## الدرس السادس :

سوف نتناول في هذه المحاضرة المواضيع التالية :

**١- أعضاء البيانات ودوال الإعداد set، ودوال التحصيل get.**
**٢- إعطاء الكائنات قيم أولية بالبواني .**
**٣- وضع الصنف في ملف منفصل .**
**٤- فصل واجهة الصنف عن تطبيقة .**
**٥- تدقيق البيانات أثناء دالة الإعداد set.**

**بسم الله وبه نستعين**

### أعضاء البيانات ، ودوال الإعداد set ، ودوال التحصيل get

في أمثلتنا السابقة كانت جميع المتغيرات يعلن عنها في الدالة الرئيسة الـ main، وتسمى هذه المتغيرات بالمتغيرات المحلية ، والمتغيرات المحلية يحدد التعامل معها على أساس عدة امور أهمها :
١- أن المتغير المحلي يمكن أستخدامة بداية من الصف الذي أعلن عنه فيه إلى نهاية قوس إغلاق تعريف الدالة.
٢- المتغير المحلي يجب أن يكون معرف ( محدد النوع والإسم ) قبل الإستخدام.
٣- لا يمكن الوصول للمتغير المحلي من خارج الدالة التي تم الإعلان عنه فيها.
٤- عند إنتهاء الدالة التي تم الإعلان عن المعرف فيها تكون قيم هذه المتغيرات فقدت.

وتجدر الإشارة هنا أن هناك إستثناء، وهي المتغيرات المحلية الساكنة التي سوف نتناولها في الفصل السادس.
ذكرنا أيضا أن كل كائن لديه خصائص ( أعضاء بيانات ) تكون محمولة معه في كل برنامج يستخدم فيه، أيضا لديهم على الأقل دالة عضو واحدة لها نفس ظروف الخاصية ( عضو البيانات ).
دوال الأعضاء تسخدم في إدارة ومعاملة أعضاء البيانات في كل صنف ( في نفس الصنف ).
في كودنا التالي سوف نناقش صنف GradeBook يحتوي على عضو بيانات courseName يمثل إسم الدورة لكائن الصنف :

كود ١

**كود**

```
[/size]

[size="3"]1  // Fig. 3.5: fig03_05.cpp
2  // Define class GradeBook that contains a courseName data member
3  // and member functions to set and get its value;
4  // Create and manipulate a GradeBook object with these functions.
5  #include <iostream>
6  using std::cout;
7  using std::cin;
8  using std::endl;
9
10  #include <string> // program uses C++ standard string class
11  using std::string;
12  using std::getline;
13
14  // GradeBook class definition
```

```
class GradeBook
{
public:
   // function that sets the course name
   void setCourseName( string name )
   {
      courseName = name; // store the course name in the object
   } // end function setCourseName

   // function that gets the course name
   string getCourseName()
   {
      return courseName; // return the object's courseName
   } // end function getCourseName

   // function that displays a welcome message
   void displayMessage()
   {
      // this statement calls getCourseName to get the
      // name of the course this GradeBook represents
      cout << "Welcome to the grade book forn" << getCourseName() << "!"
         << endl;
   } // end function displayMessage
private:
   string courseName; // course name for this GradeBook
}; // end class GradeBook

// function main begins program execution
int main()
{
   string nameOfCourse; // string of characters to store the course name
   GradeBook myGradeBook; // create a GradeBook object named
myGradeBook

   // display initial value of courseName
   cout << "Initial course name is: " << myGradeBook.getCourseName()
      << endl;

   // prompt for, input and set course name
   cout << "nPlease enter the course name:" << endl;
   getline( cin, nameOfCourse ); // read a course name with blanks
   myGradeBook.setCourseName( nameOfCourse ); // set the course name

   cout << endl; // outputs a blank line
   myGradeBook.displayMessage(); // display message with new course name
   return 0; // indicate successful termination
} // end main[/size]
```

[size="3"][/size]

[size="3"]

الصنف GradeBook مع عضو بيانات ودالة إعداد set ، ودالة حصول get :

في الكود السابق، الصنف GradeBook يحتوي على إسم الدورة كعضو بيانات يمكن استخدامة والتعديل عليه خلال وقف تنفيذ البرنامج. أيضا نلاحظ هناك دوال عضو جديدة وهي setCourseName وَ

getCourseName ، بالإضافة إلى دالتنا السابقة وهي displayMessage. وظيفة الدالة setCourseName هو تخزين إسم الدورة في عضو البيانات في الكائن، والدالة getCourseName وضيفتها الحصول على إسم الدورة متى ما أردنا استخدامه. والدالة displayMessage
وظيفتها معروفة كما ذكرنا من قبل وهي طباعة رسالة الترحيب بالإضافة إلى اسم الدورة.

لممارسة البرمجة بشكل جيد :
ضع خط فاصل بين تعريف وآخر لدوال العضو في الصنف من أجل وضوح القراءة.

في السطر ٣٩ تلاحظون أنه تم الإعلان عن متغير جديد هو courseName من نوع string وهو عبارة عن عضو بيانات في الصنف GradeBook. وتلاحظون أيضا أننا استخدمنا هذا العضو في الدوال كما هو موضح في السطور ( ٩-٢٢ ، ٢٥-٢٨ ، ٣١-٣٧ ).
والفائدة من إنشاء هذا العضو courseName في الصنف GradeBook هوتمكين دوال هذا الصنف من إدارته والتعامل معه، وسوف نأتي لتفصيل هذا إن شاء الله .

