CST-227 Activity 6: File IO and LINQ Guide

Part 1 - File IO in C# 2

Reading and Writing Data to a Class 6

Error Checking 10

Part 2 - LINQ Demo 13

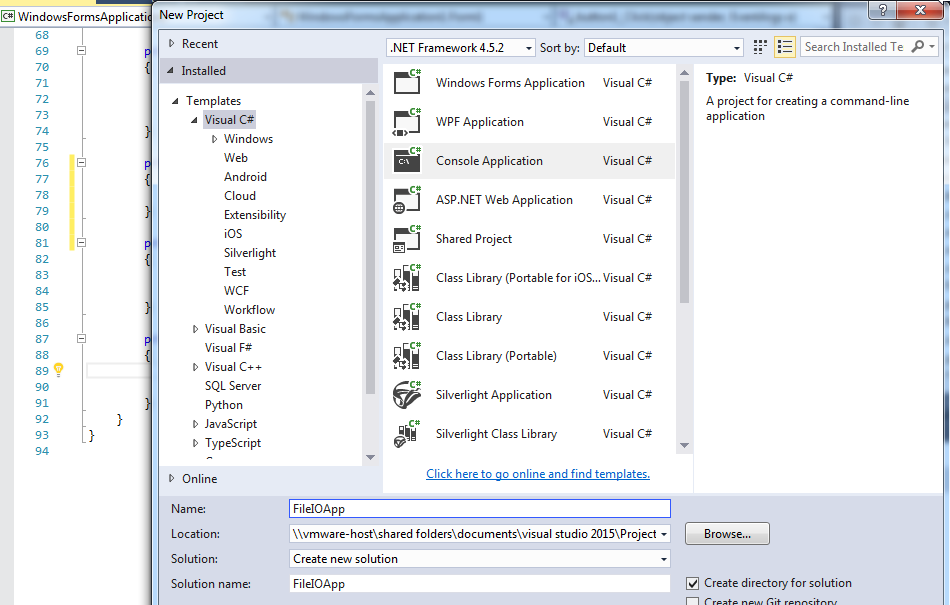
# Part 1 - File IO in C#

**Objective:** In this activity you will demonstrate: how to

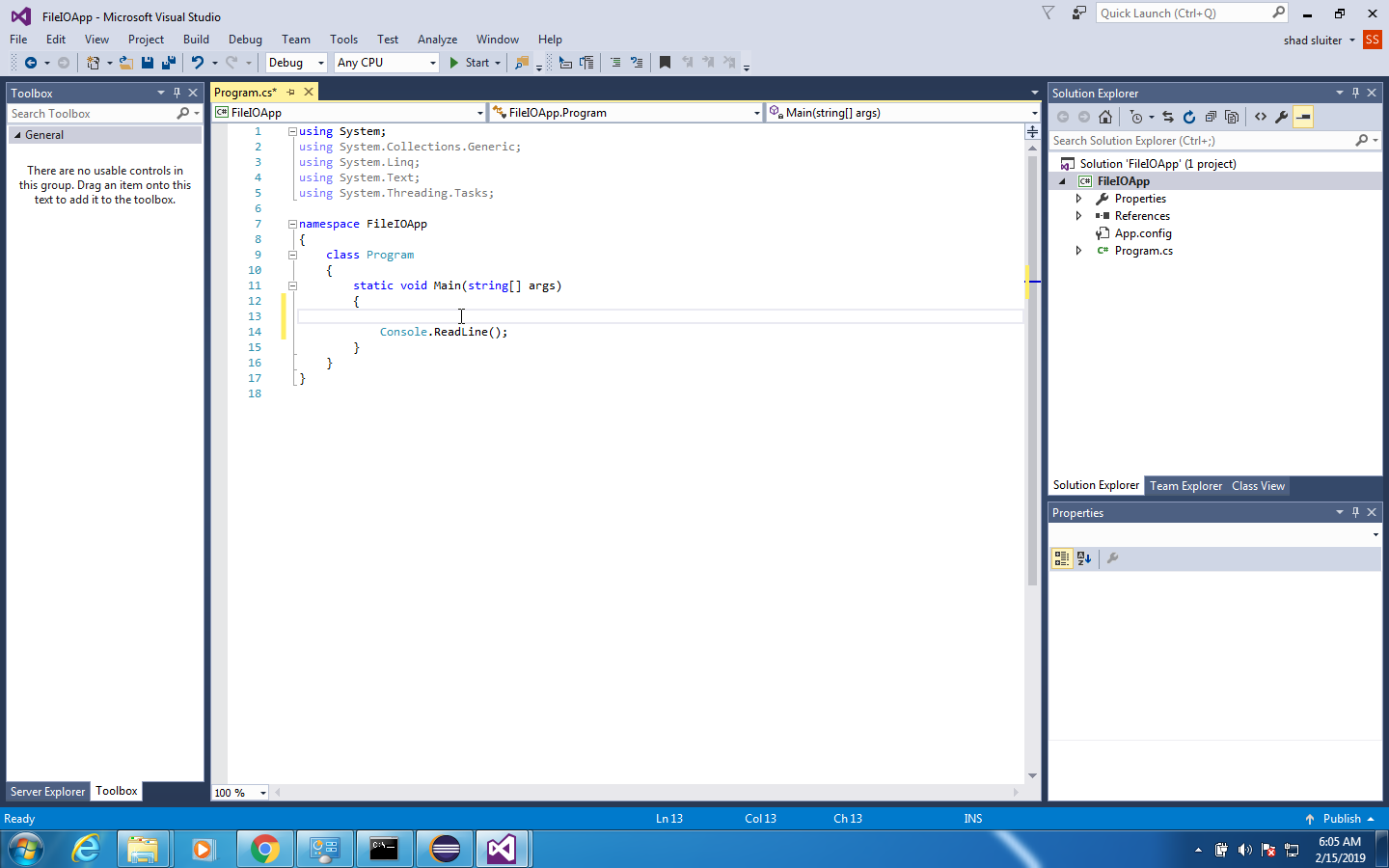
1. Read and write to a text file.
2. Translate lines of text into properties of objects in a list.
3. Check for errors.
4. Adapt the technique to a GUI application.

**Instructions**

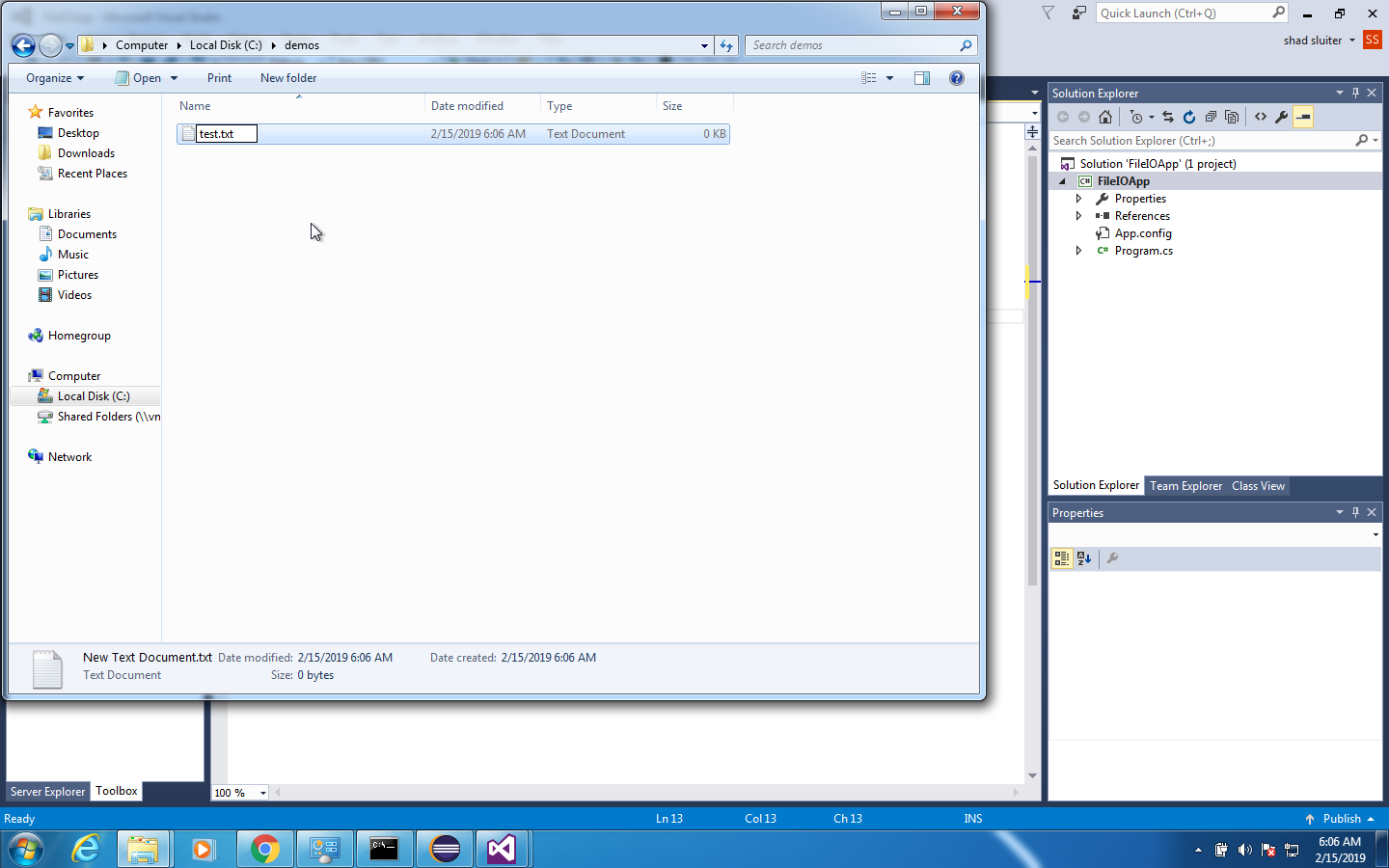
1. Create a new project called "TextFileDataAccessDemo"



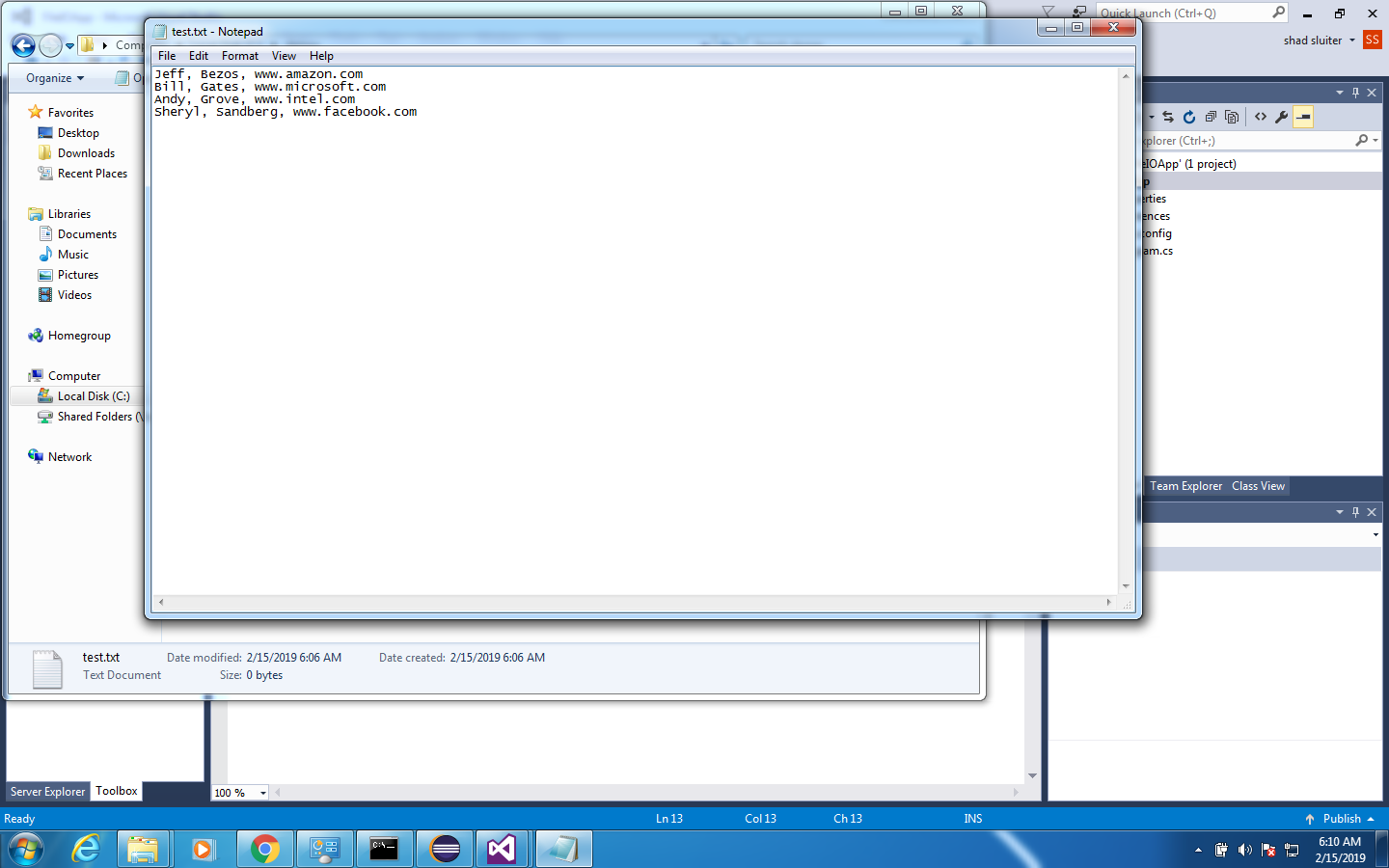
1. Start with Console.ReadLine() command to pause the program after execution.



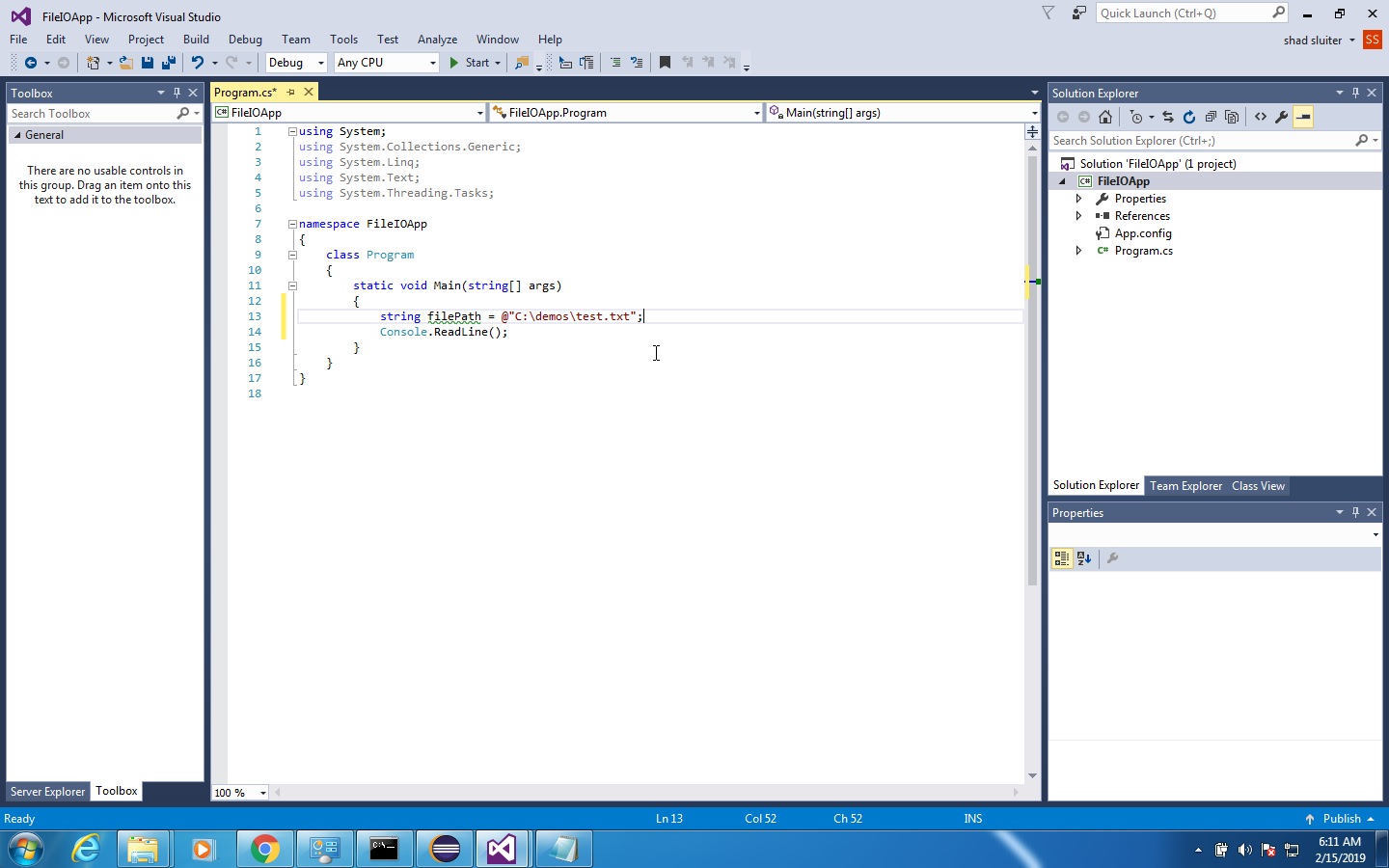
1. In the file explorer, create a text file. I am putting the file in c:\demos.



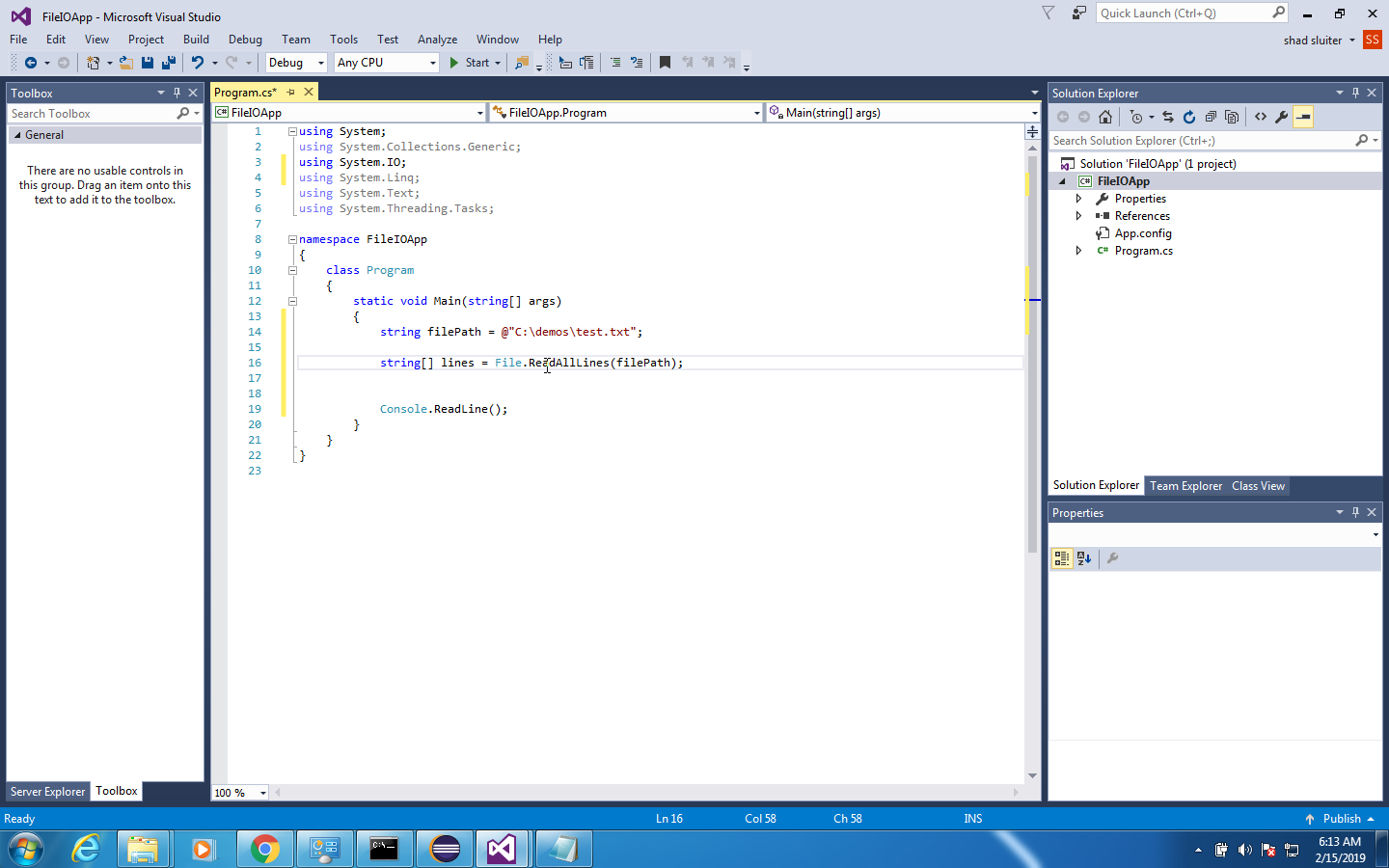
1. Let's add some data to the file. I am using a list of first name, last name and a URL separated by commas. Create a similar set of data.



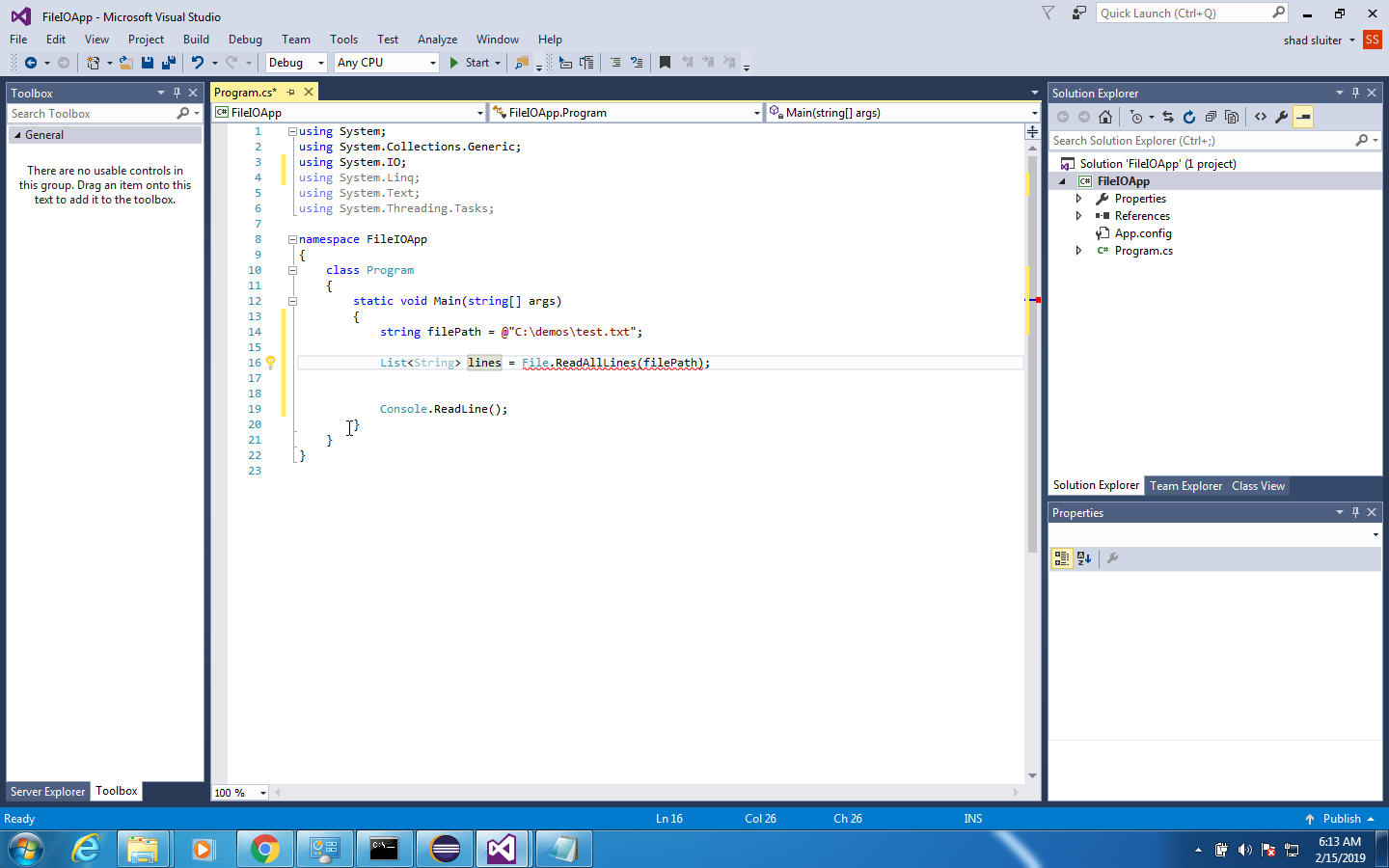
1. Add a reference to this new file. The @ symbol means "string literal follows. No escape characters needed."



