20191564 김신건

Main

데이터를 생성하고, 정렬함수를 호출하면서 시간을 측정하는 코드입니다.

```
public class Homework {
    public static void main(String[] args) {

        // 배열 초기화
        Employee[] employees1 = new Employee[11000];
        Employee[] employees2 = new Employee[110000];
        Employee[] employees3 = new Employee[11000];
        Employee[] employees4 = new Employee[110000];

        Employee[] employees12 = new Employee[11000];
        Employee[] employees22 = new Employee[110000];
        Employee[] employees32 = new Employee[11000];
        Employee[] employees42 = new Employee[110000];

        // 데이터 생성
        for(int x=0; x<10000; x++) {
            employees1[x] = new HourlyEmployee("a","b",x+1,1,1,2000,1.0,1.0);
            employees12[x] = new HourlyEmployee("a","b",x+1,1,1,2000,1.0,1.0);
            employees3[x] = new HourlyEmployee("a","b",x+1,1,1,2000,1.0,1.0);
            employees32[x] = new HourlyEmployee("a","b",x+1,1,1,2000,1.0,1.0);

        }
        for(int x=0; x<100000; x++) {
            employees2[x] = new HourlyEmployee("a","b",x+1,1,1,2000,1.0,1.0);
            employees22[x] = new HourlyEmployee("a","b",x+1,1,1,2000,1.0,1.0);
            employees4[x] = new HourlyEmployee("a","b",x+1,1,1,2000,1.0,1.0);
            employees42[x] = new HourlyEmployee("a","b",x+1,1,1,2000,1.0,1.0);
        }

        // 데이터 랜덤
        for(int x=0; x<5000; x++) {
            int randIdx = (int)(Math.random()*5000)+4999;
            Employee tmp = employees3[x];
            employees3[x] = employees3[randIdx];
            employees3[randIdx] = tmp;

            tmp = employees32[x];
            employees32[x] = employees32[randIdx];
            employees32[randIdx] = tmp;
        }
        for(int x=0; x<50000; x++) {
            int randIdx = (int)(Math.random()*50000)+49999;
            Employee tmp = employees4[x];
            employees4[x] = employees4[randIdx];
```

```
            employees4[randIdx] = tmp;

            tmp = employees42[x];
            employees42[x] = employees42[randIdx];
            employees42[randIdx] = tmp;
        }

        // Merge Sort
        long startTime = System.currentTimeMillis();
        long endTime = 0;
        MergeSort.mergeSort(employees1, 0, 9999);

        endTime = System.currentTimeMillis();
        long MergeTime1 = endTime - startTime;

        startTime = System.currentTimeMillis();
        MergeSort.mergeSort(employees2, 0, 99999);
        endTime = System.currentTimeMillis();
        long MergeTime2 = endTime - startTime;

        startTime = System.currentTimeMillis();
        MergeSort.mergeSort(employees3, 0, 9999);
        endTime = System.currentTimeMillis();
        long MergeTime3 = endTime - startTime;

        startTime = System.currentTimeMillis();
        MergeSort.mergeSort(employees4, 0, 99999);
        endTime = System.currentTimeMillis();
        long MergeTime4 = endTime - startTime;

        //Selectiopon Sort

        startTime = System.currentTimeMillis();
        SelectionSort.selectionSort(employees12, 10000);
        endTime = System.currentTimeMillis();
        long SelectionTime1 = endTime - startTime;

        startTime = System.currentTimeMillis();
        SelectionSort.selectionSort(employees22, 100000);
        endTime = System.currentTimeMillis();
        long SelectionTime2 = endTime - startTime;

        startTime = System.currentTimeMillis();
        SelectionSort.selectionSort(employees32, 10000);
        endTime = System.currentTimeMillis();
        long SelectionTime3 = endTime - startTime;

        startTime = System.currentTimeMillis();
        SelectionSort.selectionSort(employees42, 100000);
        endTime = System.currentTimeMillis();
        long SelectionTime4 = endTime - startTime;

        System.out.println("---- Original Sorted Data ----\n");
        System.out.println("Merge Sort(size 10000): "+MergeTime1);
```

```
        System.out.println("Merge Sort(size 100000): "+MergeTime2+"\n");

        System.out.println("---- Random Data ----\n");
        System.out.println("Merge Sort(size 10000): "+MergeTime3);
        System.out.println("Merge Sort(size 100000): "+MergeTime4+"\n");

        System.out.println("---- Original Sorted Data ----\n");
        System.out.println("Selection Sort(size 10000): "+SelectionTime1);
        System.out.println("Selection Sort(size 100000): "+SelectionTime2+"\n");

        System.out.println("---- Random Data ----\n");
        System.out.println("Selection Sort(size 10000): "+SelectionTime3);
        System.out.println("Selection Sort(size 100000): "+SelectionTime4+"\n");

    }
}
```

