```java
/**
 * COMP215-Programming Project 2: Multiple Sort Analysis.
 * INSERTIONSORT is the Java implementation of the insertion sort algorithm.
 * This implementation is based on psuedocode from "CLRS Algorithms."
 * This class is timed using the STOPWATCH class.
 * @author Andrew Parsons
 * @version 21 January 2017
 */

class InsertionSort implements Sorter {

    /* --- INSTANCE VARIABLES --- */
    private Stopwatch stopwatch;
    private long elapsedTime;
    private static boolean debug = MainApp.debug;

    /* --- METHODS --- */

    /** (package-private): SORT begins timing, runs through the algorithm, and then stops timing.
     * SORT returns a sorted array. Implementation of algorithm from "CLRS Algorithms" */
    @Override
    public Comparable[] sort(Comparable[] dataset) {
        /* PRE-CONDITION: A passed array of items.
         * The array has items that can be compared to determine their ascending order. */

        stopwatch = new Stopwatch();

        /* INVARIANT (outer): subarray A[1...(j-1)] (psuedocode is 1 relative) consists of
        elements originally in A[1...(j-1)] in sorted order. */

        for (int j = 1; j < dataset.length; j++) {
            if(debug)
                assert assertionIsSorted(dataset,0,j-1);
            Comparable key = dataset[j];
            int i = j - 1;

            /* INVARIANT (inner): subarray A[i...j] contains elements that are greater than or
            equal to the key. */

            while ((i >= 0) && (dataset[i].compareTo(key) == 1)) {
                dataset[i+1] = dataset[i];
                i--;

                if(debug && i > 0)
                    assert assertionIsGreaterOrEqual(dataset,i,j);
            }
            if(debug)
                assert assertionIsSorted(dataset,0,j);
            dataset[i+1] = key;
        }

        if(debug)
            assert assertionIsSorted(dataset,0,dataset.length-1);
        elapsedTime = stopwatch.elapsedTime();
        return dataset;
    }

    /** (package-private): getter method for timing */
    @Override
    public long getElapsedTime() {
        return elapsedTime;
    }

    /** check if subarray is sorted in ascending order (called by assert) */
    private static boolean assertionIsSorted(Comparable[] dataset, int start, int end) {
        for (int s = start; s < end; s++) {
            if (dataset[s].compareTo(dataset[s+1]) > 0) {
                return false;
            }
        } return true;
    }

    /** check if subarray contains elements greater than or equal to the key (called by assert)
    */
    private static boolean assertionIsGreaterOrEqual(Comparable [] dataset, int start,int end) {
        for (int k = start; k < end; k++) {
            if (dataset[end].compareTo(dataset[k]) > 1) {
                return false;
            }
        } return true;
    }

    @Override // DO NOT USE THIS METHOD
    public Comparable[] sort(Comparable[] dataset, int indexBegin, int indexEnd) {
        return new Comparable[0];
    }
}
```