Application Development with.NET and Visual Studio

NIKOLA PETROVSKI

17 FEB 2018

1.0 Application for managing Contacts of Objectville Paper Company

The purpose of this report is to explain the process of application development with the following software products: .NET Platform, C# and the Visual Studio IDE. It describes two video tutorials for creating two separate C# applications and a debugging process for an exercise application developed in class.

This part of the assignment demonstrates the development of a step-by-step tutorial from Chapter 1 of the book Head First C# Second Edition to create a fully functional, database-driven application for managing the contact / customer directory of the Objectville Paper Company. It shows the development process, design and conventions used for each application.

Section A:

This section is about creating Windows Forms Project and designing Windows Form. Naming rules are applicable to all program components. For example, this project has a meaningful name: ObjectVilleContactDirectory. A Solution Explorer window shows an automatically generated structure of nodes for Properties, Resources, References, Program Class (main entry point for the program) and a Form representing the first window of the application. In C#, colon (:) symbol is used to represent inheritance relation between two classes.

Changing properties is very convenient in C# by selecting a visual item on a form (e.g., form title, label, button) and then typing the new value beside the respective property name. These changes in field variable names are automatically reflected in the respective code. A Select Resource window allows the user to import files to the project by using the smart tag. Adding simple functionality like showing the name of the application and the author after clicking on a logo image is achievable by using the Events button of the Property window. CamelCase naming rules apply for Click event handler for example OnShowAboutBox. It is important to link controls and methods correctly in an event relation.

Section B:

This part shows how to design and create a database using the new Local DB database engine and Service-Based Databases. The Add New Item window allows selection of Service-based Database Visual C# Item for Local database engine. Data Connections can be open by double-clicking on the database file in the Server Explorer window. After creating, inserting records in the Contacts table and opening table definitions it retrieves the updated definition from the server.

Section C:

This part illustrates creation of data-tier of the application using Data Sets and the Visual Studio Designer. In order to associate data source to a project it is required to specify where the application will get the data from, the type of database and the connection information in the Data Source Configuration Wizard. The connection string is saved and verified in the application configuration file and the required database object is selected. Connection should be tested in the Connection Properties window.

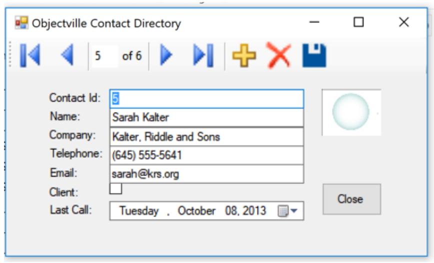
Section D:

This video shows how to implement a required functionality in an application. The grouped items are added to the main form by simple drag and drop of the Contacts from the Data Sources window. All items (visual and non-visual) should be renamed appropriately. For example, add the lbl prefix to the labels, adding _txt prefix for text fields and _contactDataSet for non-visual item. The close button should be added in the form to close the application by using a close() method.

Good documentation is essential to any project. The scope of the assignment includes an installation of Visual Studio plug-called Sandcastle but an official version is currently not available on the Internet. The Sandcastle desktop application is available through GitHub only.

Section E:

The last video in this tutorial shows how to create and test the installer for this application. Technical Documentation window describes the contents (forms, constructors, methods, fields) for this application. A Publish Wizard window sets the location, installation (setup.exe file), and update details for the application.



Contact Id: 5
Name: Sarah Kalter
Company: Kalter, Riddle and Sons
Telephone: (645) 555-5641
Email: sarah@krs.org
Client: ☐
Last Call: Tuesday , October 08, 2013

Figure 1: Final Screenshot: Objectville Paper Company Project.

2.0 Application for Consignment Shop Demo

Part two of the assignment describes the thought process of developing a Windows Forms Application in C# from determining the requirements to creating a fully functional application for running a Consignment Shop named Seconds are Better. It shows the process of analyzing objects, member's data type, relations between objects, and designing a layout of classes (Vendor, Item, Store) for this application.

Front-end design can be written on paper because there may be few updates before a right design is achieved. For this example, it is done in Visual Studio as a new project named: ConsignmentShop UI under a Solution named: ConsignmentShop. These are the steps for making the interface that application's users interact with directly:

- Main Form including all references to it must be renamed to an appropriate and meaningful name;
- Aesthetic principles must be applied to position, size, color and other visual elements;

TIP: Stack and resize the buttons using a mouse as they should be of a same height.

