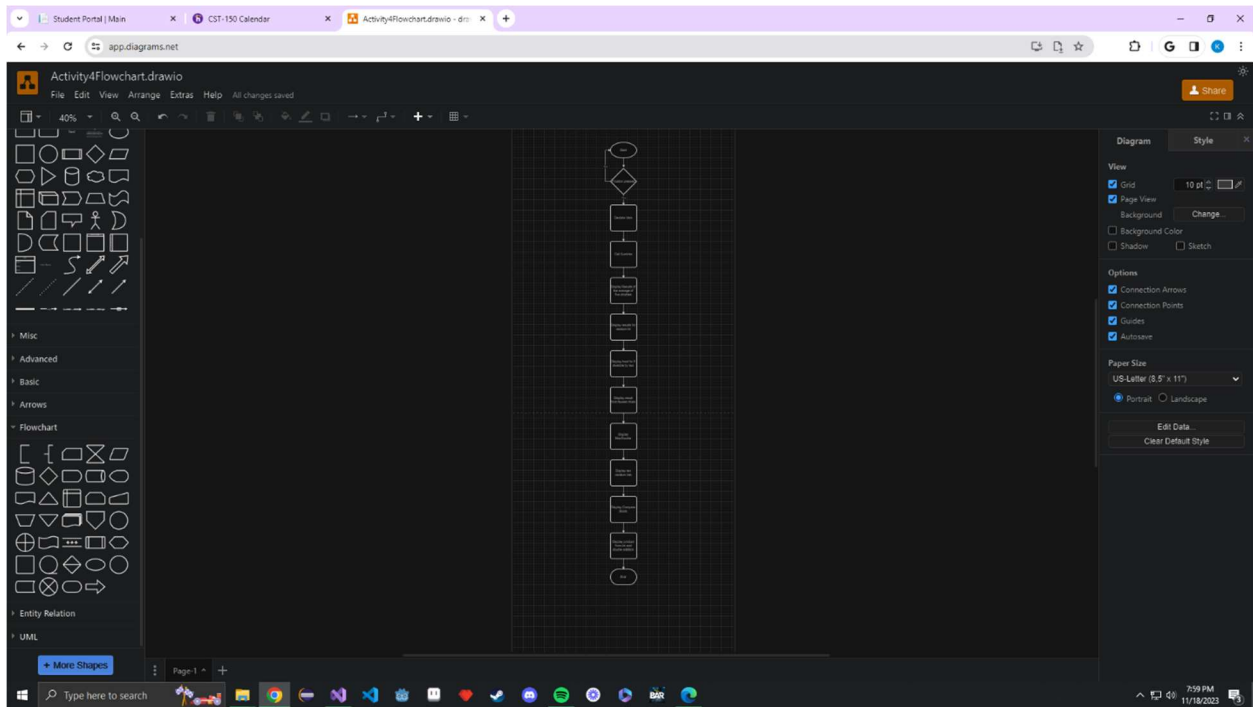


Kohl Johnson

11-18-2023

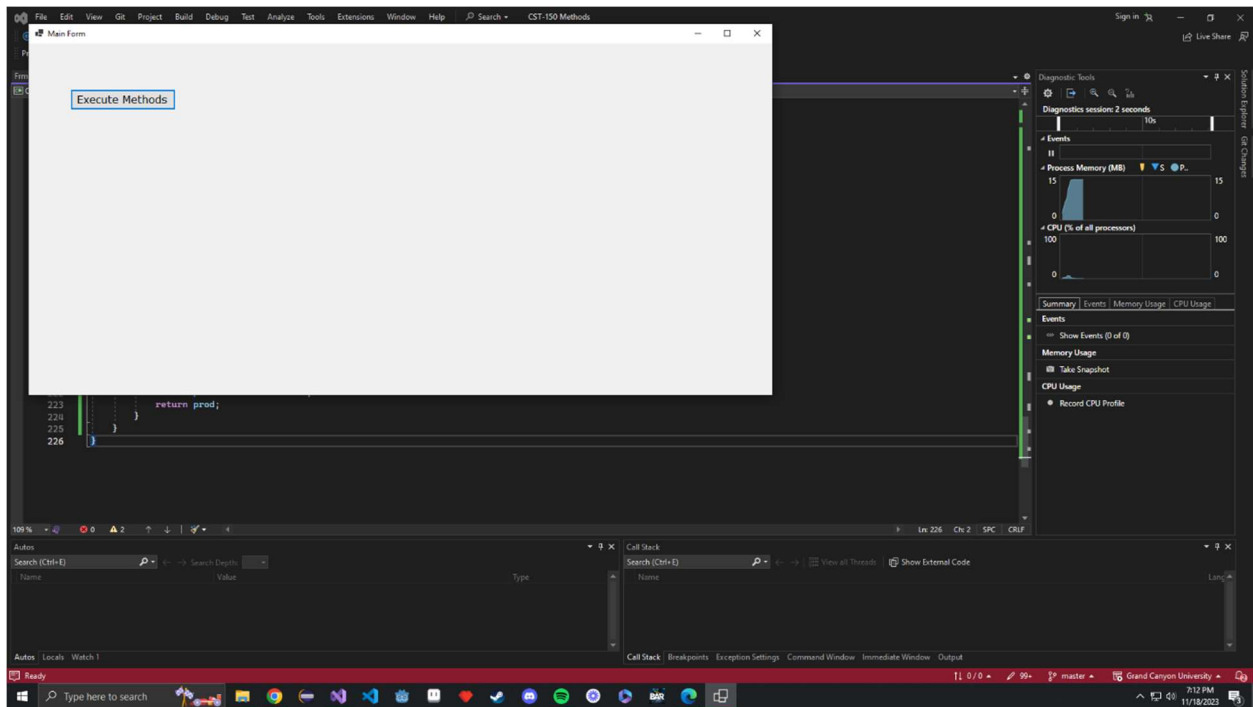
Activity 4 Part 2 Coversheet

Flowchart Screenshot



