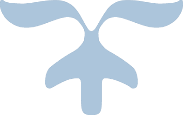
Arithmetic Game



Contents

[Toe Chart 1](#_Toc524973854)

[UML Diagram 2](#_Toc524973855)

[Debugging Evidence 3](#_Toc524973857)

[Sorting Algorithms 4](#_Toc524973858)

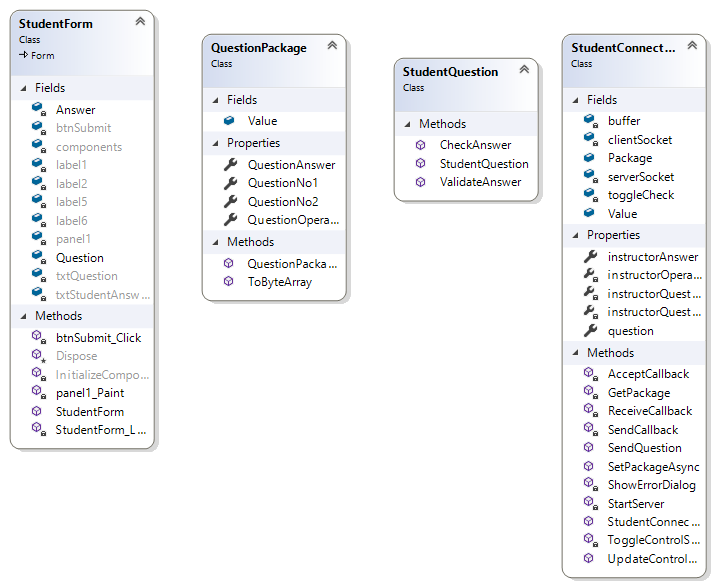
[Version Control 5](#_Toc524973859)

# Toe Chart

**Toe Chart** listing the applications functions(tasks), the object used to finish the tasks, and the event involved.

|  |  |  |
| --- | --- | --- |
| **Task** | **Object** | **Event** |
| **Get/Set Question** |  |  |
| Get/Set Number 1 | **txtQuestion1.Text** | **txtQuestion1Change** |
| Get/Set Number 2 | **txtQuestion2.Text** | **txtQuestion2Change** |
| Get/Set operator | **cmboOperator** | **txtQuestion1 + 2 Change** |
|  |  |  |
| **Determine operator** |  |  |
| Index 1 = + index 2 = - etc. Etc | **cmboOperator** | **cbmboOperator Index change** |
|  |  |  |
| **Calculate Answer** |  |  |
| Number 1 (+ - x /) Number 2 | **txtAnswer.Text** | **txtQuestion1 + 2 Change** |
|  |  |  |
| **Submit Question** |  |  |
| Send question to student application | **New InstructorConnection Class** | **btnSubmit.Click** |
| Student application to grab question | **New StudentConnection Class** | **StudentForm.Load** |
|  |  |  |
| **Display Question to Student** |  |  |
| Question received and displayed | **txtStudentQuestion.Text** | **StudentForm.Load** |
|  |  |  |
| **Add Sent Question to DataGrid** |  |  |
| Add question to DataTable | **dataTableList** | **btnSubmit.Click** |
| DataSource for grid = DataTable | **dataGridViewQuestionList** | **dataGridViewQuestionList** |
|  |  |  |
| **Send Question to Binary Tree** |  |  |
| Convert Question to Binary Tree | **New BinaryTree class** | **InstructorForm.Intitialzed** |
| Display Binary tree in one of 3 orders | **btnInOrder, btnPreOrder, BtnPostOrder** | **btnInOrder.Click btnPreorder.Click btnPostOrder.Click** |