محدد الوصول private :
عادةً تكون جميع أعضاء بيانات الصنف معرفة بعد الكلمة المفتاحية private. تعتبر هذه الكلمة محدد وصول خاص. لاحظ السطر ٣٨، هذا يعني أن عضو البيانات courseName خاص الوصول، لا يمكن الوصول له من خارج هذا الصنف، على سبيل المثال لا نستطيع أن نصله من الدالة main ولا من أي دالة أخرى، فقط يمكن الوصول إليه من خلال جسم الصنف وبالتحديد من دوال الأعضاء لنفس الصنف. حاول أن تصله من الدالة main ولاحظ ما يحدث ؟!!

**فكرة جيدة في هندسة البرمجيات :**
يجب أن تكون أعضاء البيانات معلن عنها بشكل خاص ودوال الأعضاء يجب أن يعلن عنها بشكل عام ( يمكن الوصول للأعضاء عن طريق الدوال في نفس الصنف ).

**خطأ برمجي شائع :**
محاولة الوصول إلى الأعضاء الخاصة عن طريق دوال خارجية خارج الصنف يولد خطأ في الترجمة .

الإعلان عن أعضاء البيانات بمحدد وصول خاص يسمى " إخفاء البيانات " ، لأن البيانات لا يمكن الوصول لها إلا عن طريق دوال الصنف نفسه، فبهذه الطريقة يصبح الكائن مكبسل ( مخفي ).
في كودنا السابق ( كود ١ ) ، دوال الأعضاء setCourseName وَ getCourseName وَ displayMessage هي
فقط التي تستطيع الوصول إلى عضو البيانات courseName مباشرة.

دوال العضو setCourseName وَ getCourseName :
دالة العضو setCourseName ( معرفة في السطور ١٩-٢٢ ) تستقبل وسيط واحد إسمه name ونوعة string، ولا ترجع شي. وهي تقوم بإسناد المتغير name إلى عضو البيانات courseName.
دالة العضو getCourseName ( معرفة في السطور ٢٥-٢٨ ) لا تستقبل شي وترجع بيانات من نوع string وتقوم بإرجاع القيمة courseName وهو عضو البيانات في الصنف.
ولتوضيح مسألة الإرجاع والبيانات المرجعة ، سوف نأخذ هذا المثال: عندما نذهب إلى الصراف الآلي ( ATM )
وتريد أن تستعلم عن الرصيد، أنت تدخل ( تعطي الدالة ) الرقم السري ورقم خيار الإستعلام، والصراف يخرج لك ( يعطيك ) مقدار رصيد حسابك.

**خطأ برمجي شائع :**
نسيان إرجاع قيمة لدالة من المفترض أن ترجع قيمة معينة يولد خطأ في الترجمة .

قلنا في السابق، أعضاء البيانات الخاصة لا يمكن الوصول لها إلا عن طريق الصنف نفسه، فلو تلاحظ السطرين ٢١، ٢٧ ، تجد أننا إستخدمنا courseName في هذه الدوال دون الإعلان عنهم فيها.
وهذا ما يفسر ما أشرنا إليه مسبقا. والترتيب مهم هنا ، فمثلا أن ننادي الدالة setCourseName لإعداد

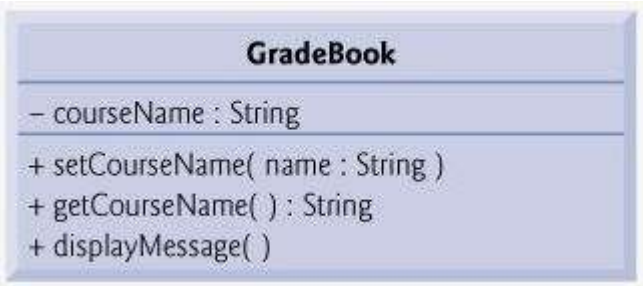
العضو ومن ثم مناداة الدالة getCourseName للحصول على القيمة المعدة، فلا يمكن تقديم الثانية على الاولى.

إختبار الصنف GradeBook ( السطور ٤٣-٦٠ ) :
السطر ٤٦ إنشاء كائن GradeBook وتسميته بـ myGradeBook
السطور ٤٩-٥٠ طباعة إسم الدورة قبل كل شي ( دوال الإعداد والحصول ) وبملاحظة أو سطر من الإخراج لا
يعرض إسم الدورة ، لأن عضو بيانات الكائن courseName فاضي في هذه اللحظة.
السطر ٥٣ يعطي رسالة للمستخدم لكي يدخل إسم الدورة.
السطر ٥٤ الدالة getline لإدخال إسم الدورة ، تم شرحها في المحاضرة الماضية.
السطر ٥٥ يتم استدعاء ( مناداة ) دالة الكائن myGradeBook وهي setCoursName وتقوم بإرسال المتغير nameOfCourse كمتغير ممرر للدالة ( في مثالنا هذا عند عملية الإستدعاء في هذا السطر فإن نسخة من المتغير nameOfCourse توضع في الوسيط.).
السطر ٥٨ إستدعاء لدالة العضو displayMessage للكائن myGradeBook لتعرض رسالة ترحيب بالإضافة إلى اسم الدورة.

هندسة الرمجيات مع دوال الـ set والـ get :
تزويد الصنف بدوال الـ set والـ get تمكن عملاء الصنف من الوصول إلى البيانات المخفية ولكن بشكل غير مباشر. العميل يعلم كيف يعدل أو يأخذ البيانات، لكنه لا يعلم كيف يتم ذلك.

**فكرة جيدة في هندسة البرمجيات :**
المبرمجين المحترفين يتوقعون أنه سوف تحدث تغييرات في البرامج ، فلذلك يقومون بتكييف أصنافهم على ذلك.