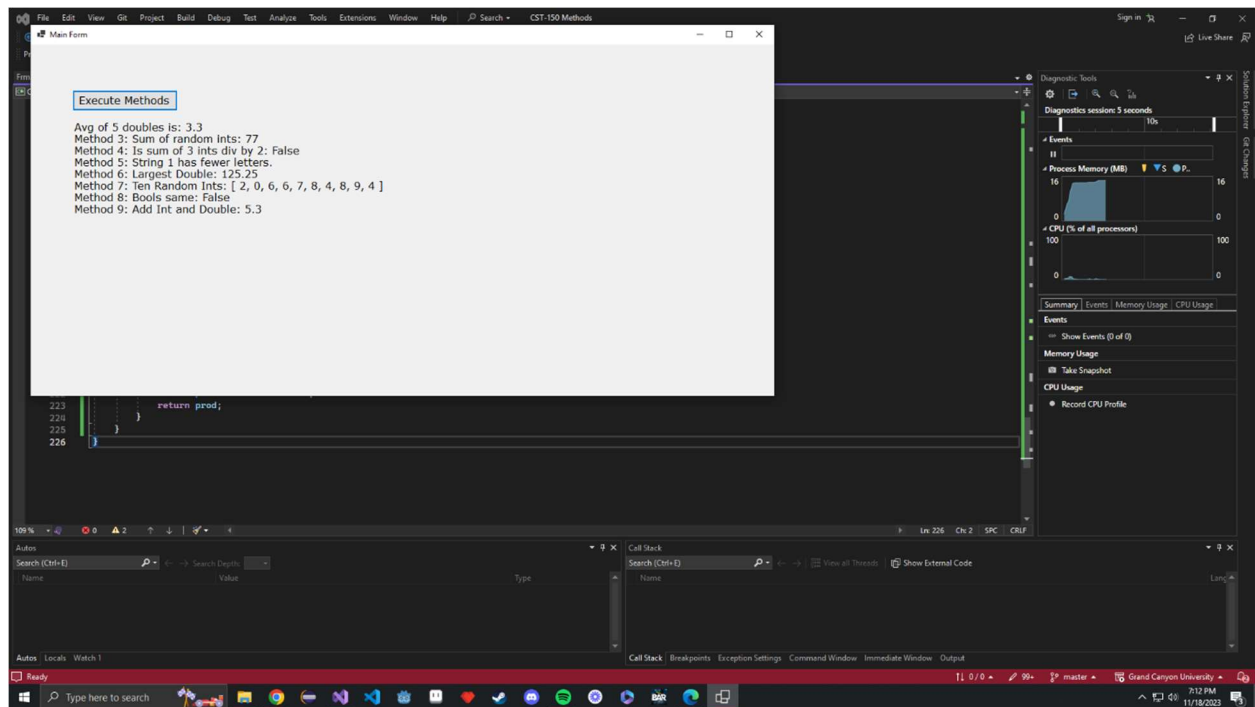
Shows the flow of the program with the button click event and then all the method calls that follow.

