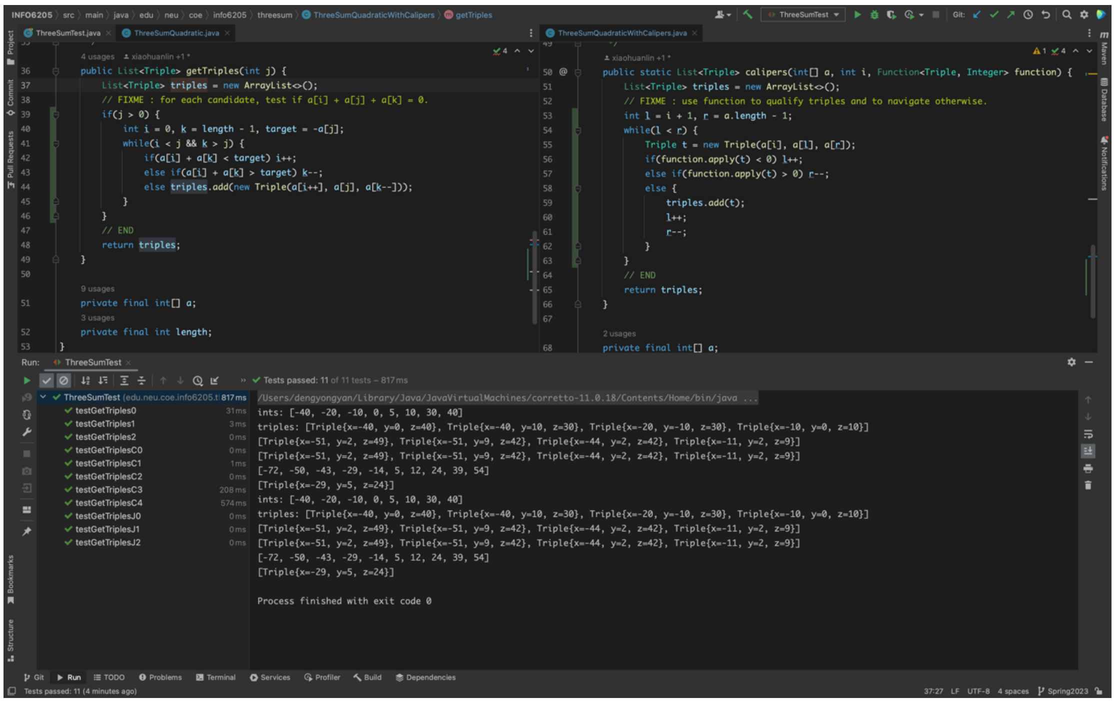
INFO6205 - Assignment02 - 3Sum Report

Here is the Unit tests result of Quadratic and QuadraticWithCalipers Methods:



```java
    public List<Triple> getTriples(int j) {
        List<Triple> triples = new ArrayList<>();
        // FIXME : for each candidate, test if a[i] + a[j] + a[k] = 0.
        if(j > 0) {
            int i = 0, k = length - 1, target = -a[j];
            while(i < j && k > j) {
                if(a[i] + a[k] < target) i++;
                else if(a[i] + a[k] > target) k--;
                else triples.add(new Triple(a[i++], a[j], a[k--]));
            }
        }
        // END
        return triples;
    }

    private final int[] a;
    private final int length;
}
```

```java
    public static List<Triple> calipers(int[] a, int i, Function<Triple, Integer> function) {
        List<Triple> triples = new ArrayList<>();
        // FIXME : use function to qualify triples and to navigate otherwise.
        int l = i + 1, r = a.length - 1;
        while(l < r) {
            Triple t = new Triple(a[i], a[l], a[r]);
            if(function.apply(t) < 0) l++;
            else if(function.apply(t) > 0) r--;
            else {
                triples.add(t);
                l++;
                r--;
            }
        }
        // END
        return triples;
    }

    private final int[] a;
```

```
Tests passed: 11 of 11 tests – 817 ms

ThreeSumTest (edu.neu.coe.info6205.t  817 ms
    testGetTriples0     31 ms
    testGetTriples1      3 ms
    testGetTriples2      0 ms
    testGetTriplesC0     0 ms
    testGetTriplesC1     1 ms
    testGetTriplesC2     0 ms
    testGetTriplesC3   208 ms
    testGetTriplesC4   574 ms
    testGetTriplesJ0     0 ms
    testGetTriplesJ1     0 ms
    testGetTriplesJ2     0 ms
```

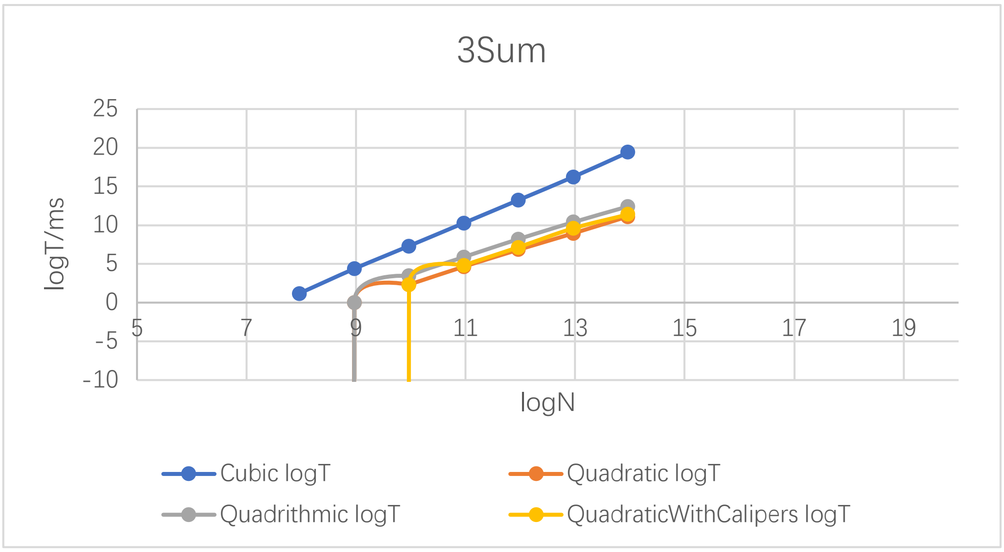
```
/Users/dengyongyan/Library/Java/JavaVirtualMachines/corretto-11.0.18/Contents/Home/bin/java ...
ints: [-40, -20, -10, 0, 5, 10, 30, 40]
triples: [Triple{x=-40, y=0, z=40}, Triple{x=-40, y=10, z=30}, Triple{x=-20, y=-10, z=30}, Triple{x=-10, y=0, z=10}]
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2, z=42}, Triple{x=-11, y=2, z=9}]
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2, z=42}, Triple{x=-11, y=2, z=9}]
[-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}, Triple{x=-20, y=-10, z=30}, Triple{x=-10, y=0, z=10}]
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2, z=42}, Triple{x=-11, y=2, z=9}]
[Triple{x=-51, y=2, z=49}, Triple{x=-51, y=9, z=42}, Triple{x=-44, y=2, z=42}, Triple{x=-11, y=2, z=9}]
[-72, -50, -43, -29, -14, 5, 12, 24, 39, 54]
[Triple{x=-29, y=5, z=24}]

Process finished with exit code 0
```

Tests passed: 11 (4 minutes ago)

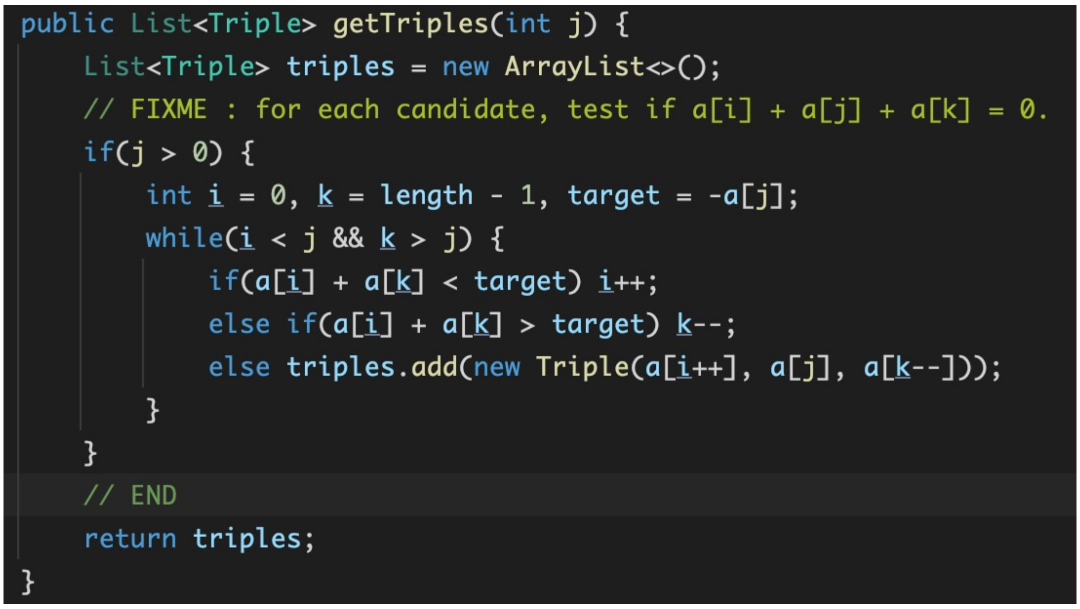
Relationship between data size N = 250, 500, 1000, 2000, 4000, 8000 and 16000 and operating time(ms) of Cubic, Quadratic, Quadrithmic and QuadriticWithCaplipers:

