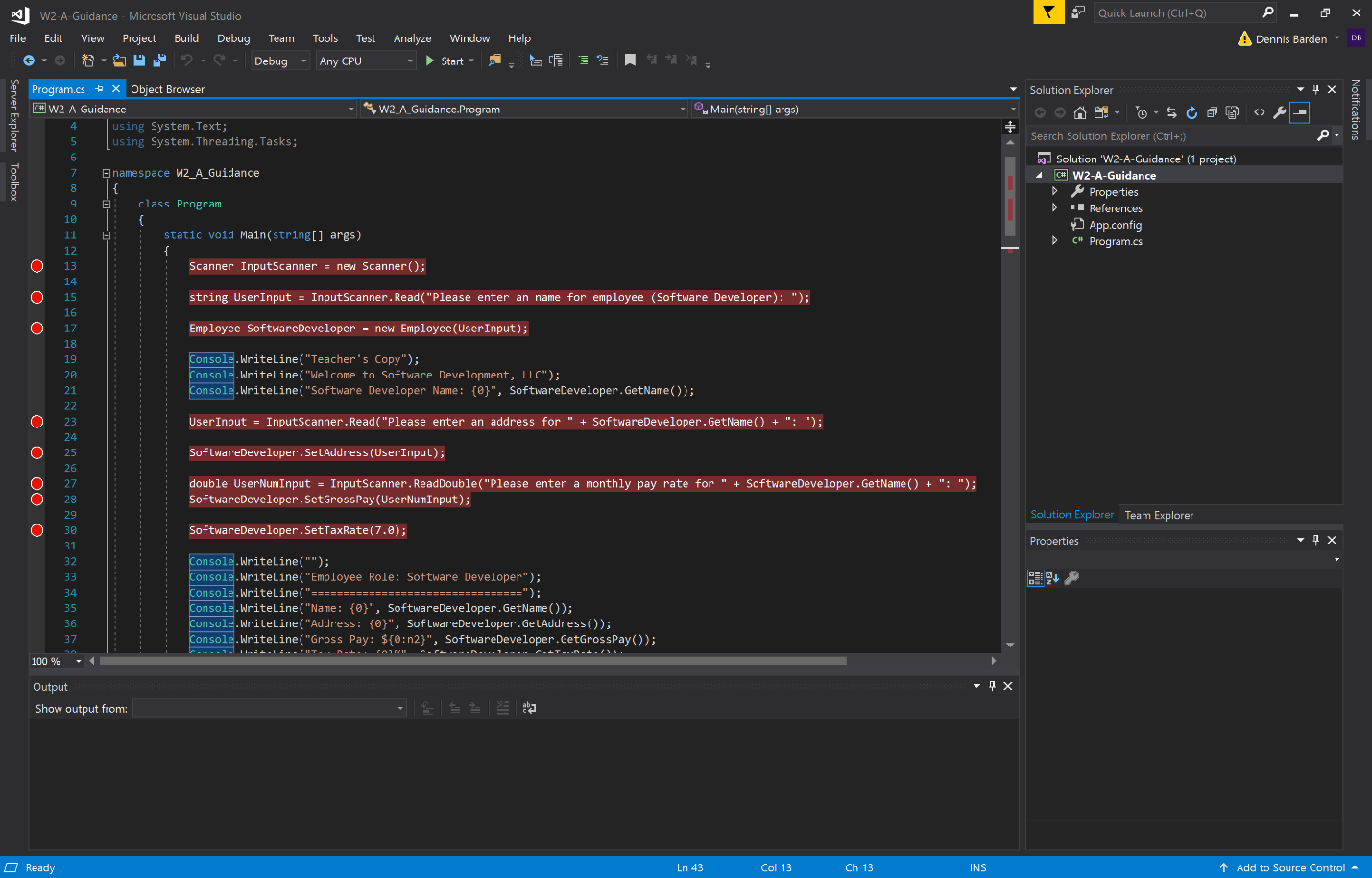
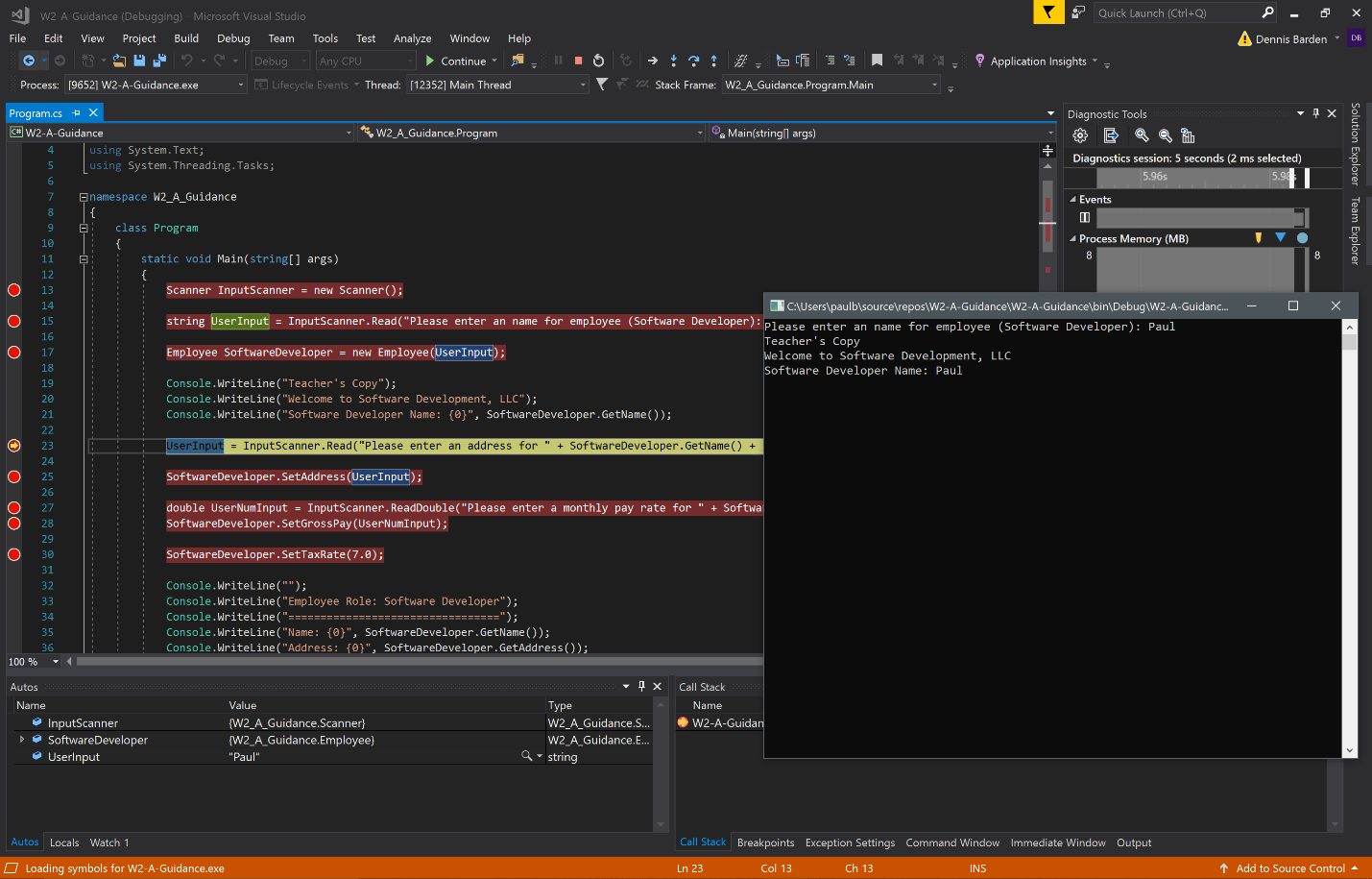
