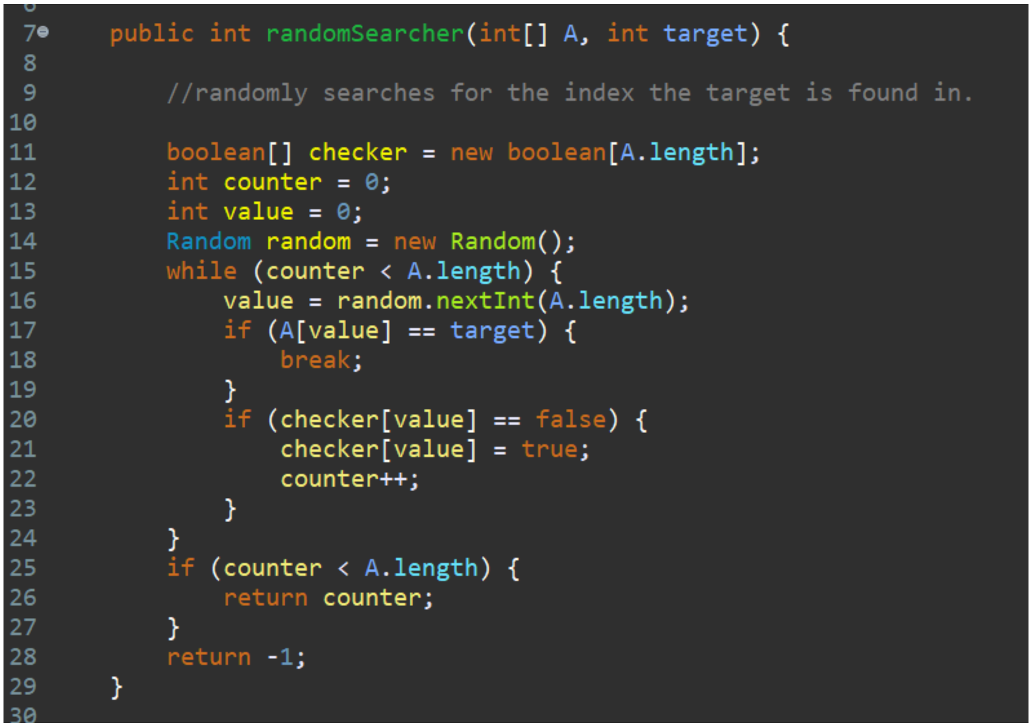
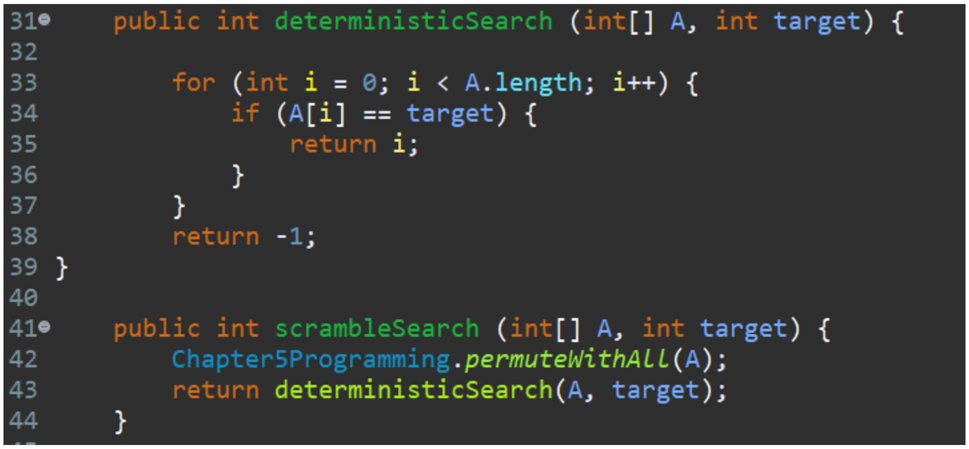
Chapter 5 Programming – Searches