- Two instances cannot have same name therefore attention must be paid in naming them;
- The natural flow of use of the application should be set in one direction (e.g., left to right or Up to down) as much as possible.

Back-end contains the class library of all information analyzed during the requirements determination process. For this example, it is done in Visual Studio as a new project of a type Class Library named: ConsignmentShopLibrary under the same Solution called: ConsignmentShop. These are the steps for making computer code that application developers interact with directly:

- Delete Class1.cs because it is automatically generated but is not applicable;
- In Add New Item – ConsignmentShopLibrary window add the three classes (Vendor, Item and Store);
- Class development rules must be followed in terms of: class access modifiers, auto or full properties;

TIP: Use the shortcuts for inserting snippers for auto property (e.g., “prop” +Tab+Tab).

TIP: Question the design to find out if any members are extra or missing.

- Attention must be paid to variable types particularly to the reference type variables;
- Wire the front-end to back-end according to the business rules set in the requirements;

TIP: Edit and refresh in a small progressive step because it helps during troubleshooting.

- Use a Reference Manage ConsignmentShopUI window and add a reference from the Solutions section that contains all available Projects to be able to reuse existing reference type variable in different places within a Solution;

NOTE: The UI needs to know about the library but the opposite direction does not work.

TIP: Use the full syntax for accessing a specific reference type from the library (e.g., ConsignmentShopLibrary.Store) or second option is to add a ”using ConsignmentShopLibrary” statement.

- Use a method called SetupData() to create or fill up data (Vendors and Items) required for further processing;
- A constructor named Vendor() sets a default commission value to 50% as required;

NOTE: InitializeComponent() method must be the first statement in a constructor.

- To keep the code DRY (Don’t Repeat Yourself), a List type of variable should be instantiated in a proper place by using a constructor (e.g., the class Store’s constructor instantiates Lists for both Vendors and Items);

TIP: Format indentation of computer code by clicking Edit -> Advanced -> Format Document.

- Associate each item to a vendor that represents the owner as well;
- A BindingSource is a type of connection between the store object and the visual box labeled “Store Items” that must be placed in a Form but NOT nested in a method inside the Form;
- All the objects property values (DataSource, DisplayMember, ValueMember) for “itemsBinding”, “itemsListBox”, “cartBinding” and shoppingCartListBox should be placed in the constructor “ConsignmentShop()” to be able to wire up a list box to its’ data source;
- “Add To Cart” button should be wired by adding an Event Property of type Click;

TIP: First select a visual item with incorrect event on the front-end, then delete the respective Event Property value to clear that event and finally delete the computer code for that Event (if it is still there) to be able to clear an event right first time without altering auto generated code.

- Add verbose comments inside an event method if the event needs complex explanation;

TIP: Use the IntelliSense feature in Visual Studio to add calls to properties and methods with only a few keystrokes. For example, SelectedItem property is an appropriate selection to select and item for the itemsListBox to store its' value in selectedItem object.

