|  |
| --- |
|  |
| Question 3  Assessment Task 2 |
| |  |  |  | | --- | --- | --- | | Ben Royans [P205225] | 3/5/20 | Programming III | |

TABLE OF CONTENTS

[Design Specifications 1](#_Toc34333118)

[Specification 1](#_Toc34333119)

[Review 1](#_Toc34333120)

[UML 2](#_Toc34333121)

[Debugging 3](#_Toc34333122)

[Testing 5](#_Toc34333123)

[Test Results 5](#_Toc34333124)

[Test Result Screenshots 6](#_Toc34333125)

[Informational Repositories 12](#_Toc34333126)

[BlackBoard 12](#_Toc34333127)

[GitHub 12](#_Toc34333128)

[Performance Screenshots 13](#_Toc34333129)

[References 15](#_Toc34333130)

# Design Specifications

## Specification

An application is required to sort the salaries of staff members. It must have the following functionalities:

* List of integers (salaries)
  + Integers must be between the value of 10,000 and 10,000,000.
  + Capacity to hold more than 1,000,000 items.
* Sorting Methods
  + 3+ Unique sorting methods.
  + Only 1 may be built in.
  + Includes timer to time the execution time of the sorting method.

## Review

A simple GUI will be constructed using WPF to launch the sorting methods. Each sorting method should be contained to it’s own class.

# UML

Below are the UML class diagrams for the classes used in this project.

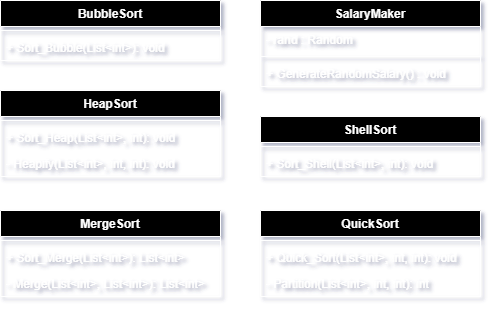


Figure 1. UML class diagrams.

# Debugging

Below are some screenshots showing the debugging process when implementing error checking on the salary amount input field.