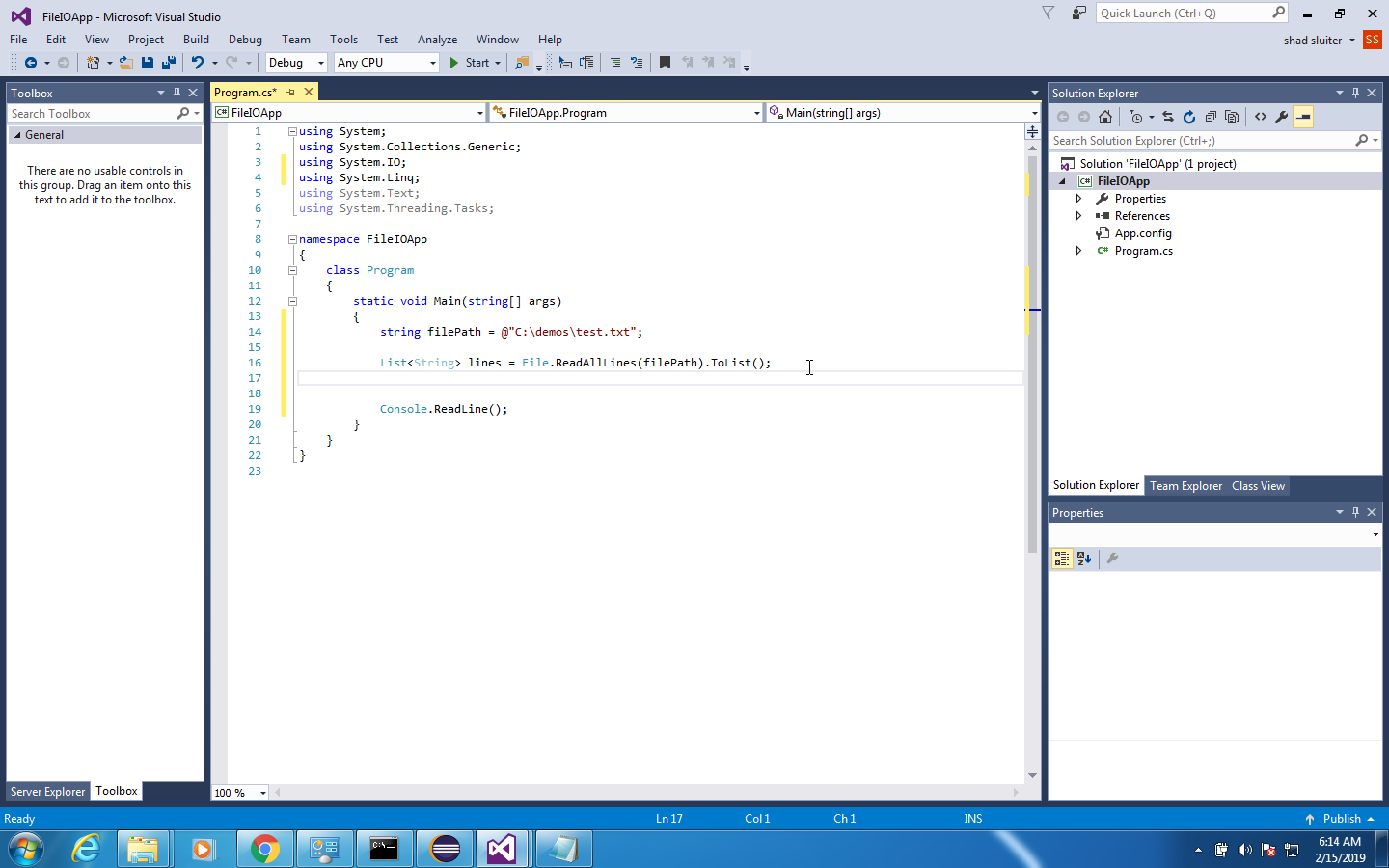
1. We want to read all lines from this text file. Notice that you will need to add "using System.IO" to use this next function.
2. We could save the result in an array…



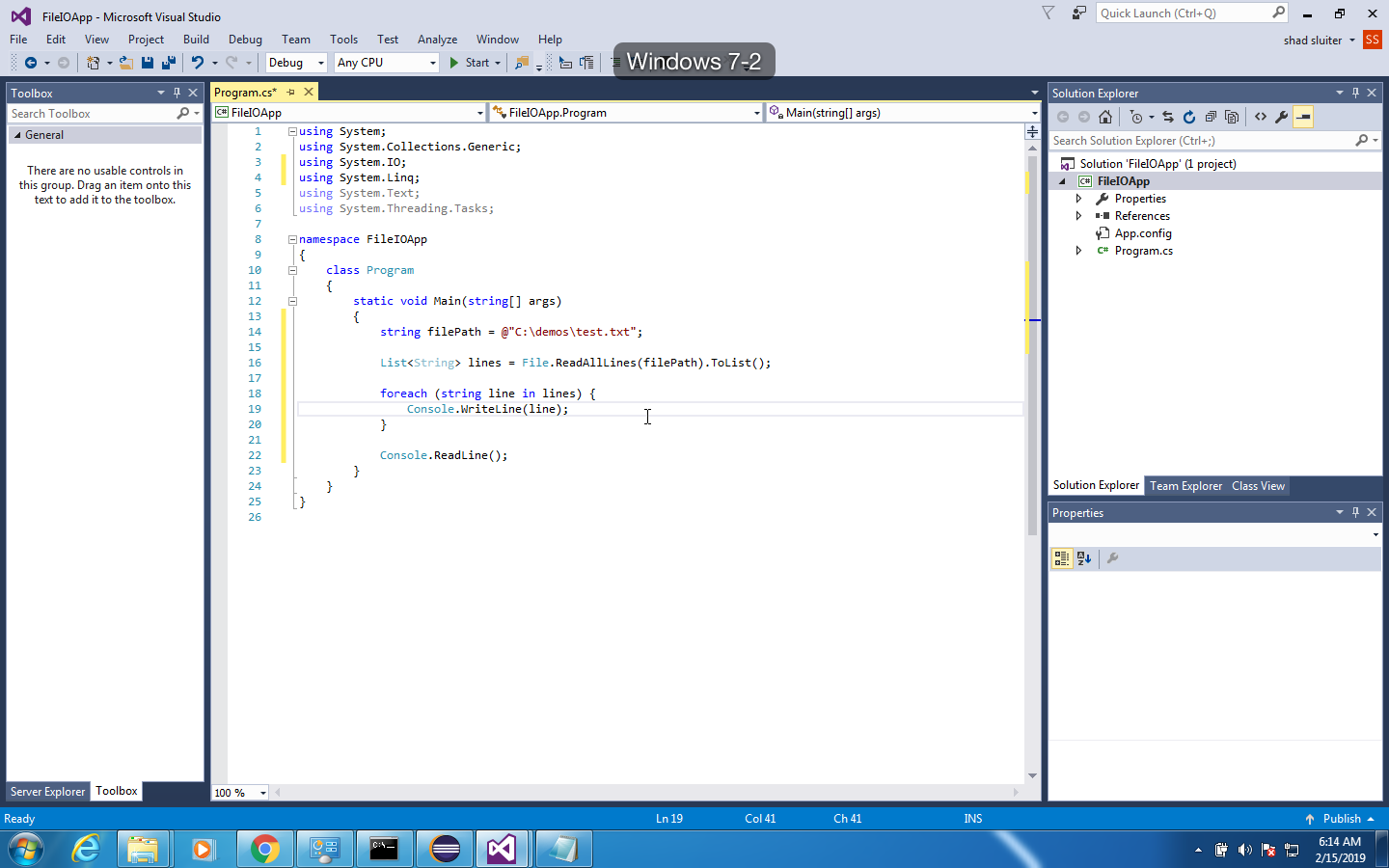
1. But a List has more options than an array. So, we will use a list instead.



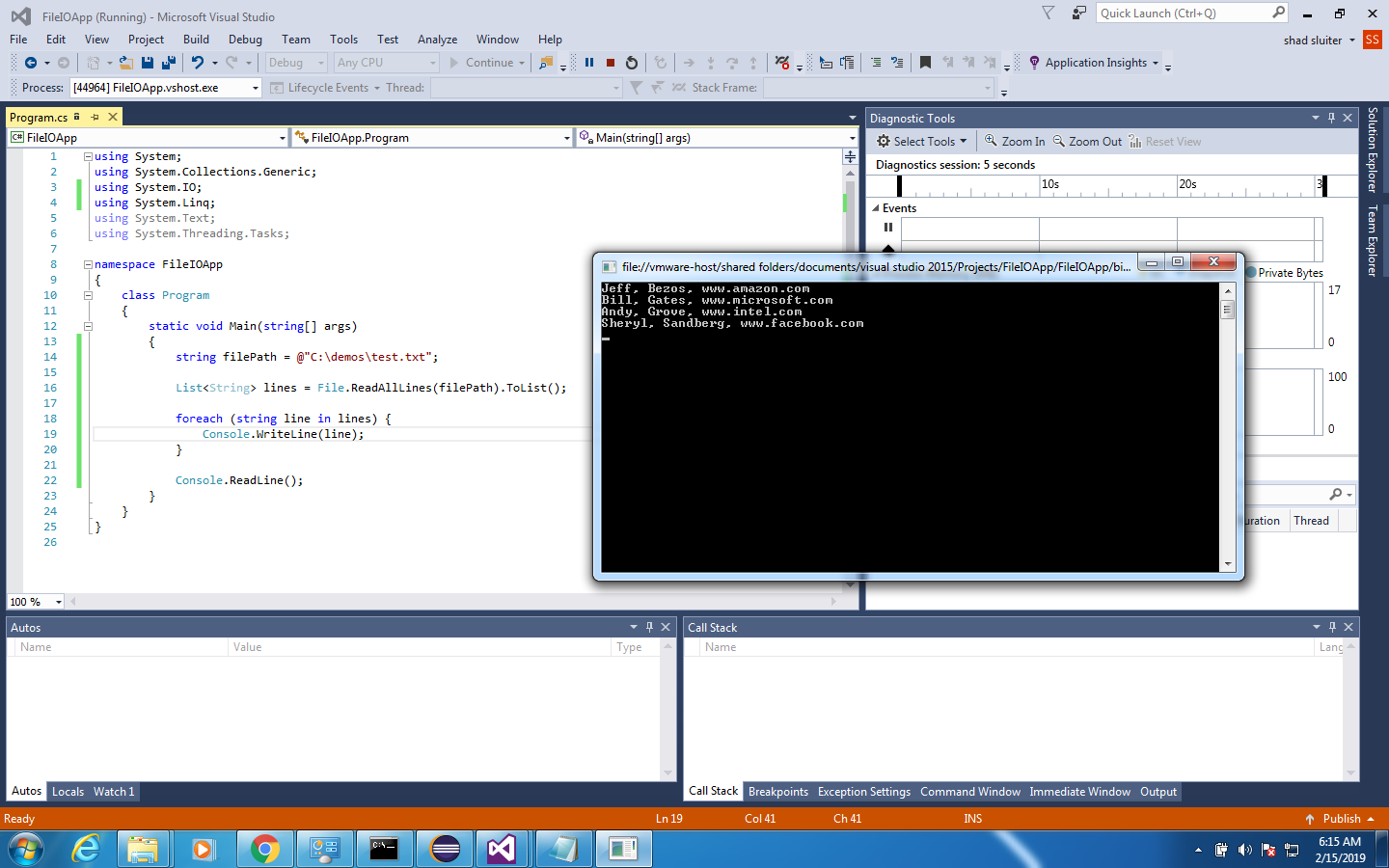
1. To use a list, we must convert the return type to a list



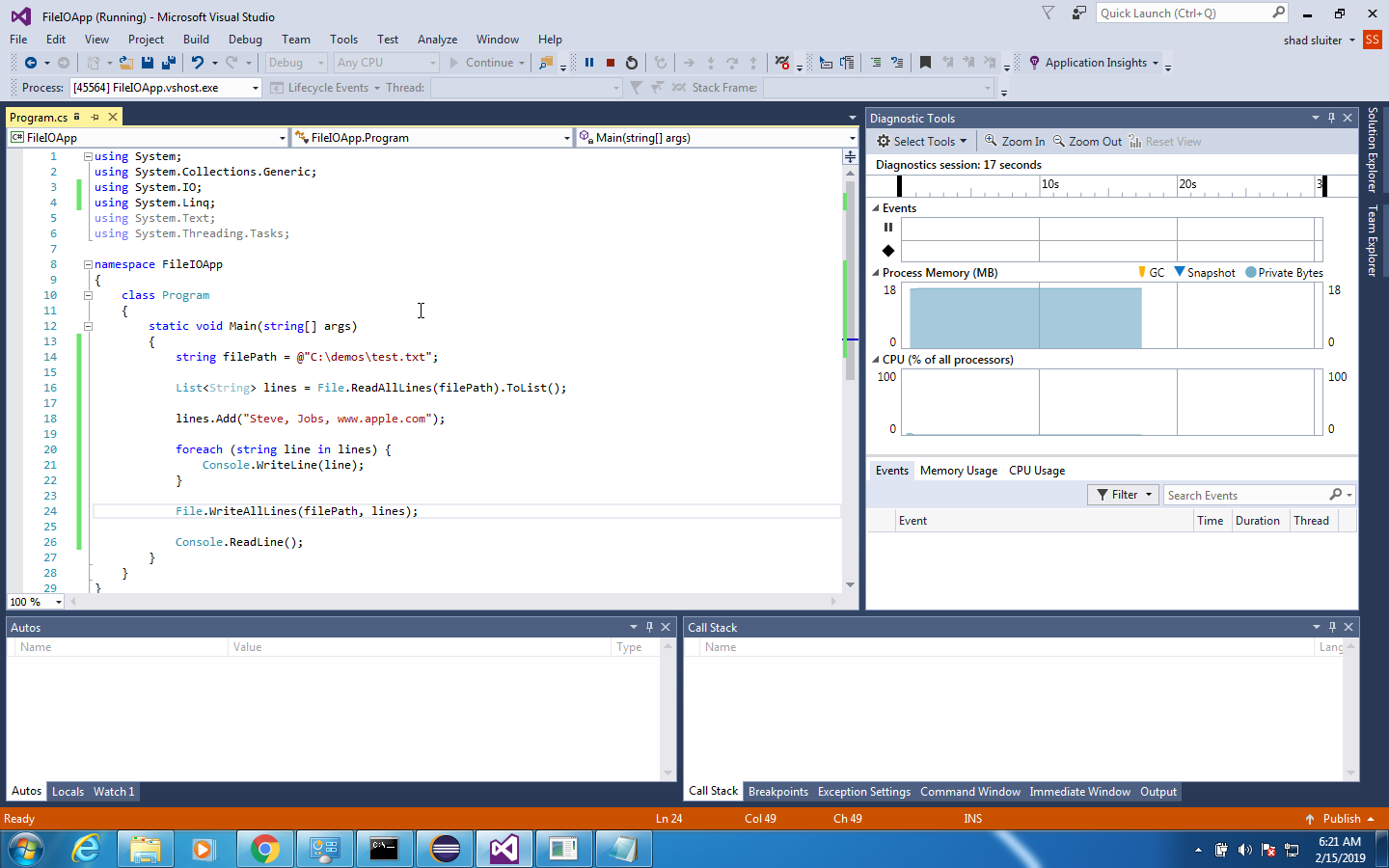
1. Read through each line and print the results to the console.