Pre-Populated Form



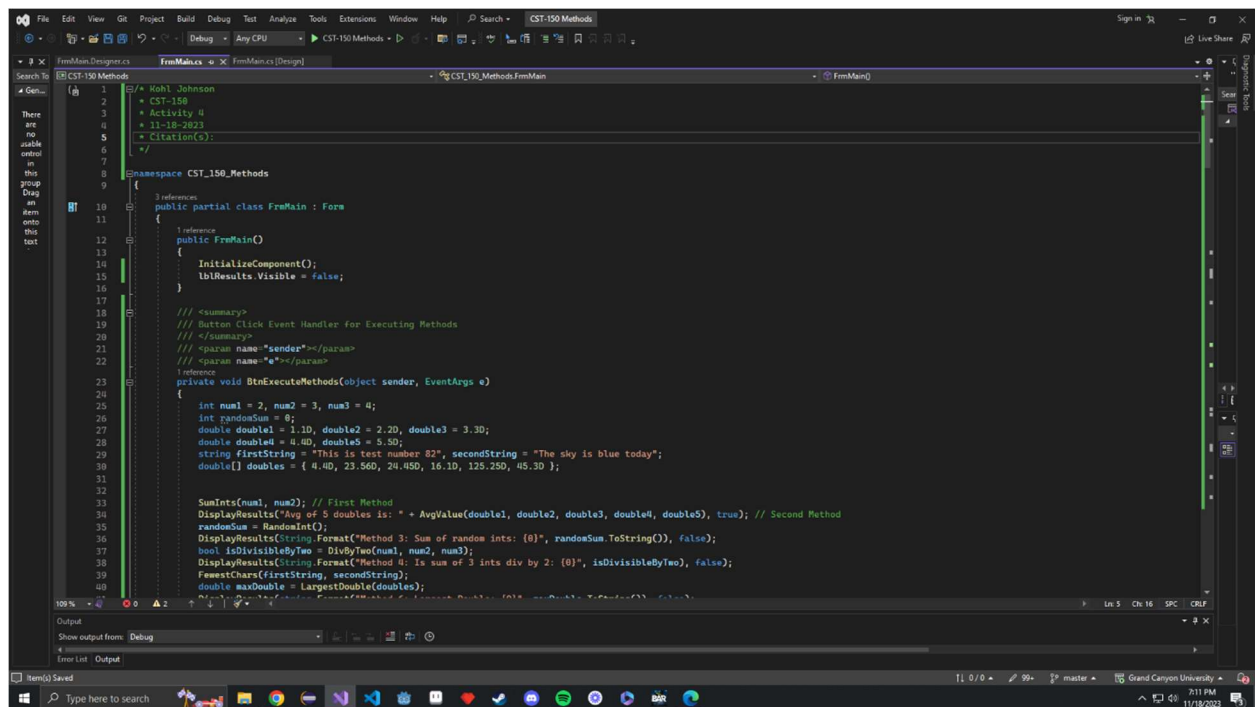
Just a single button haha. (with the label hidden of course)

Post Button Click Screenshot



Shows the results from each method, with the descriptive text.

Code Screenshot 1



Shows the first part of the program with the citation on the top.

