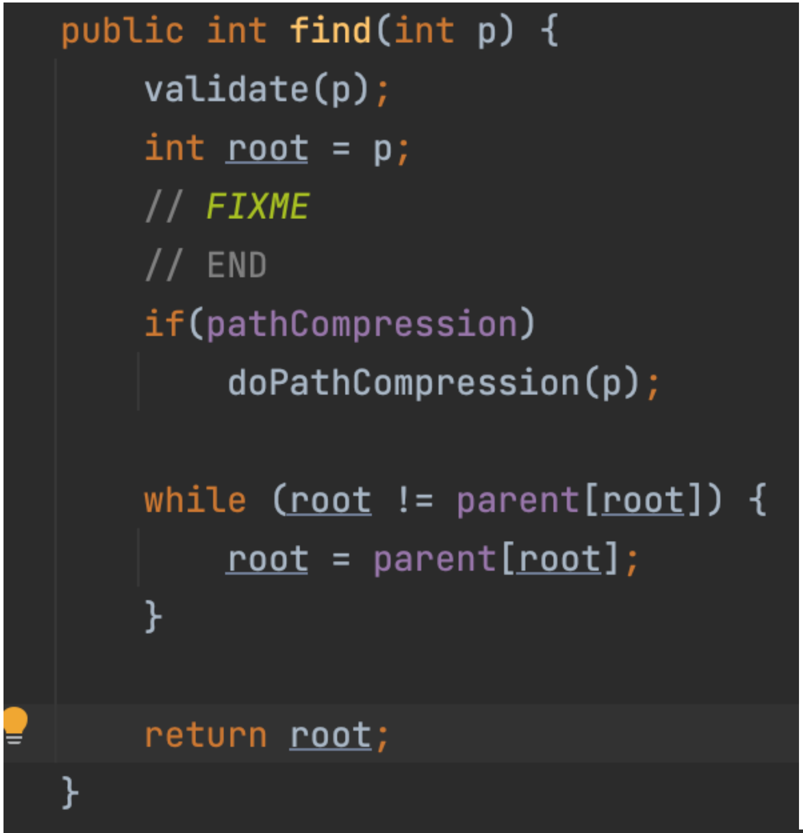
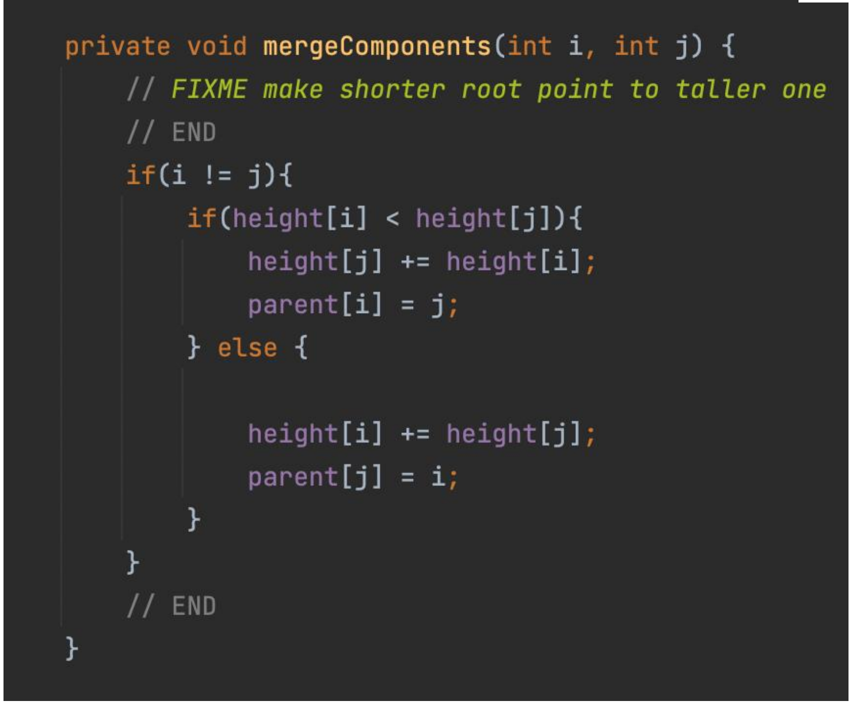
PSA Assignment 3
Siddharth Kushal
NUID 002104288