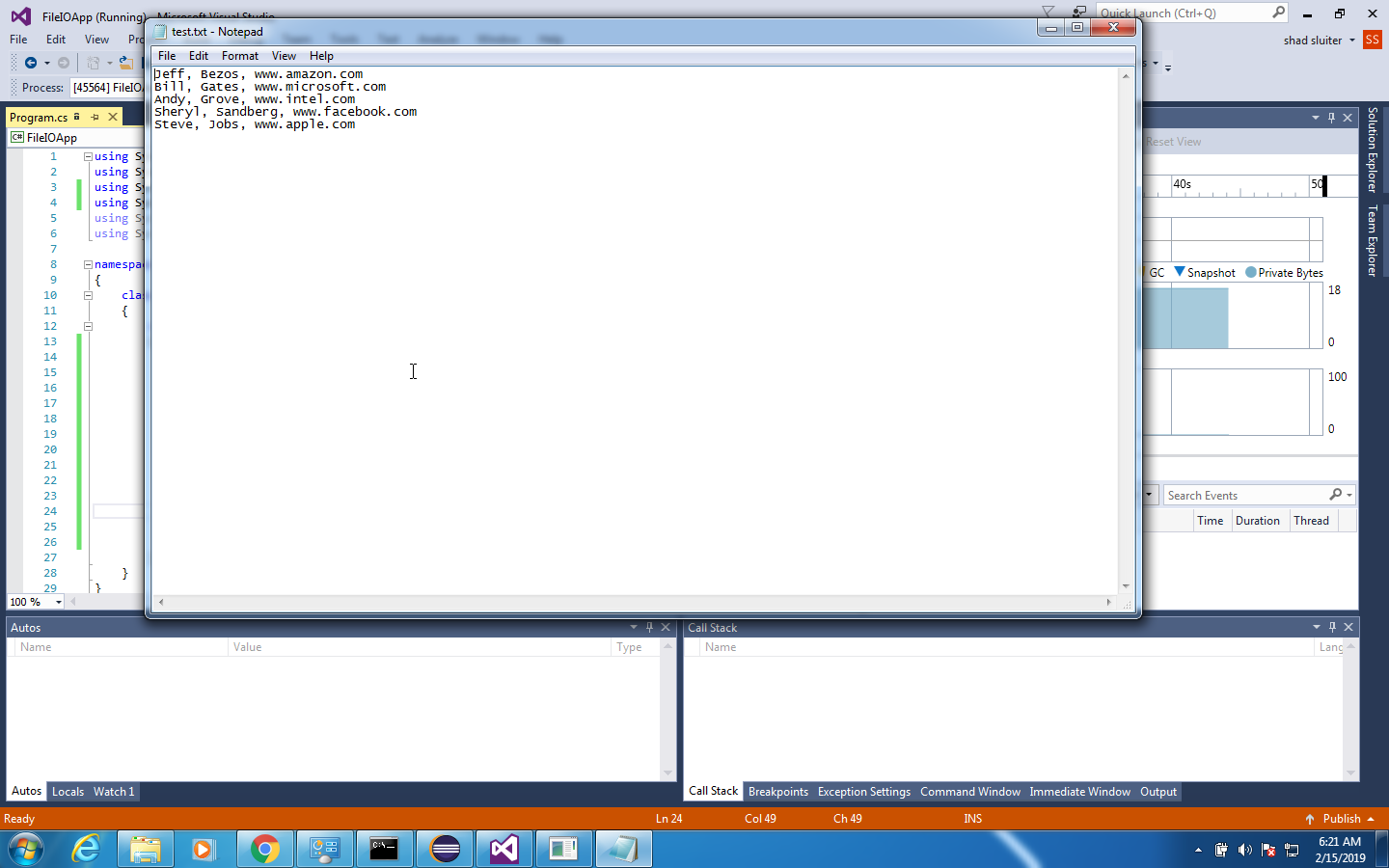
Run the program…



1. Let's add a new person and re-write the data to the file.

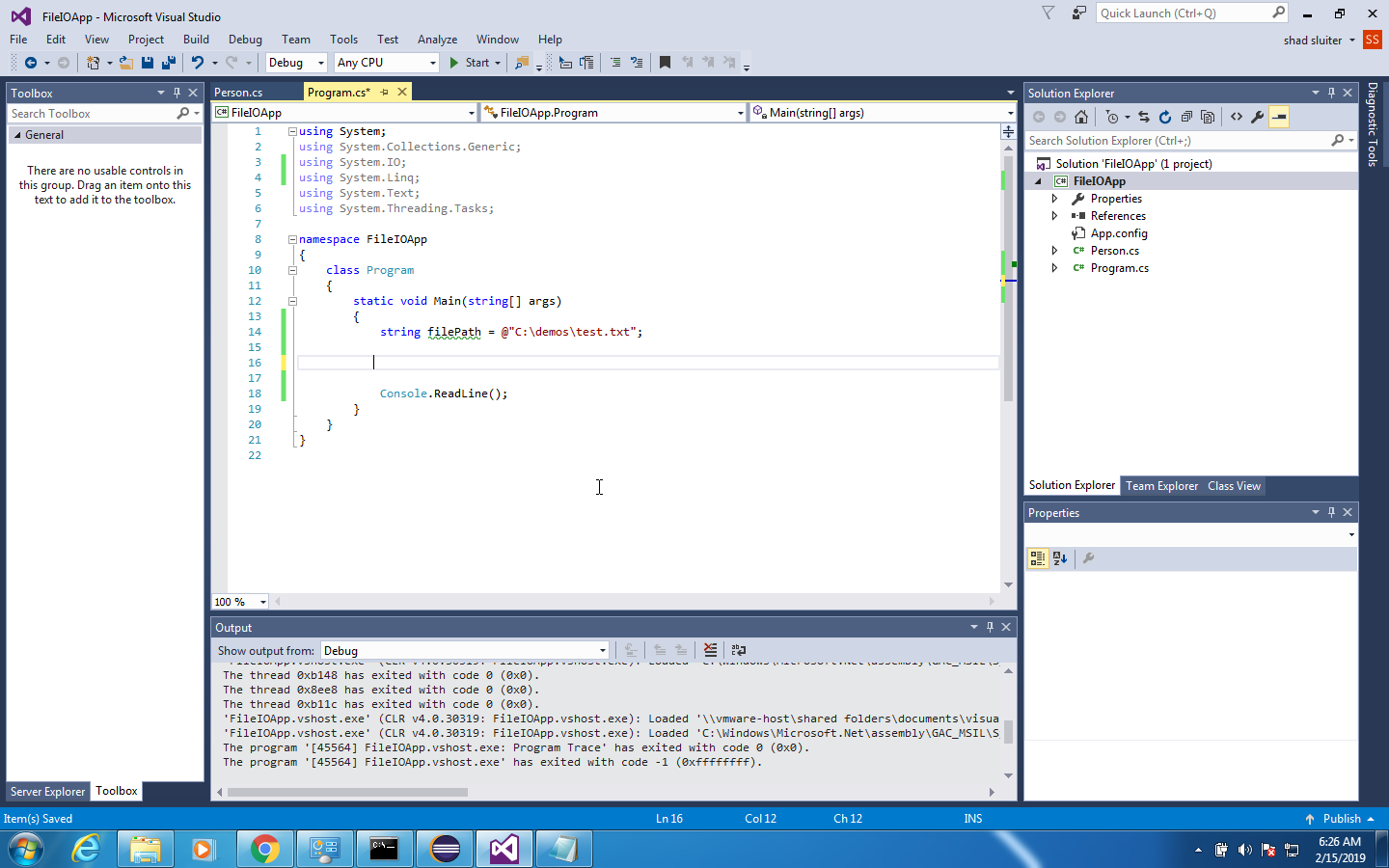


1. Now the file should show the new data.

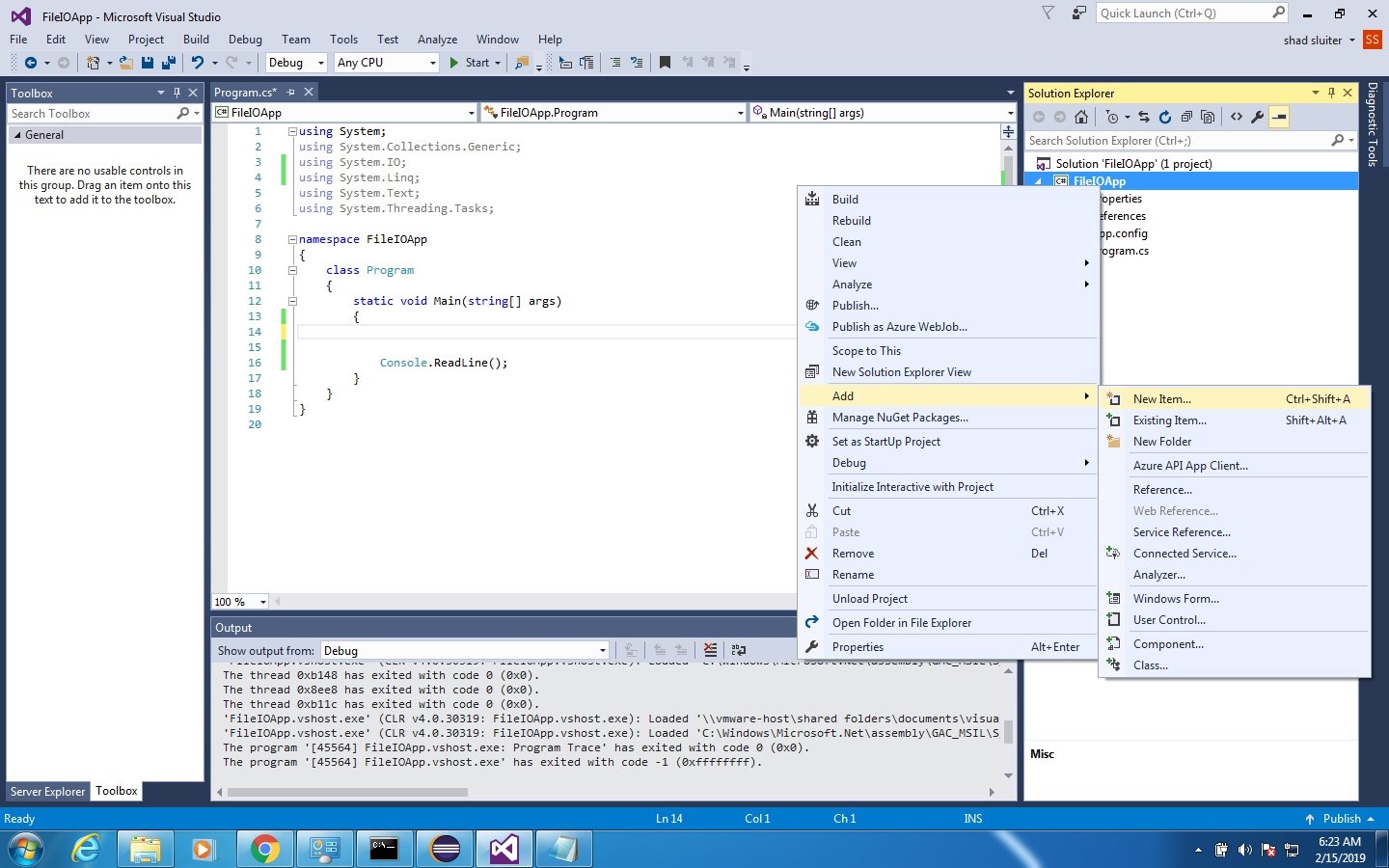


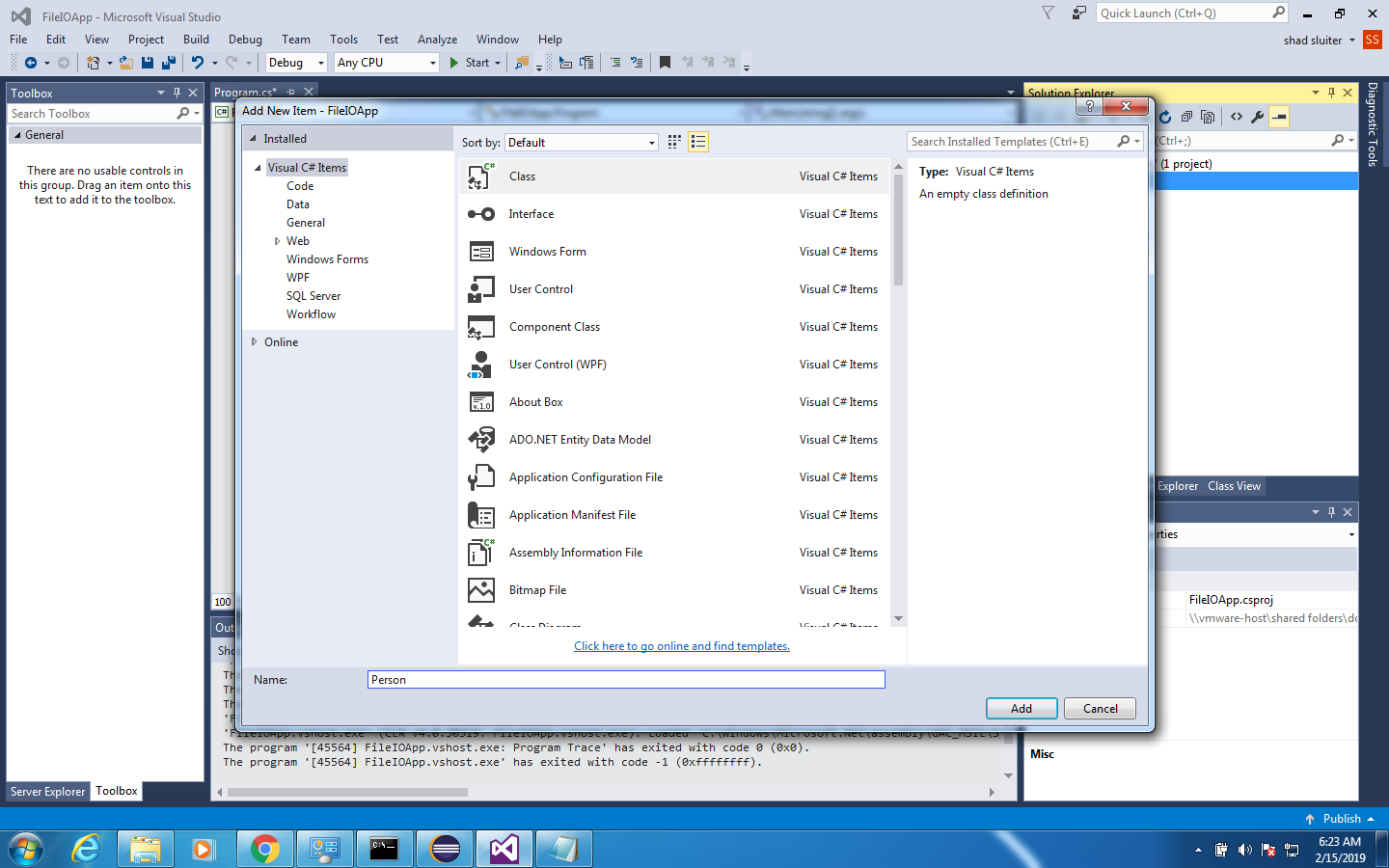
## Reading and Writing Data to a Class

1. Delete or comment the code we just made. Keep the filePath variable.

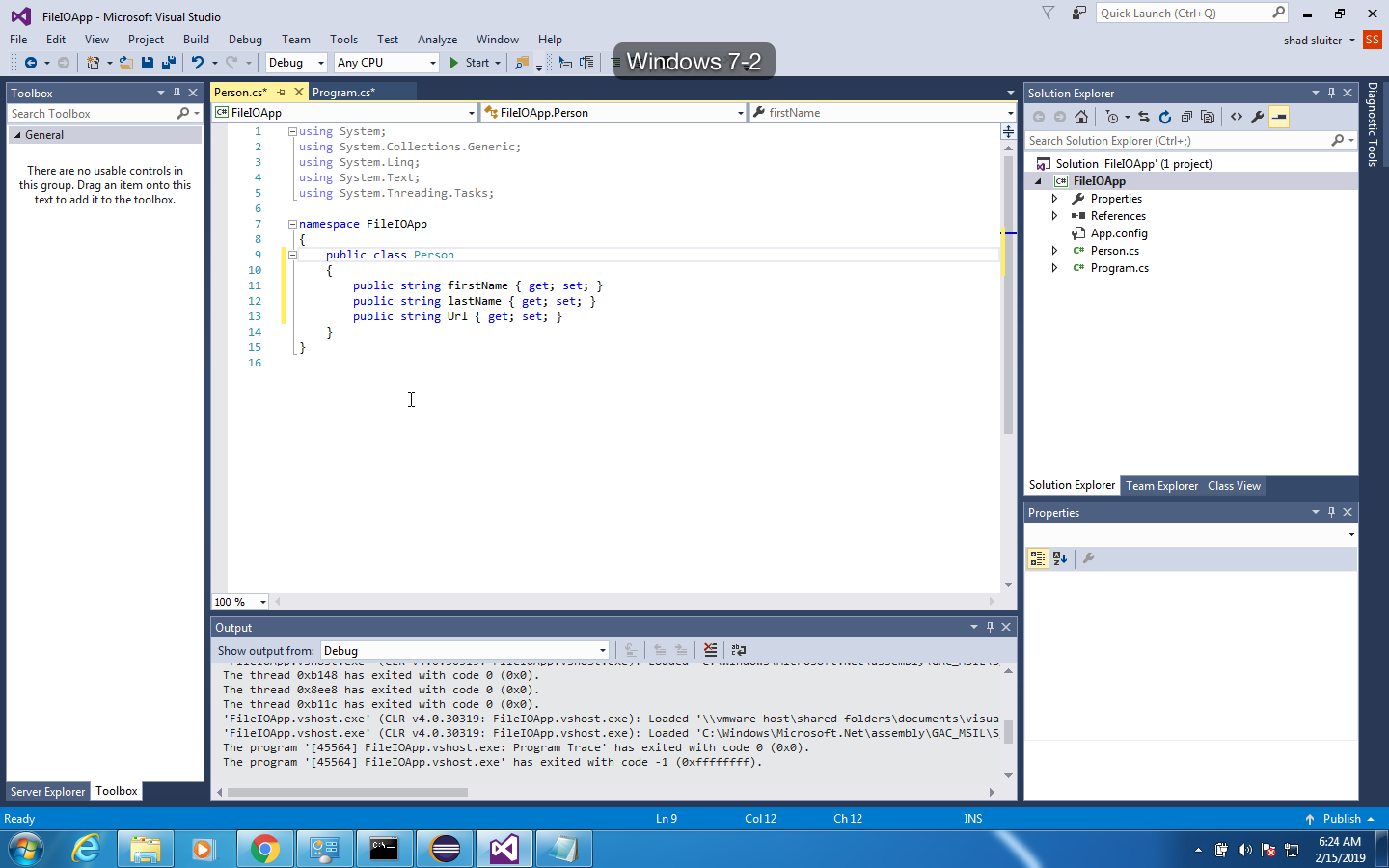


1. Add a new class to the project. Right click on the solution and choose Add > Class.

