

# IT Assignment Coversheet

Course: PROG8170 – Software Quality Assurance Techniques

Program Coordinator: David Allison

Professor/Instructor: Preethi Arattu

Assignment #: 1

Assignment Type: ☐ Individual ☐ Pair ☐ Team

Date Submitted: 07th February 2020

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student Information**  |  |  | | --- | --- | | **Name** | Tarunpreet Singh | | **Student Id** | 8668535 | |

PROG8170

Software Quality Assurance Techniques

Assignment #1: Rubric

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Mark** | **Weight** | **Criteria** |
|  |  |  | **Rectangle Class** |
| **1.** |  | **1** | Rectangle class created as a separate file |
| **2.** |  | **1** | Rectangle class length and width attributes are private |
| **3.** |  | **2** | Default and Non-Default constructor created and working properly |
| **4.** |  | **6** | Six required methods created and working properly |
|  |  |  | **Console Application** |
| **5.** |  | **3** | Initial rectangle created as described, incorrect input handled. |
| **6.** |  | **3** | Menu option 1 works as described |
| **7.** |  | **3** | Menu option 2 works as described |
| **8.** |  | **3** | Menu option 3 works as described |
| **9.** |  | **3** | Menu option 4 works as described |
| **10.** |  | **5** | Menu option 5 works as described |
| **11.** |  | **5** | Menu option 6 works as described |
| **12.** |  | **3** | Menu option 7 works as described |
|  |  |  | **Unit Tests** |
| **13.** |  | **3** | Unit test for GetLength() method |
| **14.** |  | **3** | Unit test for SetLength() method |
| **15.** |  | **3** | Unit test for GetWidth() method |
| **16.** |  | **3** | Unit test for SetWidth() method |
| **17.** |  | **3** | Unit test for GetPerimeter() method |
| **18.** |  | **3** | Unit test for GetArea() method |
| **19.** |  | **1** | Screenshot of completed unit tests run successfully |
|  |  |  | **Git** |
| **20.** |  | **3** | Screenshot showing Git repository log and required commits |
|  |  |  |  |
|  |  | **-0.50 each** | **Programming standards deductions.** |
|  |  | **-12** | **Failure to present deduction.** |
|  |  | **-12** | **Late submission (per day)** |
|  |  | **-6** | **Missing documentation from hard copy printout or not in correct order** |
|  |  |  |  |
|  |  | **60** | **Total Marks** |

