Systems Development: Object Oriented Programming

(H172 35)

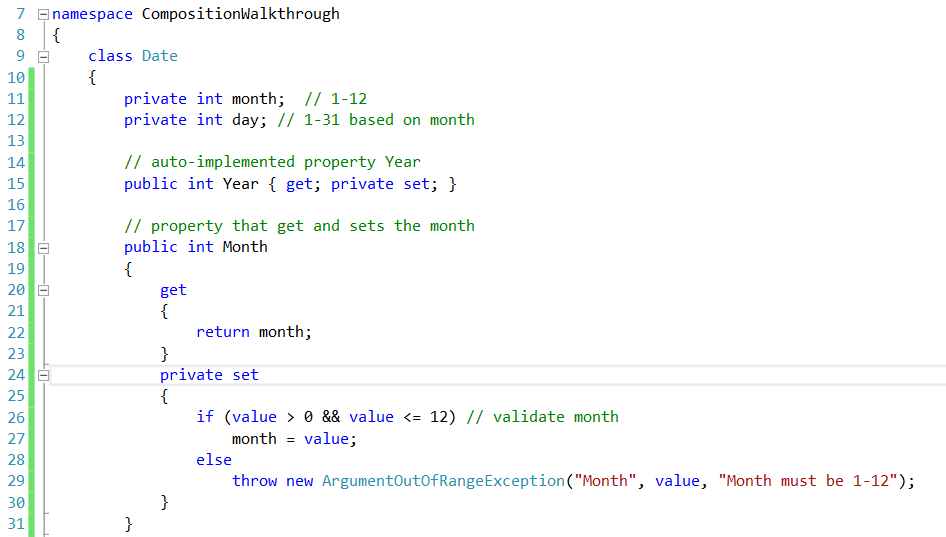
Composition (has-a) Walkthrough

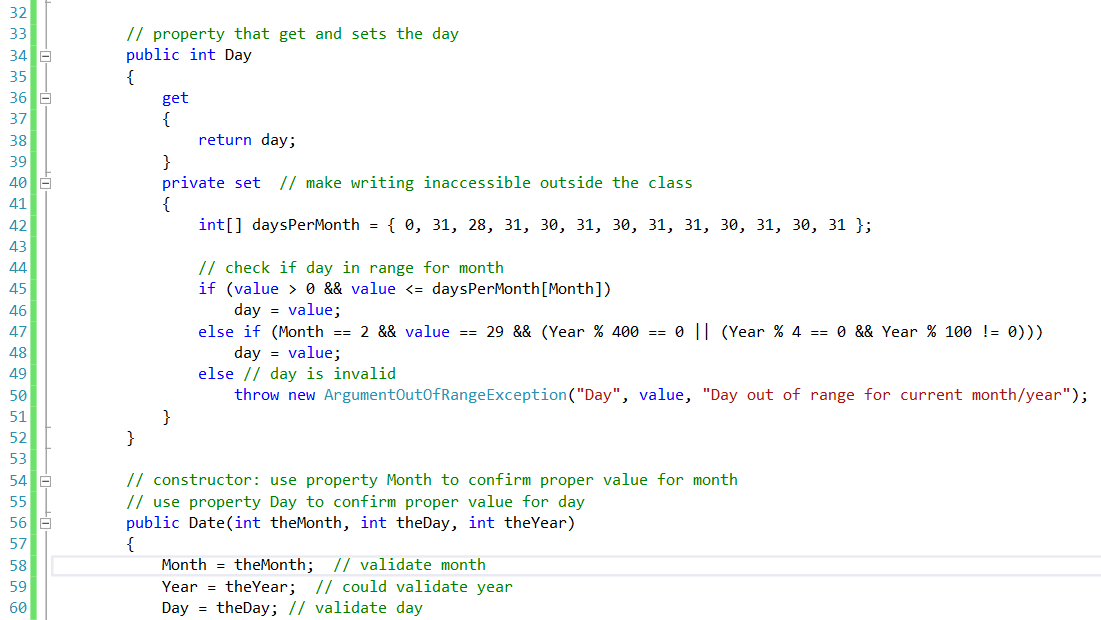


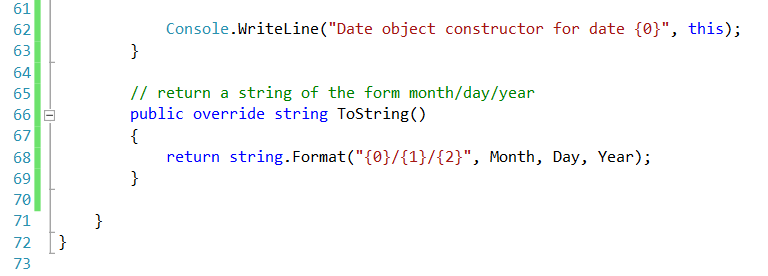
This walkthrough is to demonstrate the composition relationship (“has a”), where an application class EmployeeTest contains references to objects of other classes (Date and Employee) as member variables.