تحديث الـ UML للصنف GradeBook :
الشكل التالي يمثل الـ UML المحدث للصنف GradeBook في البرنامج السابق ( كود ١ ) بعد التعديلات واضافة
دوال الـ set والـ get وبيانات الأعضاء :




التغيير هو اضافة دالتين إضافيتين وخاصية ( عضو بيانات ) جديدة وهي courseName كما تلاحظون، في المستطيل في المنتصف وضعنا إسم الخاصية متبوعة بنقطتين ثم نوع الخاصية. يسبق إسم الخاصية إشارة ( - ) وهي تدل على أن هذه الخاصية محددة الوصول ( أي خاصة ) ليصبح تمثيلها كالتالي : - اسم الخاصية : نوع الخاصية
وهناك تغير آخر أيضا ، في دالة العضو getCourseName، كما تلاحظون بعد الأقواس هناك نقطتان عموديتان بعدهما تأتي كلمة string، وجود النقطتين بعد الأقواس يدل على أن العملية ( دالة العضو ) ترجع بيانات. الـ string هنا تمثل نوع البيانات المرجعة.

**[b]إعطاء الكائنات قيم أولية بالبواني**

[b/]

في المثال السابق ( كود ١ ) ، لاحظنا عندما انشأنا الكائن وعند طباعة عضو بياناته courseName كيف كان نا تج الطبلعة
سطر فاضي ، وهذا يدل على أن الكائن وبشكل افتراضي تم إعداده.
عند الإعلان عن الصنف تستطيع إنشاء باني ( وهو عبارة دالة عضو ) ، حيث يعطي الكائن عن انشاؤه قيمة أولية عن طريق
أحد اعضاؤه. والفرق بين بالبواني ودوال الأعضاء الأخرى هو أن البواني لا تستطيع أن ترجع قيمة ولا نوع قيمة ( حتى لو
كانت void ). في العادة ،وعند الإعلان عن البواني ، يعلن عنها في خانة الوصول العالم ( أي بعد محدد الوصول public ).

بأي حال من الأحوال ، عند انشاء كل كائن لأي صنف في الـ C++ فإن ذلك يتطلب إنشاء باني سواء كان افتراضي أو من إنشاء المستخدم.
حيث يُستدعي هذا الباني عند كل عملية إنشاء للكائن لكي يتأكد من أن الكائن تم انشاؤه بشكل صحيح. أما الباني الإفتراضي فإن المترجم يقوم
بإنشاؤه تلقائيا عندما لا يقوم المستخدم بذلك. على سبيل المثال ، في برنامجنا السابق ، عند السطر ٤٦، قمنا بإنشاء كائن لدفتر الدرجات ، في
هذه الحالة لم يقوم المستخدم بإنشاء باني ، وبشكل تلقائي وخفي ، قام المترجم بإنشاء باني افتراضي وسوف يتم استدعاؤه. ولمزيد من التوضيح
نأخذ هذا البرنامج كمثال توضيحي :
كود ٢ :

**كود**

[/size]

```
[size="3"]1  // Fig. 3.7: fig03_07.cpp
2  // Instantiating multiple objects of the GradeBook class and using
3  // the GradeBook Constructor to specify the course name
4  // when each GradeBook object is created.
5  #include <iostream>
6  using std::cout;
7  using std::endl;
8
9  #include <string> // program uses C++ standard string class
10  using std::string;
11
12  // GradeBook class definition
13  class GradeBook
14  {
15  public:
16     // Constructor initializes courseName with string supplied as argument
17     GradeBook( string name )
18     {
19        setCourseName( name ); // call set function to initialize courseName
20     } // end GradeBook constructor
21
22     // function to set the course name
23     void setCourseName( string name )
24     {
25        courseName = name; // store the course name in the object
26     } // end function setCourseName
27
28     // function to get the course name
29     string getCourseName()
30     {
```

```
31        return courseName; // return object's courseName
32      } // end function getCourseName
33
34      // display a welcome message to the GradeBook user
35      void displayMessage()
36      {
37        // call getCourseName to get the courseName
38        cout << "Welcome to the grade book forn" << getCourseName()
39          << "!" << endl;
40      } // end function displayMessage
41    private:
42      string courseName; // course name for this GradeBook
43    }; // end class GradeBook
44
45    // function main begins program execution
46    int main()
47    {
48      // create two GradeBook objects
49      GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
50      GradeBook gradeBook2( "CS102 Data Structures in C++" );
51
52      // display initial value of courseName for each GradeBook
53      cout << "gradeBook1 created for course: " <<
gradeBook1.getCourseName()
54        << "ngradeBook2 created for course: " <<
gradeBook2.getCourseName()
55        << endl;
56      return 0; // indicate successful termination
57    } // end main[/size]
```

[size="3"]

كيف نعرف الباني Constructor :

في السطور ( ١٧-٢٠ ) ، قمنا بتعريف باني للصنف GradeBook، ( لاحظ أن الباني يأخذ دائما نفس اسم الصنف ). في كثير من
الأحيان نكلف الباني بمهمة ما من خلال تزويدة ببعض الوسائط لإنجازها.
في السطر ١٧، يتضح من هذا السطر أن الباني يستقبل متغير من نوع سلسلة رمزية ( string ) ولا يقوم بإرجاع أي شي حتى لو كان ( void ).
في السطر ١٩، نلاحظ أن هناك إستدعاء ( مناداة ) للدالة setCourseName، وقد قمنا بإرسال القيمة الممررة name لها.
حيث تقوم هذه الدالة بإسناد هذه القيمة الممررة لعضو البيانات courseName كما هو واضح في السطور ( ٢٣-٢٦ ).

