~/Downloads/ArrayListExercises.java

```
1  import java.util.ArrayList;
2  import java.util.Random;
3  public class ArrayListExercises {
4
5      public static void main(String[] args) {
6
7          // You do not need to handle the User Interface (UI).
8          bulgarianSolitaire(78);
9  ;       // Instead you can run the JUnit test cases found in ArrayListExercisesTe↩
   sts.java
10
11     }
12
13     /**
14      * Removes all of the strings of even length from the given list
15      * @param listOfStrings the list of Strings (list can be empty)
16      * @return the given list with all even length strings removed
17      */
18     public static ArrayList<String> removeEvenLength(ArrayList<String>
   listOfStrings)  {
19         for(int i=listOfStrings.size()-1; i>=0; i--) {
20
21
22             if (listOfStrings.get(i).length()%2==0) {
23                 listOfStrings.remove(i);
24
25             }
26         }
27
28         return listOfStrings;   // This return statement should be last
29     }
30
31     /**
32      * Moves the minimum value in the list to the front, otherwise preserving the
   order of the elements
33      * @param listOfIntegers the lis`t of Integers (list cannot be empty)
34      * @return the given list with the minimum value in the front (zeroth element)
35      */
36     public static ArrayList<Integer> minimumToFront(ArrayList<Integer> listOfInts)
   {
37         int min = listOfInts.get(0);
38         for (int x=0; x<listOfInts.size(); x++) {
39             if (min> listOfInts.get(x)) {
40                 min=listOfInts.get(x);
41             }
42
43         }
```

```
44          listOfInts.remove(listOfInts.indexOf(min));
45          listOfInts.add(0, min);
46
47
48          return listOfInts;   // This return statement should be last
49      }
50
51      /**
52       * Removes all elements from the given list whose values are in the range min
    through max (inclusive).
53       * If no eleme)nts in range min-max are found in the list, the list's contents
    are unchanged.
54       * If an empty list is passed, the list remains empty.  Assume min < max.
55       * @param listOfInts the list of Integers (list can be empty)
56       * @param min the minimum value in the range
57       * @param max the maximum value in the range
58       * @return the given list with the range min-max removed
59       */
60      public static ArrayList<Integer> filterRange(ArrayList<Integer> listOfInts, int
    min, int max)  {
61          for (int i=(listOfInts.size()-1); i>=0;i=i-1) {
62              if ( listOfInts.get(i)<=max && listOfInts.get(i)>=min ) {
63                  listOfInts.remove(i);
64              }
65          }
66          return listOfInts;   // This return statement should be last
67      }
68
69
70      /**
71       * Models/simulates the game of Bulgarian Solitaire.
72       * @param numCards the number of cards to start with; n must be a triangular
    number (a triangular
73       * number is a number that can be written as the sum of the first n positive
    integers).
74       */
75      public static void bulgarianSolitaire(int numCards)  {
76
77          // Check if given number of cards is triangular
78          int n = (int) Math.sqrt(2*numCards);
79          if (n*(n+1)/2 != numCards)  {
80              System.out.println(numCards + " is not triangular");
81              return;
82          }
83          ArrayList<Integer> target = new ArrayList<>();
84          for (int i = 1; i <= n; i++) {
85              target.add(i);
86          }
87          System.out.println("Target: " + target);
88
```

```java
        ArrayList<Integer> start = new ArrayList<>();
        int sum = 0, remaining = numCards;
        Random rand = new Random();

        while (sum < numCards) {
            if (remaining == 1) {
                start.add(1);
                break;
            }
            int current = rand.nextInt(remaining - 1) + 1;
            start.add(current);
            sum +=  current;
            remaining -= current;
        }
        while (!start.containsAll(target)) {
            int totalSubtracted = 0;

            for (int i = start.size() - 1; i >= 0; i--) {
                int newValue = start.get(i) - 1;
                if (newValue == 0) {
                    start.remove(i);
                } else {
                    start.set(i, newValue);
                }
                totalSubtracted++;
            }
            start.add(totalSubtracted);
            System.out.println("Updated: " + start);
        }

        System.out.println("Final Target: " + target);
    }
}
```