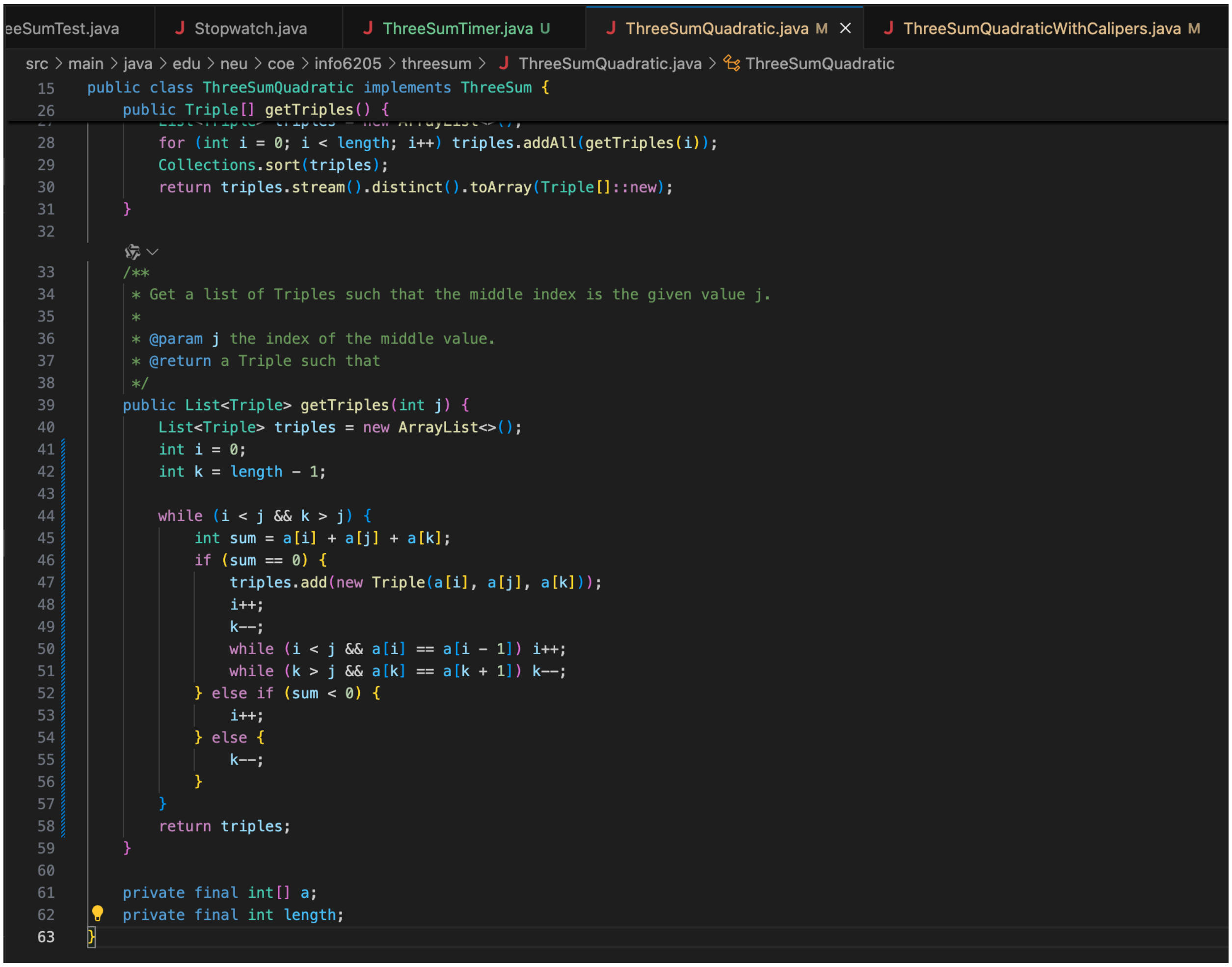
Program Structures and Algorithms

Fall 2024

NAME: Xinyi Xu

NUID: 002856992

Code Screenshots:


eeSumTest.java
Stopwatch.java
ThreeSumTimer.java U
ThreeSumQuadratic.java M
ThreeSumQuadraticWithCalipers.java M
src > main > java > edu > neu > coe > info6205 > threesum > ThreeSumQuadratic.java > ThreeSumQuadratic