فحص الصنف GradeBook :

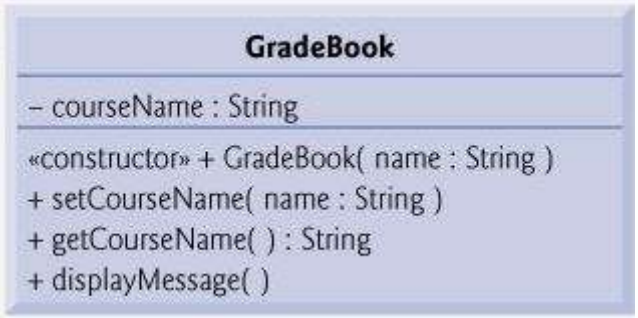
في السطر ٤٩، ننشيء الكائن ونعطيه قيمة أولية لنتأكد بذلك أن الكائن تم انشاؤه وتشغيله وأن كل شي على مايرام. الكائن اسمه gradeBook1، وقد
قمنا بإرسال القيمة الممرره " Programming ++CS101 Introduction to C " للكائن لكي تكون إسما للدورة كقيمة أولية.
في السطر ٥٠، نكرر العمليه لكائن جديد اسمه gradeBook2، وهذه المره نمرر القيمة " CS102 Data Structure in C++ " لإعطاء
اسم الدورة في هذا الكائن كقيمة اولية أيضا.
السطور ( ٥٣-٥٤ ) تستخدم الدالة getCourse لنحصل على إسم الدورة في كل كائن ونقوم بعرضه للمستخدم.

طريقتين لتزويد الصنف بباني إفتراضي :
أي باني لا يأخذ قيمة ممرره يسمى إفتراضي، الصنف يأخذ باني إفتراضي في حالتين :
١- إذا لم يحدد المستخدم باني افتراضي، المترجم بشكل تلقائي ينشيء باني إفتراضي ( عيك أن تخمن بالضبط مالذي سوف يفعله هذا الباني الإفتراضي ).
٢- يقوم المستخدم بإنشاء باني افتراضي، حيث لا يستقبل قيم ممرره. يحتوي هذا الباني عادة على عمليات اسناد أولية لأعضاء بيانات معينة.

ملاحظه : جميع البرامج السابقة التي تحتوي على أصناف، المترجم هو من تولى إنشاء بواني لهذه الأصناف.

**فكرة مفيدة في هندسة البرمجيات :**
فكرة انشاء بواني بواسطة المستخدم لكل كائن يتم انشاؤه، وذلك للتأكد من سلامة إنشاء الكائن وعدم الإعتماد على كود العميل في ذلك ( التأكد من سلامة الكائن ).

تحديث الـ UML للصنف GradeBook بعد إضافة الباني :
الـ UML يمثل برنامجنا السابق




التغيير هو إضافة باني يحتوي على وسيط واحد ( اسمه name ونوعة string )، وقد ذكرنا سابقا أن الباني يعتبر دالة عضو، لذلك وضعناه في
المستطيل السفلي في أعلى القائمة ( في المكان الذي عادةً ما يوضع فيه الباني ). ولكي نميز الباني عن دوال الأعضاء الأخرى ، وضعنا قبل اسم الباني
الكلمة Constructor بين أقواس صغيرة ، كما هو موضح في الـ UML في الأعلى .

**[b]وضع الصنف في ملف منفصل**

[b/]

الآن نريد أن نتقدم خطوة نحو تعزيز امكانية إعادة استخدام الصنف، وهي وضع الصنف في ملف منفصل حتى يمكن استخدامة من أي
برنامج ( عميل ) ، على سبيل المثال دالة الـ main. لو أي مبرمج أراد أن يستخدم صنفنا GradeBook في برنامجه سوف
لن يستطيع ان يضمّن ملف برنامجنا هذا (كود ٢) في برنامجه، وذلك لأن هناك دالة رئيسية أخرى . ( حاول أن تقوم بترجمة برنامج يحتوي
على دالتين رئيسيتين وتأمل رسائل الخطأ التي سوف تظهر لك ).

كود ٣ :

**كود**

[/size]

```
[size="3"]1  // Fig. 3.9: GradeBook.h
2  // GradeBook class definition in a separate file from main.
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include <string> // class GradeBook uses C++ standard string class
8  using std::string;
9
10  // GradeBook class definition
11  class GradeBook
12  {
13  public:
14     // Constructor initializes courseName with string supplied as argument
15     GradeBook( string name )
16     {
17        setCourseName( name ); // call set function to initialize courseName
18     } // end GradeBook Constructor
19
20     // function to set the course name
21     void setCourseName( string name )
22     {
23        courseName = name; // store the course name in the object
24     } // end function setCourseName
25
26     // function to get the course name
27     string getCourseName()
28     {
29        return courseName; // return object's courseName
30     } // end function getCourseName
31
32     // display a welcome message to the GradeBook user
33     void displayMessage()
34     {
35        // call getCourseName to get the courseName
36        cout << "Welcome to the grade book forn" << getCourseName()
37           << "!" << endl;
38     } // end function displayMessage
39  private:
40     string courseName; // course name for this GradeBook
41  }; // end class GradeBook[/size]

[size="3"][/size]

[size="3"]
```

كود ٤

**كود**

```
1  // Fig. 3.10: fig03_10.cpp
2  // Including class GradeBook from file GradeBook.h for use in main.
```

```
3  #include <iostream>
4  using std::cout;
5  using std::endl;
6
7  #include "GradeBook.h" // include definition of class GradeBook
8
9  // function main begins program execution
10  int main()
11  {
12     // create two GradeBook objects
13     GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
14     GradeBook gradeBook2( "CS102 Data Structures in C++" );
15
16     // display initial value of courseName for each GradeBook
17     cout << "gradeBook1 created for course: " <<
gradeBook1.getCourseName()
18        << "ngradeBook2 created for course: " <<
gradeBook2.getCourseName()
19        << endl;
20     return 0; // indicate successful termination
21  } // end main[/size]

[size="3"][/size]

[size="3"]
```

ملفات الترويسة ( header files ):

كل أمثلتنا ( برامجنا ) السابقة تحتوي على ملف واحد له الإمتداد .cpp ، وهو ما يسمى بملف كود المصدر source-code file ، هذا

البرنامج بشكله سوف يستخدم بشكل خاص ولكي نتيح تعميمة ليتمكن الجميع من استخدامه سوف نقوم بفصل الصنف كاملا بملف ترويسة

إسمه GradeBook.h (كود ٣ ) وإلى ملف آخر إسمه fig03_10.cpp كما هو واضح بالشكل ( كود ٤ ).

