# GlobalX Sorting Assessment — Tech Spec Document

This document describes the solution implemented here as a solution for the GlobalX recruiting assessment.

## Name Sorting Problem

The problem to be solved is to sort a list of names by Family Name and then any of one to three possible Given Names. Input to the program is a list of full names, each of the form:

[first given name] [second given name] [third given name] [family name]

The second and third given names are optional.

## The Solution

The solution provided solves the problem by reordering the parts of the full name so that a string comparison can be invoked on the full name where the family name appears before the given names. This is the simplest solution to the problem.

## Problems and Limitations

One problem with the program rests in the fact that the list of family and given names are *space* separated, but some family names have a space such as “von Neumann”. This semantic feature cannot be represented in the input format chosen. We ignore this limitation.

## Sorting Algorithm

Two sorting methods are available in the program. The default sorting method uses the classic Quick Sort algorithm. Code for this Quick Sort is included in the static generic class QuickSort<T>.

A custom QuickSort class was implemented making available the ability sort an array of names using the classic QuickSort algorithm.

## Execution Options

The default sorting algorithm used is the .NET Array Sort algorithm. However an additional custom QuickSort algorithm is supplied in a static class called QuickSort. The QuickSort algorithm can be invoked by adding an additional command option “quicksort” as a second optional command argument.

Usage

name-sorter <filename>

Optional Algorithms

quicksort QuickSort algorithm

## Solution Design

The solution is named GlobalX.Coding.Assessment.

The solution consists of seven class including the main Program class.

Program

Name

NameSorter

ArrayNameSorter

QuickNameSorter

QuickSort

The main Program class validates command arguments and prints usage info when invalid. If the command line is valid, the Program class instantiates either the ArrayNameSorter class or the QuickNameSorter class.

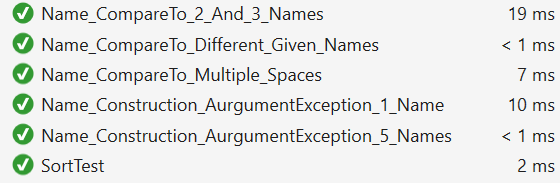
The Name class implements the IComparable interface and the CompareTo method for this interface allows sorting methods to invoke a comparison based on a reordering of FullName, so that Family Name is the first name to be ordered on then first given name, second given name etc. This reordering of the full name is exposed as a method called ToOrderedSpelling and this ordered spelling of the full name is what makes it possible to sort an array of Name objects based on the stipulated sorting criteria.

## Tests

There are 6 tests targeting the comparison operation and constructor of the Name class.

The constructor should throw an ArgumentException when the full name consists of less than 2 or more than 4 names. The constructor should also deal with names that have multiple spaces by trimming multiple spaces to a single space.

The CompareTo operation on the Name class should return -1 when the first operand alphabetically precedes the second, 0 when they are equal and 1 when the second operation alphabetically precedes the first.



## Public GIT Repository

The GIT repository for the solution can be found at

<https://github.com/CognitiveFeedback/globalx.coding.assessment>

or by simply cloning the repository using

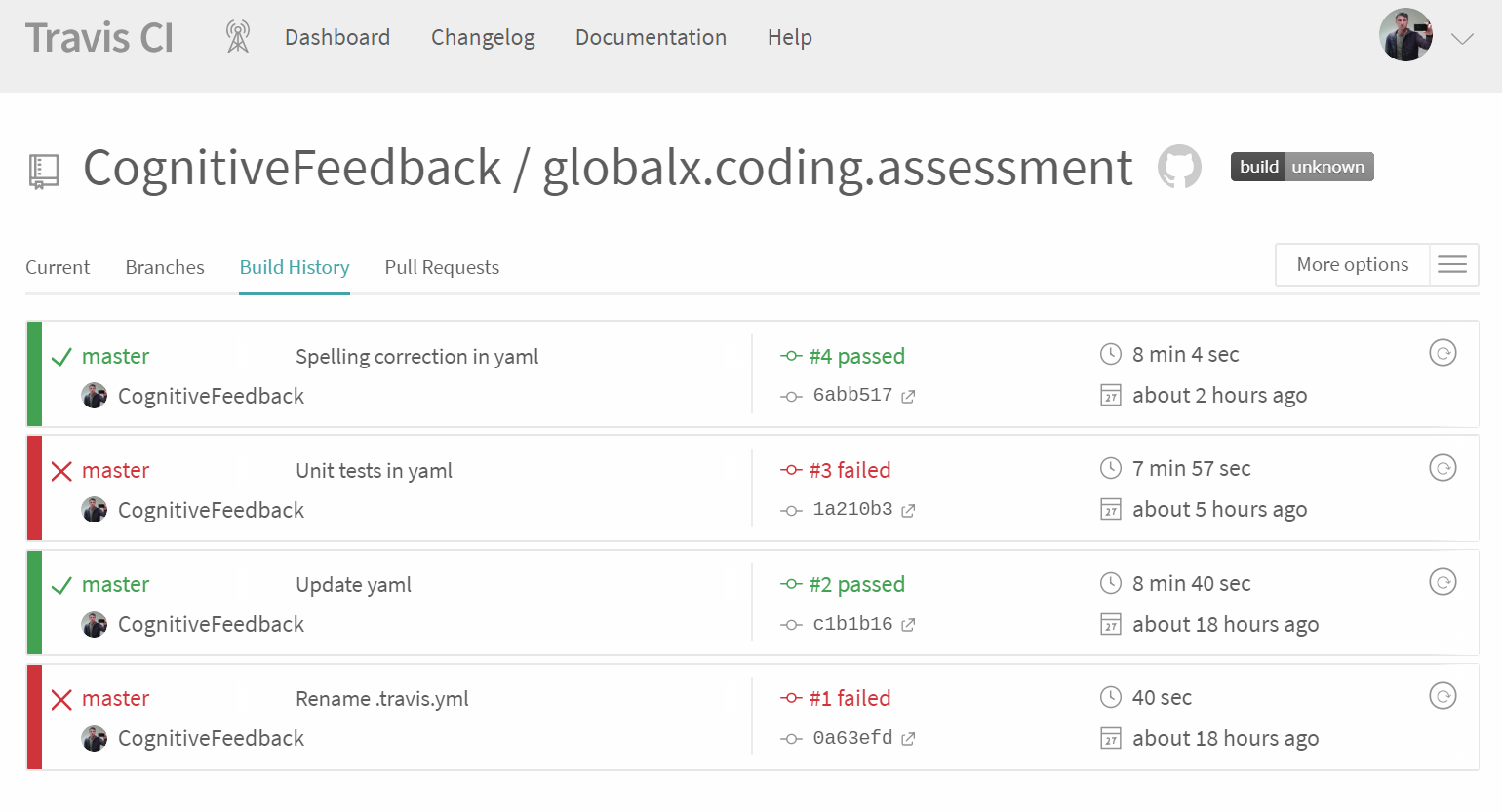
<https://github.com/CognitiveFeedback/globalx.coding.assessment.git>

## Build Pipeline in Travis CI

The Travis CI build pipeline can be found at

<https://travis-ci.org/CognitiveFeedback>

The build pipeline executes unit tests and will fail if not all unit tests pass.



## Output

The output is written to the console and also to a file named sorted-names-list.txt.