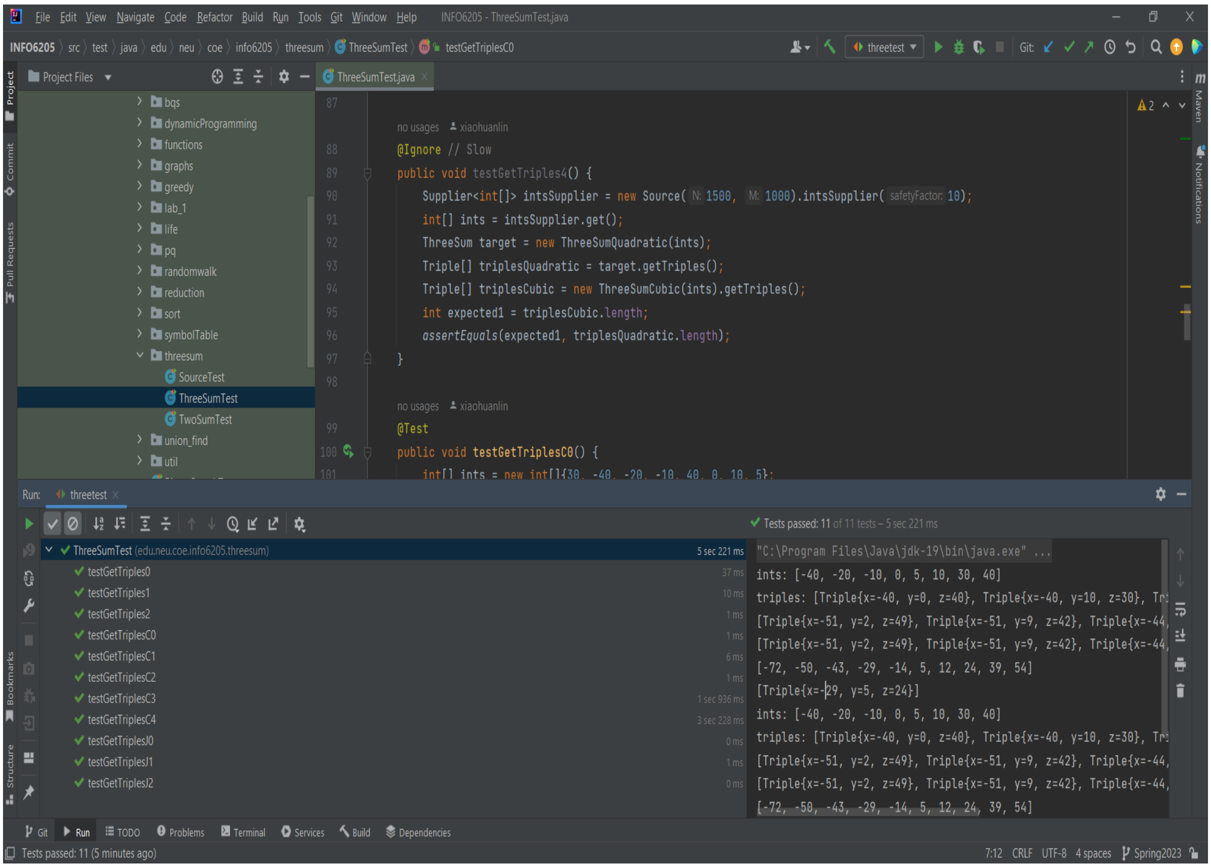
# INFO 6205-PROGRAM STRUCTURE AND ALGORITHMS ASSIGNMENT-1

**Sai Kashyap Cheruku**
**NU-ID- 002756849**

## Task:

The programmer is expected to solve the 3-sum problem using the Quadratic, Quadarithmic and Quadtratic with Calipers approaches. Testing the code by running unit tests and benchmark the time taken for each algorithm to execute successfully, using the double benchmark approach. Analyzing the behaviour and performance of different time-complexity approaches is required.

## Test Case Runs:



```
    no usages  xiaohuanlin
    @Ignore // Slow
    public void testGetTriples4() {
        Supplier<int[]> intsSupplier = new Source( N: 1500,  M: 1000).intsSupplier( safetyFactor: 10);
        int[] ints = intsSupplier.get();
        ThreeSum target = new ThreeSumQuadratic(ints);
        Triple[] triplesQuadratic = target.getTriples();
        Triple[] triplesCubic = new ThreeSumCubic(ints).getTriples();
        int expected1 = triplesCubic.length;
        assertEquals(expected1, triplesQuadratic.length);
    }

    no usages  xiaohuanlin
    @Test
    public void testGetTriplesC0() {
        int[] ints = new int[]{30, -40, -20, -10, 40, 0, 10, 5};
```

