```
/*
 * MergeSort.h
 *
 *  Created on: Oct 4, 2016
 *      Author: Kenny Do
 */

#ifndef MERGESORT_H_
#define MERGESORT_H_

#include "ComparatorInterface.h"

class MergeSort {
public:
    template<typename T>
    static void mergeSort(T arr[], int size, ComparatorInterface<T> * comp);
    /**
     * Precondition:
     *      size is greater than 0
     * Postcondition:
     *      Calls the helper method mergeSort
     */

private:
    template<typename T>
    static void merge(T arr[], int left, int middle, int right,
            ComparatorInterface<T> * comp);
    /**
     * Precondition:
     *      none
     * Postcondition:
     *      merges the arrays in sorted form
     */

    template<typename T>
    static void mergeSort(T arr[], int left, int right, ComparatorInterface<T> * comp);
    /**
     * Precondition:
     *      l < r
     * Postcondition:
     *      T arr[] is sorted by the specification of the ComparatorInterface
     */
};

template<typename T>
inline void MergeSort::mergeSort(T arr[], int size,
        ComparatorInterface<T>* comp) {
    mergeSort(arr, 0, size - 1, comp);
}

template<typename T>
inline void MergeSort::mergeSort(T arr[], int left, int right,
        ComparatorInterface<T>* comp) {
    if (left < right) {

        int m = left + (right - left) / 2;

        mergeSort(arr, left, m, comp);
        mergeSort(arr, m + 1, right, comp);

        merge(arr, left, m, right, comp);
    }
}

template<typename T>
inline void MergeSort::merge(T arr[], int left, int middle, int right,
```

```
        ComparatorInterface<T>* comp) {
    int i, j, k;
    int sizeLeft = middle - left + 1;
    int sizeRight = right - middle;

    T L[sizeLeft], R[sizeRight];

    for (i = 0; i < sizeLeft; i++)
        L[i] = arr[left + i];
    for (j = 0; j < sizeRight; j++)
        R[j] = arr[middle + 1 + j];

    i = 0;
    j = 0;
    k = left;
    while (i < sizeLeft && j < sizeRight) {
        if (comp->compare(L[i], R[j]) <= 0) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];

            j++;
        }
        k++;
    }

    while (i < sizeLeft) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < sizeRight) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
;

#endif /* MERGESORT_H_ */
```