Code Screenshot 2

```

15         lblResults.Visible = false;
16     }
17
18     /// <summary>
19     /// Button Click Event Handler for Executing Methods
20     /// </summary>
21     /// <param name="sender"></param>
22     /// <param name="e"></param>
23     [reference]
24     private void BtnExecuteMethods(object sender, EventArgs e)
25     {
26         int num1 = 2, num2 = 3, num3 = 4;
27         int randomSum = 0;
28         double double1 = 1.1D, double2 = 2.2D, double3 = 3.3D;
29         double double4 = 4.4D, double5 = 5.5D;
30         string firstString = "This is test number 82", secondString = "The sky is blue today";
31         double[] doubles = { 4.4D, 23.56D, 24.45D, 16.1D, 125.25D, 45.3D };
32
33         SumInts(num1, num2); // First Method
34         DisplayResults("Avg of 5 Doubles is: " + AvgValue(double1, double2, double3, double4, double5), true); // Second Method
35         randomSum = RandomInt();
36         DisplayResults(String.Format("Method 3: Sum of random ints: {0}", randomSum.ToString()), false);
37         bool isDivisibleByTwo = DivByTwo(num1, num2, num3);
38         DisplayResults(String.Format("Method 4: Is sum of 3 ints div by 2: {0}", isDivisibleByTwo), false);
39         FewestChars(firstString, secondString);
40         double maxDouble = LargestDouble(GenDoubles);
41         DisplayResults(string.Format("Method 6: Largest Double: {0}", maxDouble.ToString()), false);
42         int[] tenRandInts = TenRandomInts();
43         string tenInts = "{ ";
44         for (int x = 0; x < tenRandInts.Length; x++)
45         {
46             tenInts += tenRandInts[x];
47             if (x < tenRandInts.Length - 1) { tenInts += ", "; } else { tenInts += " "; }
48         }
49         DisplayResults(String.Format("Method 7: Ten Random Ints: {0}", tenInts), false);
50         bool b1 = true, b2 = false;
51         DisplayResults(String.Format("Method 8: Bools same: {0}", CompareBools(b1, b2)), false);
52         DisplayResults(String.Format("Method 9: Add Int and Double: {0}", Product(num1, double3)), false);
53         lblResults.Visible = true;
54     }
55 }

```

Shows the variable declarations and the method calls for each of the steps. Along with some of the display result calls for methods that did not return void.

Code Screenshot 3

The screenshot displays the Visual Studio Code editor with the file `FfmMain.cs` open. The code is a C# program that demonstrates various features, including XML documentation comments and method calls. The code is as follows:

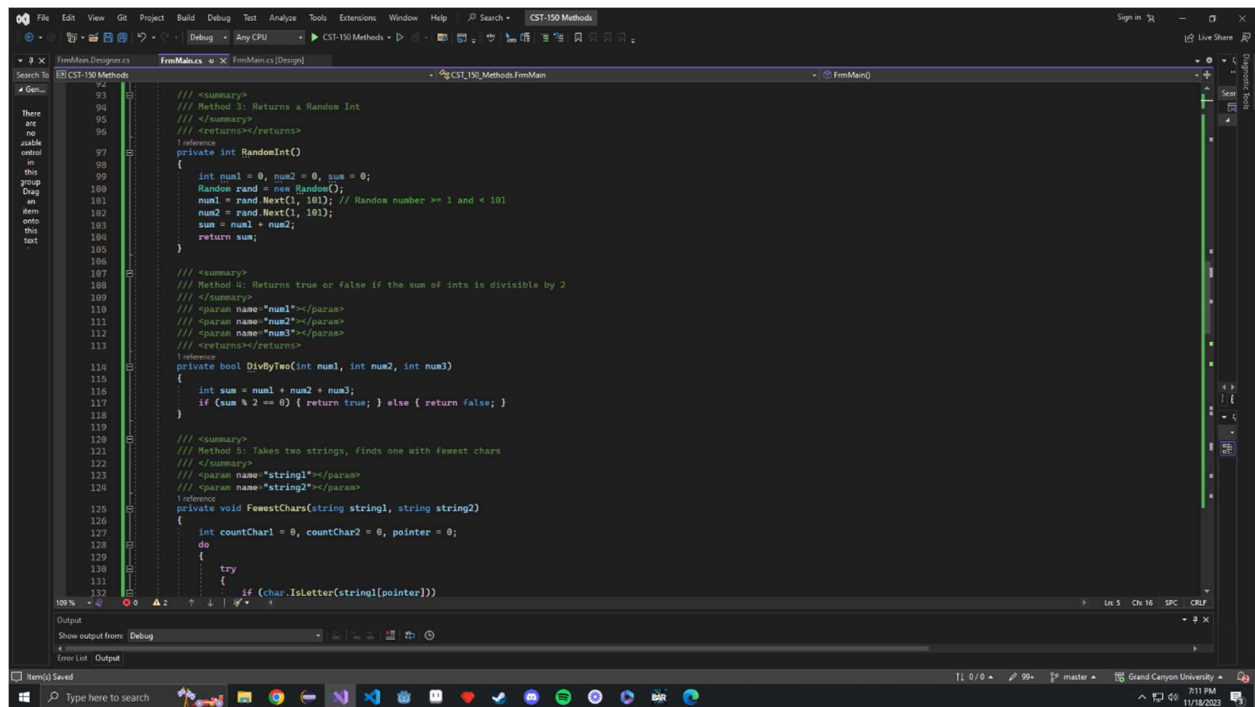
```
56  /// <summary>
57  /// Method 1: Add Two Integers Together and Display Sum with Descriptive Text
58  /// </summary>
59  /// <param name="num1"></param>
60  /// <param name="num2"></param>
61  1 reference
62  private void SumInts(int num1, int num2)
63  {
64      int sum = num1 + num2; // find the sum
65      DisplayResults("Method 1: Sum of " + num1 + " and " + num2 + " = " + sum, true); // call DisplayResults
66  }
67  /// <summary>
68  /// Displays descText to lblResults, requires the text to display and a bool for clearing the label prior
69  /// </summary>
70  /// <param name="descText"></param>
71  /// <param name="clearLabel"></param>
72  1 reference
73  private void DisplayResults(string descText, bool clearLabel)
74  {
75      if (clearLabel) { lblResults.Text = ""; } // clear the lbl if needed
76      lblResults.Text += String.Format("{0}\n", descText); // display results
77  }
78  /// <summary>
79  /// Method 2: Takes in Five Doubles and Returns the Average of those
80  /// </summary>
81  /// <param name="val1"></param>
82  /// <param name="val2"></param>
83  /// <param name="val3"></param>
84  /// <param name="val4"></param>
85  /// <param name="val5"></param>
86  1 reference
87  private double AvgValue(double val1, double val2, double val3, double val4, double val5)
88  {
89      double average = (val1 + val2 + val3 + val4 + val5) / 5.0;
90      return average;
91  }
92  /// <summary>
93  /// Method 3: Returns a Random Int
94  /// </summary>
95  1 reference
```

The Output window at the bottom shows the execution results:

```
Output
Show output from: Debug
Method 1: Sum of 1 and 2 = 3
Method 2: Average of 1.0, 2.0, 3.0, 4.0, 5.0 is 3.0
Method 3: Random Int is 42
```

Shows the code for methods one, three, and the DisplayResults function.

