```java
/*
 * HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
 * Version: Sun Sep 15 14:13:26 CEST 2024
 */

package uebung01.as.aufgabe04;

import java.util.Random;


public class InsertionSort {

  /**
   * Sorts an int-array with the Insertion-Sort algorithm.
   * @param data The array to be sorted.
   */
  public static void insertionSort(int[] data) {

    // TODO: Implement here...

  }


  public static void main(String[] args) {

    int[] array = {5, 4, 2, 3, 1};
    int[] orginalArray = array.clone();
    printArray(array);

    insertionSort(array);

    printArray(array);
    verify(orginalArray, array);

    /* Makeing some test to measure the time needed of insertionSort().
     * Creating int-arrays, beginning with length of 2^minExponent
     * until the last array with length of 2^maxExponent.
     */
    final int minExponent = 10;
    final int maxExponent = 14;
    int n = (int)Math.round(Math.pow(2, maxExponent));
    array = new int[n];
    Random rand = new Random(0);     // a Random-Generator
    for (int i = 0; i < n; i++) {
      array[i] = rand.nextInt(101); // generating Numbers: 0..100
    }
    long lastTime = Long.MAX_VALUE;
    for (int exp = minExponent; exp <= maxExponent; exp++) {
      int len = (int)Math.round(Math.pow(2, exp));
      int[] arr = new int[len];
```



```java

      final int MEASUREMENTS = 10;
      long minTime = Long.MAX_VALUE;
      for (int m = 0; m < MEASUREMENTS; m++) {
        System.arraycopy(array, 0, arr, 0, len);
        long start = System.nanoTime();
        insertionSort(arr);
        long end = System.nanoTime();
        long time = end - start;
        if (time < minTime) {
          minTime = time;
        }
        verify(array, arr);
      }
      System.out.format("Array-Size: %,6d       Time: %,7.1f ms       "
                        + "Ratio to last: %2.1f\n",
                        len, (double) minTime / (long) 1e6,
                        (double) minTime / lastTime);
      lastTime = minTime;
    }
  }


  /**
   * Prints an int-array to the console.
   * @param array The int-array.
   */
  static void printArray(int[] array) {
    System.out.print("Array["+array.length+"]: ");
    for (int i: array) {
      System.out.print(i + " ");
    }
    System.out.println("");
  }


  /**
   * Verifies that sortedArray is a correctly sorted based on originalArray.
   * @param originalArray The original array.
   * @param sortedArray The sorted array, based on originalArray.
   *                    Can be shorter than originalArray.
   */
  static void verify(int[] originalArray, int[] sortedArray) {
    int[] originalSortedArray = new int[sortedArray.length];
    System.arraycopy(originalArray, 0, originalSortedArray, 0, sortedArray.length);
    java.util.Arrays.sort(originalSortedArray);
    if ( ! java.util.Arrays.equals(originalSortedArray, sortedArray)) {
      try {Thread.sleep(200);} catch(Exception e) {e = null;}
      System.err.println("ERROR: wrong sorted!");
      System.exit(1);
    }
  }

}


/* Session-Log:

$ java -Xint InsertionSort
Array[5]: 5 4 2 3 1
Array[5]: 1 2 3 4 5
Array-Size:  1,024       Time:     3.8 ms       Ratio to last: 0.0
Array-Size:  2,048       Time:    14.7 ms       Ratio to last: 3.9
Array-Size:  4,096       Time:    58.7 ms       Ratio to last: 4.0
Array-Size:  8,192       Time:   234.1 ms       Ratio to last: 4.0
Array-Size: 16,384       Time:   942.6 ms       Ratio to last: 4.0

*/
```