SortAnalysis.java

```java
import java.util.Random;

class ArraySortTracker {
    private long[] a;
    private int nElems;
    private int comparisons;
    private int copies;
    private int swaps;

    public ArraySortTracker(int max) {
        a = new long[max];
        nElems = 0;
        comparisons = 0;
        copies = 0;
        swaps = 0;
    }

    public void insert(long value) {
        a[nElems] = value;
        nElems++;
    }

    public void fillRandom(int size) {
        Random rand = new Random();
        for (int i = 0; i < size; i++) {
            insert(rand.nextInt(100000)); // Random numbers up to 100000
        }
    }

    public void resetCounters() {
        comparisons = 0;
        copies = 0;
        swaps = 0;
    }

    public void display() {
        for (int j = 0; j < nElems; j++) {
            System.out.print(a[j] + " ");
        }
        System.out.println("");
    }

    public void bubbleSort() {
        int out, in;
        resetCounters(); // Reset metrics before sorting

        for (out = nElems - 1; out > 1; out--) {
            for (in = 0; in < out; in++) {
```

```
49                  comparisons++; // Count comparison
50                  if (a[in] > a[in + 1]) {
51                      swap(in, in + 1);
52                  }
53              }
54          }
55
56          System.out.println("Bubble Sort - Comparisons: " + comparisons + ", Swaps: " + swaps);
57      }
58
59      private void swap(int one, int two) {
60          long temp = a[one];
61          a[one] = a[two];
62          a[two] = temp;
63          swaps++;
64          copies += 3; // One swap involves 3 copies
65      }
66
67      public void selectionSort() {
68          int out, in, min;
69          resetCounters(); // Reset metrics before sorting
70
71          for (out = 0; out < nElems - 1; out++) {
72              min = out;
73              for (in = out + 1; in < nElems; in++) {
74                  comparisons++;
75                  if (a[in] < a[min]) {
76                      min = in;
77                  }
78              }
79              if (out != min) {
80                  swap(out, min);
81              }
82          }
83
84          System.out.println("Selection Sort - Comparisons: " + comparisons + ", Swaps: " +
    swaps);
85      }
86
87      public void insertionSort() {
88          int in, out;
89          resetCounters(); // Reset metrics before sorting
90
91          for (out = 1; out < nElems; out++) {
92              long temp = a[out];
93              in = out;
94              while (in > 0 && a[in - 1] >= temp) {
95                  comparisons++;
96                  a[in] = a[in - 1];
97                  in--;
```

```
                copies++; // Each shift is a copy
            }
            a[in] = temp;
            copies++; // Insert temp back into array
        }

        System.out.println("Insertion Sort - Comparisons: " + comparisons + ", Copies: " +
copies);
    }
}

public class SortAnalysis {
    public static void main(String[] args) {
        int[] sizes = {10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000};

        for (int size : sizes) {
            ArraySortTracker arr = new ArraySortTracker(size);
            arr.fillRandom(size);

            System.out.println("\nArray size: " + size);
            arr.bubbleSort();
            arr.selectionSort();
            arr.insertionSort();
        }
    }
}

```