Complete each of the following exercises from the Deitel book, using the indicated C# project names. You should use a .NET Framework Console application type for each project. All projects should be part of a single solution named: **Cs2Apps.**

**How to setup multiple Projects in one Visual Studio Solution**

When creating the first Project, make sure to ***uncheck the box*** for putting the solution and the project in the same folder. This will create a separate folder with the solution file by itself and the first project in a folder under that. For each subsequent project, use the "**File->Add->New Project**" to add each project to the \*\*same\*\* solution. If you need to review, watch video VS-Solution-containing-multiple-Projects.

To switch projects among those listed in the Solution Explorer window, right click on the project and select "**Set as StartUp Project**". The selected startup project will display as bold text in the Solution Explorer window.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  
Program1 - (from ch4) - Employee Class (Project Name: EmployeeDB) -** Create a class called Employee that includes three pieces of information as either instance variables or auto-implemented properties - first name (type string), last name (type string) and a monthly salary (decimal). Your class should have a constructor that initializes the three values. Provide a property with a get and set block for any instance variables. If code attempts to set the monthly salary negative, the set block should leave the instance variable unchanged. Write a test driver program for Employee that demonstrates the Employee class capabilities (use **Program.cs**for this). Create 3 Employee objects and display each object’s *yearly* salary. Then give each Employee a 10% raise and display each Employee’s yearly salary again.

Show the results of the test run in a plain text file zipped with the rest of the project.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  
Program2 - (from ch6) - Displaying a Bar Chart (Project Name: BarChart) -** One interesting application of computers is to display graphs and bar charts. Write an app that reads three numbers between 1 and 30. For each number that’s read, your app should display the same number of adjacent asterisks. For example, if your app reads the number 7, it should display 7 stars: **\*\*\*\*\*\*\***

Execute test runs and make output charts of 3, 5, 8, 13, and 21 stars.  
Show the results of this test in a plain text file zipped with the rest of the project.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  
Program3 - (from ch7) - Guess-the-Number Game (Project Name: GuessTheNumber) -** Write an app that plays “Guess the Number” as follows: Your app chooses the number to be guessed by selecting a random integer in the range 1 to 1000. The app displays the prompt "Guess a number between 1 and 1000: ". The player inputs a first guess. If the player’s guess is incorrect, your app should display **Too high. Try again.** or **Too low. Try again.** to help the player “zero in” on the correct answer. The app should prompt the user for the next guess. When the user enters the correct answer, display **Congratulations. You guessed the number!** and allow the user to choose whether to play again. [*Note:* The guessing technique employed in this problem is using a similar strategy to a binary search, which is discussed in Chapter 18]

Execute 3 or 4 separate test runs.  
Show the results of this test in a plain text file zipped with the rest of the project.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  
Program4 - (from ch7) - Computer-Assisted Instruction (Project Name: AITeacher)** - The use of computers in education is referred to as *computer-assisted instruction* (*CAI*). Write a program that will help an elementary-school student learn multiplication. Use a Random object to produce two positive one-digit integers. The program should then prompt the user with a question, such as: How much is 6 times 7?

The student then inputs the answer. Next, the program checks the student’s answer. If it is correct, display the message **"Very good!"** and ask another multiplication question. If the answer is wrong, display the message **"No. Please try again."** and let the student try the same question repeatedly until the student gets it right. A separate method should be used to generate each new question. This method should be called once when the app begins execution and each time the user answers the question correctly.

Execute 3 or 4 separate test runs.  
Show the results of this test in a plain text file zipped with the rest of the project.

Graphical user interface, text, application, email

Description automatically generated