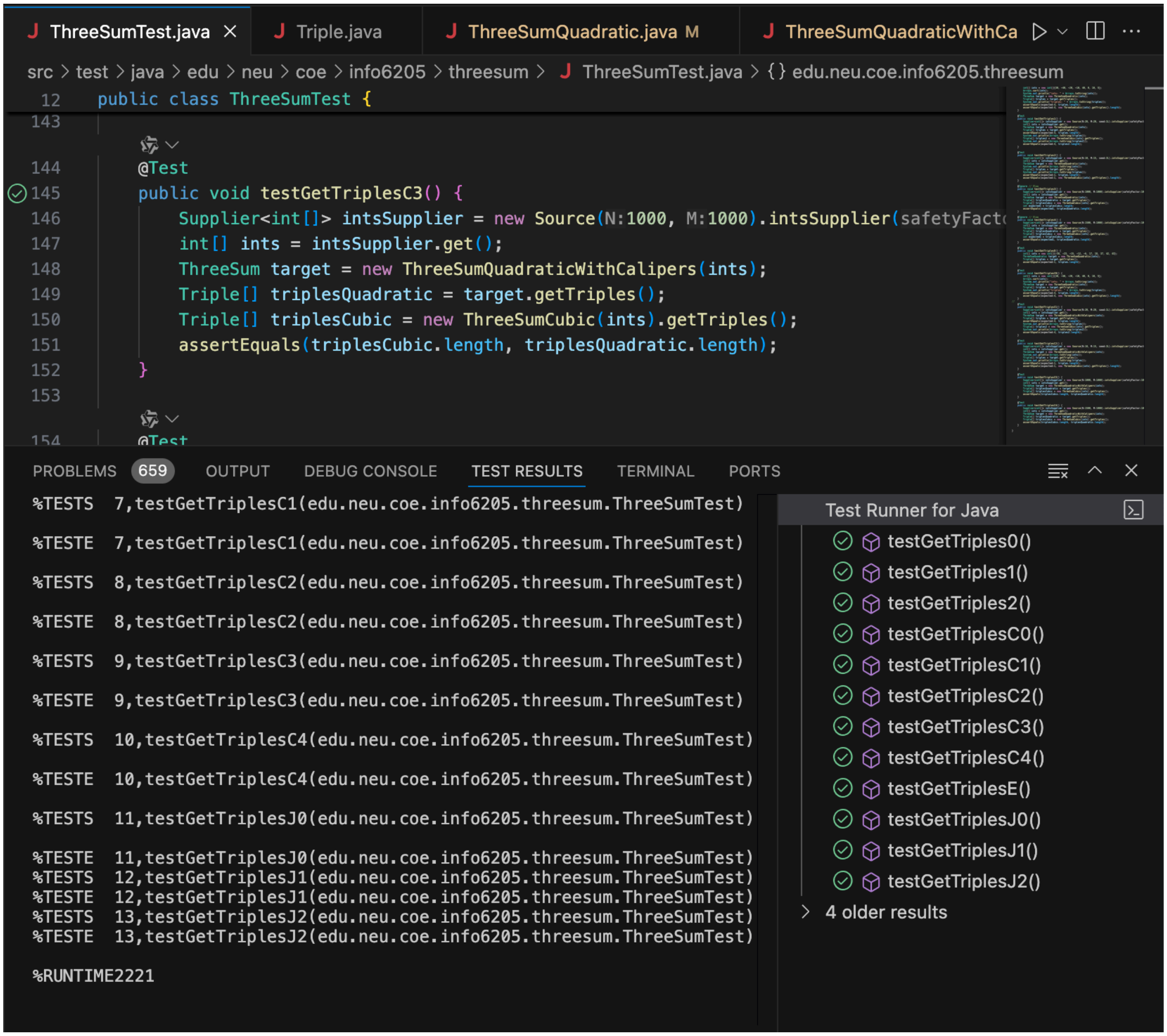
```java
public class ThreeSumQuadratic implements ThreeSum {
    public Triple[] getTriples() {
        List<Triple> triples = new ArrayList<>();
        for (int i = 0; i < length; i++) triples.addAll(getTriples(i));
        Collections.sort(triples);
        return triples.stream().distinct().toArray(Triple[]::new);
    }

    /**
     * Get a list of Triples such that the middle index is the given value j.
     *
     * @param j the index of the middle value.
     * @return a Triple such that
     */
    public List<Triple> getTriples(int j) {
        List<Triple> triples = new ArrayList<>();
        int i = 0;
        int k = length - 1;

        while (i < j && k > j) {
            int sum = a[i] + a[j] + a[k];
            if (sum == 0) {
                triples.add(new Triple(a[i], a[j], a[k]));
                i++;
                k--;
                while (i < j && a[i] == a[i - 1]) i++;
                while (k > j && a[k] == a[k + 1]) k--;
            } else if (sum < 0) {
                i++;
            } else {
                k--;
            }
        }
        return triples;
    }

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

Unit Test Screenshots:



