



OM&IS Gazette

Two Weeks Until Finals:

Yes, this week and next week- then finals! Refrain from cruising through these two weeks. Make sure you are diligent with your work, specific in your scheduling, and considerate with your well-being. Don't celebrate until you cross the finish line.

OMISpiration

“Being gifted doesn't mean you've been given something. It means, you have something to give.”

~Please find this~

31

Events

T-Shirt Tuesday—Wear your OM&IS apparel on Tuesdays for a chance to win a **FREE GIFT CARD!**

- AITP Elections - [Click here for details](#)

JOB POSTINGS

Visit this section each week for the latest OMIS job postings! Every week brings with it new opportunities. Make sure to check for attachments and links within these newsletters for additional job descriptions and details.

Scholarship Opportunity for Spring 2019 NIU Internship Students

The deadline for the spring 2019 awards is 4:30 p.m. on Wednesday, May 1, 2019.

[Click here for more details and the forms](#)



Graduation Receptions

Master's - Friday, May 10th from 4:00 PM to 5:30 PM at Barsema Hall Atrium - [More details](#)

Bachelor's - Saturday, May 11th from 11:00 AM to 12:30 PM at Barsema Hall Atrium - [More details](#)

Company Hangouts

Tuesday ~ April 23rd
10:00 a.m. - 02:00p.m.

AAR Corp. will be recruiting all undergraduate majors for full-time opportunities.

INTERNSHIP OPPORTUNITIES

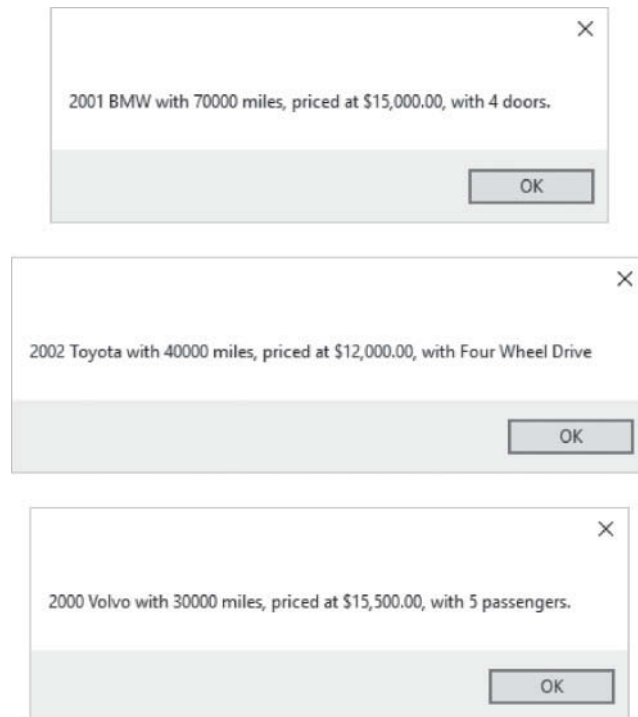
Visit this section each week for the latest OMIS Internship Opportunities!



Line 44 creates an instance of the `SportUtility` class, referenced by the `mySUV` variable. Lines 45–49 assign values to the objects `Make`, `Model`, `Mileage`, `Price`, and `Passengers` properties.

Lines 52–55 display the properties of the `myCar` object in the topmost message box shown in Figure 10-6. Lines 58–61 display the properties of the `myTruck` object in the middle message box shown in Figure 10-6. Lines 64–67 display the properties of the `mySUV` object in the bottom message box shown in Figure 10-6.

Figure 10-6 Message boxes displayed by the *Car Truck SUV Demo* project



In Tutorial 10-1, you create a base class for a savings account, and a derived class for a certificate of deposit.

Tutorial 10-1:

Creating and Testing the `SavingsAccount` and `CDAccount` Classes



VideoNote

Tutorial 10-1:
Creating and
Testing the
`SavingsAccount`
and
`CDAccount`
Classes

Bank Financial Systems, Inc., develops financial software for banks and credit unions. The company is developing a new system that manages customer accounts. One of your tasks is to develop a class that represents a savings account. The data that must be held by an object of this class is

- The account number
- The interest rate
- The account balance

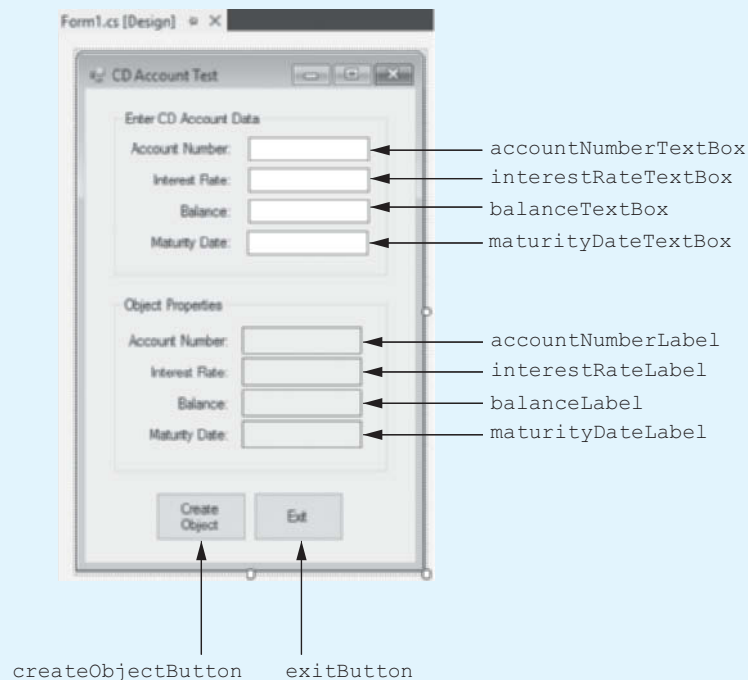
You must also develop a class that represents a certificate of deposit (CD) account. The data that must be held by an object of this class is