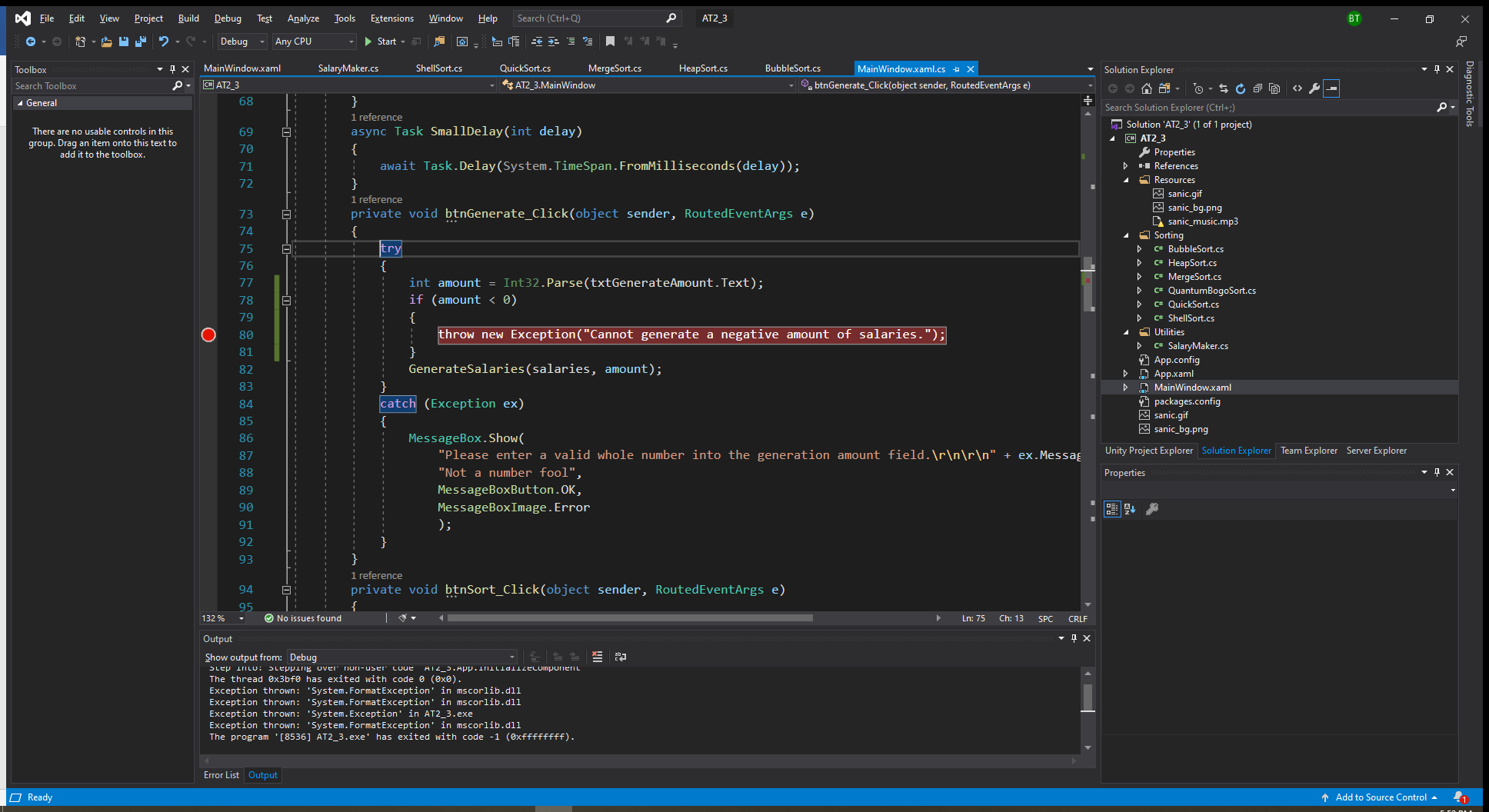


Figure 2. Placing a breakpoint on the newly implemented error checking mechanism.

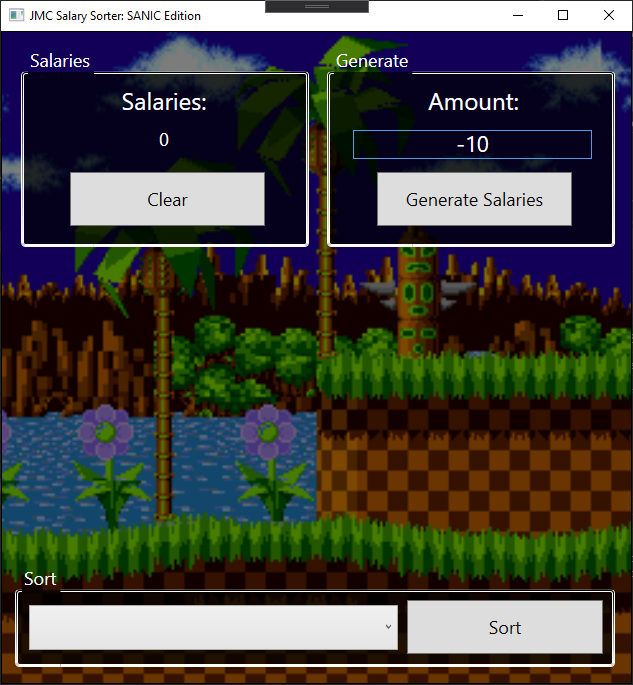


Figure 3. Using a negative number of salaries to generate.