```java
public class ThreeSumTest {

    @Test
    public void testGetTriplesC3() {
        Supplier<int[]> intsSupplier = new Source(N:1000, M:1000).intsSupplier(safetyFact
        int[] ints = intsSupplier.get();
        ThreeSum target = new ThreeSumQuadraticWithCalipers(ints);
        Triple[] triplesQuadratic = target.getTriples();
        Triple[] triplesCubic = new ThreeSumCubic(ints).getTriples();
        assertEquals(triplesCubic.length, triplesQuadratic.length);
    }

    @Test
```

```
%TESTS  7,testGetTriplesC1(edu.neu.coe.info6205.threesum.ThreeSumTest)

%TESTE  7,testGetTriplesC1(edu.neu.coe.info6205.threesum.ThreeSumTest)

%TESTS  8,testGetTriplesC2(edu.neu.coe.info6205.threesum.ThreeSumTest)

%TESTE  8,testGetTriplesC2(edu.neu.coe.info6205.threesum.ThreeSumTest)

%TESTS  9,testGetTriplesC3(edu.neu.coe.info6205.threesum.ThreeSumTest)

%TESTE  9,testGetTriplesC3(edu.neu.coe.info6205.threesum.ThreeSumTest)

%TESTS  10,testGetTriplesC4(edu.neu.coe.info6205.threesum.ThreeSumTest)

%TESTE  10,testGetTriplesC4(edu.neu.coe.info6205.threesum.ThreeSumTest)

%TESTS  11,testGetTriplesJ0(edu.neu.coe.info6205.threesum.ThreeSumTest)

%TESTE  11,testGetTriplesJ0(edu.neu.coe.info6205.threesum.ThreeSumTest)
%TESTS  12,testGetTriplesJ1(edu.neu.coe.info6205.threesum.ThreeSumTest)
%TESTE  12,testGetTriplesJ1(edu.neu.coe.info6205.threesum.ThreeSumTest)
%TESTS  13,testGetTriplesJ2(edu.neu.coe.info6205.threesum.ThreeSumTest)
%TESTE  13,testGetTriplesJ2(edu.neu.coe.info6205.threesum.ThreeSumTest)

%RUNTIME2221
```

- ✓ testGetTriples0()
- ✓ testGetTriples1()
- ✓ testGetTriples2()
- ✓ testGetTriplesC0()
- ✓ testGetTriplesC1()
- ✓ testGetTriplesC2()
- ✓ testGetTriplesC3()
- ✓ testGetTriplesC4()
- ✓ testGetTriplesE()
- ✓ testGetTriplesJ0()
- ✓ testGetTriplesJ1()
- ✓ testGetTriplesJ2()
- › 4 older results

Observations:

N	Cubic (ms)	Quadrithmic (ms)	Quadratic (ms)	QuadraticWithCalipers (ms)
50	24	3	3	10
100	5	3	5	4
200	13	6	10	8
400	68	25	46	21
800	372	147	139	78
1600	1970	223	389	571
3200	20536	1245	746	3695

Explanation:

The quadratic method first sorts the array, which helps organize the numbers. Then, it uses the two-pointer technique to find the triples quickly. By combining these two strategies, the method significantly reduces the number of checks needed.