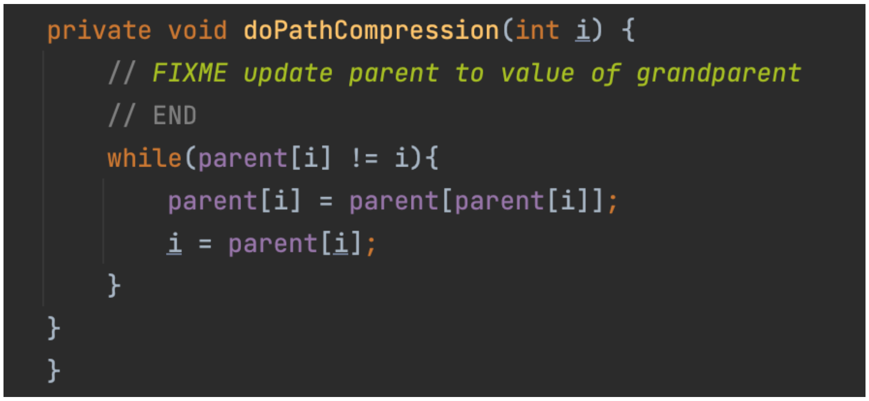
Step 1:



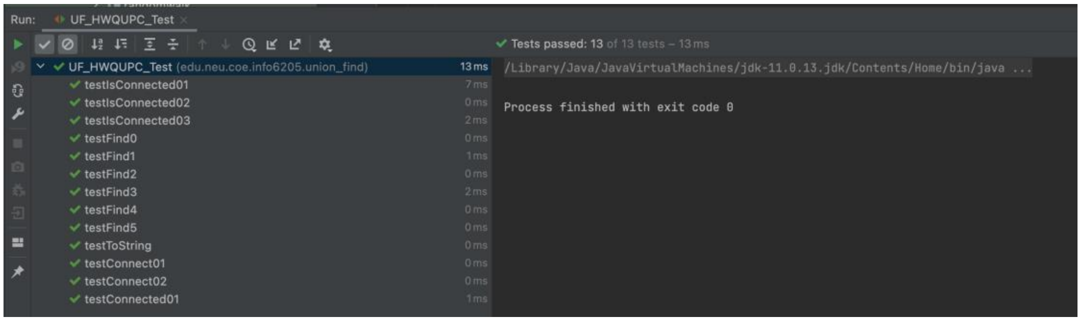
```
public int find(int p) {
    validate(p);
    int root = p;
    // FIXME
    // END
    if(pathCompression)
        doPathCompression(p);

    while (root != parent[root]) {
        root = parent[root];
    }

    return root;
}
```



```
private void mergeComponents(int i, int j) {
    // FIXME make shorter root point to taller one
    // END
    if(i != j){
        if(height[i] < height[j]){
            height[j] += height[i];
            parent[i] = j;
        } else {

            height[i] += height[j];
            parent[j] = i;
        }
    }
    // END
}
```