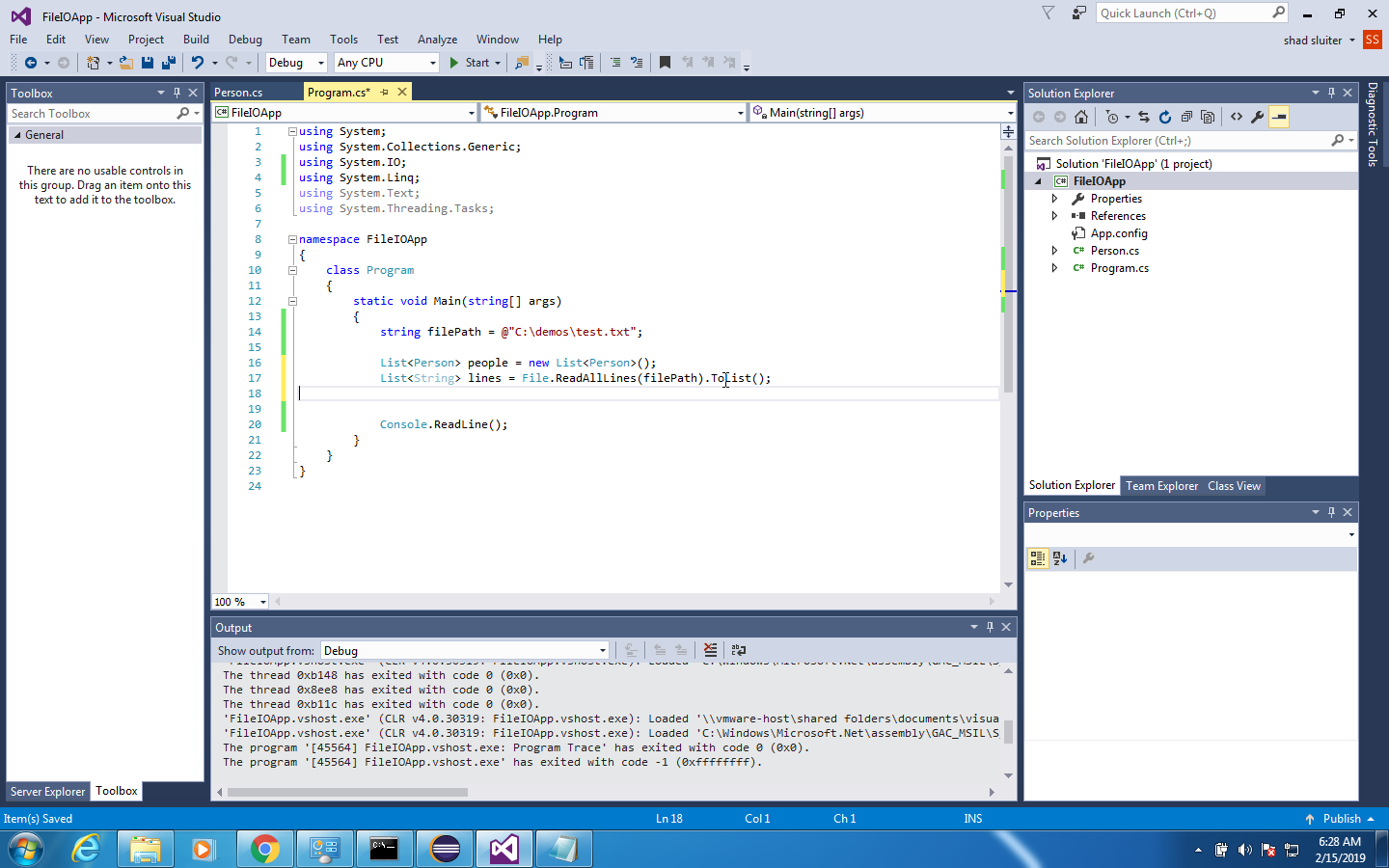




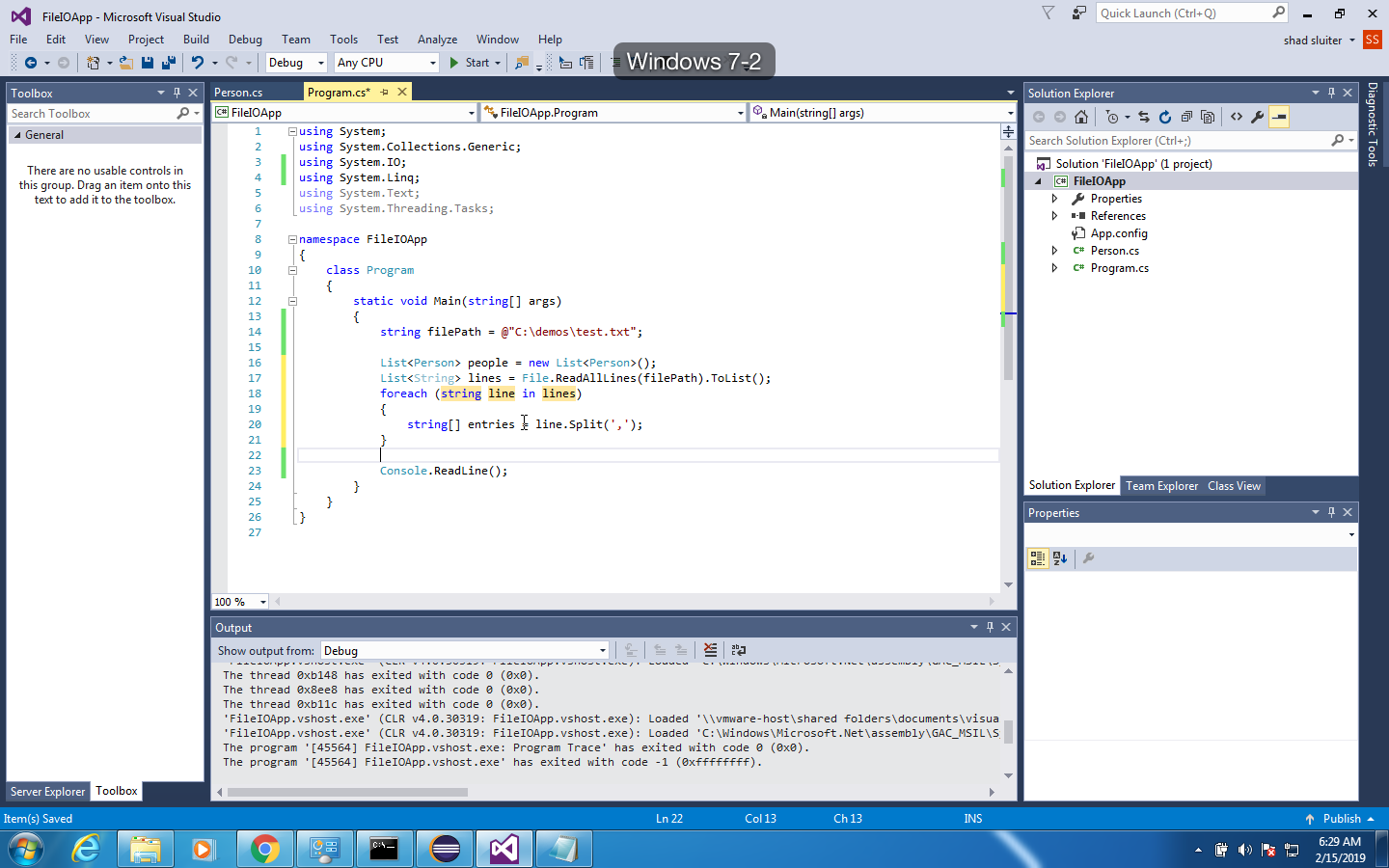
1. Create a Person class with three properties. Set the class to public visibility.



1. Create a new list of type person. Read all lines from the text file and store them in a list.



1. For each line read, split the text by using the comma as a delimiter. Store the results in an array.



1. Now create a new person object from the array items.