N	Cubic/ms	Quadratic/ms	Quadrithmic/ms	QuadraticWithCalipers/ms
250	2.29	0	0	0
500	20.66	1	1	0
1000	159	5	11	5
2000	1240	25	59	28
4000	9719.8	114	295	143
8000	77161	492	1340	775
16000	694120	2238	5479	2669



__Explanation of why the quadratic method(s) work.__

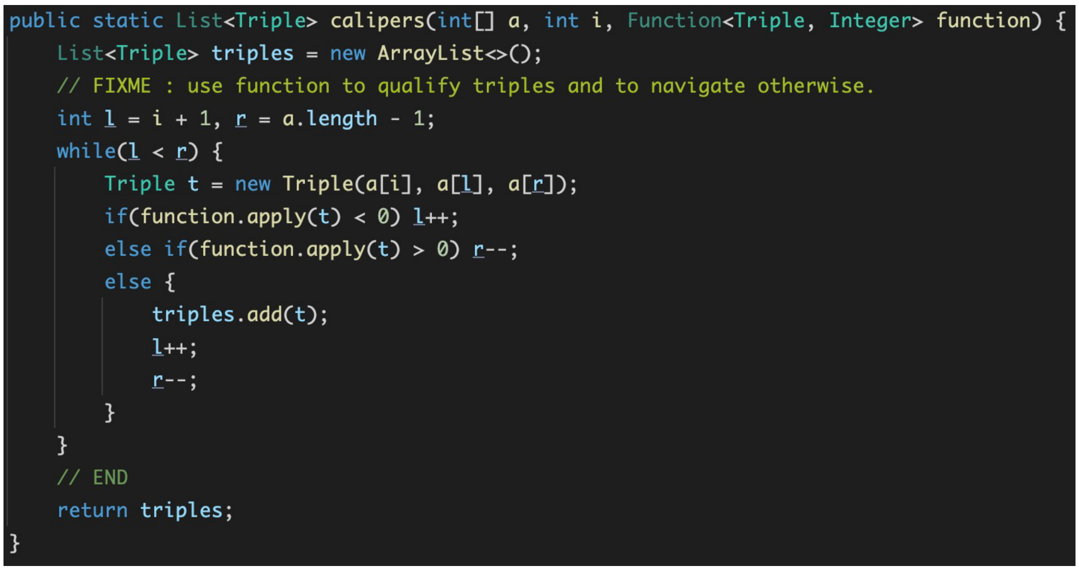
For 3SumQuadratic method, two pointers “i” and “k” will iterate in range [0, j - 1] and [j + 1, length - 1]. Because of the sorted array, if the sum of pointer “i” and “k” is smaller than -a[j], then the sum needs to be added up, so i++. If the sum of two pointers is bigger than -a[j], then the sum needs to be minus off, so j--.



```
public List<Triple> getTriples(int j) {
    List<Triple> triples = new ArrayList<>();
    // FIXME : for each candidate, test if a[i] + a[j] + a[k] = 0.
    if(j > 0) {
        int i = 0, k = length - 1, target = -a[j];
        while(i < j && k > j) {
            if(a[i] + a[k] < target) i++;
            else if(a[i] + a[k] > target) k--;
            else triples.add(new Triple(a[i++], a[j], a[k--]));
        }
    }
    // END
    return triples;
}
```

3SumQuadratic

For 3SumQuadraticWithCalipers, two pointers “l” and “r” will move in range [i + 1, r - 1] and [l + 1, length - 1] and follow the role that is same as 3SumQuadratic.



```java
public static List<Triple> calipers(int[] a, int i, Function<Triple, Integer> function) {
    List<Triple> triples = new ArrayList<>();
    // FIXME : use function to qualify triples and to navigate otherwise.
    int l = i + 1, r = a.length - 1;
    while(l < r) {
        Triple t = new Triple(a[i], a[l], a[r]);
        if(function.apply(t) < 0) l++;
        else if(function.apply(t) > 0) r--;
        else {
            triples.add(t);
            l++;
            r--;
        }
    }
    // END
    return triples;
}
```

3SumQuadraticWithCalipers