Program Structures and Algorithms
Spring 2024

NAME: Gautham Venkata Krishna Prasad
NUID: 002249901
GITHUB LINK: https://github.com/gauthamkris7neu/INFO6205Assignment

Task:
Solve 3-SUM using the *Quadrithmic*, *Quadratic*, and (bonus point) *quadraticWithCalipers*.

Relationship Conclusion (Brief Explanation of Why the Quadratic Method(s) Work) :

The quadratic methods work effectively for this problem because they leverage the sorted nature of the array to reduce the problem space. Rather that checking all possible triplets, we focus only on viable candidates so it reduces the number of operations required drastically. The two pointer and calipers techniques make sure that each element is involved in the sum calculation only a linear number of times.

This method works mainly because we transform the problem into a series of steps where each step can be solved in linear time and the total number of steps is also linear, resulting in quadratic performance which is significantly better than cubic for large datasets.

Evidence to support that conclusion (Spreadsheet Showing Timing Observations) :

With the help of the main method and stopwatch class recorded the timings..
Source Class was used to generate the array with random values..

```java
public static void main(String[] args) {
    int n = 100;
    for (int i = 0; i < 5; i++) {
        Source source = new Source(n, n, seed: 0L);
        Supplier<int[]> supplier = source.intsSupplier( safetyFactor: 10);
        int [] a = supplier.get();
        Stopwatch stopwatch = new Stopwatch();
        ThreeSumQuadraticWithCalipers threesumAlgo = new ThreeSumQuadraticWithCalipers(a);
        threesumAlgo.getTriples();
        long time = stopwatch.lap();
        System.out.println("N = " + n + ", Time = " + time + " ms");
        n *= 2;
    }
}
```

N(Number of Elements in the array)	Quadrithmic	Quadratic	Cubic	Quadratic with Calipers
100	1ms	1ms	2ms	2ms
200	0ms	0ms	2ms	0ms
400	1ms	0ms	2ms	1ms
800	5ms	1ms	14ms	2ms
1600	24ms	2ms	418ms	2ms

Unit Test Screenshots: