Sorting Competition

Group 2
Skye Antinozzi and Jocelyn Bayer

Sorting Competition Results

- Disqualified at competition for forgetting to include deep cloning
  - Had deep cloning been added for the competition the data would not have already been sorted correctly outside of the timer
  - Even with the deep cloning issue, the algorithm still sorted correctly even though it was not being timed

Data Representation and Algorithm

- Data Representation
  - Custom Point objects (Not java.awt.Point)
- Algorithm for sorting
  - First used mergesort implementation provided by Nic McPhee
  - Tried the Java API Arrays.sort(T[], Comparator<? super T>) method
  - Finally settled on the Java API Arrays.sort(Object[])

Handling Distances

- Distance formula originally used
- Scrapped the square root operator to eliminate overhead
- The result was a calculation that could only result in integer values
    - Threshold was never used and the data still sorted correctly

Running Time and Non-constant memory

- Running Time
  - Worst case: O(n log(n))
  - Average case: Θ(n log(n))
- Non-constant memory
  - Points array of size n for each non-reference point
  - n/2 object references for randomly ordered input arrays

Algorithm	Time (ms)
Nic's Mergesort	~550ms
Comparator Mergesort	~150ms
Comparable Mergesort	~35ms

Correctness

- The only problem mentioned by the group doing the correctness analysis was that we forget to use the deep clone method.
- Overall, the group checking our solution was satisfied that our solution was correct

Code Optimizations

- Using prefix decrement (--i instead of i)
- Loop conditions based on 0 rather than an upper-bound
- Removal of method calls by inlining
- Removal of method calls by finding other mathematical solutions
- Removal of floating point values by using integers

Potential Improvements

- Experimentation with return types
  - Return Point[] vs int[][]
- Clever ways to determine which Point within a pair of Points is closest to some reference point
- Exploration of Java bytecode optimization on a sorting algorithm