

4

Introduction to Classes and Objects

© 2006 Pearson Education, Inc. All rights reserved.

- ### OBJECTIVES
- In this chapter you will learn:
- How to implement a class's attributes as instance variables and properties.
 - The differences between instance variables of a class and local variables of a method.
 - How to use a constructor to ensure that an object's data is initialized when the object is created.
- © 2006 Pearson Education, Inc. All rights reserved.

- ### 4.1 Introduction
- **Classes**
 - Properties
 - Methods
 - Constructors
- © 2006 Pearson Education, Inc. All rights reserved.

- ### 4.5 Instance Variables and Properties
- **Variables declared in the body of method**
 - Called local variables
 - Can only be used within that method
 - **Variables declared in a class declaration**
 - Called fields or instance variables
 - Each object of the class has a separate instance of the variable
- © 2006 Pearson Education, Inc. All rights reserved.

- ### 4.5 Instance Variables and Properties (Cont.)
- **private keyword**
 - Used for most instance variables
 - `private` variables and methods are accessible only to methods of the class in which they are declared
 - Declaring instance variables `private` is known as information hiding
- © 2006 Pearson Education, Inc. All rights reserved.

- ### 4.5 Instance Variables and Properties (Cont.)
- **Property Declaration**
 - Declaration consist of an access modifier, type, and name
 - `get` and `set` allows you to access and modify private variables outside of the class, respectively
 - Contain a `get` accessor, `set` accessor, or both
 - After defining a property, you can use it like a variable
 - **get and set Accessors**
 - `get` accessor contains a `return` statement
 - `set` accessor contains a `variable = value`
 - `value` implicitly declared
- © 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 4.7: GradeBook.cs
2 // GradeBook class that contains a courseName instance variable,
3 // and a property to get and set its value.
4 using System;
5
6 public class GradeBook
7 {
8     private string courseName; // course name for this GradeBook
9
10    // property to get and set the course name
11    public string CourseName
12    {
13        get
14        {
15            return courseName;
16        } // end get
17        set
18        {
19            courseName = value;
20        } // end set
21    } // end property CourseName

```

Outline

GradeBook.cs
(1 of 2)

Instance variable courseName

get accessor for property courseName

set accessor for property courseName

© 2006 Pearson Education, Inc. All rights reserved.

```

22
23 // display a welcome message to the GradeBook user
24 public void DisplayMessage()
25 {
26     // use property CourseName to get the
27     // name of the course that this GradeBook represents
28     Console.WriteLine("Welcome to the grade book for\n{0}",
29         CourseName); // display property CourseName
30 } // end method DisplayMessage
31 } // end class GradeBook

```

Outline

GradeBook.cs
(2 of 2)

Call the get accessor

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 4.8: GradeBookTest.cs
2 // Create and manipulate a GradeBook object.
3 using System;
4
5 public class GradeBookTest
6 {
7     // Main method begins program execution
8     public static void Main( string[] args )
9     {
10        // create a GradeBook object and assign it to myGradeBook
11        GradeBook myGradeBook = new GradeBook();
12
13        // display initial value of CourseName
14        Console.WriteLine("Initial course name is: '{0}'\n",
15            myGradeBook.CourseName);
16
17        // prompt for and read course name
18        Console.WriteLine("Please enter the course name: ");
19        string theName = Console.ReadLine(); // read a line of text
20        myGradeBook.CourseName = theName; // set name using a property
21        Console.WriteLine(); // output a blank line

```

Outline

GradeBookTest.cs
(1 of 2)

Call get accessor of CourseName

Call set accessor of CourseName

© 2006 Pearson Education, Inc. All rights reserved.

```

22
23 // display welcome message after specifying course name
24 myGradeBook.DisplayMessage();
25 } // end Main
26 } // end class GradeBookTest

```

Outline

GradeBookTest.cs
(2 of 2)

Call DisplayMessage

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!

© 2006 Pearson Education, Inc. All rights reserved.

4.7 Software Engineering with Properties and set and get Accessors

- Public variable can be read or written by any property or method
- Private variables can only be access indirectly through the class's non-private properties
 - Class able to control how the data is set or returned
 - Allows for data validation
- Properties of a class should use class's own methods to manipulate the class's private instance variables
 - Creates more robust class

© 2006 Pearson Education, Inc. All rights reserved.

4.9 Initializing Objects with Constructors

- Constructors
 - Initialize an object of a class
 - Called when keyword new is followed by the class name and parentheses
 - Constructors can also take arguments
 - Constructor header similar to regular method header except the name is always the class name

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 4.12: GradeBook.cs
2 // GradeBook class with a constructor to initialize the course name.
3 using System;
4
5 public class GradeBook
6 {
7     private string courseName; // course name for this GradeBook
8
9     // constructor initializes courseName with string supplied as argument
10    public GradeBook( string name )
11    {
12        CourseName = name; // initialize courseName using property
13    } // end constructor
14
15    // property to get and set the course name
16    public string CourseName
17    {
18        get
19        {
20            return courseName;
21        } // end get
22        set
23        {
24            courseName = value;
25        } // end set
26    } // end property CourseName

```

Outline
GradeBook.cs
(1 of 2)

Constructor to initialize courseName variable

© 2006 Pearson Education, Inc. All rights reserved.

```

27 // display a welcome message to the GradeBook user
28 public void DisplayMessage()
29 {
30     // use property CourseName to get the
31     // name of the course that this GradeBook represents
32     Console.WriteLine( "Welcome to the grade book for\n{0}",
33         CourseName );
34 } // end method DisplayMessage
35 } // end class GradeBook

```

Outline
GradeBook.cs
(2 of 2)

© 2006 Pearson Education, Inc. All rights reserved.

```

1 // Fig. 4.13: GradeBookTest.cs
2 // GradeBook constructor used to specify the course name at the
3 // time each GradeBook object is created.
4 using System;
5
6 public class GradeBookTest
7 {
8     // Main method begins program execution
9     public static void Main( string[] args )
10    {
11        // create GradeBook object
12        GradeBook gradeBook1 = new GradeBook( // invokes constructor
13            "CS101 Introduction to C# Programming" );
14        GradeBook gradeBook2 = new GradeBook( // invokes constructor
15            "CS102 Data Structures in C#" );

```

Outline
GradeBookTest.cs

Call constructor to create first grade book object

Create second grade book object

© 2006 Pearson Education, Inc. All rights reserved.

```

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

```

Outline
GradeBookTest.cs
(2 of 2)

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

© 2006 Pearson Education, Inc. All rights reserved.