```java
import java.util.ArrayList;
import java.util.Random;
public class ArrayListExercises {

 public static void main(String[] args) {

  int n = 66;
  bulgarianSolitaire(n);

 }

 /**
  * Removes all of the strings of even length from the given list
  * @param listOfStrings the list of Strings (list can be empty)
  * @return the given list with all even length strings removed
  */
 public static ArrayList<String> removeEvenLength(ArrayList<String> listOfStrings)  {
  for (int x = 0; x < listOfStrings.size(); x++) {
   if (listOfStrings.get(x).length() % 2 == 0) {
    listOfStrings.remove(x);
    x -= 1;
   }
  }
  return listOfStrings;   // This return statement should be last
 }

 /**
  * Moves the minimum value in the list to the front, otherwise preserving the order of the elements
  * @param listOfIntegers the list of Integers (list cannot be empty)
  * @return the given list with the minimum value in the front (zeroth element)
  */
 public static ArrayList<Integer> minimumToFront(ArrayList<Integer> listOfInts)  {
  int min = listOfInts.get(0);
  for (int x = 0; x < listOfInts.size(); x++) {
   if (min > listOfInts.get(x)) {
    min = listOfInts.get(x);
   }
  }
  listOfInts.remove(listOfInts.indexOf(min));
  listOfInts.add(0,min);
  return listOfInts;   // This return statement should be last
 }

 /**
  * Removes all elements from the given list whose values are in the range min through max (inclusive).
  * If no elements in range min-max are found in the list, the list's contents are unchanged.
  * If an empty list is passed, the list remains empty.  Assume min < max.
  * @param listOfInts the list of Integers (list can be empty)
  * @param min the minimum value in the range
  * @param max the maximum value in the range
  * @return the given list with the range min-max removed
  */
 public static ArrayList<Integer> filterRange(ArrayList<Integer> listOfInts, int min, int max)  {
  listOfInts.removeIf(x -> (x <= max && x >= min));
  return listOfInts;   // This return statement should be last
 }


 /**
  * Models/simulates the game of Bulgarian Solitaire.
  * @param numCards the number of cards to start with; n must be a triangular number (a triangular
  * number is a number that can be written as the sum of the first n positive integers).
  */
 public static void bulgarianSolitaire(int numCards)  {

  // Check if given number of cards is triangular
  int n = (int) Math.sqrt(2*numCards);
  if (n*(n+1)/2 != numCards)  {
   System.out.println(numCards + " is not triangular");
```

```
   return;
  }
  Random y = new Random();
  ArrayList <Integer> variableName = new ArrayList <Integer> ();
  ArrayList <Integer> variableName2 = new ArrayList <Integer> ();
  int sum = 0;
  int last = 0;
  int sum2 = 0;
  int sum3 = 0;
  for (int x = 1; x <= numCards; x++) {
   variableName.add(x-1,y.nextInt(numCards)+1);
   sum += variableName.get(x-1);
   if (sum >= numCards) {
    last = numCards - sum + variableName.get(variableName.size() - 1);
    variableName.set(variableName.size() - 1, last);
    break;
   }
  }
  for (int x = 1; x <= numCards; x++) {
   if (sum2 >= numCards) {
    break;
   }
   sum2 += x;
   variableName2.add(x);
  }
  System.out.println(variableName);
  while (!variableName.containsAll(variableName2)) {
   for (int x = 0; x < variableName.size(); x++) {
    variableName.set(x,variableName.get(x)-1);
    sum3++;
   }
   for (int x = 0; x < variableName.size(); x++) {
    if (variableName.get(x) == 0) {
     variableName.remove(x);
     x--;
    }
   }
   variableName.add(variableName.size(), sum3);
   System.out.println(variableName);
   sum3 = 0;
  }
 }
}
```