- The account number
- The interest rate
- The account balance
- The account maturity date

As you analyze these requirements, you realize that a CD account is really a specialized version of a savings account. The class that represents a CD will hold all the same data as the class that represents a savings account, plus an extra property for the maturity date. You decide to create a `SavingsAccount` class to represent a savings account and then create a class that is derived from `SavingsAccount`, named `CDAccount`, to represent a CD account.

To test the classes, you will use them in an application that lets the user enter data about a CD account, creates an object of the `CDAccount` class, sets the object's properties to the data that the user entered, and then displays the object's data. The application's form, which has already been created for you, is shown in Figure 10-7.

Figure 10-7 The *CD Account Test* application's form



Step 1: Start Visual Studio. Open the project named *CD Account Test* in the *Chap10* folder of the Student Sample Programs.

Step 2: Click *Project* on the Visual Studio menu bar and then select *Add Class. . .* The *Add New Item* window should appear. Make sure *Class* is selected as the type of item. Change the default filename to *SavingsAccount.cs* and then click the *Add* button. This adds a source code file named *SavingsAccount.cs* to the project.

Step 3: The *SavingsAccount.cs* file should now be displayed in the code editor. Complete the code for the `SavingsAccount` class by typing lines 11–43 in Program 10-3. The class code is very straightforward. It declares properties named

AccountNumber, InterestRate, and Balance, as well as the backing fields for the properties. The class also has a parameterless constructor that initializes the backing fields to default values.

Step 4: Click *Project* on the Visual Studio menu bar and then select *Add Class*. . . . The *Add New Item* window should appear. Make sure *Class* is selected as the type of item. Change the default filename to *CDAccount.cs* and then click the *Add* button. This adds a source code file named *CDAccount.cs* to the project.

Step 5: The *CDAccount.cs* file should now be displayed in the code editor. Complete the code for the *CDAccount* class as follows:

- At the end of the class header type `: SavingsAccount` to indicate that this class is derived from the *SavingsAccount* class. (See line 9 in Program 10-4.)
- Type the code shown in lines 11–25 in Program 10-4. The class code is very straightforward. It declares a property named *MaturityDate* as well as the backing field for the property. The class also has a parameterless constructor that initializes the backing field to an empty string.

Step 6: Open the *Form1.cs* file in the code editor. Type the comments and code for the *GetCDDData* method, shown in lines 20–57 of Program 10-5. Notice in line 23 that the method has a *CDAccount* parameter named *account*. When we call this method, we pass a *CDAccount* object to it as an argument.

The purpose of the *GetCDDData* method is to read the data that the user has entered into the form's text boxes and store that data in the *account* object's properties. The values that have been entered for the interest rate and the balance are validated.

Step 7: Next, you create the Click event handlers for the Button controls. Switch your view to the *Form1* form in the *Designer*. Double-click the *createObjectButton* control. This opens the *Form1.cs* file in the code editor, and you will see an empty event handler named *createObjectButton_Click*. Complete the event handler by typing the code shown in lines 61–71 in Program 10-5. Let's review this code:

Line 62: This statement creates a *CDAccount* object in memory, referenced by a variable named *myAccount*.

Line 65: This statement calls the *GetCDDData* method, passing the *myAccount* object as an argument. After the method executes, the *myAccount* object's properties will be set to the values entered by the user.

Lines 68–71: These statements display the values of the *myAccount* object's properties.

Step 8: Switch your view back to the *Form1* form in the *Designer* and double-click the *exitButton* control. In the code editor you will see an empty event handler named *exitButton_Click*. Complete the event handler by typing the code shown in lines 76–77 in Program 10-5.



Step 9: Save the project. Then, press  on the keyboard or click the *Start Debugging* button () on the toolbar to compile and run the application. When the application runs, enter some sample data in the *TextBox* controls and click the *Create Object* button. You should see the values that you entered displayed in the *Object Properties* group box. Figure 10-8 shows an example. Click the *Exit* button when you are finished.

Figure 10-8 The *CD Account Test* application



NOTE: The `MaturityDate` property in the `CDAccount` class is implemented as a `string`. If you prefer, you can use the `DateTime` data type provided by the .NET Framework. We have not covered the `DateTime` data type in this book, but if you feel adventurous, you can explore it on your own and devise a way to implement the `MaturityDate` property as a `DateTime` instead of a `string`.

Program 10-3 Completed code for the `SavingsAccount.cs` file in the *CD Account Test* application

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace CD_Account_Test
8 {
9     class SavingsAccount
10    {
11        // Fields
12        private string _accountNumber;
13        private decimal _interestRate;
14        private decimal _balance;
15
16        // Constructor
17        public SavingsAccount()
18        {
19            _accountNumber = "";
20            _interestRate = 0;
21            _balance = 0;
22        }
23
24        // AccountNumber property

```