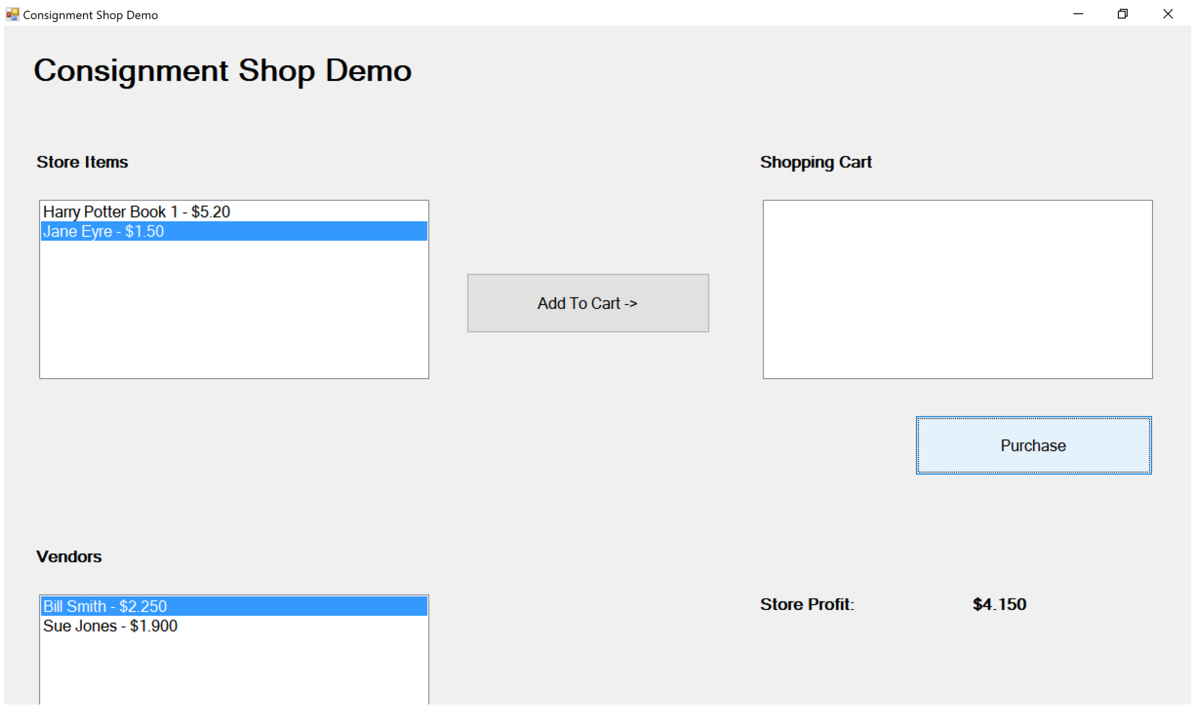
- Debugging tool in C# provides verification of the state of variables in specific code places.
- A ResetBindings property is required to every time an item changes (e.g., shopping cart).
- Purchase button has an event code that has a loop (e.g., calculates payment due), Clear() method, two ResetBindings (required for items update) and storeProfit calculations.
- The list of store items can be updated by using SQL clause to specify a result set only;

NOTE: The place of the SQL (lambda) expression is critical to functionality of the application.

- Vendors visual box can be included in the main form to meet the application requirements. A duplicate of Store Items is a starting point for creating the Vendors box.

NOTE: Earlier naming rules for all existing Vendor properties and new property called PaymentDue are applicable to this object as well.

- Final requirement is to determine the amount payed to the store. It is accomplished by adding a "Store Profit" label. The label should be linked to a variable of type decimal.
- After the essentials of the application are done it should be tested and then shipped.



Harry Potter Book 1 - $5.20
Jane Eyre - $1.50

Bill Smith - $2.250
Sue Jones - $1.900

Store Profit: $4.150

Figure 2: Final Screenshot: Seconds are Better Company Project.

3.0 Professional Development Process using Visual Studio

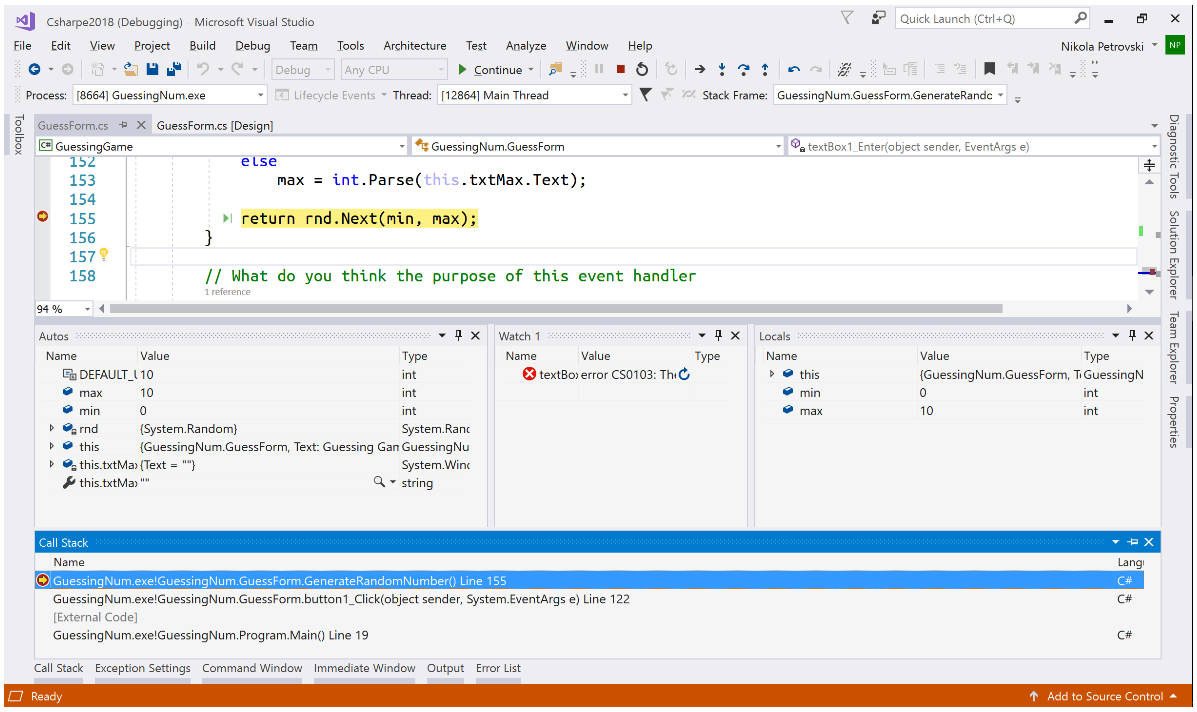
This section verifies assumptions about certain computer code effects using the debugger tool in Visual Studio.NET 2017 and the GuessingGame model of an exercise developed in class for this purpose. The debugging consists of these steps:

- placing a break in code;
- starting the debugging line-by-line;
- watching the values of variables in the code
- driving the debugger (not letting the debugger take its's way into a loop inadvertently).

describe the assumption being verified, **how** it was verified and **what** was the outcome of the verification through the debugger:

Section A:

The function of method "GenerateRandomNumber()" is creation of the random number. The number is required for processing of the application. Debugging step-by-step until the line of code with the break that creates the random number generator provides a method to track the changes in variables. This assumption is verified by using the debugger.


Csharpe2018 (Debugging) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Nikola Petrovski
Debug Any CPU Continue
Process: [8664] GuessingNum.exe
Lifecycle Events Thread: [12864] Main Thread
Stack Frame: GuessingNum.GuessForm.GenerateRandc
GuessForm.cs GuessForm.cs [Design]
GuessingGame GuessingNum.GuessForm textBox1_Enter(object sender, EventArgs e)
94 %
Toolbox
Diagnostic Tools Solution Explorer Team Explorer Properties
Call Stack Exception Settings Command Window Immediate Window Output Error List
Ready
Add to Source Control

```
            else
                max = int.Parse(this.txtMax.Text);

            return rnd.Next(min, max);
        }

        // What do you think the purpose of this event handler
        1 reference
```

Autos

Name	Value	Type
DEFAULT_(	10	int
max	10	int
min	0	int
rnd	{System.Random}	System.Ranc
this	{GuessingNum.GuessForm, Text: Guessing Gan	GuessingNu
this.txtMa›	{Text = ""}	System.Winc
this.txtMa›	""	string

