

Systems Development: Object Oriented Programming

(H172 35)

Introducing Classes

GradeBook Walkthrough

Step 4 – initialising objects with constructors

When a GradeBook object is created, its instance variable courseName is initialized to null by default. What if you want to provide an actual course name when you create a GradeBook object, so that it can have an initial value other than null?

Each class can provide a **constructor** method that can be used to initialise an object of a class when the object is created. In fact, C# requires a constructor call for every object that’s created and fortunately the compiler automatically provides a public default constructor (with no parameters) for each class that does not explicitly define a constructor, so every class has a constructor. The default constructor does not modify the default values of the instance variables.

When is a constructor called? - The **new** operator calls the class’s constructor to perform the initialisation. The constructor call is indicated by the class name, followed by parentheses. For example:

GradeBook myGradeBook = new **GradeBook()**;

**new** is used to create a GradeBook object. The empty parentheses after “new GradeBook()” indicate a call without arguments to the class’s constructor.

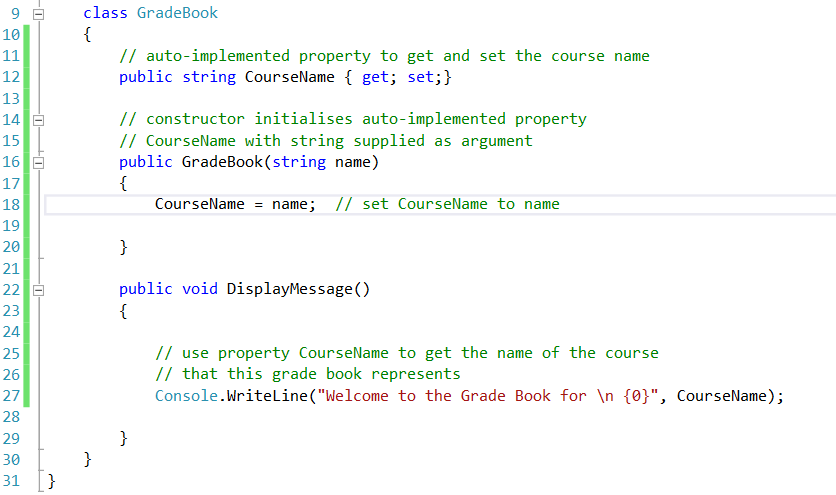
Custom initialisation with constructors

When you declare a class, you can provide your own constructor (or several constructors, as you’ll learn later) to specify custom initialisation for objects of your class. For example, you might want to specify a course name for a GradeBook object when the object is created, as in

GradeBook myGradeBook =

new GradeBook( "CS101 Introduction to C# Programming" );

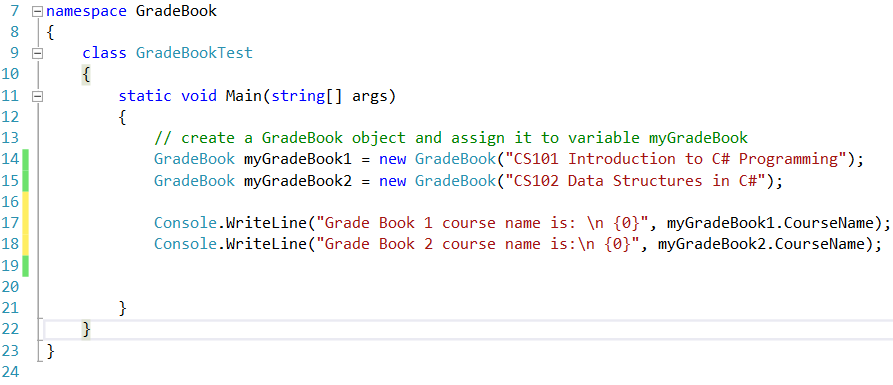
In this case, the argument "CS101 Introduction to C# Programming" is passed to the GradeBook object’s constructor and used to initialize the CourseName. Each time you create a new GradeBook object, you can provide a different course name. The preceding statement requires that the class provide a constructor with a string parameter. The code below shows a modified GradeBook class with such a constructor, update your GradeBook class in GradeBook.cs to have this custom constructor:

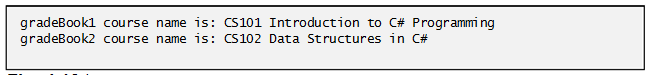


Lines 16–20 declare the constructor for class GradeBook. A constructor must have the same name as its class. Like a method, a constructor specifies in its parameter list the data it requires to perform its task. When you use new to create an object, you place this data in the parentheses that follow the class name. **Unlike a method, a constructor doesn’t specify a return type, not even void**. Line 16 indicates that class GradeBook’s constructor has a parameter called *name* of type string. In line 18, the *name* passed to the constructor is used to initialize auto-implemented property CourseName via its set accessor.

Initialising GradeBook objects with a custom constructor

Now we need to update our client code, GradeBookTest.cs, so that when a new GradeBook object is created, we use the custom constructor. Update your GradeBookTest.cs as follows:





Line 14 creates and initialises a GradeBook object. The constructor of class GradeBook is called with the argument "CS101 Introduction to C# Programming" to initialise the course name. The object-creation expression to the right of = in line 14 returns a reference to the new object, which is assigned to variable gradeBook1. Line 15 repeats this process for another GradeBook object, this time passing the argument "CS102 Data Structures in C#" to initialise the course name for gradeBook2. Lines 17–18 use each object’s Course-Name property to obtain the course names and show that they were indeed initialised when the objects were created. Previously, you learned that each instance (i.e., object) of a class contains its own copy of the class’s instance variables. The output confirms that each GradeBook maintains its own course name.