## Problem

You are given a stack of boarding cards for various transportations that will take you from a point A to point B via several stops on the way. All of the boarding cards are out of order and you don't know where your journey starts, or where it ends. Write an API that lets you sort this kind of list

## Assumption/Understanding

I am given a stack of boarding cards for different means of transportation around the world. All these cards may be completely different from each other in terms of content, details, language etc But how ever all these cards must have an arrival and departure location. To make things simpler I assume that the arrival and departure locations will be names of cities only.

These stack of cards need to sorted only based on the arrival location and departure location. The sorted cards then need to be displayed to the user.

Since this is just an API I will display the result just by typing a command in the command prompt and displaying the result.

## Tacking the problem

Below are list of how I have tackled the problem

* Create an interface and declare all methods that are necessary to have the boarding cards work correctly .Therefore if anyone wants to create a boarding card class it must implement this interface
* Abstract Class: For this problem I know that the sort will be done only on the departure location and the arrival location. These departure location and arrival location could be entered in any language .But I need a way to map same locations entered in different languages to the same place. To do that I write these methods in the abstract class. This abstract class will implement the interface, define all the methods it has to and keep the rest abstract so that they can be defined later in the Boarding classes that extend this abstract class.
* Boarding Classes: Creating any boarding class must extend the abstract class which will in turn get all the methods declared in the interface. The created boarding class will have to define all methods declared abstract in the abstract class and can add its own methods of required
* The reason I followed the above structure is if in future there is another element that is to be added to sort for some new kinds of cards , we could easily create a new abstract class that extends this current abstract class ,add the new element in the sort and all these new cards can extend the new abstract class.
* Secondly if the way locations are defined changes (location is needed for sorting) .Then we can easily do it in the abstract class and all the classes will be changed
* Also if there is a requirement to create a completely new API. Then the new programmer can just implement the interface and define all the methods in his own way
* [Code documents can be found here](doc/namespaces/Cls.html)

## Testing of the code

I am use a test driven development (TDD) and therefore I have defined the outputs of different sections of my code before I start the development. (PHPUNIT testing is what I will be using)

For this project I feel there are only 3 major places I need to test the output .Since the rest are just simple getters and setters

Below is the list of what I will be testing

* Setting departure location
* Setting arrival location
* Setting the list of boarding class for the sort.

Since the arrival and departure TDD testing are the same for all the boarding card classes. I created an abstract “PHPUNIT” class that defines the unit test for the departure location and arrival location. All the unit testing classes for each boarding class must extend this abstract class. ([Code documentation can be found here](doc/namespaces/Test.html))

## Sorting Algorithm

The sorting algorithm I have written is very similar to a bubble sort but here every next sorted element will bubble up in the list. The order of complexity for this algorithm is

Best case O (n), Average case O (n ²), Worst case O (n ²)

I considered not using a divide and conquer algorithm for the sort because of the following reasons

* Since I am doing a swap only when the departure or arrival of one card is equal to the arrival or departure of another card .The likely hood of 2 cards being sorted at lower levels of the divide and conquer tree is very un likely and the average time complexity would be case as the bubble sort but with an extra over head of building a divide and conquer tree
* Since I would be writing a recursive code the stack would be used up and since the sorting elements are classes they would be a load on the memory.
* [Code documentation can be found here](doc/classes/Cls.CardSort.html)

## Things to know

Below are a few things that you need to know to understand the code better

* Auto loading: I am using the vendor auto loading classes that I installed along with PHPUNIT. I installed these classes using composer.
* PHPUNIT: This is the unit testing library I will be using to do all my unit testing
* Directories: Below is the list of directories and what they contain
  1. Cls : All the class files
  2. Test: All the test files
  3. Doc: All the documentation that were generated using phpDocumentor
  4. Vendor: All the Auto load classes
* Files : Below are some files you may want to know about
  1. Main.php : This is the main file the will run the API (has to be run via command line)
  2. Composer.json ,composer.loc : Used for the installation of PHP UNIT
  3. Phpunit.xml : Used to run the unit tests from the test directory
  4. Readme.docx : This current file you are reading
* PHP : I used PHP version 5.6.8

## How to run

Open command prompt and go into the directory where you found this readme file.

To run the test execute the command “phpunit”

To run the API execute the command “php Main.php” or “php Main.php L” (L will give the full details)

## Code documentation

[This is the documentation generated using phpDocumentor](doc/index.html)

## Improvements

The major drawback of this current code is that it cannot support languages written in fonts other than English. To solve this problem I would consider using an external language files. In these language files we add all the cities in the correct font with a mapping code.

When the boarding card is created we load the correct language file and get the mapping code for the arrival and departure cities. These mapping codes are then used in the sorting algorithms

## Conclusion

Working on this project I tried to show case my ability to understand real world problems and use my skills as an object oriented programmer to create good software solutions. I hope my efforts were good enough to please your expectations.