
```
[5, 4, 1]
[4, 3, 3]
[2, 2, 3, 3]
[1, 1, 2, 2, 4]
[1, 1, 3, 5]
[2, 4, 4]
[1, 3, 3, 3]
[2, 2, 2, 4]
[1, 1, 1, 3, 4]
[2, 3, 5]
[1, 2, 4, 3]
```

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
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);

	}


	/**
	 * 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;
            }

            //Create the end goal of the game (when the output is 1, 2,
3, 4, 5, ... k)
            ArrayList<Integer> endGoal = new ArrayList<Integer>();

            int runSum = 0;

            for (int i = 1; runSum + i <= numCards; i++) {
                  endGoal.add(i);
                  runSum += i;
            }

            //Randomly divide the number of cards across into piles given
the number of cards
            ArrayList<Integer> pileOfCards = new ArrayList<Integer>();

            Random rand = new Random();

            int remaining = numCards;

            while (remaining>0) {

                  int pile = rand.nextInt(remaining) + 1;
                  pileOfCards.add(pile);
                  remaining -= pile;

            }

            System.out.println(pileOfCards);

            //Until the array pileOfCards matches endGoal, keep
continuing the pattern
            while (!pileOfCards.containsAll(endGoal)) {

                  int leftovers = 0;


                  for (int i = 0; i<pileOfCards.size(); i++){

                        //Subtract 1 from the number of cards in each pile

                        pileOfCards.set(i, pileOfCards.get(i)-1);

                        //Count up the leftover or removed cards that will
go in the new pile
```

```
                        leftovers ++;

                        //Take out any piles from the array that have 0
cards in them
                        if (pileOfCards.get(i)==0) {
                              pileOfCards.remove(i);
                              i = i-1;
                        }
                  }

                  //Make a new pile with all the removed/leftover cards
                  pileOfCards.add(leftovers);

                  System.out.println(pileOfCards);

                  //Sort the array list so it can effectively be compared
with the array endGoal
                  Collections.sort(pileOfCards);



            }

      }
}
```