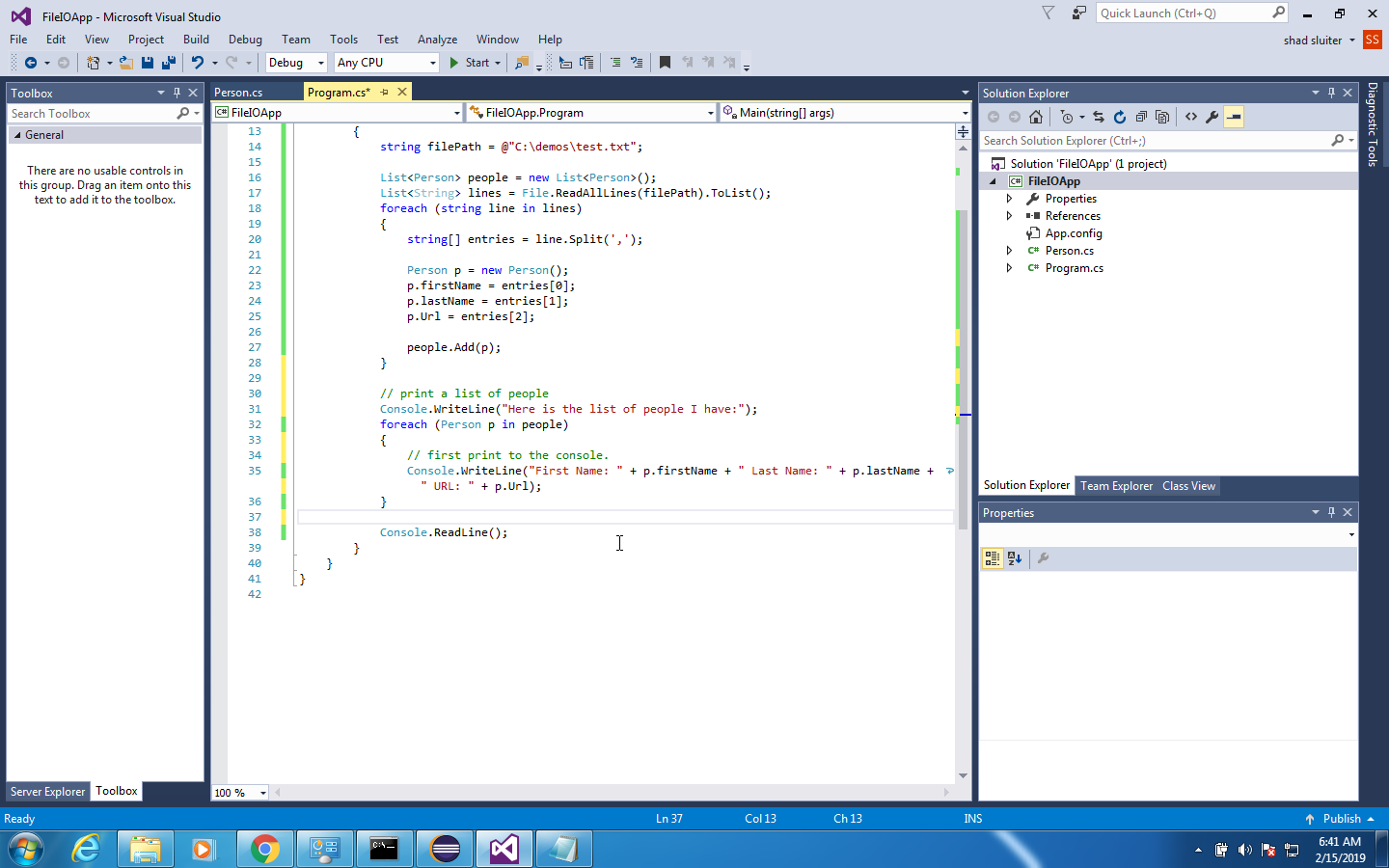


## 

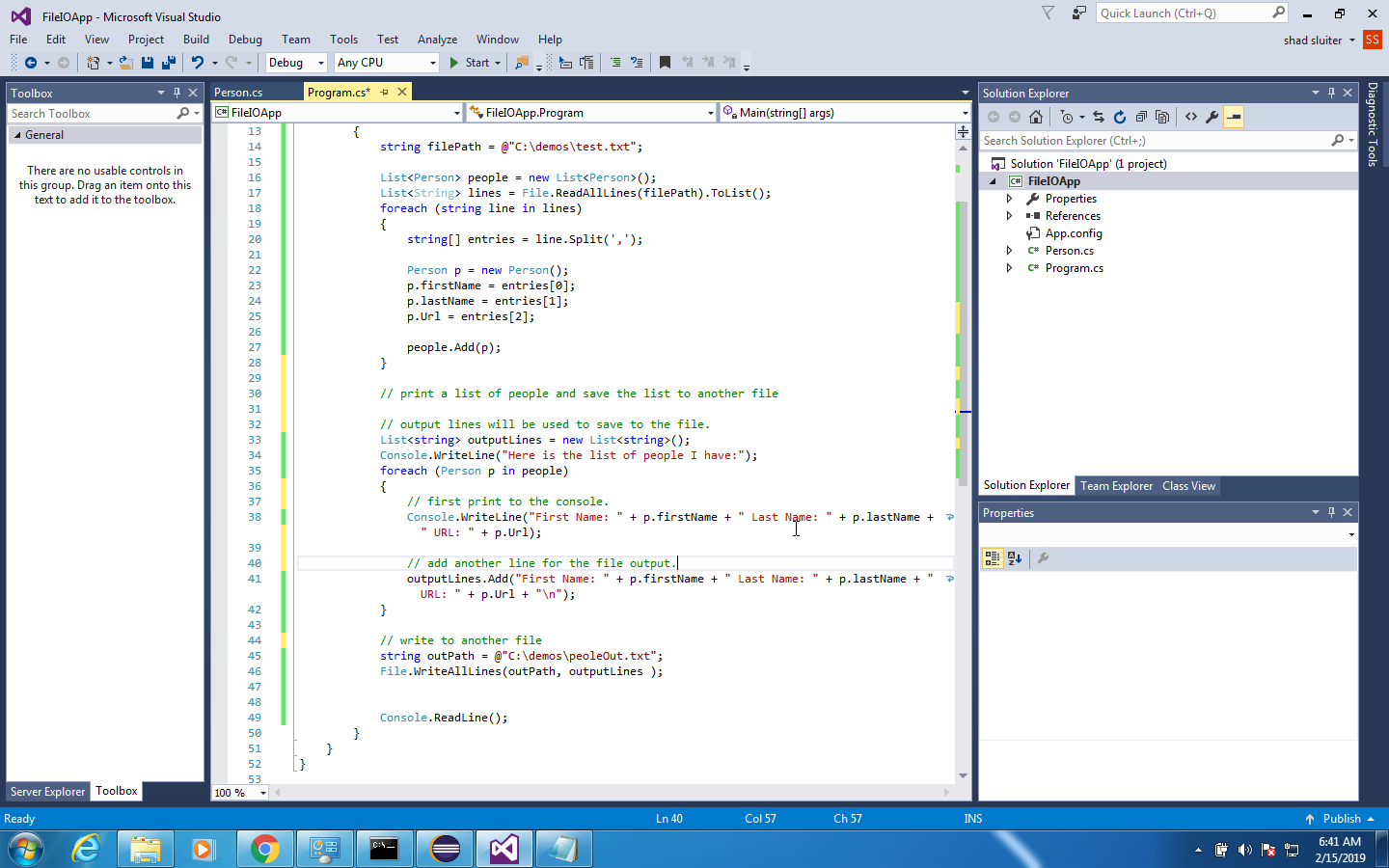
## Error Checking

This code will work only if the text data is perfectly formatted. I will leave a challenge to you to check for errors to avoid program faults. (1) Does the file read complete correctly? (2) Is the data formatted correctly? Each line must have three items, or the program will crash.

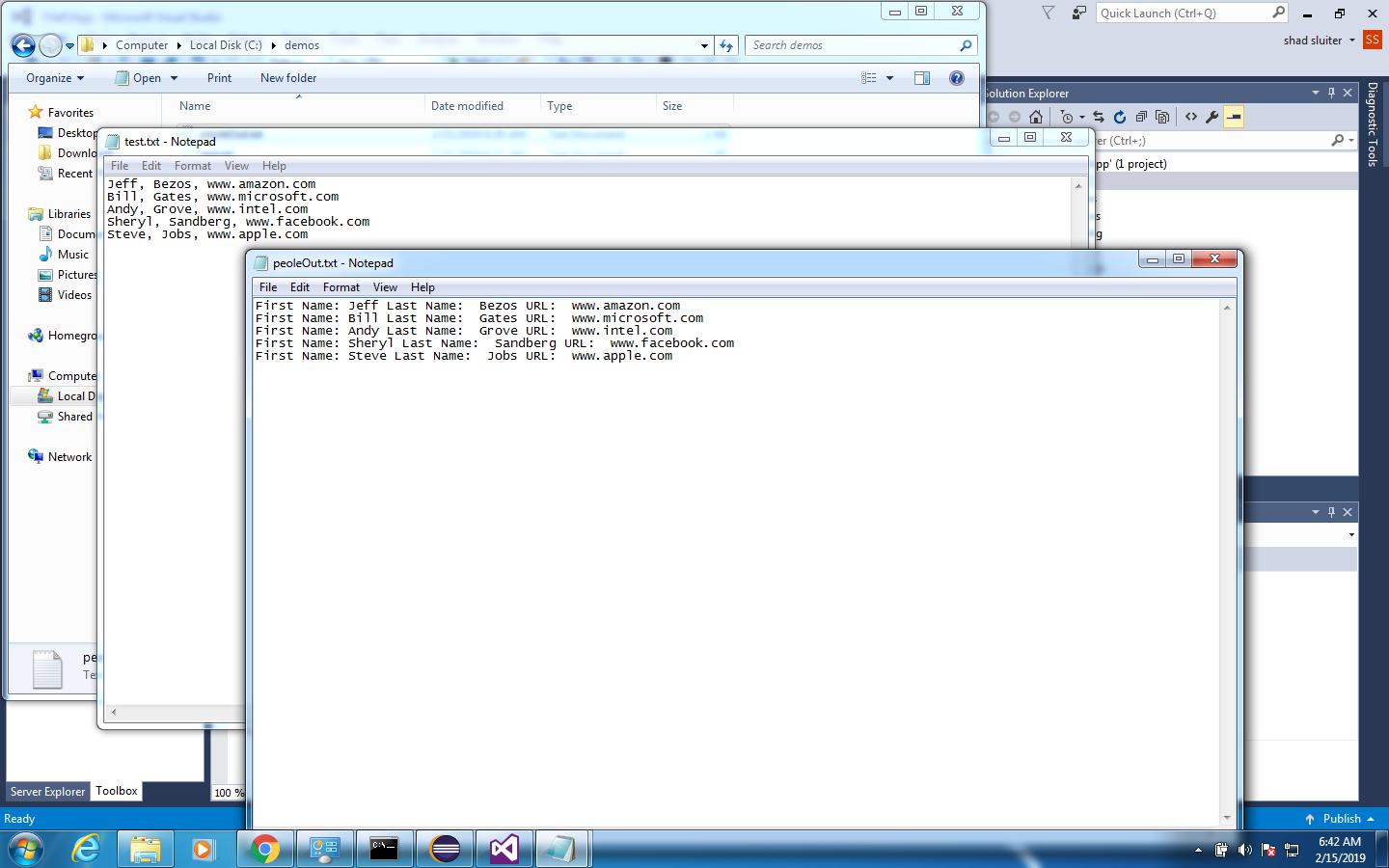
1. Print the results to the console. Show each person on one line as follows.