Code Screenshot 4

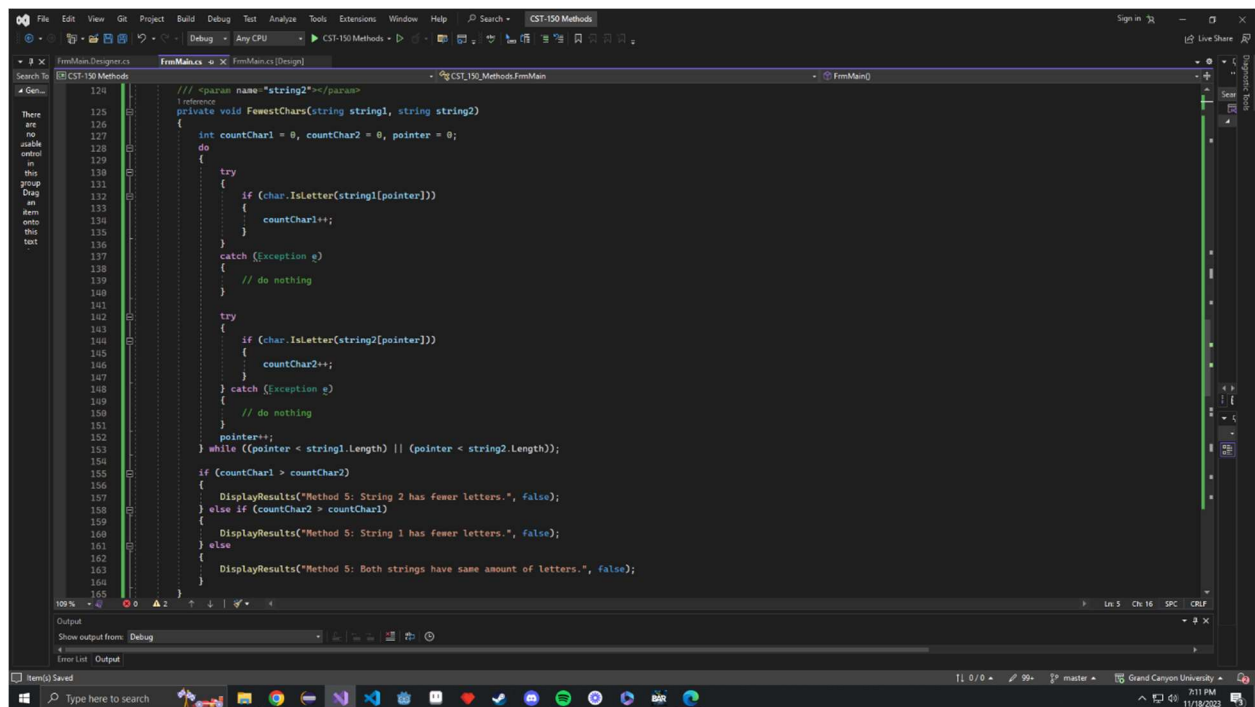


This screenshot shows the Visual Studio IDE with the 'FrmMain.cs' file open. The code editor displays the following C# code:

```
197 // summary
198 // Method 3: Returns a Random Int
199 // summary
200 // returns
201 reference int RandomInt()
202 {
203     int num1 = 0, num2 = 0, sum = 0;
204     Random rand = new Random();
205     num1 = rand.Next(1, 101); // Random number >= 1 and < 101
206     num2 = rand.Next(1, 101);
207     sum = num1 + num2;
208     return sum;
209 }
210
211 // summary
212 // Method 4: Returns true or false if the sum of ints is divisible by 2
213 // summary
214 // param name="num1"
215 // param name="num2"
216 // param name="num3"
217 // returns
218 reference bool DivByTwo(int num1, int num2, int num3)
219 {
220     int sum = num1 + num2 + num3;
221     if (sum % 2 == 0) { return true; } else { return false; }
222 }
223
224 // summary
225 // Method 5: Takes two strings, finds one with fewest chars
226 // summary
227 // param name="string1"
228 // param name="string2"
229 reference void FewestChars(string string1, string string2)
230 {
231     int countChar1 = 0, countChar2 = 0, pointer = 0;
232     do
233     {
234         try
235         {
236             if (char.IsLetter(string1[pointer]))
237             {
238                 countChar1++;
239             }
240             catch (Exception e)
241             {
242                 // do nothing
243             }
244             try
245             {
246                 if (char.IsLetter(string2[pointer]))
247                 {
248                     countChar2++;
249                 }
250                 catch (Exception e)
251                 {
252                     // do nothing
253                 }
254             }
255             pointer++;
256         } while ((pointer < string1.Length) || (pointer < string2.Length));
257         if (countChar1 > countChar2)
258         {
259             DisplayResults("Method 5: String 2 has fewer letters.", false);
260         }
261         else if (countChar2 > countChar1)
262         {
263             DisplayResults("Method 5: String 1 has fewer letters.", false);
264         }
265         else
266         {
267             DisplayResults("Method 5: Both strings have same amount of letters.", false);
268         }
269     }
```

Shows the code for method three, four, and part of method five.