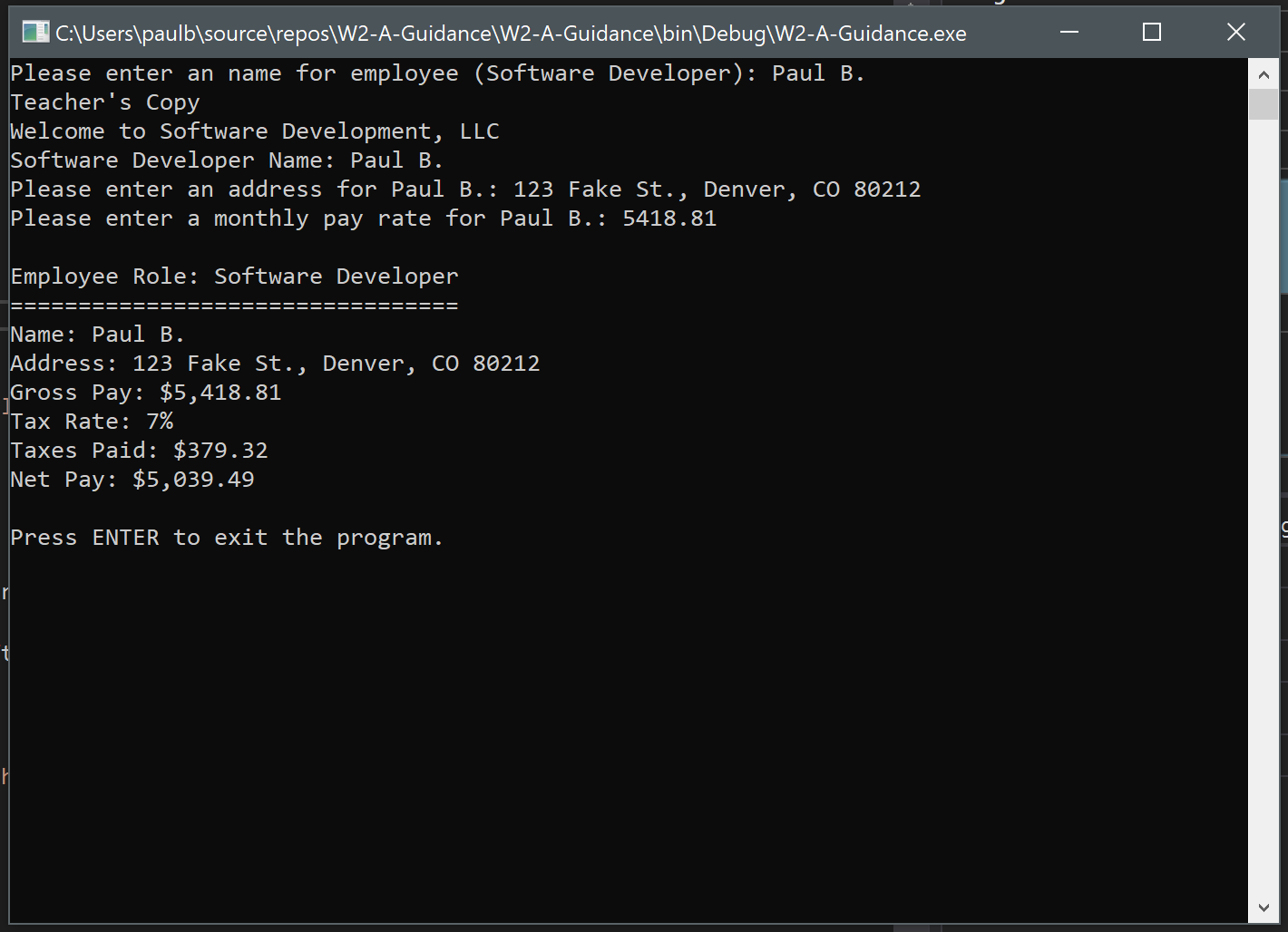