# **Source Code**

**Program.cs**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Assignment1  {  class Program  {  static void Main(string[] args)  {  //get user input of length and width  int[] attributes = GetRectangleAttributes();  /\*create an instance of the rectangle with the attributes provided by the user  \* attributes[0] = length  \* attributes[1] = width  \*/  Rectangle rectangle = new Rectangle(attributes[0],attributes[1]);  //Call method to run menu section  Menu(rectangle);  Console.ReadKey();  }  private static int[] GetRectangleAttributes()  {  int length = 0;  int width = 0;  //get length of the rectangle  length = GetLengthFromUser();  //get the width of the rectangle  width = GetWidthFromUser();  return new int[] {length,width};  }  private static int GetWidthFromUser()  {  int width = 0;  do  {  try  {  //get the width of the rectangle  Console.Write("\nEnter the width of the rectangle :");  width = Int32.Parse(Console.ReadLine());  //check if the user has entered 0  if (width <= 0)  {  throw new Exception();  }  break;  }  catch (Exception e) //something unexpected happened  {  Console.Write("\nPlease enter a positive integer value greater than 0 less than 2,147,483,648\n\n");  }  } while (true);  return width;  }  private static int GetLengthFromUser()  {  int length = 0;  do  {  try  {  //get the length of the rectangle  Console.Write("\nEnter the length of the rectangle :");  length = Int32.Parse(Console.ReadLine());  //check if the user has entered 0  if (length <= 0)  {  throw new Exception();  }  break;  }  catch (Exception e) //something unexpected happened  {  Console.Write("\nPlease enter a positive integer value greater than 0 less than 2,147,483,648\n\n");  }  } while (true);  return length;  }  public static void Menu(Rectangle rectangle)  {  int choice = 0;  string[] options = {  "Get Rectangle Length",  "Change Rectangle Length",  "Get Rectangle Width",  "Change Rectangle Width",  "Get Rectangle Perimeter",  "Get Rectangle Area",  "Exit"  };  while(true)  {  ShowMenu(options);  try  {  Console.Write("\n\nEnter a valid choice - ");  //get the user input  choice = Int32.Parse(Console.ReadLine());  //check if user has entered a