```java
import java.util.ArrayList;
import java.util.Random;

public class ArrayListExercises {

  public static void main(String[] args) {

    // You do not need to handle the User Interface (UI).
    // Instead you can run the JUnit test cases found in
ArrayListExercisesTests.java

    bulgarianSolitaire(10);

  }

  /**
   * 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 i=0; i <listOfStrings.size(); i++) {
      if((listOfStrings.get(i)).length() % 2 == 0) {
        listOfStrings.remove(i);
        i--;
      }
    }

    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 currentMin = 0;
    int nextUp = 0;
    int position = 0;

    currentMin = listOfInts.get(0);

    for(int i = 0; i < listOfInts.size() - 1; i++) {
      int j = 0;
      nextUp = listOfInts.get(i+1);
      if(currentMin > nextUp) {
        currentMin = nextUp;
        position = i+1;
      }

    }

    listOfInts.remove(position);
    listOfInts.add(0, currentMin);

    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)  {

    for(int i = 0; i < listOfInts.size(); i++) {
      if(listOfInts.get(i) >= min && listOfInts.get(i) <= max) {
        listOfInts.remove(i);
        i--;
      }
    }

    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;
    }
    //initialize arraylist
    ArrayList <Integer> cards = new ArrayList<>();
    ArrayList <Integer> desiredDeck = new ArrayList<>();

    //generate random number of decks
    Random rand = new Random();
    int numDecks = rand.nextInt(numCards);

    //distribute cards among all decks
    int remainingCards = numCards;


    while (remainingCards > 0) {
      int addCards = rand.nextInt(remainingCards) + 1;
      cards.add(addCards);
      remainingCards-=addCards;
    }


    //before loop: array to check if sorting is done

    int num = numCards;
    int bob = 1;

    while(num > 0) {
      desiredDeck.add(bob);
      num -= bob;
      bob++;

      System.out.println(desiredDeck);
    }

    //subtract one card from each deck and add a new deck sequence

    while(!cards.containsAll(desiredDeck)) {
      int holdSize = cards.size();

      for(int i = 0; i < cards.size();i++) {
        cards.set(i, cards.get(i) - 1);

        //remove zeroes
        if(cards.get(i) == 0) {
          cards.remove(i);
          i--;
        }
      }

      //add changing size of deck
      cards.add(holdSize);

    }

    //print final order of cards
    System.out.println(cards);

  }
}
```