deposit amount read from user
19
20         // prompt and obtain user input
21         Console.Write("Enter deposit amount for account1: ");
22         depositAmount = Convert.ToDecimal(Console.ReadLine());
23         Console.WriteLine("adding {0:C} to account1 balance\n",
24            depositAmount);
25         account1.Credit(depositAmount); // add to account1 balance
26
27         // display balances
28         Console.WriteLine("account1 balance: {0:C}",
29            account1.Balance);
30         Console.WriteLine("account2 balance: {0:C}\n",
31            account2.Balance);
32
33         // prompt and obtain user input
34         Console.Write("Enter deposit amount for account2: ");
35         depositAmount = Convert.ToDecimal(Console.ReadLine());
36         Console.WriteLine("adding {0:C} to account2 balance\n",
37            depositAmount);
38         account2.Credit(depositAmount); // add to account2 balance
39
40         // display balances
```

```
41          Console.WriteLine("account1 balance: {0:C}", account1.Balance);
42          Console.WriteLine("account2 balance: {0:C}", account2.Balance);
43      } // end Main
44  } // end class AccountTest
```

```
account1 balance: $50.00
account2 balance: $0.00

Enter deposit amount for account1: 49.99
adding $49.99 to account1 balance

account1 balance: $99.99
account2 balance: $0.00

Enter deposit amount for account2: 123.21
adding $123.21 to account2 balance

account1 balance: $99.99
account2 balance: $123.21
```

الشكل 12.4 التعامل مع الصنف حساب مصرفي

الصنف AccountTest ينشأ غرضان من الصنف Account ( السطور 9 – 10 ) ويحدد قيم ابتدائية لهما ( 50.00 و 7.53- ).

في الأمثلة السابقة كانت الفائدة من استخدام الخاصية ضمن الباني لتعيين القيم الابتدائية غير واضحة ، بينما تتبين الفائدة هنا بالتحقق من قيمة الوسيط عبر استدعاء متحكم التعديل للخاصية Balance.

يقوم الباني ببساطة بالتحقق عن طريق إسناد القيمة الممررة للخاصية Balance بدون كتابة رماز إضافي للتحقق ، فعندما تمرر القيمة الابتدائية المراد تعيينها ( السطر 9 من الشكل 11.4 ) يقوم الباني بدوره بتمرير تلك القيمة لمتحكم التعديل في الخاصة والذي يقوم بعملية التحقق.

لذلك لن تتغير قيمة متحول الغرض في الغرض account2 ( السطر 10 ) لأن قيمة المعامل أصغر من الصفر وبالتالي ستترك على حالها ( قيمتها الافتراضية 0.00 ).

نستخدم في السطر 13 عنصر التنسيق {0:C} لتنسيق الخرج على شكل قيمة مالية ، والحرف C ( currency ) في عنصر التنسيق نسميه موصف تنسيق ( format specifier ) ، حيث يحدد تنسيق العملة من خلال الإعدادات الإقليمية لنظام التشغيل.

والشكل 13.4 يعرض قائمة لبعض موصفات التنسيق.

| موصف التنسيق ( format specifier ) | الشرح | مثال | |
|---|---|---|---|
| C or c | عملة ( Currency ) | Console.Write("{0:C}", 2.5);<br>Console.Write("{0:C}", -2.5); | $2.50<br>($2.50) |
| D or d | عدد كسري ( Decimal) | Console.Write("{0:D5}", 25); | 00025 |
| E or e | علمي (Scientific) | Console.Write("{0:D5}", 25); | 25.00000E+005 |
| F or f | تقريب الفاصلة (Fixed-point) | Console.Write("{0:F2}", 25);<br>Console.Write("{0:F0}", 25); | 25.00<br>25 |
| G or g | عام ( General) | Console.Write("{0:G}", 2.5); | 2.5 |
| N or n | عدد ( Number) | Console.Write("{0:N}", 2500000); | 2,500,000.00 |
| X or x | ست عشري (Hexadecimal) | Console.Write("{0:X}", 250);<br>Console.Write("{0:X}", 0xffff); | FA<br>FFFF |

الشكل 13.4 موصفات التنسيق النصية

السطر 18 نصرح عن متحول محلي depositAmount لنخزن فيه مبلغ الإيداع المدخل من قبل المستخدم ، وبخلاف متحول الغرض balance في الصنف Account المتحول المحلي ( local variable ) depositAmount للوسيلة Main() لا يعطى قيمة ابتدائية بشكل افتراضي وأيضاً المتحول المحلي يمكن استخدامه فقط ضمن الوسيلة المصرح عنه ضمنها.

على كل حال : هذا المتحول ليس بحاجة لأن يأخذ قيمة ابتدائية هنا لأن قيمته سوف تحدد من قبل مدخلات المستخدم ، والمترجم بدوره لن يسمح لأي متحول محلي بأن تقرأ قيمته ما لم تعيين قيمة ابتدائية له.

**المتحكمان set و get مع تغيير معرفات الوصول :**

بشكل افتراضي يمتلك متحكما التعديل والإرجاع في الخاصية نفس معرف وصول الخاصية ، فعلى سبيل المثال : بالنسبة لخاصية عامة ( public ) فإن متحكماها سيكونان أيضاً ( public ) وإنه لمن الممكن أن نصرح عن المتحكمين بمعرفي وصول مختلفين عن معرف وصول الخاصية وفي هذه الحالة واحد منهما يجب أن يكون معرفه نفس معرف وصول الخاصية التي ينتمي إليها.

على سبيل المثال : في الخاصية العامة public ، متحكم الإرجاع get يمكن أن يكون عاماً ومتحكم التعديل set يمكن جعله خاصاً private.

سوق نتكلم عن هذه الميزة بالتفصيل في درس قادم إن شاء الله تعالى.

**أخطاء برمجية شائعة :**

يجب أن يتحقق متحكم التعديل ( accessor set ) الذي مهمته تعديل متحولات الغرض الخاصة من القيم الجديدة ، فإذا كانت غير صحيحة يجب عدم تعديل قيمة المتحول والإشارة ( طباعة رسالة مثلاً ) بأن هناك خطأ في الإدخال.