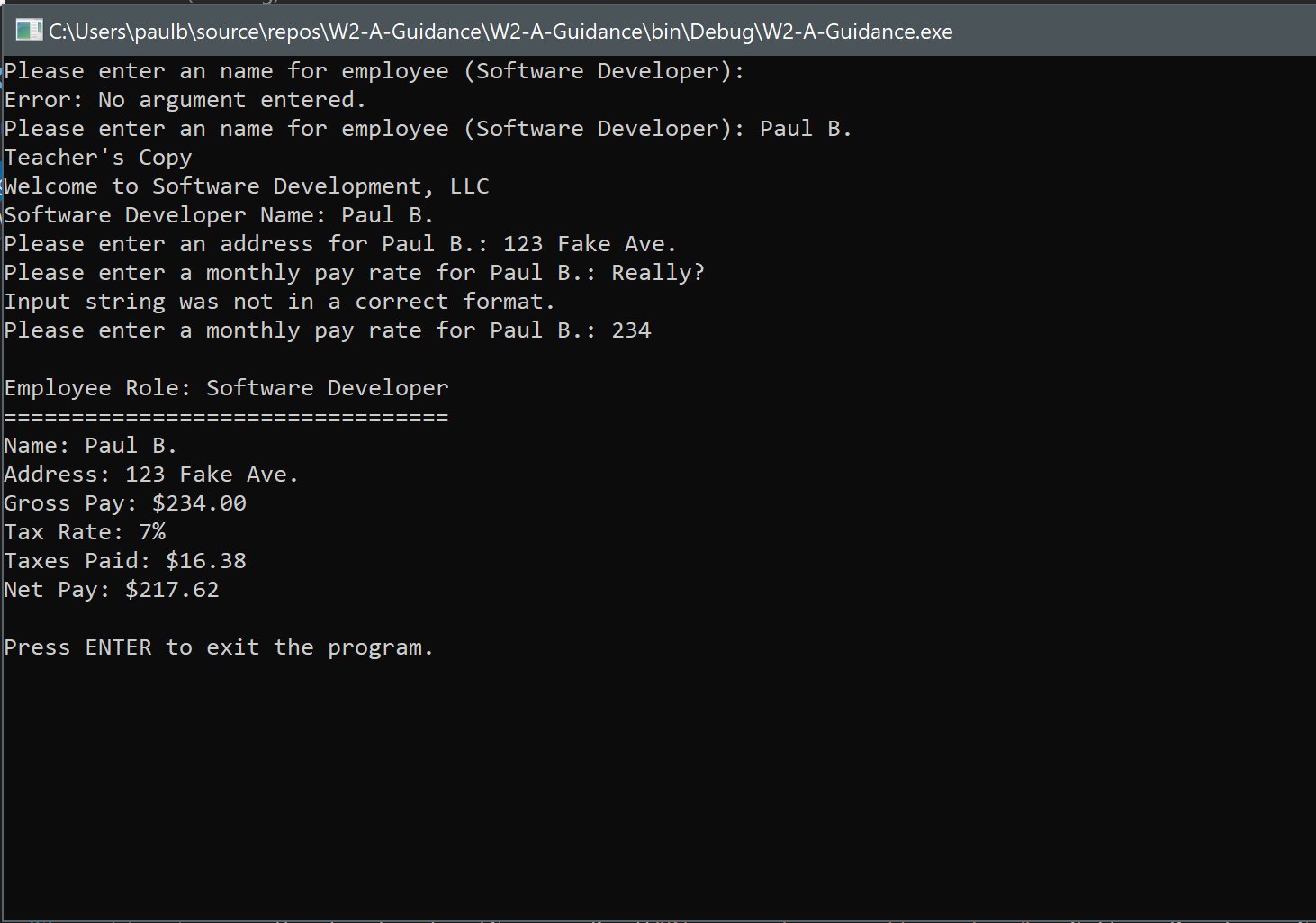
Demonstration of breakpoints and debugging:





Demonstration of the program running (with intended flow and sample of input validation):





Source code is below; comments are in **BOLD** to explain the application logic:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace W2\_A\_Guidance

{

class Program

{

static void Main(string[] args)

{

**// Create a new Scanner object (defined on line 139) which handles the collection**

**// of user input and includes limited validation for types string and double**

Scanner InputScanner = new Scanner();

**// Use the Scanner object to read user input. The user prompt is passed in.**

**// InputScanner() returns a string which is assigned to the UserInput string variable**

string UserInput = InputScanner.Read("Please enter an name for employee (Software Developer): ");

**// Create a new Employee object using the name assigned to the UserInput variable above**

Employee SoftwareDeveloper = new Employee(UserInput);

Console.WriteLine("Teacher's Copy");

Console.WriteLine("Welcome to Software Development, LLC");

Console.WriteLine("Software Developer Name: {0}", SoftwareDeveloper.GetName());

**// InputScanner() returns a string which is assigned to the UserInput string variable**

UserInput = InputScanner.Read("Please enter an address for " + SoftwareDeveloper.GetName() + ": ");

**// Use UserInput to assign a new address to the employee**

SoftwareDeveloper.SetAddress(UserInput);

**// InputScanner() returns a double which is assigned to the UserNumInput double variable**

double UserNumInput = InputScanner.ReadDouble("Please enter a monthly pay rate for " + SoftwareDeveloper.GetName() + ": ");

SoftwareDeveloper.SetGrossPay(UserNumInput);

**// Sets a tax rate**

SoftwareDeveloper.SetTaxRate(7.0);