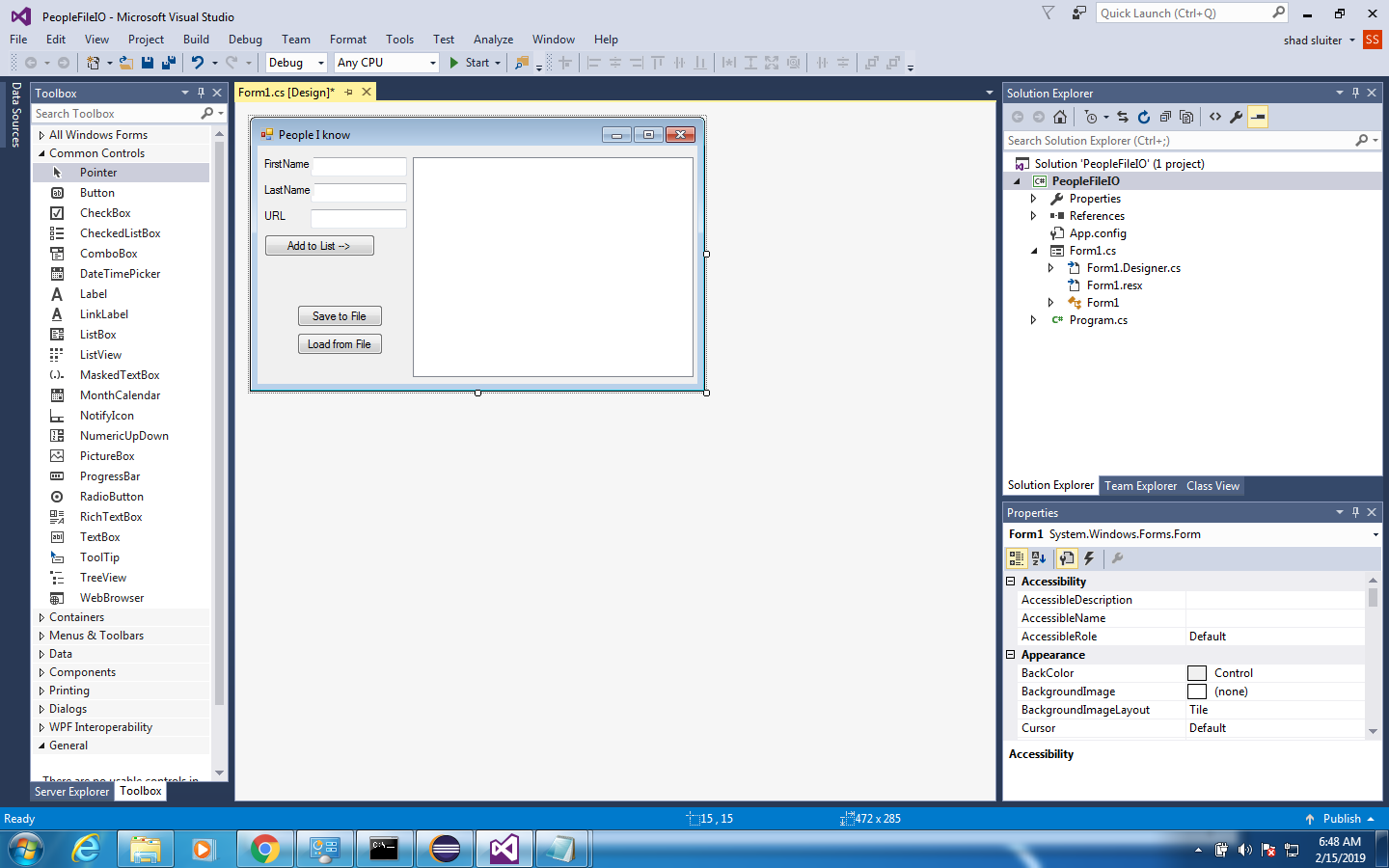
1. Add a new person, add it to the list of Persons and write the results back to another file.



1. You should see the new person is in the text file. I included the extra headings "First Name: LastName: and URL:" just to emphasize the fact that the two files are distinct. Depending on your needs, adding extra labels may make the text file more difficult to parse by other programs.



**Challenges**

1. Fix the potential errors. Instead of crashing, check to see that each line of the text file has the correct number of columns. If a line does not have 3 items (2 commas), then print a message.
2. Adapt the program to a GUI.
   1. Create an input form and submit button.
   2. The submit button should create a new Person and add it to a list in memory.
   3. Display the list in a list control. Recall from the first activity from this semester how to display a list in a list control. We stored Car objects in the list box.
   4. Create a Save button that will write the results to a text file.
   5. Create a Load button that will read the text file and parse the data.

**Deliverables**

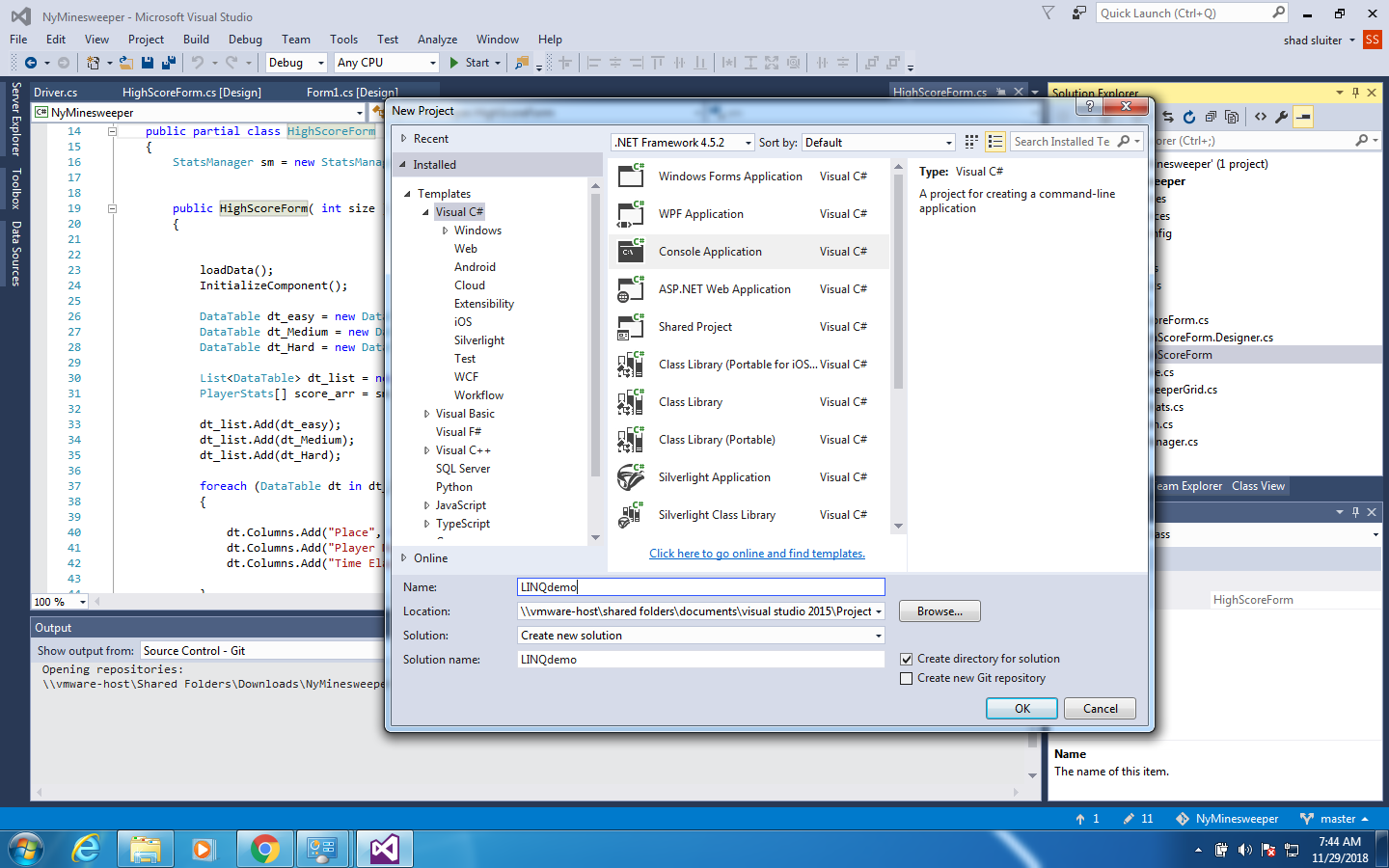
1. Create a Word document showing the app in action, demonstrating the features in the tutorial, error checking and the GUI version of the program.
2. ZIP file of the contents of the project folder and the text file you created.

# Part 2 - LINQ Demo

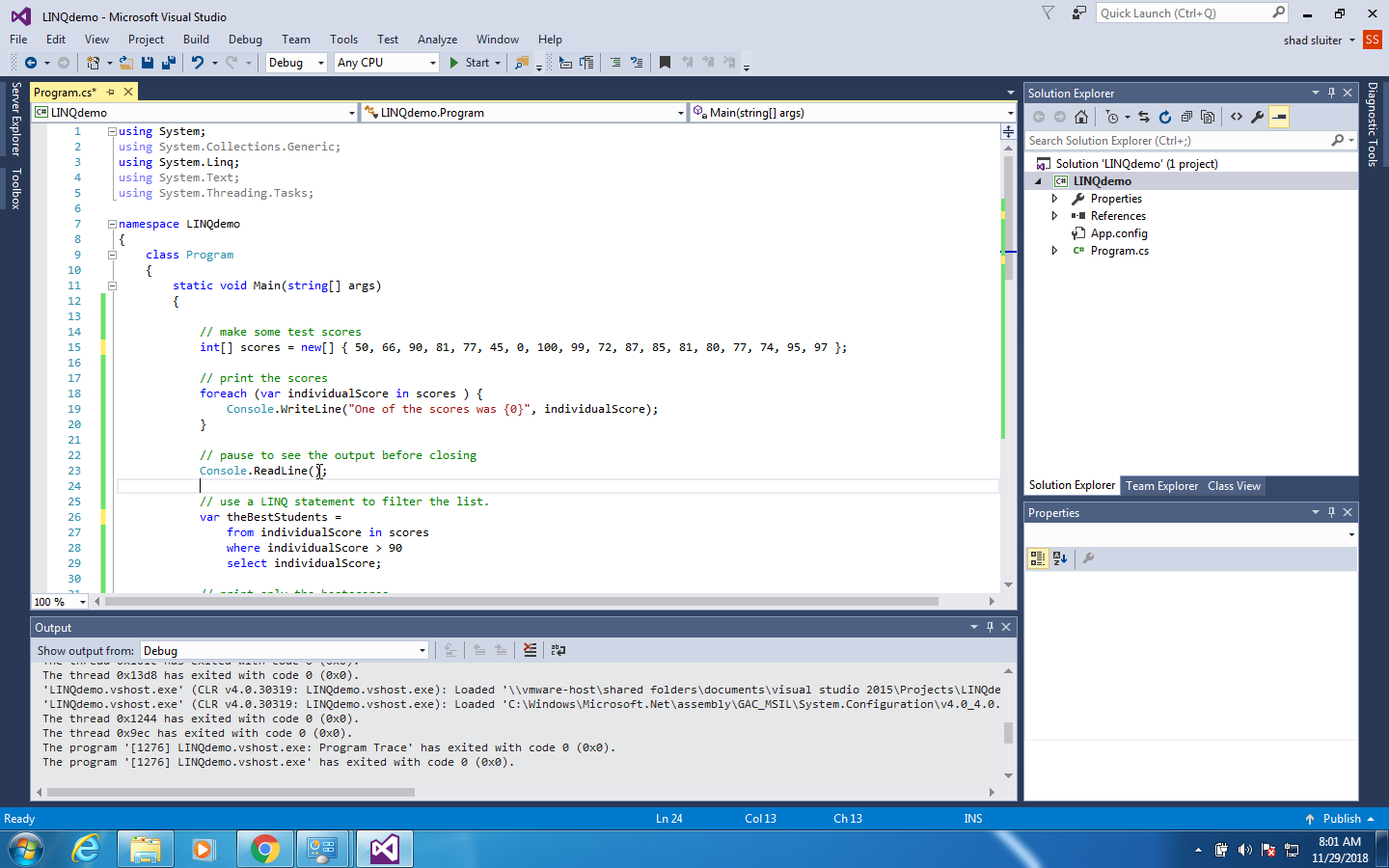
**Objective:** Demonstrate the use of LINQ in sorting and filtering lists of data. The example shown here is for a simple array. You can adapt LINQ to be used with arrayLists of objects if the object has a compareTo method implemented.