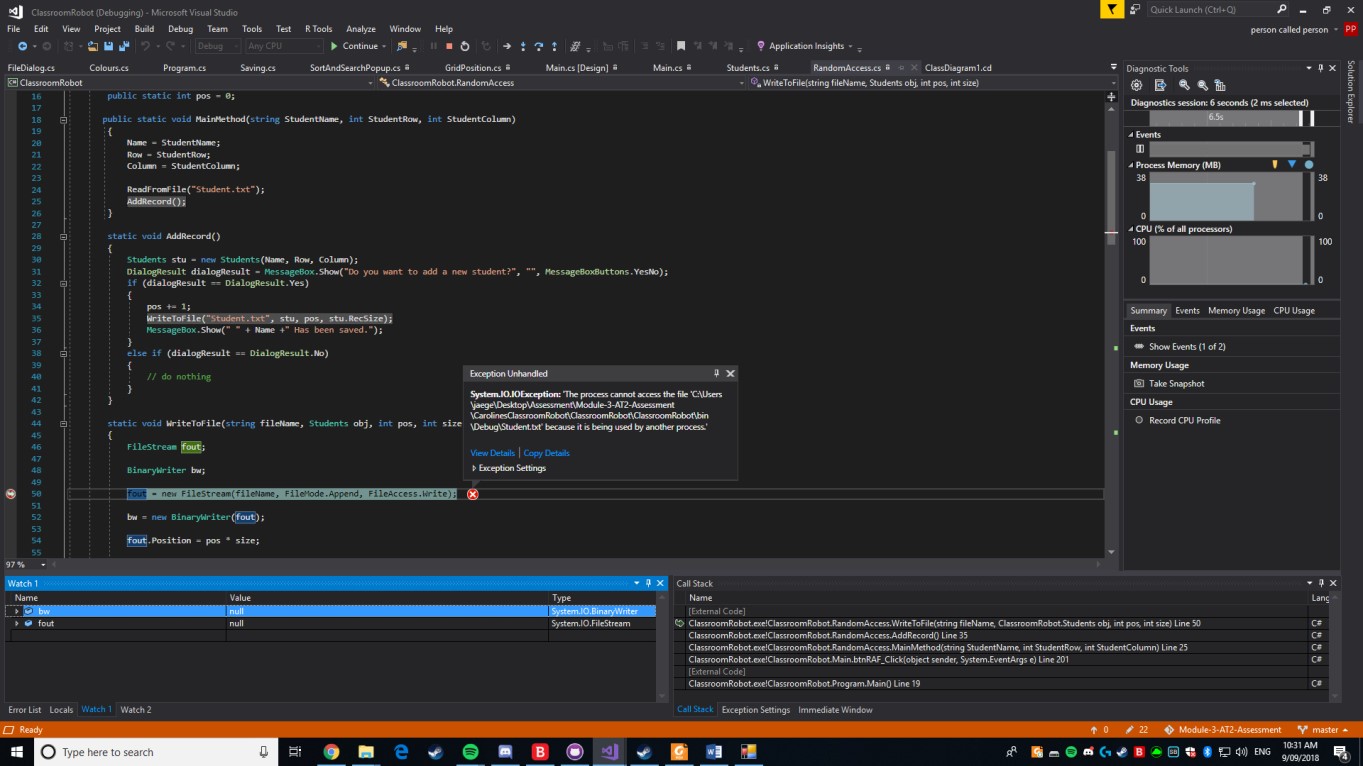
# UML Diagram

**Provided** UML Class Diagrams showing all classes, fields, properties and methods for both projects.

# 

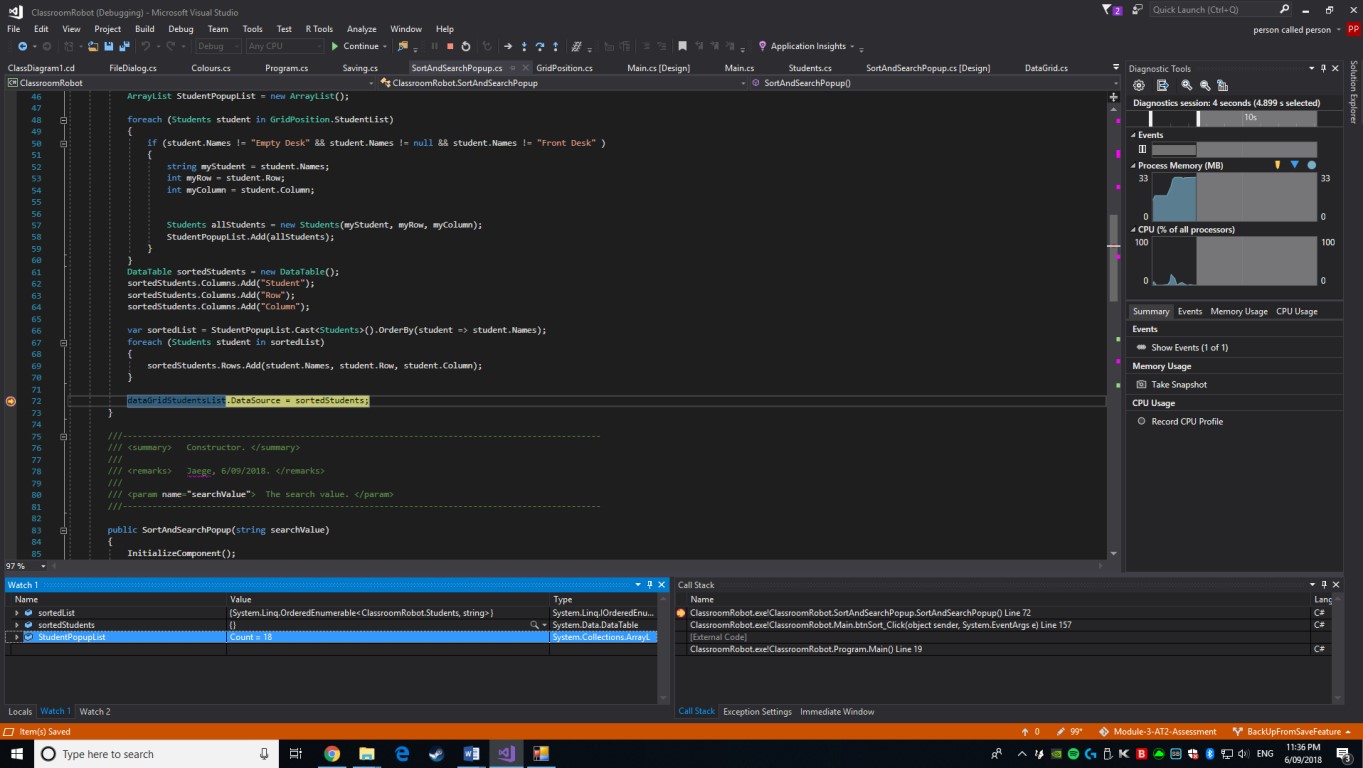
# Debugging Evidence

Evidence of debugging



Handling exception, by placing watches and running through the code to determine where the fault lay.

Checking that the variable is receiving a value.



Checking that the variable is receiving a value.