Code Screenshot 5

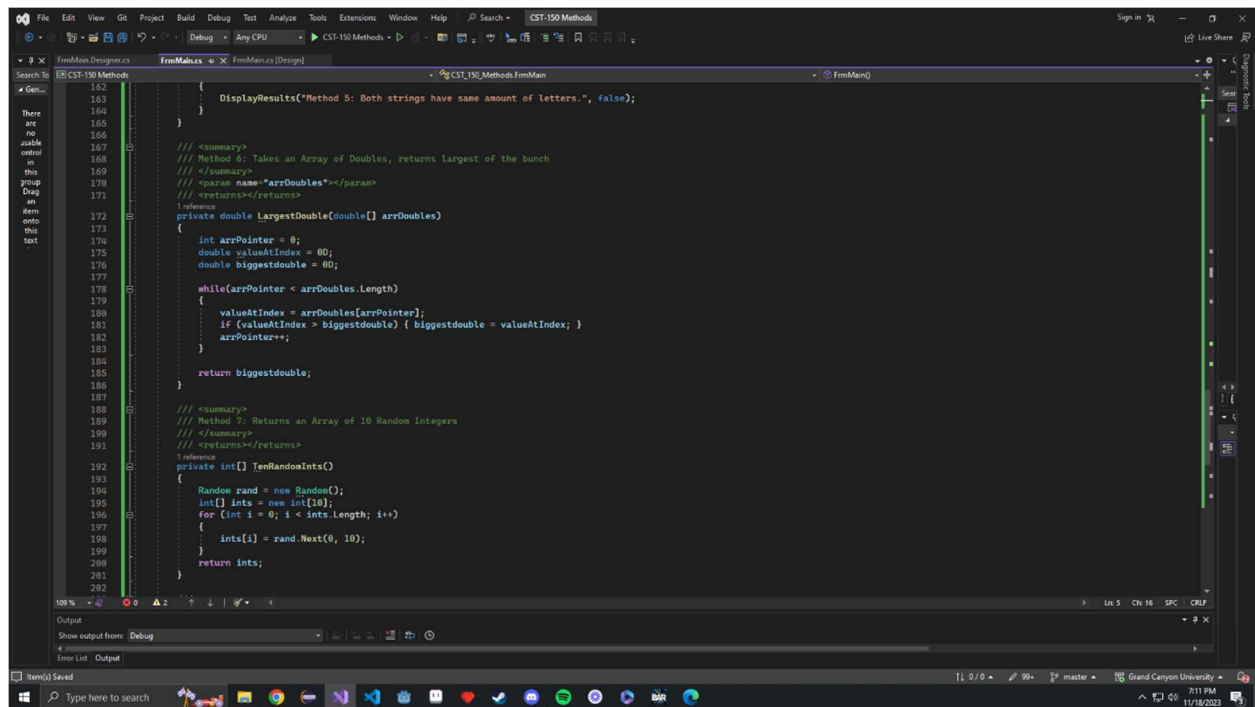


This screenshot shows the Visual Studio IDE with the 'FrmMain.cs' file open. The code editor displays the following C# code:

```
270     }
271     while ((pointer < string1.Length) || (pointer < string2.Length));
272     if (countChar1 > countChar2)
273     {
274         DisplayResults("Method 5: String 2 has fewer letters.", false);
275     }
276     else if (countChar2 > countChar1)
277     {
278         DisplayResults("Method 5: String 1 has fewer letters.", false);
279     }
280     else
281     {
282         DisplayResults("Method 5: Both strings have same amount of letters.", false);
283     }
284 }
```

Shows the rest of the code for method five.