Watch 1

Name	Value	Type
textBo›	error CS0103: Th	

Locals

Name	Value	Type
this	{GuessingNum.GuessForm, T	GuessingN
min	0	int
max	10	int

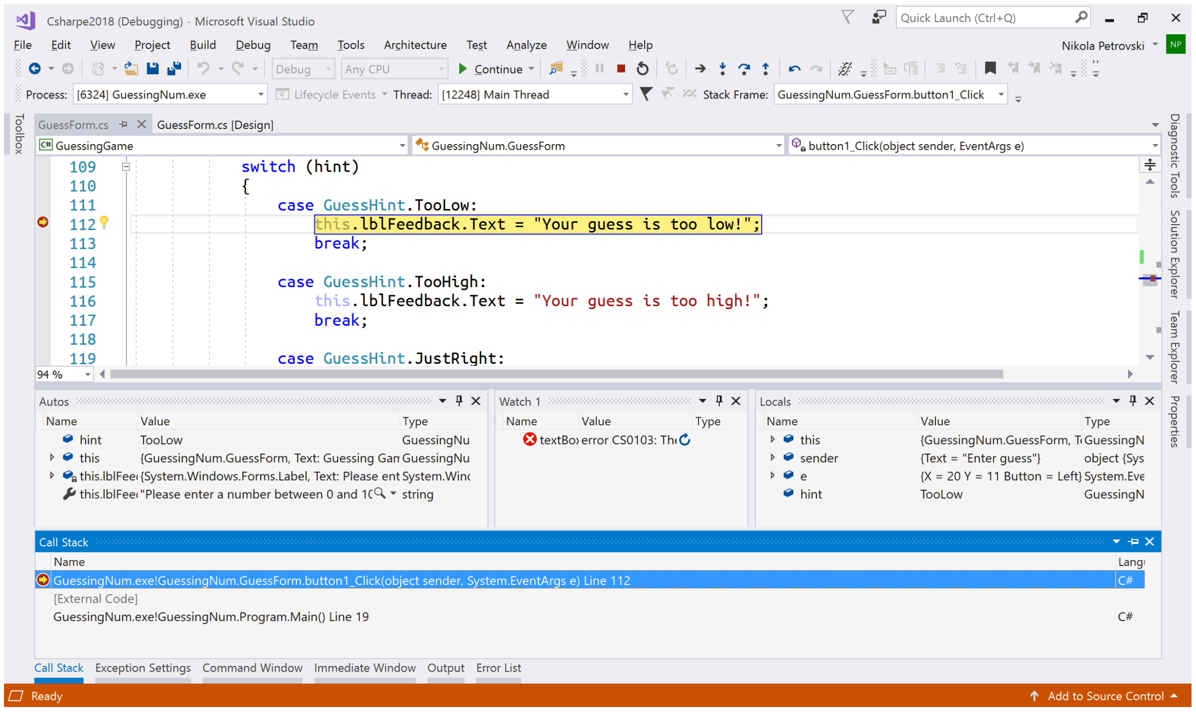
Call Stack

Name	Lang
GuessingNum.exe!GuessingNum.GuessForm.GenerateRandomNumber() Line 155	C#
GuessingNum.exe!GuessingNum.GuessForm.button1_Click(object sender, System.EventArgs e) Line 122	C#
[External Code]	
GuessingNum.exe!GuessingNum.Program.Main() Line 19	C#

Figure 1: the debugger on the line of code that creates the random number generator

Section B:

The function of the switch statement is to make a decision of whether the guess is too low, too high or correct. When one of the cases is disabled it throws and error when that is the case based on the user input. Using the debugger, the variable values can be manipulated and this assumption verified.



```
switch (hint)
{
    case GuessHint.TooLow:
        this.lblFeedback.Text = "Your guess is too low!";
        break;

    case GuessHint.TooHigh:
        this.lblFeedback.Text = "Your guess is too high!";
        break;

    case GuessHint.JustRight:
```

Autos

Name	Value	Type
hint	TooLow	GuessingNu
this	{GuessingNum.GuessForm, Text: Guessing Gan	GuessingNu
this.lblFee	{System.Windows.Forms.Label, Text: Please ent	System.Winc
this.lblFee	"Please enter a number between 0 and 1(	string

Watch 1

Name	Value	Type
textBo	error CS0103: The	

Locals

Name	Value	Type
this	{GuessingNum.GuessForm, T	GuessingN
sender	{Text = "Enter guess"}	object {Sys
e	{X = 20 Y = 11 Button = Left}	System.Eve
hint	TooLow	GuessingN