valid input  if (choice <= 0 || choice > options.Length)  {  throw new Exception();  }  //show output according to user input  ShowOutput(rectangle,choice);  }  catch(Exception e)  {  Console.Write("\n\nPlease enter a valid choice!");  }  }  }  private static void ShowOutput(Rectangle rectangle, int choice)  {  //give output according to the input provided  switch (choice)  {  case 1: Console.Write("\n\nLength of Rectangle : " + rectangle.GetLength());  break;  case 2: int length = rectangle.SetLength(GetLengthFromUser());  Console.Write("\n\nRectangle's length set to : " + length);  break;  case 3: Console.Write("\n\nWidth of Rectangle : " + rectangle.GetWidth());  break;  case 4: int width = rectangle.SetWidth(GetWidthFromUser());  Console.Write("\n\nRectangle's width set to : " + width);  break;  case 5: Console.Write("\n\nPerimeter of the rectangle : " + rectangle.GetPerimeter());  break;  case 6: Console.Write("\n\nArea of the rectangle " + rectangle.GetArea());  break;  case 7: Environment.Exit(0);  break;  }  }  private static void ShowMenu(string[] options)  {  Console.Write("\n\n Menu: \n");  //show the menu to the user  for (int i =0;i<options.Length;i++)  {  Console.WriteLine(i+1 + "." + options[i]);  }  }  }  } |

**Rectangle.cs**

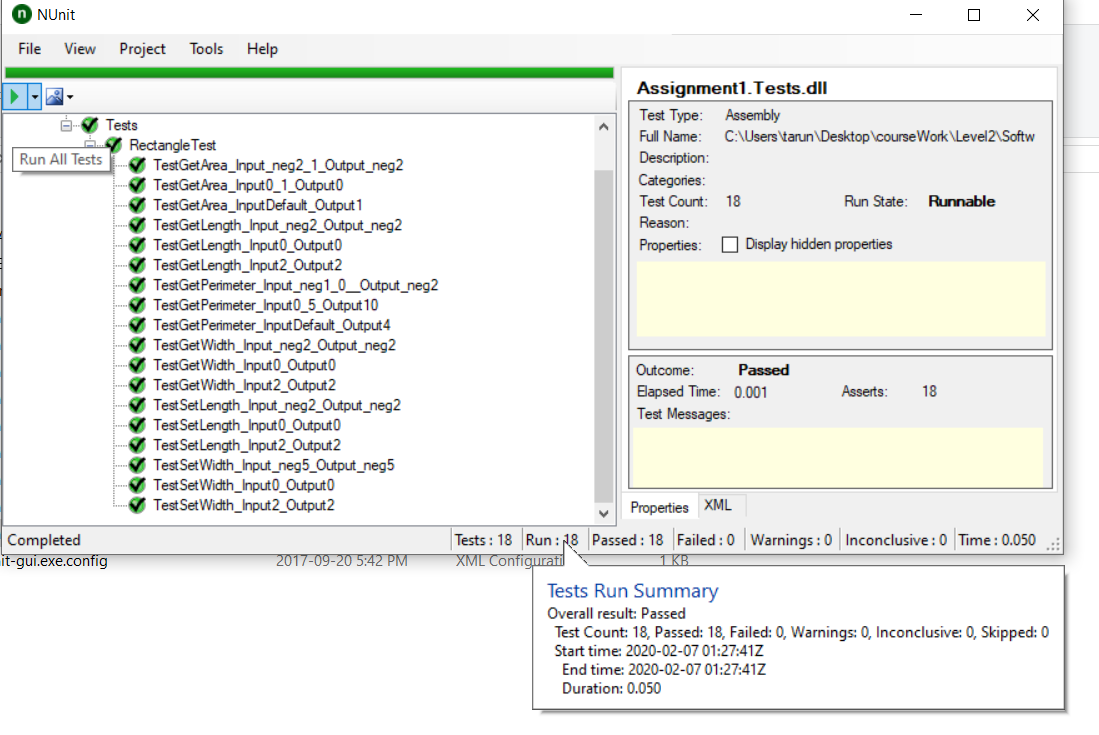
|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Assignment1  {  public class Rectangle  {  //variables to store attributes of a rectangle  private int length;  private int width;  //default constrcutor  public Rectangle()  {  length = 1;  width = 1;  }  //parametarised constructor  public Rectangle(int length, int width)  {  this.length = length;  this.width = width;  }    //getters  public int GetLength()  {  return this.length;  }  public int GetWidth()  {  return this.width;  }  //setters  public int SetLength(int length)  {  this.length = length;  return this.length;  }  public int SetWidth(int width)  {  this.width = width;  return this.width;  }    //member functions  //get the perimeter of the rectangle  public int GetPerimeter()  {  return 2 \* (this.length + this.width);  }  //get the area of the retangle  public int GetArea()  {  return this.length \* this.width;  }  }  } |