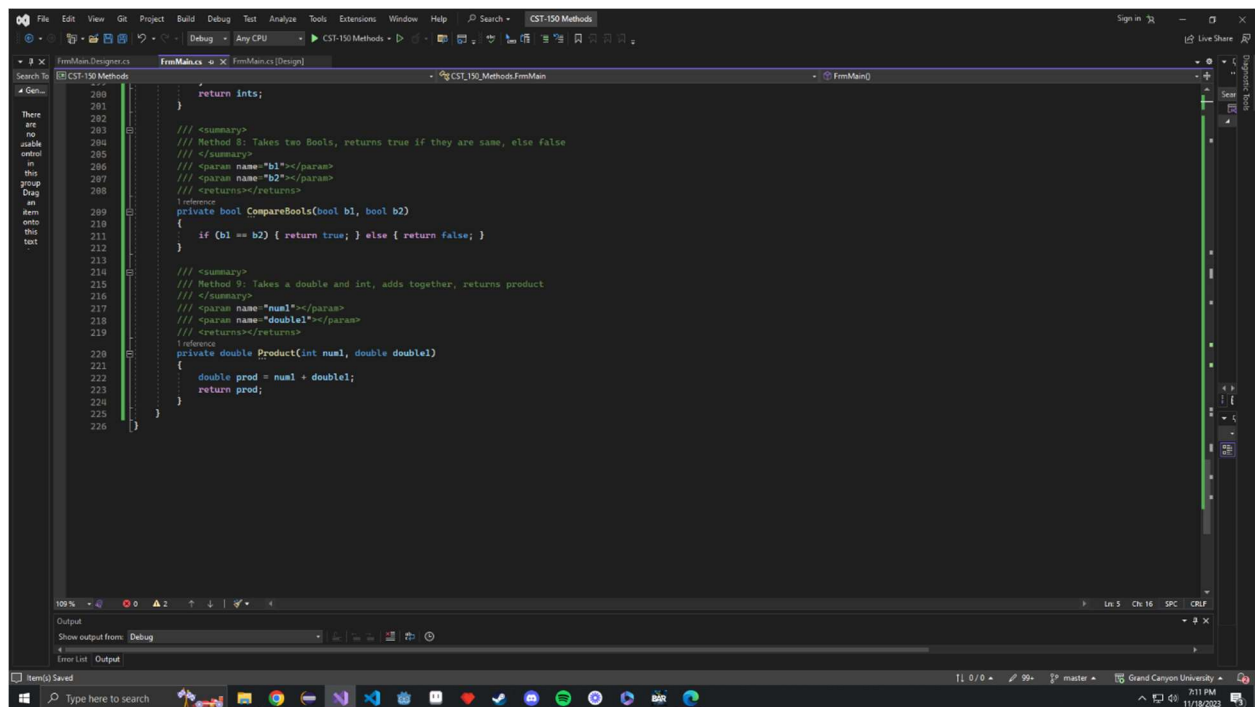
Code Screenshot 6



```
162 {
163     DisplayResults("Method 5: Both strings have same amount of letters.", false);
164 }
165
166
167 /// <summary>
168 /// Method 6: Takes an Array of Doubles, returns largest of the bunch
169 /// </summary>
170 /// <param name="arrDoubles"></param>
171 /// <returns></returns>
172 reference
173 private double LargestDouble(double[] arrDoubles)
174 {
175     int arrPointer = 0;
176     double valueAtIndex = 60;
177     double biggestDouble = 60;
178
179     while(arrPointer < arrDoubles.Length)
180     {
181         valueAtIndex = arrDoubles[arrPointer];
182         if (valueAtIndex > biggestDouble) { biggestDouble = valueAtIndex; }
183         arrPointer++;
184     }
185
186     return biggestDouble;
187 }
188
189 /// <summary>
190 /// Method 7: Returns an Array of 10 Random Integers
191 /// </summary>
192 /// <returns></returns>
193 reference
194 private int[] TenRandomInts()
195 {
196     Random rand = new Random();
197     int[] ints = new int[10];
198     for (int i = 0; i < ints.Length; i++)
199     {
200         ints[i] = rand.Next(0, 10);
201     }
202     return ints;
203 }
```

Shows the code for methods six and seven.

Code Screenshot 7



```
204 }
205
206 /// <summary>
207 /// Method 8: Takes two Booleans, returns true if they are same, else false
208 /// </summary>
209 /// <param name="b1"></param>
210 /// <param name="b2"></param>
211 /// <returns></returns>
212 reference
213 private bool CompareBools(bool b1, bool b2)
214 {
215     if (b1 == b2) { return true; } else { return false; }
216 }
217
218 /// <summary>
219 /// Method 9: Takes a double and int, adds together, returns product
220 /// </summary>
221 /// <param name="num1"></param>
222 /// <param name="double1"></param>
223 /// <returns></returns>
224 reference
225 private double Product(int num1, double double1)
226 {
227     double prod = num1 * double1;
228     return prod;
229 }
230 }
```

Shows the code for the final methods, number eight and nine.