كما تلاحظون في إسم الملف GradeBook.h أنه ينتهي بالاحقة .h وهي تدل على أن هذا الملف ملف ترويسة.

في (كود ٣ ) نلاحظ أن الملف يحتوي فقط على الصنف ( السطور ١١- ٤١ )، في السطور ( ٣-٨ ) الصلاحيات المعطاه

لهذهالنصف بإستخدام مساحات الأسما والمكتبة < string > التي تحتوي على كائنات سلاسل الرموز.

في الملف الثاني ( كود ٤ ) توجد الدالة الرئيسية ( mian ) - الدالة السواقة - وفيه استخدمنا موجه المعالج الأولي #include وذلك

لتضمين ملف صنف الـ GradeBook كما تلاحظون في السطر ٧.

المبرمجين في الحقيقة عملهم ليس انشاء برامج بشكل دائم ، بل يقومون بإنشاء أصناف في ملفات منفصله ويقومون بفحصها عن طريق

البرنامج السواق " البرنامج الذي يحتوي على دالة الـ main " ، حيث يقومون بتضمين اصنافهم في هذه البرنامج عن طريق موجه

المعالج الأولي كما في السطر ٧ ( كود ٤ ) ويستخدم البرنامج الصنف كما لو كان مكتوب في نفس الكود، وايضا تستطيع هذه الأصناف تضمين

هذه الأصناف في أي برنامج نريد عن طريق هذه الموجه.

كيفية تحديد مكان ملفات الترويسة ( files header ):

عندما يواجه المعالج الأولي ملف ترويسة مضمن فإنه يقوم بتحديد مكانه عن طريق مساره الموضح بين علامتي التنصيص، فإذا كان

المسار غير محدد " أي مكتوب إسم الملف فقط " فإنه سوف يبحث عنه بنفس المجلد.

**خطأ برمجي شائع :**
إستخدام العلامات <> لتضمين ملف ترويسة من تعريف المستخدم ، وهذا خطأ ، حيث هذه العلامات خاصة بتضمين ملفات الترويسة
التي تكون مكتوبة في مكتبة الـ C++ القياسية فقط. فيجب أن تكتب إذن علامتي التصنيس " " بدلا من ذلك.

مسألة مهمه في هذه الطريقة :
في حقيقة الأمر أن هذه الطريقة تكشف تطبيقات الصنف بشكل كامل، ويكون مكشوف لجميع من تعامل معه هذا الصنف، وهذه طريقة
لا تصلح في تقنية إخفاء البيانات.
ما يحتاجه عملاء الصنف هو معرفة أسماء دوال الأعضاء وماذا تستقبل وماذا تقوم بإرجاعة، ولا يحتاج كود العميل معرفة تفاصيل
تطبيق هذه الدوال. إذن كود العميل يعرف تفاصيل الصنف، ربما يقوم بتأسيس كود يعتمد هذه التفاصيل وهذه خطأ فادح، يجب أن
لا يستطيع العملاء أن يقومون بهذا التغيير.
في موضوعنا القادم سوف نقوم بشرح طريقة فصل واجهة الصنف عن تفاصيل تطبيق دوالها لتلاشي العيب السابق.

**[b]فصل واجهة الصنف عن تطبيقة**

[b/]

لكي نعرف ماهية الواجهة سوف نذكر مثال الراديو ونشبه بالنصف لتوضيح ذلك.
نرى في واجهة أي راديو أزرار تحكمات تمكن المستخدم من رفع الصوت وتغيير المحطه ، والإختيار ما بين الـ FM والـ AM ، مختلف
أنواع اجهزة الراديو تحمل هذه الواجه ولكن بأشكال مختلفه.
المستخدم يعلم كيف يرفع الصوت ويغير المحطه ولكن لا يعلم تفاصيل ذلك، بشكل مشابه ، واجهة الصنف كذلك. فهي توضح للعميل
ماهي دوال الأعضاء وطريقة طلبها ( ماذا تستقبل وماذا ترجع ).

كود ٥

**كود**

[/size]

```
[size="3"]1  // Fig. 3.11: GradeBook.h
2  // GradeBook class definition. This file presents GradeBook's public
3  // interface without revealing the implementations of GradeBook's member
4  // functions, which are defined in GradeBook.cpp.
5  #include <string> // class GradeBook uses C++ standard string class
6  using std::string;
7
8  // GradeBook class definition
9  class GradeBook
10  {
11  public:
12     GradeBook( string ); // Constructor that initializes courseName
13     void setCourseName( string ); // function that sets the course name
```

```
   string getCourseName(); // function that gets the course name
   void displayMessage(); // function that displays a welcome message
private:
   string courseName; // course name for this GradeBook
}; // end class GradeBook
```