**// Display all employee information stored**

Console.WriteLine("");

Console.WriteLine("Employee Role: Software Developer");

Console.WriteLine("=================================");

Console.WriteLine("Name: {0}", SoftwareDeveloper.GetName());

Console.WriteLine("Address: {0}", SoftwareDeveloper.GetAddress());

Console.WriteLine("Gross Pay: ${0:n2}", SoftwareDeveloper.GetGrossPay());

Console.WriteLine("Tax Rate: {0}%", SoftwareDeveloper.GetTaxRate());

Console.WriteLine("Taxes Paid: ${0:n2}", SoftwareDeveloper.GetTaxesPaid());

Console.WriteLine("Net Pay: ${0:n2}", SoftwareDeveloper.GetNetPay());

Console.WriteLine("");

**// Added to prevent the application from closing until user has a chance to view information** output above

Console.WriteLine("Press ENTER to exit the program.");

string NewInput = Console.ReadLine();

}

}

**// Defines the Employee object**

class Employee

{

**// Attributes belonging to the instance object**

private string Name;

private string Address;

private double GrossPay;

private double NetPay;

private double TaxRate;

**// Initializes the Employee object using a string which is then assigned to the name instance variable**

public Employee(string name)

{

this.Name = name;

}

**// Returns the name as a string**

public string GetName()

{

return this.Name;

}

**// Sets a new employee name**

public void SetName(string NewName)

{

this.Name = NewName;

}

**// Returns the address as a string**

public string GetAddress()

{

return this.Address;

}

**// Sets a new employee address**

public void SetAddress(string NewAddress)

{

this.Address = NewAddress;

}

**// Returns the gross pay as a double**

public double GetGrossPay()

{

return this.GrossPay;

}

**// Sets a new employee monthly gross pay**

public void SetGrossPay(double AdjustedGross)

{

this.GrossPay = AdjustedGross;

}

**// Returns the net pay as a double**

public double GetNetPay()

{

this.NetPay = this.GrossPay - GetTaxesPaid();

return this.NetPay;

}

**// Returns the tax rate as a double**

public double GetTaxRate()

{

return TaxRate;

}

**// Sets a new employee tax rate as a percentage**

public void SetTaxRate(double Rate)

{

this.TaxRate = Rate;

}

**// Returns the tax amount deducted from the monthly pay**

public double GetTaxesPaid()

{

double TaxAmount = this.GrossPay \* (this.TaxRate / 100);

return TaxAmount;

}

}

**// Defines the Scanner object**

class Scanner {

**// Instantiates the Scanner**

public Scanner() { }

**// Used to return a string collected from user on the console**

**// Takes the user message prompt passed in as a parameter**

public string Read(string Prompt)

{

**// Initialize UserInput as an empty string**

string UserInput = "";

**// This loop is performed until the user enters a valid string**

while (UserInput == "")

{

try

{

**// Write the prompt to the console and collect user input from the same line**

Console.Write(Prompt);

string ProcessInput = Console.ReadLine();

**// If nothing is entered, an exception will be thrown and its message will be displayed on the console**

if (ProcessInput == "")

{

throw new Exception("Error: No argument entered.");

}

**// If something is entered, assign it to the string object**

else

{

UserInput = ProcessInput;

}

}

**// Generic exceptions are caught and displayed, though in a full**

**// program this can be an opportunity to correct errors**

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

**// Returns user input as a string**

return UserInput;

}

**// Used to return a double collected from user on the console**

**// Takes the user message prompt passed in as a parameter**

public double ReadDouble(string Prompt)

{

**// Initialize UserInput as zero**

double UserInput = 0;

**// This loop is performed until the user enters a valid double**

while (UserInput == 0)

{

try

{

**// Write the prompt to the console and collect user input from the same line**

Console.Write(Prompt);

double ProcessInput = Convert.ToDouble(Console.ReadLine());

**// Executes when user enters a value 0 or less**

if (ProcessInput < 0 || ProcessInput == 0)

{

throw new Exception("Error: Number must be greater than zero.");

}

**// Executes if user enters number more than 0**

else if (ProcessInput > 0)

{

UserInput = ProcessInput;

}

**// Otherwise an exception is thrown**

else

{

throw new Exception("Error: No valid argument entered.");

}

}

**// Catches generic exceptions and displays message on the console**

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

**// returns input as a double**

return UserInput;

}

}

}