Student Name: Ivon Harris   
  
Student Number: 472832219

**ICTPRG418: AT1 Birthday Tracker - Documentation**

**1. Testing**

1.1 Create at least one (1) Unit test in Visual Studio based on a test case described in the table below (1.2). Select a functional test (one which tests a specific functionality that is required by the user for this application – not a trivial one).

Note: This unit test may be used for ICTPRG529 AT1

Copy + Paste your code here:

No test case is provided for the question below as this is the Birthday Tracker.

[TestMethod]

public void TestMethod1()

{

string name = "Ivon";

string likes = "Chocolate";

string dislikes = "Wine";

string birthDay = "31";

string birthMonth = "6";

bool expectedResult = false;

bool actualResult = ValidateData(name, likes, dislikes, birthDay, birthMonth);

Assert.AreEqual(expectedResult, actualResult);

}

Using the following method for test:

public bool ValidateData(string name, string likes, string dislikes, string birthDay, string birthMonth)

{

bool checkBox = true;

string errorMessage = "ERROR(S) Encountered\n";

int birthdayDay;

int birthdayMonth;

//string errorMessage = "ERROR(S) Encountered\n";

// check friend textbox

if (String.IsNullOrEmpty(name))

{

errorMessage += "Friend name is required\n";

checkBox = false;

}

// check last name text box

if (String.IsNullOrEmpty(likes))

{

errorMessage += "Likes is required\n";

checkBox = false;

}

//check dislikes text box

if (String.IsNullOrEmpty(dislikes))

{

errorMessage += "dislikes is required\n";

checkBox = false;

}

//check day of birth day text box

if (String.IsNullOrEmpty(birthDay))

{

errorMessage += "must have a birth 'day' \n";

checkBox = false;

}

//check birth month text box

if (String.IsNullOrEmpty(birthMonth))

{

errorMessage += "must have a birth 'month'\n";

checkBox = false;

}

if (checkBox == true)

{

try

{

birthdayDay = Int32.Parse(birthDay);

birthdayMonth = Int32.Parse(birthMonth);

switch (birthdayMonth)

{

case 1: case 3: case 5: case 7: case 8: case 10:

case 12:

if (birthdayDay < 1 || birthdayDay > 31)

{

errorMessage += "Check month error, only 1 - 31 days in these months \n";

checkBox = false;

}

break;

case 4: case 6: case 9: case 11:

if (birthdayDay < 1 || birthdayDay > 30)

{

errorMessage += "Check month error, only 1 - 30 days in these months \n";

checkBox = false;

}

break;

case 2:

if (birthdayDay < 1 || birthdayDay > 29)

{

errorMessage += "Check month error, only 1 - 29 days in this months \n";

checkBox = false;

}

break;

}// end switch

} // end try

catch (Exception ex)

{

errorMessage += "Birth day and month must be numeric\n";

checkBox = false;

}

}

if (checkBox == false)

{

//MessageBox.Show(errorMessage, "Errors!");

Console.WriteLine(errorMessage);

}

return checkBox;

}

[TestMethod]

public void TestMethod1()

{

string name = "Ivon";

string likes = "Chocolate";

string dislikes = "Wine";

string birthDay = "31";

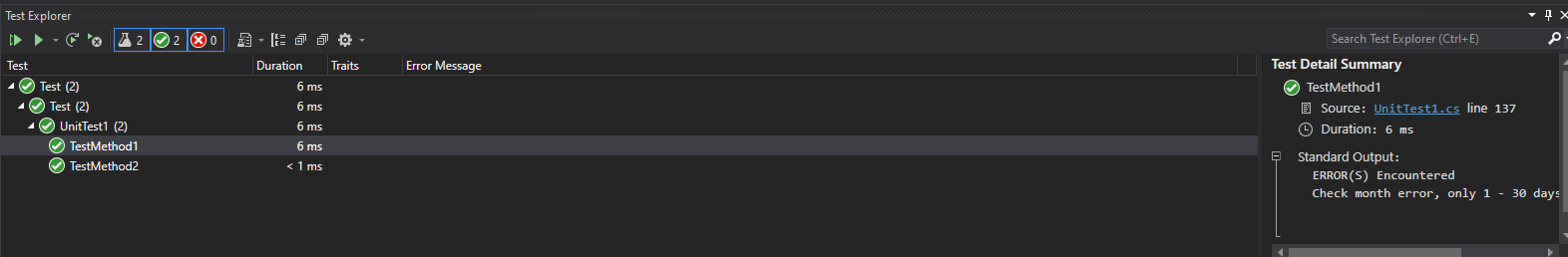
string birthMonth = "6";

bool expectedResult = false;

bool actualResult = ValidateData(name, likes, dislikes, birthDay, birthMonth);