كود ٦

**كود**

```
// Fig. 3.12: GradeBook.cpp
// GradeBook member-function definitions. This file contains
// implementations of the member functions prototyped in GradeBook.h.
#include <iostream>
using std::cout;
using std::endl;

#include "GradeBook.h" // include definition of class GradeBook

// Constructor initializes courseName with string supplied as argument
GradeBook::GradeBook( string name )
{
   setCourseName( name ); // call set function to initialize courseName
} // end GradeBook Constructor

// function to set the course name
void GradeBook::setCourseName( string name )
{
   courseName = name; // store the course name in the object
} // end function setCourseName

// function to get the course name
string GradeBook::getCourseName()
{
   return courseName; // return object's courseName
} // end function getCourseName

// display a welcome message to the GradeBook user
void GradeBook::displayMessage()
{
   // call getCourseName to get the courseName
   cout << "Welcome to the grade book forn" << getCourseName()
      << "!" << endl;
} // end function displayMessage [/size]

[size="3"]
```

كود ٧

**كود**

[/size]

```
[size="3"]1  // Fig. 3.13: fig03_13.cpp
2  // GradeBook class demonstration after separating
3  // its interface from its implementation.
4  #include <iostream>
5  using std::cout;
6  using std::endl;
7
8  #include "GradeBook.h" // include definition of class GradeBook
9
10  // function main begins program execution
11  int main()
12  {
13     // create two GradeBook objects
14     GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
15     GradeBook gradeBook2( "CS102 Data Structures in C++" );
16
17     // display initial value of courseName for each GradeBook
18     cout << "gradeBook1 created for course: " <<
gradeBook1.getCourseName()
19        << "ngradeBook2 created for course: " <<
gradeBook2.getCourseName()
20        << endl;
21     return 0; // indicate successful termination
22  } // end main[/size]

[size="3"]
```

فصل الواجهه عن التطبيق :

في موضوعنا السابق قمنا بفصل الصنف عن الدالة الـ main . والآن لتحقيق خاصية الكبسلة " التغليف " نريد أن نفصل واجهة
الصنف عن تفاصيل دوال أعضائه " التطبيق " ذلك لمنع كود العميل من إستخدام " وتغيير " هذه التفاصيل.
نقوم بفصل واجهة الصنف عن تطبيقها بشطر " تقسيم " تعريف الصنف في ( الكود ٣ ) إلى ملفين ، ملف
الترويسة GradeBook.h ( الكود ٥ ) الذي تكون فيه الواجهه " تعرييف الصنف " وملف كود
المصدر GradeBook.cpp ( الكود ٦ ) وفيه تفاصيل إجراءات الدوال " التطبيق ".
لاحظو أن ملف التطبيق هو عبارة عن كود مصدر .cpp وإسمه نفس إسم الصنف وهذا ما جرت عليه العادة.
وأخيرا، ( الكود ٧ ) وهو الملف السواق الذي يقوم بفحص هذه الأصناف.

GradeBook.h تعريف واجهة الصنف مع التوصيفات الأولية للدوال ( prototypes ) :

( الكود ٥ ) يحتوي على واجهة الصنف " GradeBook.h " ولو قارنا هذه الواجهه بالكود ( كود ٣ )
لوجدنا أن في هذه الواجه
استخدمنا توصيف أولي لدوال الأعضاء بدلا من كتابة كامل تفاصيل الدالة كما في الكود ٣.
في الكود ٥ السطر ١٣ يعتبر توصيف أولي ( prototype ) وهي ما يحتاجه كود العميل فقط ( إسم الدالة
والوسائط المستقبله
والقيم المرجعه ) وهكذا الحال مع باقي السطور ( ١٢، ١٤، ١٥).
والفرق بين ترويسات الدالة ( كما في السطور ١٥، ٢١، ٢٧، ٣٣، في الكود ٣ ) والتوصيفات الأولية للدوال ( كما في
السطور ١٢، ١٣، ١٤، ١٥، في الكود ٥ ) شيئين :

١- التوصيف الأولي يحتوي على فاصلة منقوطه في نهاية السطر .
٢- التوصيف الأولي للدالة ، تسمية الوسائط فيه غير مهمه كما لو كان ترويسة دالة.

**خطأ برمجي شائع :**
نسيان الفاصلة المنقوطة في نهاية التوصيف الأولي للدالة يولد خطأ نصي .

**لممارسة البرمجة بشكل جيد :**
كثير من المبرمجين يضعون أسماء للوسائط في التوصيفات الأولية وذلك لغرض التوثيقي والتوضيحي .

**فكرة جيدة في منع الأخطاء:**
يقوم بالكثير من المبرمجين بنسخ السطر الأول لتعريف الدالة " ترويسة الدالة " ووضعه بدلا من التوصيف الأولي للدالة ووضع الفاصلة
المنقوطة ، وذلك لمنع التظليل.

GradeBook.cpp : تعريف دوال الأعضاء في ملف كود المصدر للتطبيق :
في الكود ٦ لاحظ أن كل إسم دالة ( ترويسة الدالة ) تكون مسبوقة بإسم الصنف ثم :: ، هذه البادئة تلحق كل تعريف لدالة العضو
بصنفها الخاص بها ، وبدونها لن يتعرف المترجم عليها، وسوف يعتبرها دوال حره أو مفقودة .