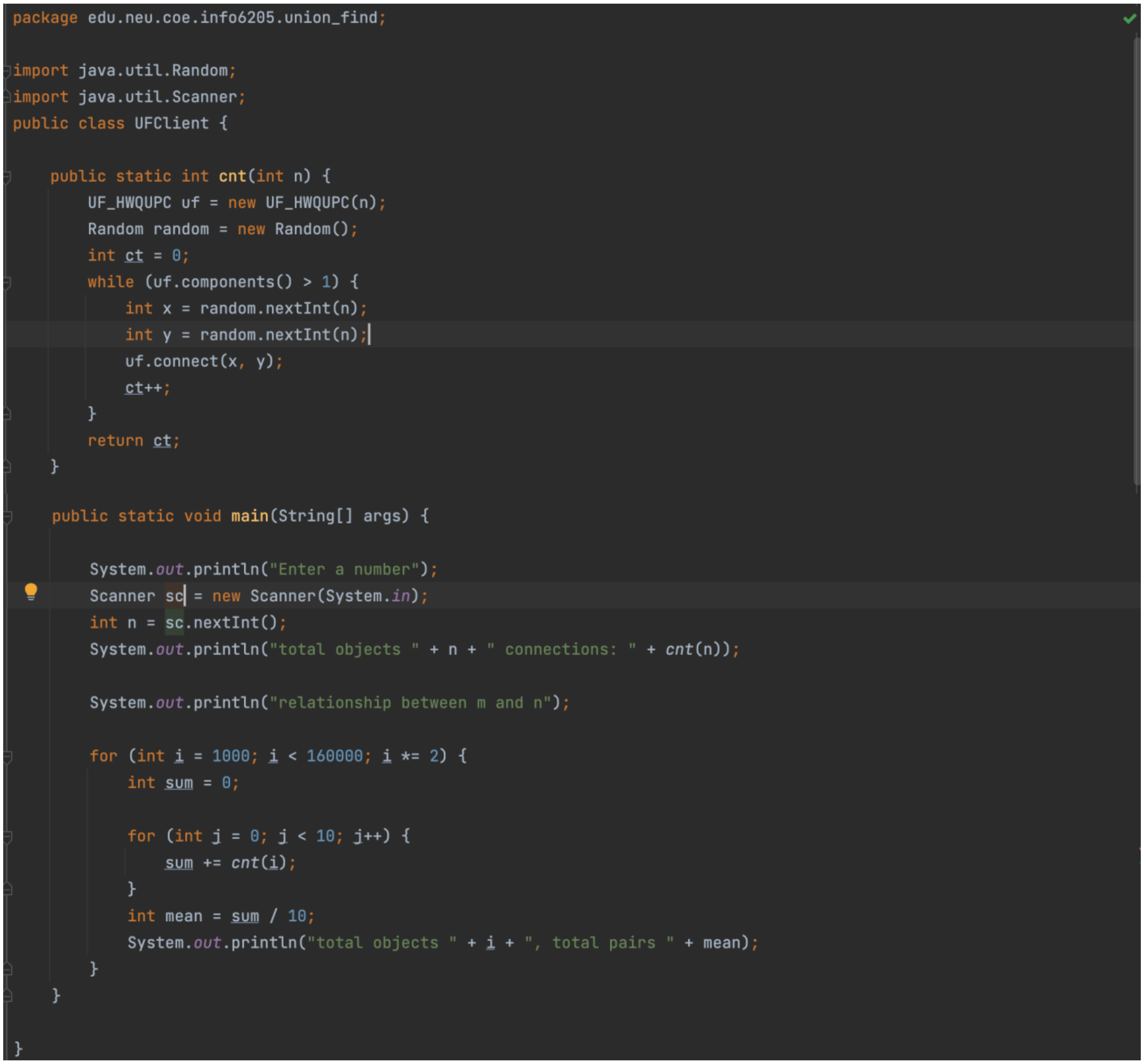


```java
private void doPathCompression(int i) {
    // FIXME update parent to value of grandparent
    // END
    while(parent[i] != i){
        parent[i] = parent[parent[i]];
        i = parent[i];
    }
}
}
```



```
✓ Tests passed: 13 of 13 tests – 13 ms

✓ UF_HWQUPC_Test (edu.neu.coe.info6205.union_find)   13 ms
    ✓ testIsConnected01                                7 ms
    ✓ testIsConnected02                                0 ms
    ✓ testIsConnected03                                2 ms
    ✓ testFind0                                        0 ms
    ✓ testFind1                                        1 ms
    ✓ testFind2                                        0 ms
    ✓ testFind3                                        2 ms
    ✓ testFind4                                        0 ms
    ✓ testFind5                                        0 ms
    ✓ testToString                                     0 ms
    ✓ testConnect01                                    0 ms
    ✓ testConnect02                                    0 ms
    ✓ testConnected01                                  1 ms

/Library/Java/JavaVirtualMachines/jdk-11.0.13.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```

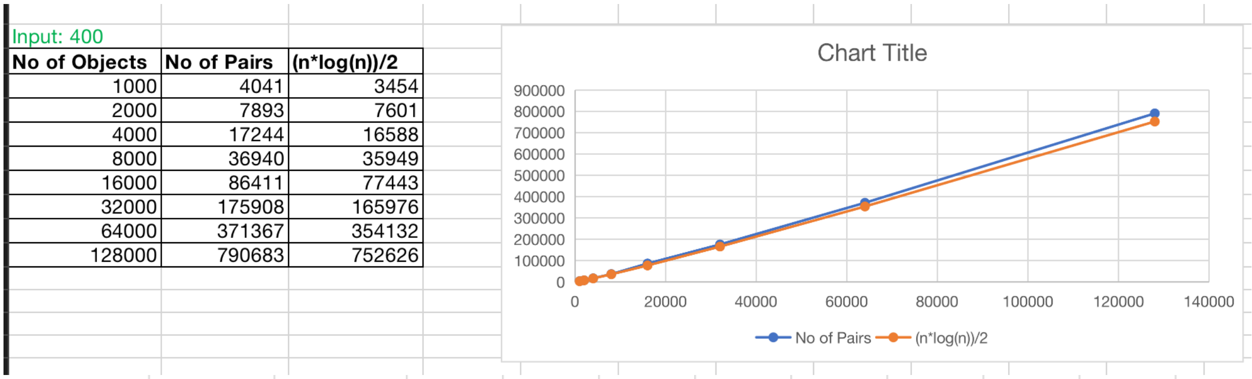
Step 2:



```java
package edu.neu.coe.info6205.union_find;

import java.util.Random;
import java.util.Scanner;
public class UFClient {

    public static int cnt(int n) {
        UF_HWQUPC uf = new UF_HWQUPC(n);
        Random random = new Random();
        int ct = 0;
        while (uf.components() > 1) {
            int x = random.nextInt(n);
            int y = random.nextInt(n);
            uf.connect(x, y);
            ct++;
        }
        return ct;
    }

    public static void main(String[] args) {

        System.out.println("Enter a number");
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        System.out.println("total objects " + n + " connections: " + cnt(n));

        System.out.println("relationship between m and n");

        for (int i = 1000; i < 160000; i *= 2) {
            int sum = 0;

            for (int j = 0; j < 10; j++) {
                sum += cnt(i);
            }
            int mean = sum / 10;
            System.out.println("total objects " + i + ", total pairs " + mean);
        }
    }

}
```

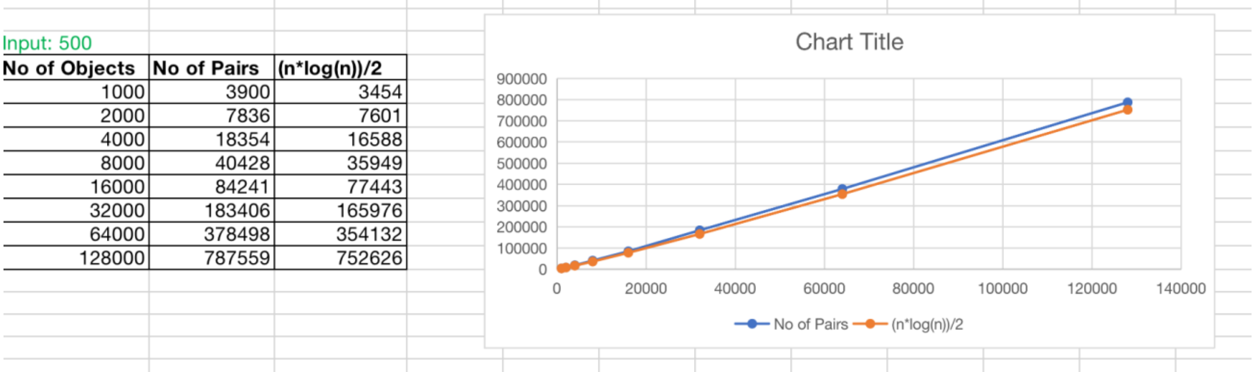
Run the UFClient.java file. To test the count technique, we may use the terminal to provide a number. To evaluate the link between m and n, we can run more n values and double their values using the doubling method, each 10 times.

Using the approach described above, we mapped the two outputs. First, the command line input was set to 400, and then the command line input was set to 500. Both of their outputs were mapped in a graphical fashion. The following is the output and graphical depiction.



Input: 400

No of Objects	No of Pairs	(n*log(n))/2
1000	4041	3454
2000	7893	7601
4000	17244	16588
8000	36940	35949
16000	86411	77443
32000	175908	165976
64000	371367	354132
128000	790683	752626



Input: 500

No of Objects	No of Pairs	(n*log(n))/2
1000	3900	3454
2000	7836	7601
4000	18354	16588
8000	40428	35949
16000	84241	77443
32000	183406	165976
64000	378498	354132
128000	787559	752626

Relationship Conclusion:

We can conclude from above scenario that

Both lines are around the same length. The connection between m and n may be deduced as follows:

m = ½ (n log(n))