**RectangleTest.cs**

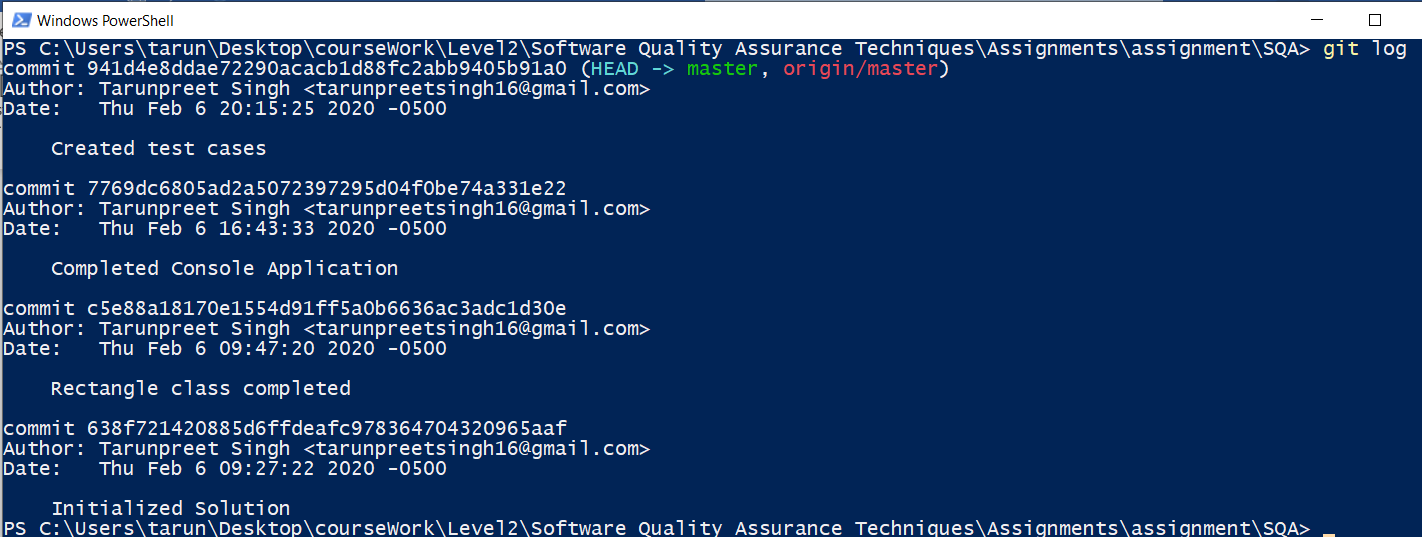
|  |
| --- |
| using NUnit.Framework;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Assignment1.Tests  {  [TestFixture]  class RectangleTest  {  Rectangle rectangle;  /\*Test cases for GetPerimeter() method\*/  /\*Test case 1  \* Revision : 1.0  \* Description : To test whether the perimeter function is returning the correct value  \* Input : Default => length = 1, width = 1  \* Expected : 4  \*/  [Test]  public void TestGetPerimeter\_InputDefault\_Output4()  {  rectangle = new Rectangle();  int perimeter = rectangle.GetPerimeter();  Assert.AreEqual(4,perimeter);  }  /\*Test case 2  \* Revision : 1.0  \* Description : To test whether the perimeter function is returning the correct value  \* when length = 0  \* Input : length = 0, width = 5  \* Expected : 10  \*/  [Test]  public void TestGetPerimeter\_Input0\_5\_Output10()  {  rectangle = new Rectangle(0,5);  int perimeter = rectangle.GetPerimeter();  Assert.AreEqual(10,perimeter);  }  /\*Test case 3  \* Revision : 1.0  \* Description : To test whether the perimeter function is returning  \* the correct value when passed with engative values  \* Input : length = -5, width = 0  \* Expected : -2  \*/  [Test]  public void TestGetPerimeter\_Input\_neg1\_0\_\_Output\_neg2()  {  rectangle = new Rectangle(-1,0);  int perimeter = rectangle.GetPerimeter();  Assert.AreEqual(-2,perimeter);  }  /\*Test cases for GetArea() method\*/  /\*Test case 1  \* Revision : 1.0  \* Description : To test whether the area function is returning the correct value  \* when user isn't entering anything  \* Input : Default => length = 1, width = 1  \* Expected : 1  \*/  [Test]  public void TestGetArea\_InputDefault\_Output1()  {  rectangle = new Rectangle();  int area = rectangle.GetArea();  Assert.AreEqual(1, area);  }    /\*Test case 2  \* Revision : 1.0  \* Description : To test whether the area function is returning the correct value  \* when the length = 0  \* Input :length = 0, width = 1  \* Expected : 0  \*/  [Test]  public void TestGetArea\_Input0\_1\_Output0()  {  rectangle = new Rectangle(0,1);  int area = rectangle.GetArea();  Assert.AreEqual(0, area);  }    /\*Test case 3  \* Revision : 1.0  \* Description : To test whether the area function is returning the correct value  \* when negative values are passed  \* Input : length = -2, width = 1  \* Expected : -2  \*/  [Test]  public void TestGetArea\_Input\_neg2\_1\_Output\_neg2()  {  rectangle = new Rectangle(-2,1);  int area = rectangle.GetArea();  Assert.AreEqual(-2, area);  }  /\*Test cases for SetLength() method\*/  /\*Test case 1  \* Revision : 1.0  \* Description : To test whether the SetLength() method is working fine or not by passing positive value  \* Input : length = 2  \* Expected : 2  \*/  [Test]  public void TestSetLength\_Input2\_Output2()  {  rectangle = new Rectangle();  int lengthSet = rectangle.SetLength(2);  Assert.AreEqual(2, lengthSet);  }  /\*Test case 2  \* Revision : 1.0  \* Description : To test whether the SetLength() method is working fine or not by passing negative value  \* Input : length = -5  \* Expected : -5  \*/  [Test]  public void TestSetLength\_Input\_neg2\_Output\_neg2()  {  rectangle = new Rectangle();  int lengthSet = rectangle.SetLength(-5);  Assert.AreEqual(-5, lengthSet);  }  /\*Test case 3  \* Revision : 1.0  \* Description : To test whether the SetLength() method is working fine or not by passing 0  \* Input :length = 0  \* Expected : 0  \*/  [Test]  public void TestSetLength\_Input0\_Output0()  {  rectangle = new Rectangle();  int lengthSet = rectangle.SetLength(0);  Assert.AreEqual(0, lengthSet);  }    /\*Test cases for SetWidth() method\*/  /\*Test case 1  \* Revision : 1.0  \* Description : To test whether the SetWidth() method is working fine or not by passing positive value  \* Input : width = 2  \* Expected : 2  \*/  [Test]  public void TestSetWidth\_Input2\_Output2()  {  rectangle = new Rectangle();  int widthSet = rectangle.SetWidth(2);  Assert.AreEqual(2, widthSet);  }  /\*Test case 2  \* Revision : 1.0  \* Description : To test whether the SetWidth() method is working fine or not by passing negative value  \* Input : width = -5  \* Expected : -5  \*/  [Test]  public void TestSetWidth\_Input\_neg5\_Output\_neg5()  {  rectangle = new Rectangle();  int widthSet = rectangle.SetWidth(-5);  Assert.AreEqual(-5, widthSet);  }  /\*Test case 3  \* Revision : 1.0  \* Description : To test whether the SetWidth() method is working fine or not by passing 0  \* Input :width = 0  \* Expected : 0  \*/  [Test]  public void TestSetWidth\_Input0\_Output0()  {  rectangle = new Rectangle();  int widthSet = rectangle.SetWidth(0);  Assert.AreEqual(0, widthSet);  }  /\*Test cases for GetLength() method\*/  /\*Test case 1  \* Revision : 1.0  \* Description : To test whether the GetLength() method is working fine or not by passing positive value  \* Input : length = 2  \* Expected : 2  \*/  [Test]  public void TestGetLength\_Input2\_Output2()  {  rectangle = new Rectangle();  rectangle.SetLength(2);  int length = rectangle.GetLength();  Assert.AreEqual(2, length);  }  /\*Test case 2  \* Revision : 1.0  \* Description : To test whether the GetLength() method is working fine or not by passing negative value  \* Input : length = -2  \* Expected : -2  \*/  [Test]  public void TestGetLength\_Input\_neg2\_Output\_neg2()  {  rectangle = new Rectangle();  rectangle.SetLength(-2);  int length = rectangle.GetLength();  Assert.AreEqual(-2, length);  }  /\*Test case 3  \* Revision : 1.0  \* Description : To test whether the GetLength() method is working fine or not by passing 0  \* Input : length = 0  \* Expected : 0  \*/  [Test]  public void TestGetLength\_Input0\_Output0()  {  rectangle = new Rectangle();  rectangle.SetLength(0);  int length = rectangle.GetLength();  Assert.AreEqual(0, length);  }  /\*Test cases for GetWidth() method\*/  /\*Test case 1  \* Revision : 1.0  \* Description : To test whether the GetWidth() method is working fine or not by passing positive value  \* Input : width = 2  \* Expected : 2  \*/  [Test]  public void TestGetWidth\_Input2\_Output2()  {  rectangle = new Rectangle();  rectangle.SetWidth(2);  int width = rectangle.GetWidth();  Assert.AreEqual(2, width);  }  /\*Test case 2  \* Revision : 1.0  \* Description : To test whether the GetWidth() method is working fine or not by passing negative value  \* Input : length = -2  \* Expected : -2  \*/  [Test]  public void TestGetWidth\_Input\_neg2\_Output\_neg2()  {  rectangle = new Rectangle();  rectangle.SetWidth(-2);  int width= rectangle.GetWidth();  Assert.AreEqual(-2, width);  }  /\*Test case 3  \* Revision : 1.0  \* Description : To test whether the GetWidth() method is working fine or not by passing 0  \* Input : length = 0  \* Expected : 0  \*/  [Test]  public void TestGetWidth\_Input0\_Output0()  {  rectangle = new Rectangle();  rectangle.SetWidth(0);  int width = rectangle.GetWidth();  Assert.AreEqual(0, width);  }  }  } |

**Screenshots**

* **NUnit GUI with unit test cases**

****

* **Git log**

****