This coding test is a **design** *evaluation*. It does NOT matter if you are a fresh graduate from school or you have many years of experience working as programmers.

The question in this test is simple, and you can easily write code that meets the requirement, eg. produce correct output/result.

**BUT** our goal is ***NOT*** to see you submit code that just works. More importantly for us is to see if you can follow the instruction carefully; your code must be clearly written and with *empathy* to your potential team members.

So, what do we mean by *empathy*?   
Empathy here is you *care* about how easy your code is to be *understood* by the next programmer who reads it.

Before you continue reading the question in this test, please take a good time to look at:

Hint #1: **Object Oriented Design (OOP)**.   
Demonstrate your ability to compose a quality code that follows *SOLID* principles. <https://en.wikipedia.org/wiki/SOLID> (make sure you read this first!)

HINT #2:   
**A *good* programmer** can complete this test with a correct solution. ***A very* *good* programmer** can explain **why** he/she wrote it that way.

# The Goal: Name Sorter

Build a name sorter. Given a set of names, order that set first by last name, then by any given names the person may have. A name must have at least 1 given name and may have up to 3 given names.

*Example Usage*

Given a file called *unsorted-names-list.txt* containing the following list of names;

Orson Milka Iddins  
Erna Dorey Battelle  
Flori Chaunce Franzel  
Odetta Sue Kaspar  
Roy Ketti Kopfen  
Madel Bordie Mapplebeck  
Selle Bellison  
Leonerd Adda Mitchell Monaghan  
Debra Micheli  
Hailey Avie Annakin

Executing the program in the following way;

name-sorter ./unsorted-names-list.txt

Should result the sorted names to screen;

Hailey Avie Annakin  
Erna Dorey Battelle  
Selle Bellison  
Flori Chaunce Franzel  
Orson Milka Iddins  
Odetta Sue Kaspar  
Roy Ketti Kopfen  
Madel Bordie Mapplebeck  
Debra Micheli  
Leonerd Adda Mitchell Monaghan

and also put the results into a file in the working directory called *sorted-names-list.txt*.

# Assessment Criteria

We will execute your submission against a list with a thousand names. Your submission must meet the following criteria:

* The solution should be available for review on *github*.
* It must read the unsorted list of names from a file called *unsorted-names-list.txt*
* The names should be sorted correctly.
* It must print the sorted list of names to screen.
* It must write/overwrite the sorted list of names to a file called *sorted-names-list.txt*
* **Unit tests** must exist.
* Minimal, practical documentation is needed.

*Awesome, but not mandatory criteria:*Create a build pipeline like Travis or AppVeyor or Azure DevOps that execute build and test steps.

# Submission

When you are done let us know the URL of the repo, be prepared to answer any follow up questions about your **design and solution**.

Don’t feel *overwhelmed* by this test, give your best try that makes you proud to say…

“*I wrote this code*”

Good luck!