**خطأ برمجي شائع :**
نسيان البادئه ( إسم الصنف :: ) يولد خطأ ترجمه.

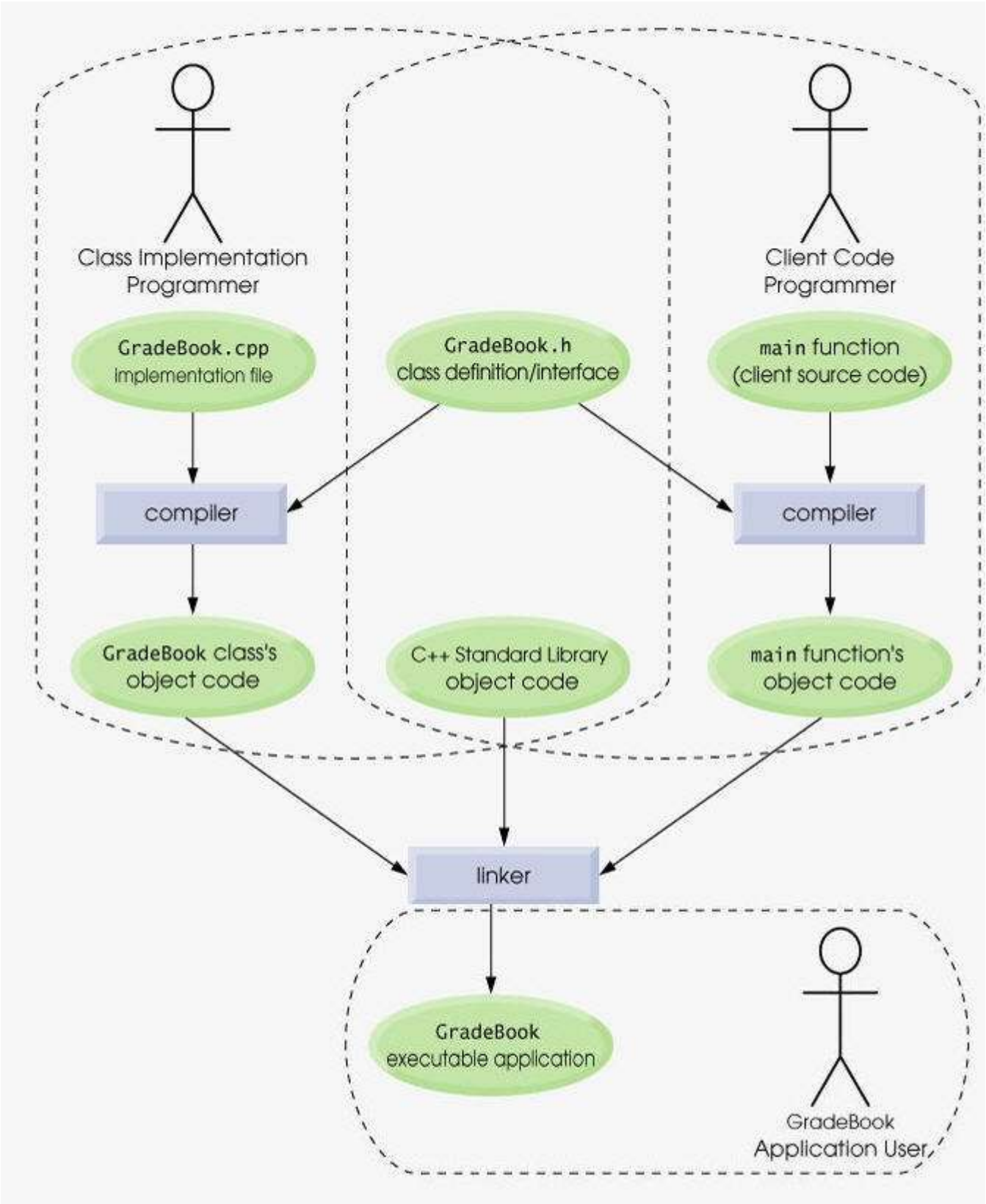
لاحظ في السطر ٨ ، وهو إدراج واجهة الصنف لكي نشير بذلك لدوال الأعضاء على أنها جز من هذا الصنف .

إختبار الصنف :
لا يوجد بهِ شي جديد عن سابقه ، وإذا كان هناك غير هذا عليك بإكتشافه بنفسك.

عملية الترجمة والربط :
الرسم التالي يوضح لنا عملية الربط والترجمة التي ينتج فيها نلا تطبيق GradeBook قابل للتنفيذ :

**خطأ!**

تم تحميل الدرس من شبكة المنهل التعليمية
http://111000.net




لإنشاء برنامج بتقنية برمجة الكائنات الموجهه ، في غالب الأحيان يقوم ببرمجته مبرمجين ، مبرمج كود العميل ، ومبرمج تطبيق الصنف ، وناتج
هذا ، تطبيق جاهز للإستخدام من قبل المستخدم.
مبرمج تطبيق الصنف مسؤول عن إنشاء صنف GradeBook وذلك بإنشاء ملف الترويسة

للمراسلة
w@111000.net

GradeBook.h وكود الكائن
للصنف GradeBook الذي يحتوي على تعليمات لغة الآلة التي تقوم بتمثيل دوال عضو الـ GradeBook.
وبالتالي مبرمج كود
العميل لا يعلم أي تفاصيل عما حدث.
مبرمج كود العميل لكي يستفيد من الصنف ويستطيع إستخدامه يقوم بإدراج هذه الصنف عن طريق موجه
المعالج الأولي #include.
عملية الربط في آخر خطوة ، بعد عملية الترجمة وأثناء عملية الربط يقوم بربط هذه الأجزاء التالية معا :
١- كود الكائن للدالة main.
٢- كود الكائن لتطبيقات دوال أعضاء الصنف GradeBook.
٣- كود الكائن الخاص بالمكتبات القياسية المستخدمة في البرنامج " مكتبة الـ string".

الناتج عن هذا ، تطبيق دفتر درجات جاهز للإستخدام.