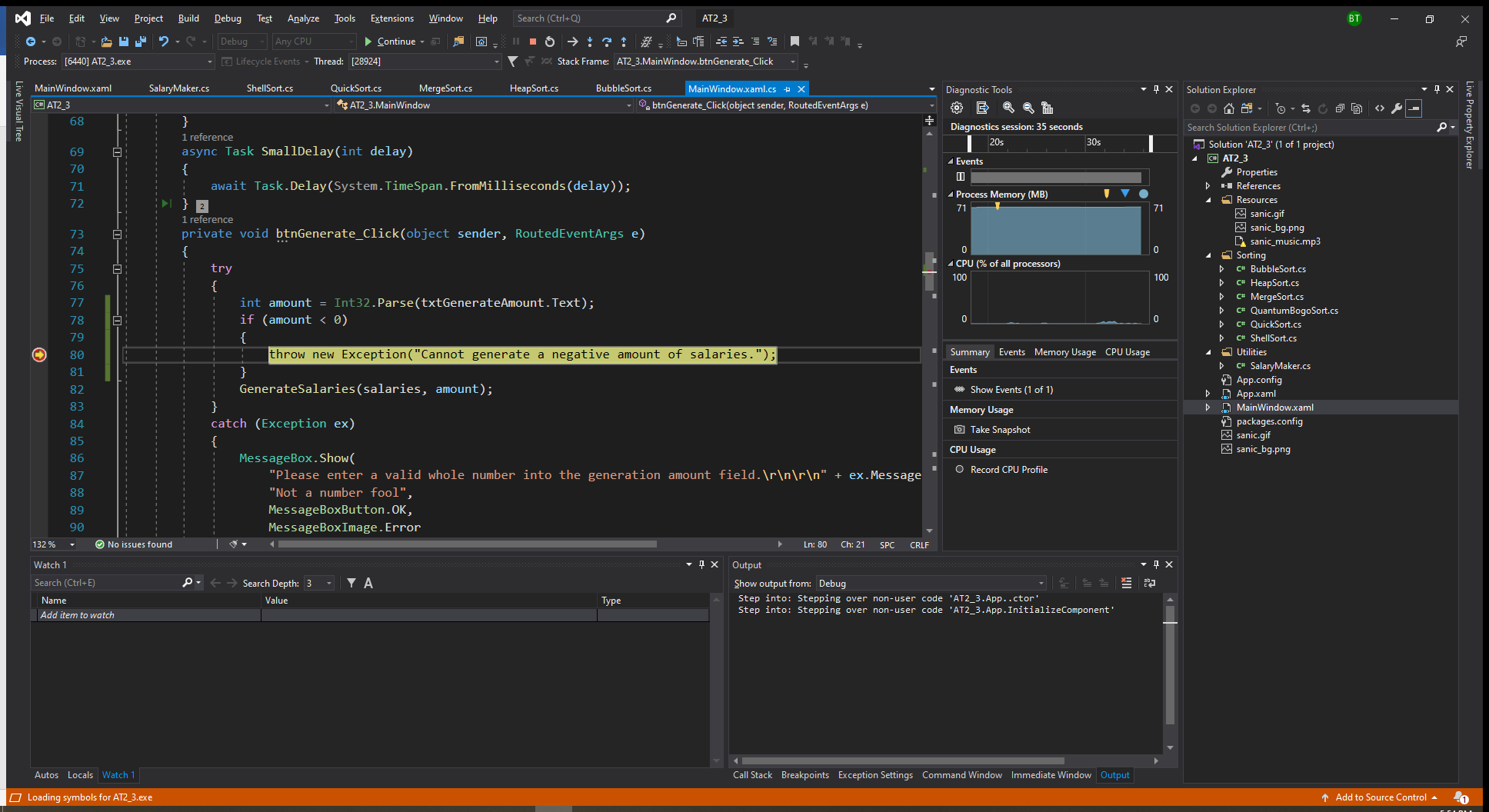


Figure 4. The breakpoint shows the code is being executed as intended.

# Testing

## Test Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Case #** | **Functionality Tested** | **Data/Procedure**  **Used** | **Expected**  **Outcome** | **Pass / Fail** |
| 1 | Sort Button | No sorting method selected. | Error message. | PASS |
| 2 | Sort Button | Merge sort selected. | Sorted dialog message containing time taken to complete. | PASS |
| 3 | Sort Button | Heap sort selected. | Sorted dialog message containing time taken to complete. | PASS |
| 4 | Sort Button | Quick sort selected. | Sorted dialog message containing time taken to complete. | PASS |
| 5 | Sort Button | Shell sort selected. | Sorted dialog message containing time taken to complete. | PASS |
| 6 | Sort Button | Bubble sort selected. | Sorted dialog message containing time taken to complete. | PASS |
| 7 | Sort Button | Quantum Bogo sort selected. | Sorted dialog message containing time taken to complete. | PASS |
| 8 | Generate Salaries Button | A positive whole number used in input field. | Salaries generated. | PASS |
| 9 | Generate Salaries Button | A character used in input field. | Error message. | PASS |
| 10 | Generate Salaries Button | A negative whole number used in input field. | Error message. | PASS |
| 11 | Generate Salaries Button | A decimal number used in input field. | Error message. | PASS |
| 12 | Clear Button | Salaries count is above 0. | Salaries reduced to 0. | PASS |
| 13 | Clear Button | Salaries count is 0. | Salaries reduced to 0. | PASS |

## Test Result Screenshots

|  |  |
| --- | --- |
| **Case #** | **Screenshot** |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |

# Informational Repositories

## BlackBoard

The assessment specifications and requirements have been supplied through BlackBoard. BlackBoard also provides the learning content used for this assessment task and will provide the methods for deployment (handing in the assessment).

## GitHub

GitHub will be used as version control software for the project. Regular and systematic commits to either the master or branches of the code will provide a safety net for unexpected errors on the working application.

Public Link:

# Performance Screenshots



Figure 5. After initialisation.



Figure 6. During a Bubble sort of 30,000 salaries.

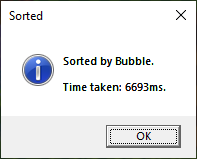


Figure 7. Dialog shown on completion of the sorting.

# References

Boyini, K. (2018, July 24). *Bubble Sort program in C#*. Retrieved from tutorialspoint: https://www.tutorialspoint.com/Bubble-Sort-program-in-Chash

John, G. (2018, Novemeber 23). *Shell Sort program in C#*. Retrieved from tutorialspoint: https://www.tutorialspoint.com/shell-sort-program-in-chash

Reddy, A. (2018, November 23). *Heap Sort in C#*. Retrieved from tutorialspoint: https://www.tutorialspoint.com/heap-sort-in-chash

Shanker, A. (2019, April 6). *Merge Sort Algorithm in C#*. Retrieved from C# Corner: https://www.c-sharpcorner.com/blogs/merge-sorting-algorithm-in-c-sharp1

Sharma, A. (2017, August 31). *Quick Sort Algorithm In C#*. Retrieved from C# Corner: https://www.c-sharpcorner.com/blogs/quick-sort-algorithm-in-c-sharp

w3resource. (2020, February 26). *C# Sharp Searching and Sorting Algorithm Exercises: Merge sort*. Retrieved from w3resource: https://www.w3resource.com/csharp-exercises/searching-and-sorting-algorithm/searching-and-sorting-algorithm-exercise-7.php