We will first look at the individual classes:

Class Date







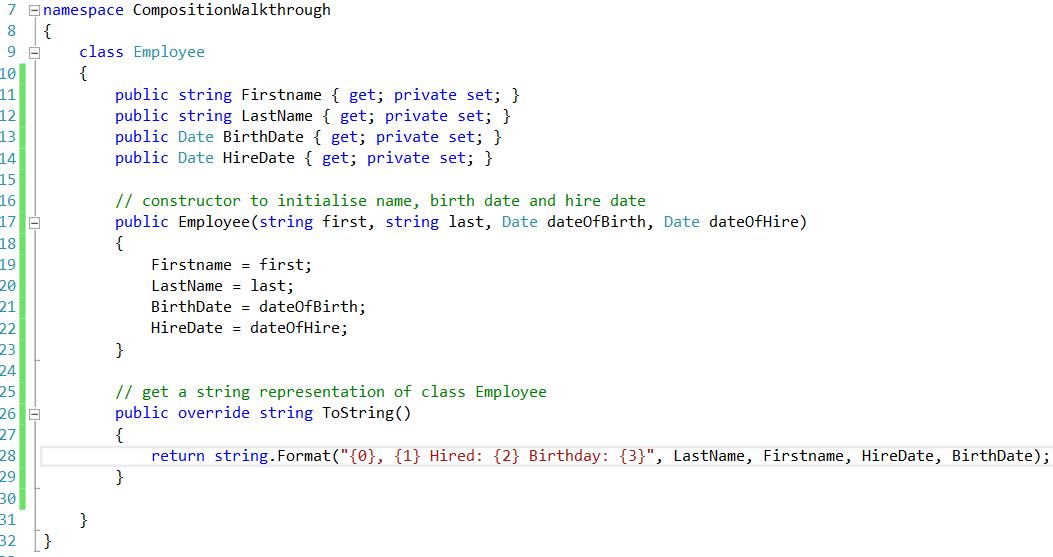
Class Date declares instance variables month and day (lines 11–12) and auto-implemented property Year (line 15) to represent a date. The constructor receives three int parameters. Line 58 invokes the set accessor of property Month (lines 24–30) to validate the month—if the value is out-of-range the accessor throws an ex-ception. Line 59 uses property Year to set the year. Since Year is an auto-implemented property, we’re assuming in this example that the value for Year is correct. Line 60 uses property Day (lines 34–52), which validates and assigns the value for day based on the current month and Year (by using properties Month and Year in turn to obtain the values of month and Year). The order of initialization is important, because the set accessor of property Day validates the value for day based on the assumption that month and Year are correct. Line 45 determines whether the day is correct based on the number of days in the particular Month. If the day is not correct, line 47 determines whether the Month is February, the day is 29 and the Year is a leap year. Otherwise, if the parameter value does not contain a correct value for day, the set accessor throws an exception. Line 62 in the constructor outputs the *this* reference as a string. Since this is a reference to the current Date object, the object’s To-String method (lines 66–69) is called implicitly to obtain the object’s string representation.

(Remember that each object inherits from the System.Object class which contains a ToString method, therefore for us to provide our own implementation of the ToString method we need to “*override*” the existing version”)

Class Date’s private set accessor

Class Date uses access modifiers to ensure that clients of the class must use the appropriate methods and properties to access private data. In particular, the properties Year, Month and Day declare private set accessors (lines 15, 24 and 40, respectively) to restrict the use of the set accessors to **members** of the class. We declare these private for the same reasons that we declare the instance variables private—to simplify code maintenance and control access to the class’s data. Although the constructor, method and properties in class Date still have all the advantages of using the set accessors to perform validation, clients of the class must use the class’s constructor to initialize the data in a Date object. The get accessors of properties Year, Month and Day are implicitly declared public because their prop-erties are declared public—when there’s no access modifier before a get or set accessor, the accessor inherits the access modifier preceding the property name.

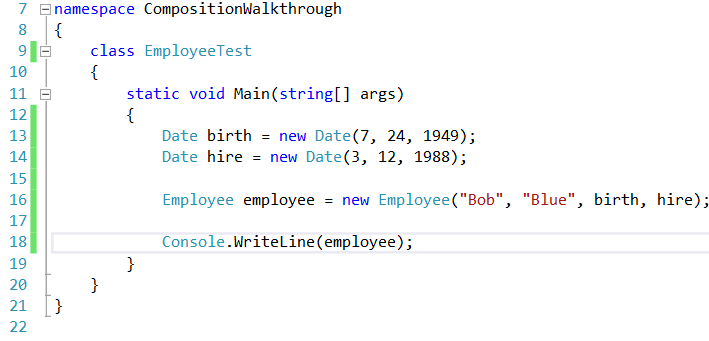
Class Employee



Class Employee has public auto-implemented properties FirstName, Last-

Name, BirthDate and HireDate. BirthDate and HireDate (lines 13–14) manipulate Date objects, demonstrating that a class can have **references to objects of other classes as members**. This, of course, is also true of the properties FirstName and LastName, which manipulate String objects. The Employee constructor (lines 17–23) takes four parameters—first, last, dateOfBirth and dateOfHire. The objects referenced by parameters dateOfBirth and dateOfHire are assigned to the Employee object’s BirthDate and HireDate properties, respectively. When class Employee’s ToString method is called, it returns a string containing the string representations of the two Date objects. Each of these strings is obtained with an implicit call to the Date class’s ToString method.

Class EmployeeTest



Class EmployeeTest creates two Date objects (lines 13–14) to represent an Em-ployee’s birthday and hire date, respectively. Line 16 creates an Employee and initializes its instance variables by passing to the constructor two strings (representing the Employee’s first and last names) and two Date objects (representing the birthday and hire date). Line 18 implicitly invokes the Employee’s ToString method to display the values of its instance variables and demonstrate that the object was initialized properly.