Before starting this code, I recommend reading about LINQ. Here is a good reference <http://www.tutorialsteacher.com/linq/linq-tutorials>

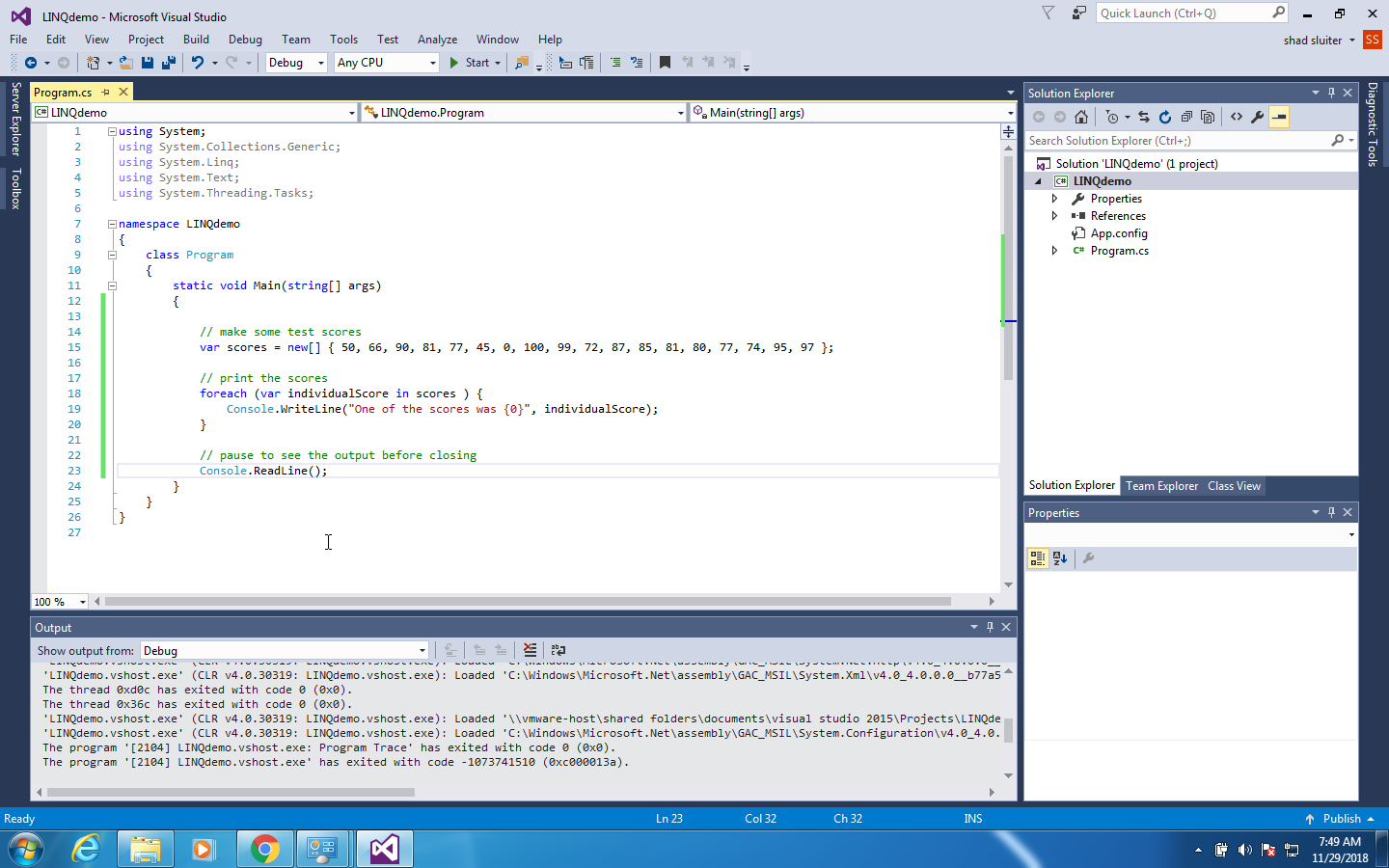
1. Start a new project.



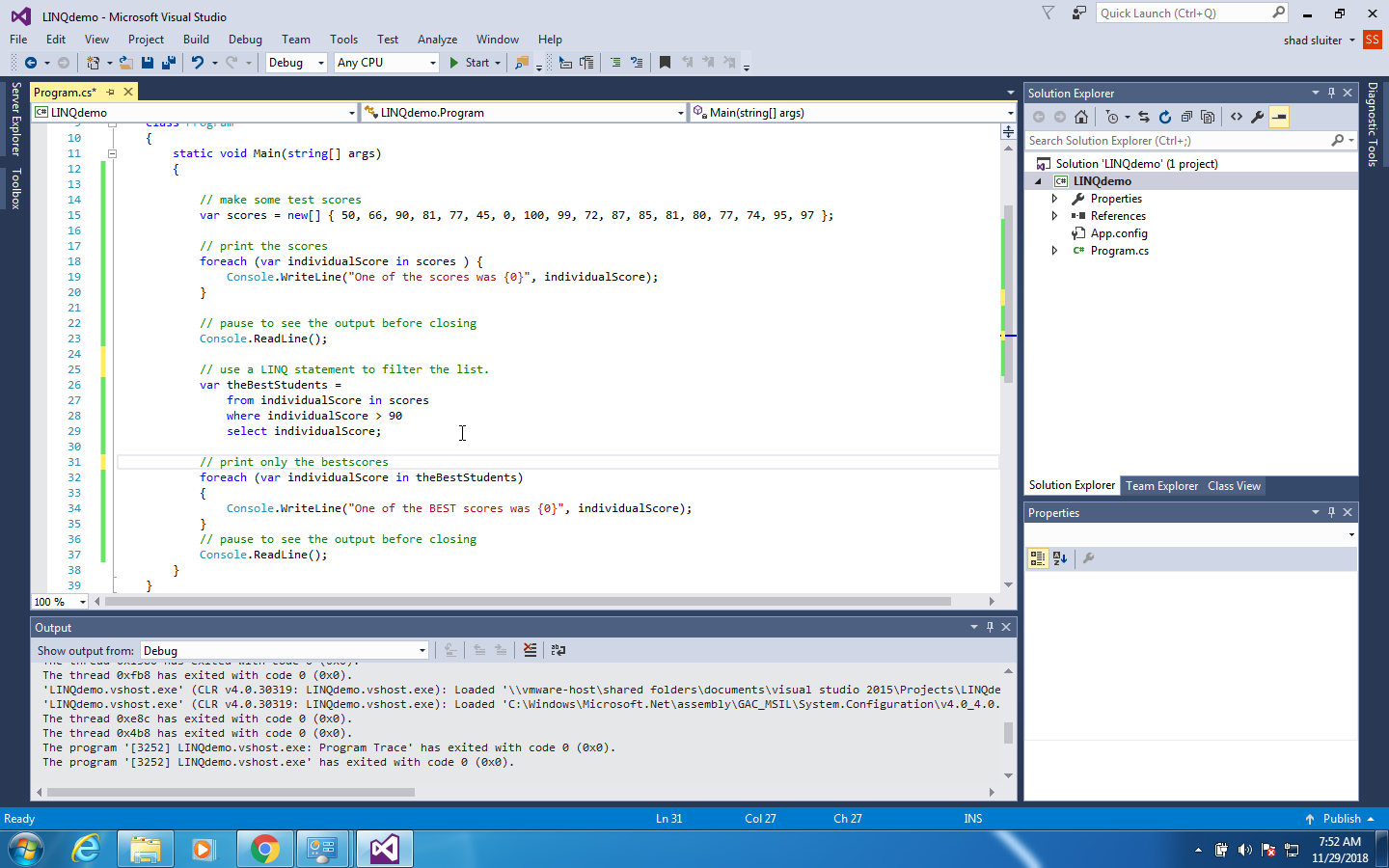
1. Add an array of test scores.



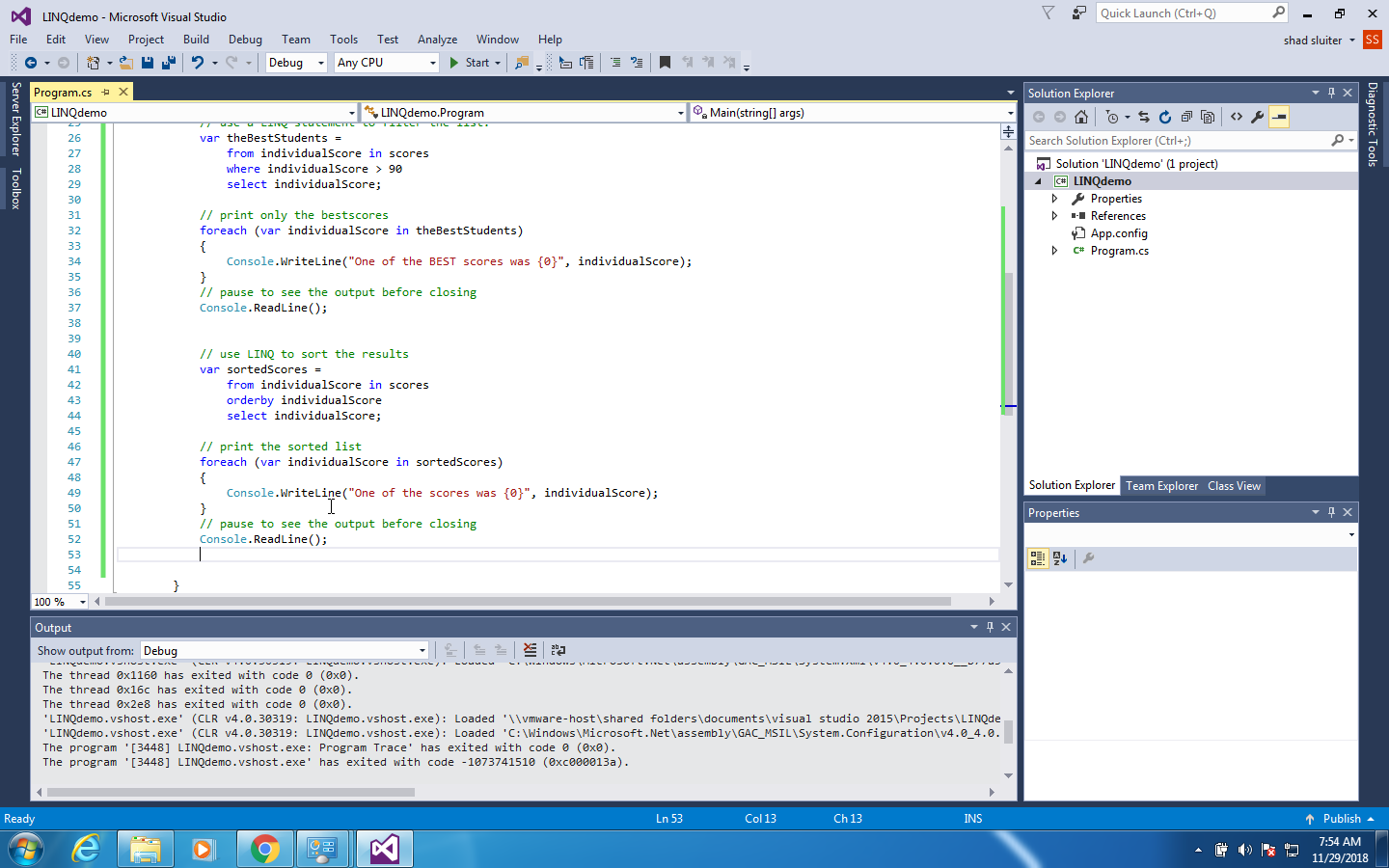
1. Print the scores with a foreach loop.



1. Filter out only the A students and print them to the screen.



1. Now create a list of sorted scores.



**Challenge**:

1. Print a list of only the B students (80% to 89%) in ascending order.
2. Create an arraylist of students and demonstrate the use of LINQ on the new list. Each student object should have at least three properties: name, age and grade. Add the iComparable implementation to the Student class and create a compareTo() method in the student class in order to be able to sort the list.

For help on iComparable and compareTo see the following link <https://www.dotnetperls.com/icomparable>

**Deliverables**:

1. Zip file containing all source code.
2. Word document containing screenshots of the application being run. Be sure to demonstrate all features that were created in the tutorial as well as the challenges.