✓ Tests passed: 11 of 11 tests – 5 sec 221 ms

```
✓ ThreeSumTest (edu.neu.coe.info6205.threesum)   5 sec 221 ms
    ✓ testGetTriples0                            37 ms
    ✓ testGetTriples1                            10 ms
    ✓ testGetTriples2                             1 ms
    ✓ testGetTriplesC0                            1 ms
    ✓ testGetTriplesC1                            6 ms
    ✓ testGetTriplesC2                            1 ms
    ✓ testGetTriplesC3                   1 sec 936 ms
    ✓ testGetTriplesC4                   3 sec 228 ms
    ✓ testGetTriplesJ0                            0 ms
    ✓ testGetTriplesJ1                            1 ms
    ✓ testGetTriplesJ2                            0 ms
```

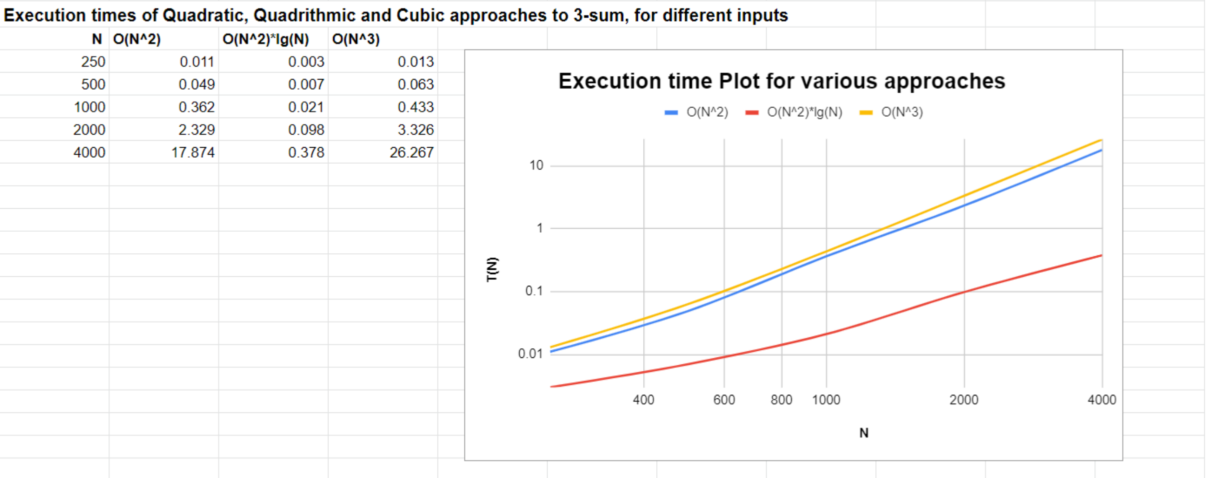
```
"C:\Program Files\Java\jdk-19\bin\java.exe" ...
ints: [-40, -20, -10, 0, 5, 10, 30, 40]
triples: [Triple{x=-40, y=0, z=40}, Triple{x=-40, y=10, z=30}, Tri
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44,
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44,
[-72, -50, -43, -29, -14, 5, 12, 24, 39, 54]
[Triple{x=-29, y=5, z=24}]
ints: [-40, -20, -10, 0, 5, 10, 30, 40]
triples: [Triple{x=-40, y=0, z=40}, Triple{x=-40, y=10, z=30}, Tri
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44,
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44,
[-72, -50, -43, -29, -14, 5, 12, 24, 39, 54]
```

Tests passed: 11 (5 minutes ago)

**Benchmarking Results:**

Tests for obtaining the execution time have been carried out for the three approaches. The input array size has been doubled for every successive three-algorithm test run. In case of the cubic complexity algorithm, a limit on the permissible input size is set to avoid long execution times.

The timing is performed by using the Stopwatch class present in the repository.



**Execution times of Quadratic, Quadrithmic and Cubic approaches to 3-sum, for different inputs**

| N | O(N^2) | O(N^2)*lg(N) | O(N^3) |
|---|---|---|---|
| 250 | 0.011 | 0.003 | 0.013 |
| 500 | 0.049 | 0.007 | 0.063 |
| 1000 | 0.362 | 0.021 | 0.433 |
| 2000 | 2.329 | 0.098 | 3.326 |
| 4000 | 17.874 | 0.378 | 26.267 |

*Note: Execution time calculated in seconds.*

**More on the Quadratic Approach:**

In the quadratic approaches, solutions are arrived at by sequentially deriving the solution spaces pertaining to an input array. The sequence followed is by fixing an index in the triplet and finding the other two indices which satisfy the zero sum condition. As the arrays are sorted, depending on which index is fixed, searching can begin outwards in both directions from the fixed index. The middle element is fixed before the search operation in quadratic approach, whereas the first element is fixed in the quadratic with calipers approach. Both the approaches exhibit a time complexity of O(N^2).