Варіант - 13

```java
import java.util.Random;
public class Main {
    static private long factorial (long n) {
        if (n == 0 || n == 1)
            return 1;
        int memory = 1;
        for (int i = 1; i <= n; i++) {
            memory*= i;
        }
        return memory;
    }
    static private long permutation_no_repeat(long n) {
        return factorial(n);
    }
    static private long permutation_repeat(long n, long array[]) {
        long result = permutation_no_repeat(n);
        for (long i: array) {
            result /= factorial(i);
        }
        return result;
    }
    static private long arrangement_no_repeat (long n, long k) {
        return factorial(n)/factorial(n-k);
    }
    static private long arrangement_repeat (long n, long k) {
        return (long)Math.pow(n,k);
    }
    static private long combination_no_repeat (long n, long k) {
        return arrangement_no_repeat(n, k)/factorial(k);
    }
    static private long combination_repeat (long n, long k) {
        return combination_no_repeat(n+k-1, k);
    }
    static private double test() {
        int tests_number = 1000000;
        Random tester = new Random();
        int required_numbers_counter = 0;
        for(int i = 0; i < tests_number; i++) {
            int [] test_num = {tester.nextInt(9)+1,
tester.nextInt(10),tester.nextInt(10),

tester.nextInt(10),tester.nextInt(10),tester.nextInt(10)};
            int eight_count = 0;
            int two_count = 0;
            for (int j: test_num) {
                if (j == 2) {
                    two_count++;
                }
                if (j == 8) {
                    eight_count++;
                }
            }
            if (eight_count == 1 && two_count == 2) {
                required_numbers_counter++;
            }
        }
        return (double)required_numbers_counter/tests_number;
    }
    public static void main(String[] args) {
        long digits_num = 6;
        long eight_repetion = 1;
        long two_repetion = 2;
        //long [] array = {eight_repetion, two_repetion, digits_num-eight_repetion-
two_repetion};
        //long [] array_2 = {eight_repetion, two_repetion, digits_num-eight_repetion-
two_repetion-1};
```

```
        /*long required_numbers =
permutation_repeat(digits_num,array)*arrangement_repeat(8,digits_num-eight_repetion-
two_repetion)-            old formula (correct)
                 permutation_repeat(digits_num-
1,array_2)*arrangement_repeat(8,digits_num-1-eight_repetion-two_repetion);*/
        long required_numbers = combination_no_repeat(digits_num,
two_repetion)*combination_no_repeat(digits_num-two_repetion, eight_repetion)*
                 arrangement_repeat(8,digits_num-eight_repetion-two_repetion) -
combination_no_repeat(digits_num-1, two_repetion)*combination_no_repeat(digits_num-1-
two_repetion, eight_repetion)*
                 arrangement_repeat(8,digits_num-1-eight_repetion-two_repetion);
        long all_numbers = 9*arrangement_repeat(10,digits_num-1);
        System.out.println("Відповідних чисел порахована кількість: " + required_numbers
+
                 "\nВсього шестицифрових чисел: " + all_numbers +
                 "\nКінцева імовірність: " + (double)required_numbers/
(double)all_numbers);
        System.out.println("\nОцінка статистична кількості відповідних чисел: " +
test());
    }
}
```