**[b]تدقيق البيانات أثناء دالة الإعداد set**

[b/]

عند استخدام دالة setCourseName في المثال ( كود ٣ ) ، استخدمناها فقط لإعداد عضو البيانات
courseName وإسناد
قيمة له " سلسلة رموز " ، في الكود التالي نريد أن ندقق في سلسلة الرموز المستقبله ونرى هل عدد رموزها
أقل من او
تساوي ٢٥ رمز ( تسمى هذه العملية بالـ validation ) :

كود ٨

**كود**

```
// Fig. 3.15: GradeBook.h
// GradeBook class definition presents the public interface of
// the class. Member-function definitions appear in GradeBook.cpp.
#include <string> // program uses C++ standard string class
using std::string;

// GradeBook class definition
class GradeBook
{
public:
   GradeBook( string ); // Constructor that initializes a GradeBook object
   void setCourseName( string ); // function that sets the course name
   string getCourseName(); // function that gets the course name
   void displayMessage(); // function that displays a welcome message
private:
   string courseName; // course name for this GradeBook
}; // end class GradeBook
```

كود ٩

**كود**

```
// Fig. 3.16: GradeBook.cpp
// Implementations of the GradeBook member-function definitions.
// The setCourseName function performs validation.
#include <iostream>
using std::cout;
using std::endl;

#include "GradeBook.h" // include definition of class GradeBook

// Constructor initializes courseName with string supplied as argument
GradeBook::GradeBook( string name )
{
   setCourseName( name ); // validate and store courseName
} // end GradeBook Constructor

// function that sets the course name;
// ensures that the course name has at most 25 characters
void GradeBook::setCourseName( string name )
{
   if ( name.length() <= 25 ) // if name has 25 or fewer characters
      courseName = name; // store the course name in the object

   if ( name.length() > 25 ) // if name has more than 25 characters
   {
      // set courseName to first 25 characters of parameter name
      courseName = name.substr( 0, 25 ); // start at 0, length of 25

      cout << "Name "" << name << "" exceeds maximum length (25).n"
         << "Limiting courseName to first 25 characters.n" << endl;
   } // end if
} // end function setCourseName

// function to get the course name
string GradeBook::getCourseName()
{
   return courseName; // return object's courseName
} // end function getCourseName

// display a welcome message to the GradeBook user
void GradeBook::displayMessage()
{
   // call getCourseName to get the courseName
   cout << "Welcome to the grade book forn" << getCourseName()
      << "!" << endl;
} // end function displayMessage
```

كود ١٠

**كود**

```
// Fig. 3.17: fig03_17.cpp
// Create and manipulate a GradeBook object; illustrate validation.
#include <iostream>
using std::cout;
using std::endl;
```

```
6
7  #include "GradeBook.h" // include definition of class GradeBook
8
9  // function main begins program execution
10  int main()
11  {
12     // create two GradeBook objects;
13     // initial course name of gradeBook1 is too long
14     GradeBook gradeBook1( "CS101 Introduction to Programming in C++" );
15     GradeBook gradeBook2( "CS102 C++ Data Structures" );
16
17     // display each GradeBook's courseName
18     cout << "gradeBook1's initial course name is: "
19        << gradeBook1.getCourseName()
20        << "ngradeBook2's initial course name is: "
21        << gradeBook2.getCourseName() << endl;
22
23     // modify myGradeBook's courseName (with a valid-length string)
24     gradeBook1.setCourseName( "CS101 C++ Programming" );
25
26     // display each GradeBook's courseName
27     cout << "ngradeBook1's course name is: "
28        << gradeBook1.getCourseName()
29        << "ngradeBook2's course name is: "
30        << gradeBook2.getCourseName() << endl;
31     return 0; // indicate successful termination
32  } // end main
```

عملية التدقيق تتم في ملف التطبيق ( الكود ٩ ) تحديدا في دالة العضو setCourseName:

في السطر ٢٠ عبارة الـ if تحتوي على التعبير التالي name.length() ( قلنا فيما سبق ان جميع المتغيرات المعرفه على انها سلاسل

رموز " string " تعتبر كائنات معرفة ضمن مكتبة الـ C++ القياسية ) نحن حددنا فيما سبق أن الوسيط name من نوع سلسلة

رمزيه.، لذلك ، تكون name عبارة عن كائن ضمن مكتبة الـ C++ القياسية تحتوي على دالة أعضاء اسمها length لتحديد طول السلسلة

الرموزية ( أي عدد الرموز ).

في السطر ٢٣ ، عبارة If اخرى تحتوي على نفس الدالة السابقة ، في حال تحقق هذا الشرط ( وهوي أن السلسلة اطول من ٢٥ ) تقوم الدالة في

السطر ٢٤ name.substr(0 25 ) بإرجاع كائن string جديد بواسطة نسخ جز من السلسلة name يبدأ من

الموضع ٠ إلى ٢٥ وإسنادة لعضو البيانات courseName.

إختبار الصنف GradeBook في الكود ١٠:

في كود ١٠ إختبار للصنف بعد إجراء التدقيق في الدالة setCourseName.

في السطر ١٤، قمنا بإنشاء كائن إسمه gradeBook1 وعن طريق هذا الإنشاء قمنا بإستدعاء الباني الذي يقوم ايضا بإستدعاء

الدالة setCoursName التي تقوم بعلمية التدقيق التي قمنا بشرحها سابقا.

**فكرة جيدة في هندسة البرمجيات :**

عملية التدقيق تدل على دقة البرنامج وصحتة ، لذلك ينصح بهذه العمليات عند إعداد دوال الأعضاء للتحقق من أن البرنامج يعمل بكفاءة.

تم تحميل الدرس من شبكة المنهل التعليمية
http://111000.net

**http://www.arabteam2000-forum.com**

للمراسلة
w@111000.net