Call Stack

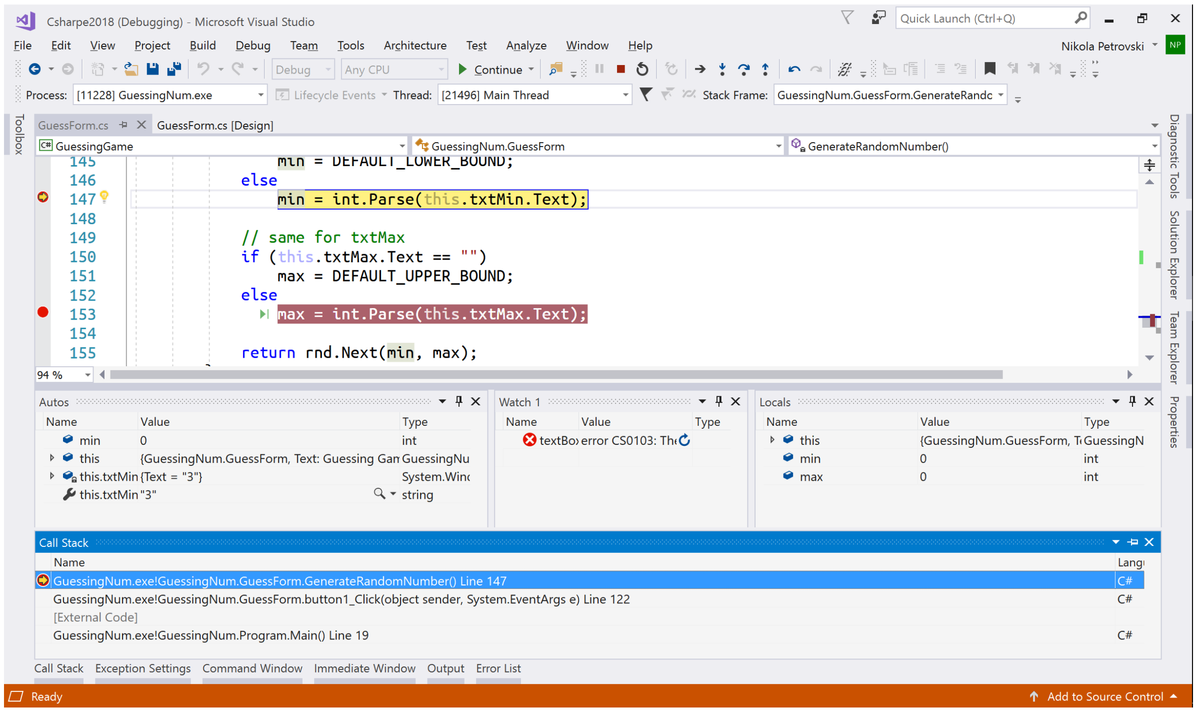
Name	Lang
GuessingNum.exe!GuessingNum.GuessForm.button1_Click(object sender, System.EventArgs e) Line 112	C#
[External Code]	
GuessingNum.exe!GuessingNum.Program.Main() Line 19	C#

Csharpe2018 (Debugging) - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Process: [6324] GuessingNum.exe
Thread: [12248] Main Thread
Stack Frame: GuessingNum.GuessForm.button1_Click
GuessForm.cs
GuessForm.cs [Design]
Continue
Call Stack Exception Settings Command Window Immediate Window Output Error List
Ready
Add to Source Control

Figure 2: the debugger inside the fragment of code making the decision

Section C:

The function of the code for initialization of the guess range is to store the values provided by the user. If the lines shown below are marked with breaks then Visual Studio shows how the range is affected. The debugger allows to change the values and run the program and hover over the objects to track the changes as required.


Csharpe2018 (Debugging) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Nikola Petrovski
Debug
Any CPU
Continue

Process: [11228] GuessingNum.exe Lifecycle Events Thread: [21496] Main Thread Stack Frame: GuessingNum.GuessForm.GenerateRandc

GuessForm.cs GuessForm.cs [Design]

GuessingGame GuessingNum.GuessForm GenerateRandomNumber()

```
            min = DEFAULT_LOWER_BOUND;
        else
            min = int.Parse(this.txtMin.Text);

        // same for txtMax
        if (this.txtMax.Text == "")
            max = DEFAULT_UPPER_BOUND;
        else
            max = int.Parse(this.txtMax.Text);

        return rnd.Next(min, max);
```

94 %

Autos

Name	Value	Type
min	0	int
this	{GuessingNum.GuessForm, Text: Guessing Gan	GuessingNu
this.txtMin	{Text = "3"}	System.Winc
this.txtMin	"3"	string

Watch 1

Name	Value	Type
textBo›	error CS0103: The	

Locals

Name	Value	Type
this	{GuessingNum.GuessForm, T	GuessingN
min	0	int
max	0	int

Call Stack

Name	Lang
GuessingNum.exe!GuessingNum.GuessForm.GenerateRandomNumber() Line 147	C#
GuessingNum.exe!GuessingNum.GuessForm.button1_Click(object sender, System.EventArgs e) Line 122	C#
[External Code]	
GuessingNum.exe!GuessingNum.Program.Main() Line 19	C#

Figure 3: the debugger on the line(s) of code that initializes the guess range

Note: The screen shots are one of more of "watch", "autos", "call stack" and "locals" windows in addition to the main coding window (showing the breakpoint(s));

4.0 A 5-minute video demonstrating the required debugging process

Bonus: Create and post a 5-minute video demonstrating the required debugging process you have gone through for Part 3. In the video, identify assumptions and how they are verified using the Visual Studio debugger.
[to be published]