Merge sort

내림차순으로 바꾸기 위해서, merge를 하는 과정에서 `(ar[beginHalf1].compareTo(ar[beginHalf2]) > 0)` 이 부분의 부호가 변경되었습니다.

```
public class MergeSort {

    public static <T extends Comparable<? super T>> void mergeSort(T ar[], int
first, int last) {
        @SuppressWarnings("unchecked")
        T[] tempArray =(T[]) new Comparable<?>[ar.length];
        mergeSort(ar, tempArray, first, last);
    }

    public static <T extends Comparable<? super T>> void mergeSort(T ar[], T
tempArray[], int first, int last) {
        int mid = (first + last)/2;
        if(first < last) {
            mergeSort(ar, tempArray, first, mid);
            mergeSort(ar, tempArray, mid + 1, last);
            merge(ar,tempArray,first, last);
        }
    }
    public static <T extends Comparable<? super T>> void merge(T ar[], T
tempArray[], int first, int last) {
        int mid = (first + last) / 2;
        int beginHalf1 = first;
        int endHalf1 = mid;

        int beginHalf2 = mid + 1;
        int endHalf2 = last;

        int index = first;
        while((beginHalf1 <= endHalf1) && (beginHalf2 <= endHalf2)) {
```

```
            // 내림차순으로 바꾸기 위해 compareTo 비교문의 부호 변경
            if(ar[beginHalf1].compareTo(ar[beginHalf2]) > 0) {
                tempArray[index] = ar[beginHalf1];
                beginHalf1++;
            }
            else {
                tempArray[index] = ar[beginHalf2];
                beginHalf2++;
            }
            index++;
        }
        while(beginHalf1 <= endHalf1) {
            tempArray[index] = ar[beginHalf1];
            beginHalf1++;
            index++;
        }
        while(beginHalf2 <= endHalf2) {
            tempArray[index] = ar[beginHalf2];
            beginHalf2++;
            index++;
        }

        for(int x=first; x <= last; x++) {
            ar[x] = tempArray[x];
        }
    }
}
```

Selection Sort

내림차순으로 바꾸기 위해서 `if(ar[beginHalf1].compareTo(ar[beginHalf2]) > 0)` 이부분의 부호가 변경되었습니다.

```
public class SelectionSort {
    public static <T extends Comparable<? super T>> void selectionSort(T ar[],int
n) {
        for(int index = 0; index < n-1; index++) {
            int indexOfNextSmallest = getIndexOfSmallest(ar, index, n-1);
            swap(ar, index, indexOfNextSmallest);
        }
    }

    public static <T extends Comparable<? super T>> int getIndexOfSmallest(T[] ar,
int first, int last) {
        T min = ar[first];
        int indexOfMin = first;
        for(int index = first +1; index <=last; index++) {
            // 내림차순으로 바꾸기 위해 compareTo 비교문의 부호 변경
            if(ar[beginHalf1].compareTo(ar[beginHalf2]) > 0)
                min = ar[index];
                indexOfMin = index;
```

```
                }
            }
            return indexOfMin;
        }

        private static void swap(Object[] a, int i, int j) {
            Object temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }
```