ASSIGNMENTS:

|  |
| --- |
| **Lesson 1: Computer Basics and the History of Programming** |
| Write out an algorithm for how to tie your shoes.  First, do this by drawing a flowchart.  When you are satisfied that your algorithm is correct, give it to someone else and see if they can use your directions to tie their shoes.  Next, do the same thing using pseudocode (English-like statements).  Which one was easier for you? Which one was easier for your user? |
| **Lesson 2: The C# Compiler and Your First C# Program** |
| Write a class named *Favorites*.  This class should contain a Main method that has statements in it to display your name on the screen. The program should also display your favorite movie on a second line and your favorite color on a third line.  When you are finished typing the file, save it, compile it and run it.   |  | | --- | | https://api.ed2go.com/CourseBuilder/2.0/images/resources/prod/global/images/info.png**Unzipping a File**  In the remaining lessons of the course you'll be able to download the Solution for each Assignment. These Solutions for this course are *zipped*, meaning it's compressed to make easier to download.  To download the file, **right-click** the link and choose **Save Target As** from the menu. Save the zip file to a location on your computer that you will remember.  After downloading the file, you'll have to unzip it to get out the document. **Here's how using Windows XP**:   1. After the file has finished downloading, locate the zip file. 2. Right-click the **Favorites.zip** icon, and from the menu that pops up, click **Open With**. 3. You'll see another menu. Click **Compressed (zipped) Folders**.   how to unzip   1. This will open a new window showing the file that is in the Zip folder. In the left hand pane of this window, click **Extract all files**.   how to unzip- part 2   1. This will extract the file from the ZIP file, and you'll see the **Favorites.cs** document in your folder.   **Here's how using Windows Vista or Windows 7**:   1. After the file has finished downloading, locate the Favorites.zip file. 2. Right-click the Favorites.zip icon, and from the menu that pops up, click Extract All.   Extract All   1. A dialog box appears, letting you choose the folder where you want to extract the files. Click on a folder and then click the Extract button.   how to unzip in Windows Vista   1. This will extract the .CS file from the ZIP file, and you'll see two new document in your folder.   Unzipping files is pretty simple! If you have any trouble with it, let me know in the Discussion Area. Please be sure to save and unzip the files in each lesson, as you'll want to see the solutions to the assignments. |   See Example folder: **Favorites.cs** |
| **Lesson 3: Data Types and Mathematical Operators** |
| A friend of yours has a job installing carpet. To better serve their customers, your friend asks you to write a program that will calculate the cost of carpeting a client's floor. Write a program that will ask the user for the length and width of the room in feet. Next, ask the user for the cost of the carpet per square foot in dollars and cents. Using this information, calculate and display in a friendly, descriptive sentence, the area of the room (area = length \* width) and the cost of carpeting that room (cost = area \* price per square foot).  See Example folder: **FloorArea.cs** |
| **Lesson 4: Value Returning Methods** |
| Write a program that will calculate the hypotenuse of a right triangle. Have your Main method get the lengths of the two sides. It should then call on a method called *calcHypotenuse*. This method should take two doubles as parameters and return the hypotenuse as a double. Remember that to calculate the hypotenuse, you add the square of the first side to the square of the second side. Take the square root of that sum to get the hypotenuse. For example, to calculate a right triangle with sides of length 3 and 4:  hypotenuse = square root ( 3^2 + 4^2 ) = 5  See Example folder: **RightTriangle.cs** |
| **Lesson 5: Void Methods and Overloading** |
| Write a program that will help to balance your checkbook. To help practice the ability to pass variables by reference, you should write the following methods:   * A method that reads in a double from the keyboard. Remember, all input is read in as a string, and then it must be converted to the appropriate data type. This method should be a void method that takes a reference to a double variable as a parameter. * A void method that takes the beginning balance, the total deposits, the total withdrawals, and a reference to the variable containing the ending balance.   See Example folder: **CheckBook.cs** |
| **Lesson 6: The *if* Selection Structure** |
| Write a program that will display the user's letter grade based on their numerical average. Prompt the user for a number, and then display the corresponding letter grade based on the 10-point scale:  A 90 - 100 B 80 - 89.9 C 70 - 79.9 D 60 - 69.9 F < 60.0="">  See Example Folder: **LetterGrade.cs** |
| **Lesson 7: More About the Selection Structure** |