# Sorting Algorithms

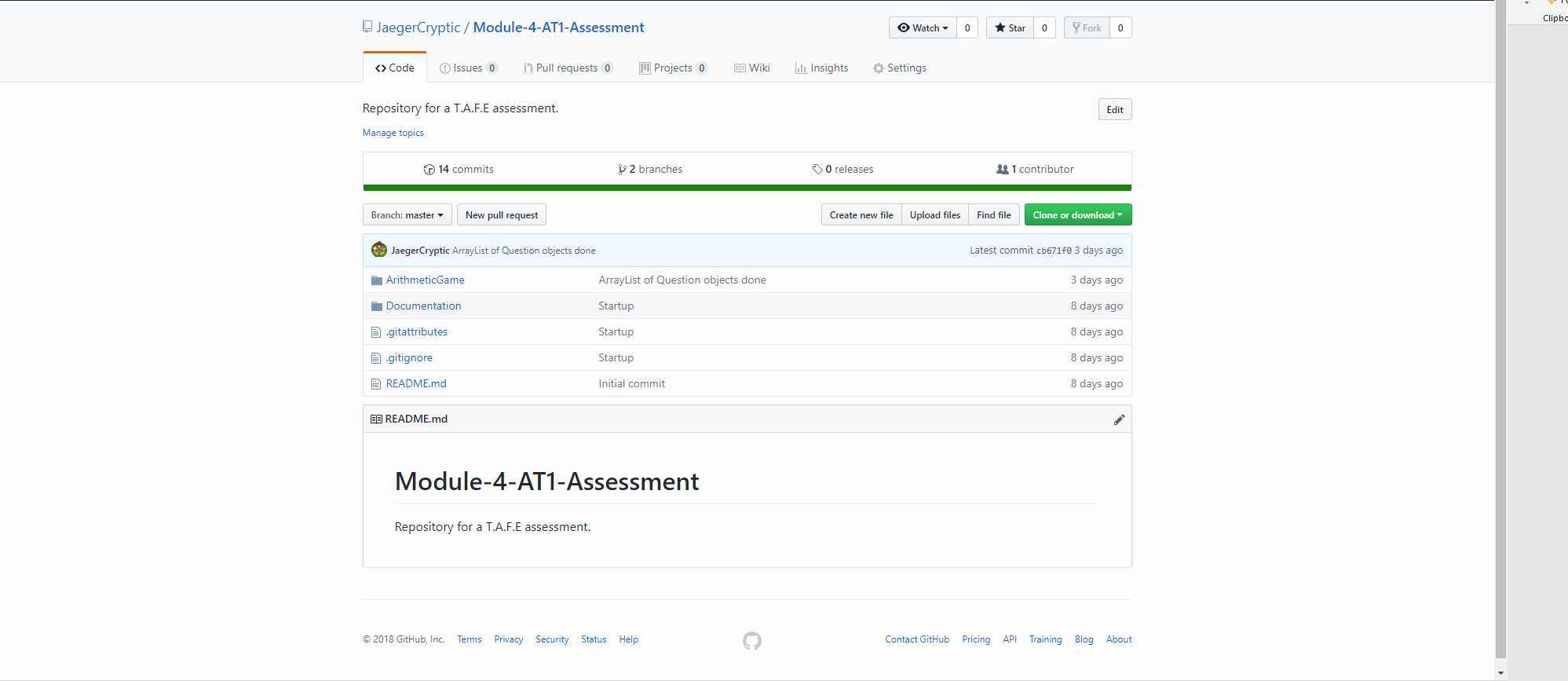
**The Bubble Sort** is considered one of the worst sorting algorithms available. This doesn’t mean it’s useless though. For one, the Bubble Sort, just like Insertion Sorts and Selection Sorts are some of the easiest Sorting algorithms to understand and utilize. While the Bubble Sort does not sort to the same effect as other Sorting methods, it is still useful. It’s best used in small to medium sized item lists, starting to lose its ability the larger the item list. A Bubble Sort works by comparing the first two values in a data set. When it compares the values if it finds that the first has a greater value than the second it swaps the values. This is repeated until the data set is sorted.

**An Insertion Sort** is another type of sort and just like the Bubble Sort; is a very basic sorting algorithm, meaning it loses effect the larger the item list. The Insertion sort shines in small data sets and unlike more advanced algorithms, this algorithm only takes a couple of lines to code. In almost all cases an Insertion sort beats a Bubble Sort. Only slightly requiring more code. An Insertion Sort works by taking the values from the selected data set and then, sorting them by value in a new data set. Another disadvantage to the Insertion Sort can be that the sorting algorithm can be taxing compared to other basic sorting algorithms.

**Lastly, the Selection Sort.** This algorithm is another basic algorithm and suffers from the same drawbacks the other basic algorithms do. While the Selection sort is considered better than a Bubble Sort in terms of efficiency, it stills fails to compare with an Insertion Sort. A Selection sort works by finding the lowest value in a data set and swaps it with the first value. This is repeated until the Selection Sort goes through the entire data set.

# Version Control

**Version Control** was used throughout this project and can be seen [Here:](https://github.com/JaegerCryptic/Module-4-AT1-Assessment)

**GitHub** in conjunction with GitHub Desktop was used while developing this project. All progress was committed and stored with the version control and backups were continuously updated and merged, ensuring the master product was kept intact.