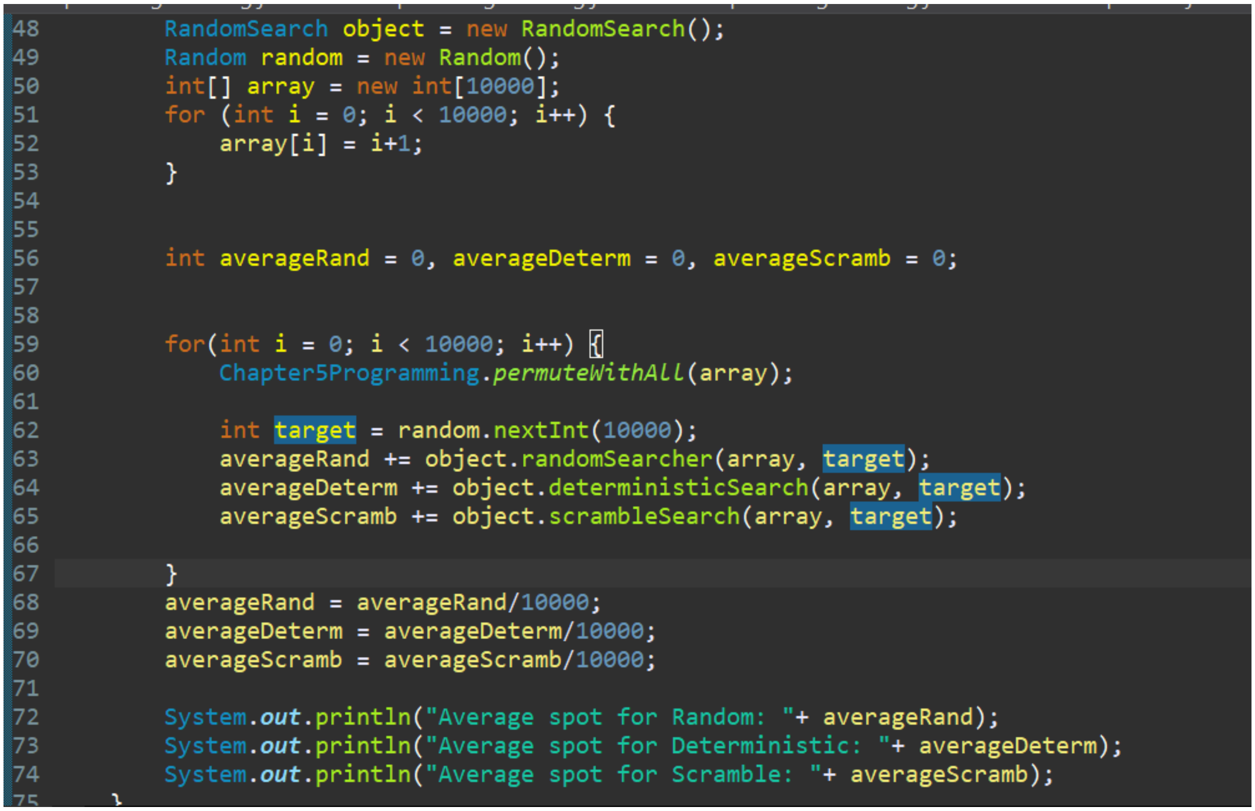
```java
    public int randomSearcher(int[] A, int target) {

        //randomly searches for the index the target is found in.

        boolean[] checker = new boolean[A.length];
        int counter = 0;
        int value = 0;
        Random random = new Random();
        while (counter < A.length) {
            value = random.nextInt(A.length);
            if (A[value] == target) {
                break;
            }
            if (checker[value] == false) {
                checker[value] = true;
                counter++;
            }
        }
        if (counter < A.length) {
            return counter;
        }
        return -1;
    }
```

This works in the same way as outlined in the instructions. It will randomly check a value, and if it is checking that value for the first time, it will switch the Boolean at that spot to true. A true Boolean means that its been checked. Once all of the values have been checked, the counter will reach the same size as the array and the target must not have been found.

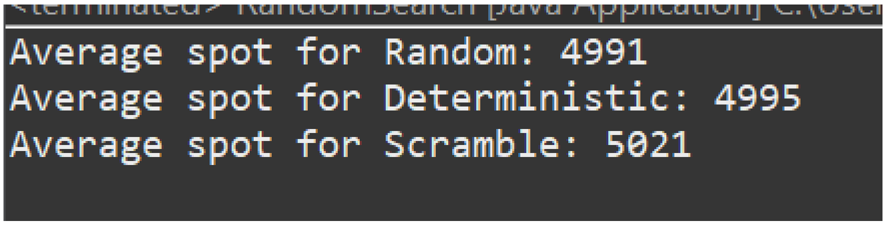


```java
    public int deterministicSearch (int[] A, int target) {

        for (int i = 0; i < A.length; i++) {
            if (A[i] == target) {
                return i;
            }
        }
        return -1;
}

    public int scrambleSearch (int[] A, int target) {
        Chapter5Programming.permuteWithAll(A);
        return deterministicSearch(A, target);
    }
```

These ones were quite simple to throw together. They both perform a linear, deterministic search. I pulled in the permuteWithAll method from the previous assignment to scramble the search.



```java
        RandomSearch object = new RandomSearch();
        Random random = new Random();
        int[] array = new int[10000];
        for (int i = 0; i < 10000; i++) {
            array[i] = i+1;
        }


        int averageRand = 0, averageDeterm = 0, averageScramb = 0;


        for(int i = 0; i < 10000; i++) {
            Chapter5Programming.permuteWithAll(array);

            int target = random.nextInt(10000);
            averageRand += object.randomSearcher(array, target);
            averageDeterm += object.deterministicSearch(array, target);
            averageScramb += object.scrambleSearch(array, target);

        }
        averageRand = averageRand/10000;
        averageDeterm = averageDeterm/10000;
        averageScramb = averageScramb/10000;

        System.out.println("Average spot for Random: "+ averageRand);
        System.out.println("Average spot for Deterministic: "+ averageDeterm);
        System.out.println("Average spot for Scramble: "+ averageScramb);
    }
```



```
<terminated> RandomSearch [Java Application] C:\Use
Average spot for Random: 4991
Average spot for Deterministic: 4995
Average spot for Scramble: 5021
```

The main method creates random sequences of an array containing 1 to 10000, and then adds the slot a random integer is found from each algorithm. Once its done this 10000 times, it divides the sums by the same number to find the average. The output suggests that the average number of spots that need to be checked is about half of the 10000 spots.

For Random Search, you must check every spot in the array to know that the target is not in there.