| Write a program that will prompt the user for their year number in school, 1 through 4. Based on this response, your program should display their class rank as Freshman (1), Sophomore (2), Junior (3), Senior (4), or Error (any other number).  To demonstrate your knowledge of the different selection structures, after prompting the user for their year number, also ask them if they want the program to use a nested if structure or a switch structure.  Make your program more organized by doing all the input and output in the Main function, and create two different value returning functions, getRankIf() and getRankSwitch().  See Example Folder: **YearInSchool.cs** |
| **Lesson 8: The Repetition Structure** |
| Write a program that uses a loop to ask the user for a list of numbers. The program should stop looping when the user enters *999*. When the user enters that value, your program should display the sum of all the values that were entered. Do not include the value 999 in the sum.  See Example Folder: **SumNumbers.cs** |
| **Lesson 9: Arrays** |
| Write a program that prompts the user for a list of up to 100 numbers to be stored in an array. The program should stop asking for numbers if the user enters the value *999* or if the array is full. When the user is finished entering values, calculate the average of only those numbers entered.  See Example Folder: **SumNumbersArray.cs** |
| **Lesson 10: Classes** |
| Finish the Time class that was started in this lesson. Do this by completing the following methods:   * **SetMinute(int m)** * **SetSecond(int s)** * **GetMinute()** * **GetSecond()** * **AddSecond()** - This method should add 1 to the second variable stored in the current object. If the second becomes 60, be sure to set the second equal to 0 and increment the minute by one. * **AddMinute()** - This method should add 1 to the minute variable stored in the current object. If the minute becomes 60, be sure to set the minute equal to 0 and increment the hour variable by one. * **AddHour()** - This method should add 1 to the hour variable stored in the current object. If the hour becomes 24, be sure to set the hour equal to 0. * **DisplayCivilian()** - This method should display the time stored in the object as civilian time. * **DisplayMilitary()** - This method should display the time stored in the object as military time. * **Equals(Time t)** - This method should compare the time stored in the current object to the time stored in the Time object *t* that is passed into the method.   See Example Folder: **Time.cs**  Now change the code in the UsingTime class to test each of these methods.  See Example Folder: **UsingTime.cs** |
| **Lesson 11: Exceptions and Files** |
| Write a program that will keep track of movies you have seen using files. In addition to storing the name of the movie (a string), also keep track of the movie's rating (int) as a number of stars between 1 and 5. To make your program more flexible, use a file to keep track of the data so that it can be added to in the future. Here is a suggestion on how to make your program work:  When Main starts, it should open the data file and read in the movie data into two arrays, one for the name and one for the rating. Next have your program display a menu with choices to   1. display movie data 2. add movie data 3. quit   When the user chooses to quit, copy the data from the arrays to the data file. To make your program more flexible, be sure to handle any exceptions that may be thrown from invalid input.  If you want to practice more of your C# skills, write a class called Movie with two instance variables. Change your Main program so that it keeps track of the data using a Movie array variable. You could also add an option to the menu to delete a movie. You now have the knowledge to make very useful programs. Have fun with this!  See Example Folder: **MovieFileData.cs** |
| **Lesson 12: Graphical User Interfaces** |
| For your final assignment, I want you to re-write the previous assignment as a GUI application. In case you have forgotten, here is the problem:  Write a program that will keep track of movies you have seen using files. In addition to storing the name of the movie (a string), also keep track of the movie's rating (int) as a number of stars between 1 and 5. To make your program more flexible, use a file to keep track of the data so that it can be added to in the future.  In the previous lesson you were asked to write a menu system on the Console Window. Now, you can instead create three buttons:   1. Display the movie data 2. Add movie data 3. Quit   When the user chooses to quit, copy the data from the arrays to the data file. To make your program more flexible, be sure to handle any exceptions that may be thrown from invalid input.  If you chose to write a Movie class in the previous lesson, then you should only need to build your interface and make the GUI talk to it.  I want to tell you about a couple of text box properties that will make this program a little easier to write:   1. Multiline. If you make the Height property bigger and correctly set the Multiline property, your text box can display more than one line of text. To set this property, just type:  <Control Name>.Multiline = true; 2. Scroll Bars. If your text box can show multiple lines, you will want a vertical scroll bar to make it easier for your user to view the text. To set this property, type:  <Control Name>.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;   [Click here for solution: **MovieGUI.zip**](https://api.ed2go.com/CourseBuilder/2.0/images/resources/prod/cpb-0/MovieGUI.zip) |