Assert.AreEqual(expectedResult, actualResult);

}



tTest result Successfully identified that there are only 30 days in June

1.2 Classroom Planner Test Case

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID#** | **Test Case Description** | **Test Steps** | **Test data** | **Expected Result** | **Actual Result** | **Pass**  **or**  **Fail** |
| NA | We are not testing classroom planner | Unittest can be applied on correct application ( see question) | <data for use in the test> |  |  |  |

**Software**: Visual Studio IDE

**Hardware** used for the test:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **OS Version** | **Device Model** | **Screen Resolution** | **Processor** | **Processor Architecture** | **RAM** | **GPU** |
| Windows 11 |  |  |  |  |  |  |

**2. Documentation**

2.1 Provide a README.md file of your software (use the provided template and modify to suit). You will need to provide the text file separately to this document as part of your zip archive submission.

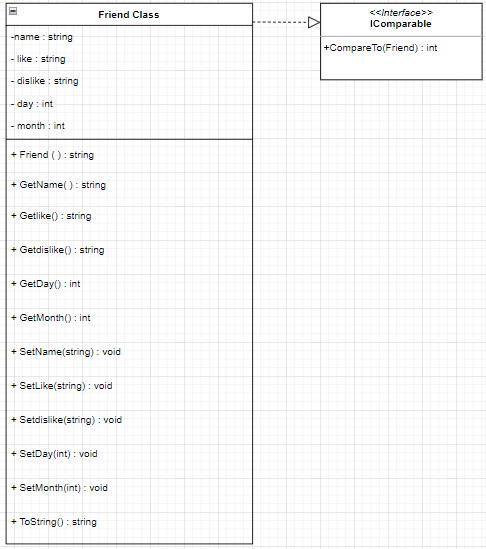
Provided as a file in project Readme.txt

2.2 Provide a copy of your MIT License for this software below (complete with your copyright details).

Provided in documentation

License.txt

2.3 Provide a UML class diagram of your Friend class (or alternative) used in the development of this application. Ensure that you correctly display all private data fields and public properties and methods for the class, including the IComparable interface.



2.4 Provide the XML file generated from within Visual Studio for the class in 2.3 above - this must include all public properties and public methods of your classes. You will need to provide the XML file separately to this document as part of your zip archive submission. As an option, you may wish to include a suitable stylesheet file so that the XML document can be viewed as a HTML file in a standard browser.

The XML document is located in the project folder.

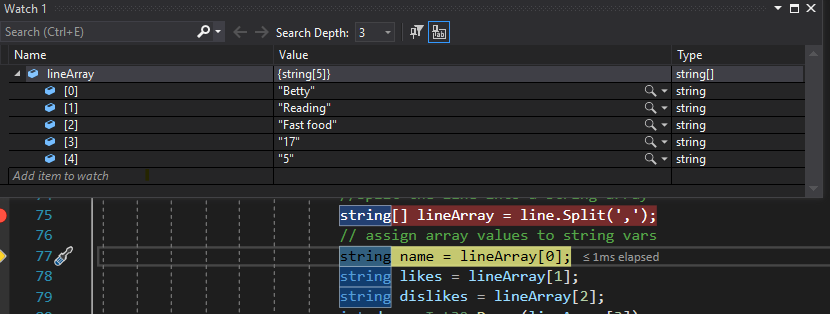
**3. Debugging**

3.1 Briefly describe what you have chosen to use as your debugging example.

Place breakpoint in line 75 at where the lineArray is created from the read-in CSV file

Checking the read of external file and breaking up the comma values

3.2 Provide a screen shot of the breakpoint(s) in Visual Studio



3.3. Provide a screen shot of the associated watches (watch variables you have chosen to keep a track of)

