```
import java.util.ArrayList;
import java.util.Random;
public class ArrayListExercises {
    public static void main(String[] args) {
        bulgarianSolitaire(10);
        // You do not need to handle the
User Interface (UI).
        // Instead you can run the JUnit test
cases found in
ArrayListExercisesTests.java

    }

    /**
     * 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 = listOfStrings.size() - 1; i
>= 0; i--) {
            if (listOfStrings.get(i).length() %
2 == 0)    {
                listOfStrings.remove(i);
            }

        }


        return listOfStrings;   // This return
statement should be last
    }
```

```java
    /**
     * 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 smallest = listOfInts.get(0);
        int count = 0;
        // if i - 1 is less than i, then update the variable to i - 1
```

```
        for(int i = 1; i < listOfInts.size(); i++)
{
            if(listOfInts.get(i) < smallest) {
            smallest = listOfInts.get(i);
            count = i;
            }
        }
        listOfInts.remove(count);
        listOfInts.add(0, smallest);
        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 = listOfInts.size() - 1; i >= 0; i--) {
```

```
            if (listOfInts.get(i) >= min &&
listOfInts.get(i) <= max)     {
                listOfInts.remove(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");
            }
        ArrayList <Integer> values = new
ArrayList<Integer>();
        ArrayList <Integer> endconfig =
new ArrayList<Integer>();
        Random randy = new Random();
        int count = 0;
        int a = 0;
        int tries = 0;
```

```
        int i = 0;
        int cards = numCards;
        int har = randy.nextInt(cards);
        endconfig.add(1);
        endconfig.add(2);
        endconfig.add(3);
        endconfig.add(4);
    for(int o = numCards; o > 0;) {
        a = randy.nextInt(o) + 1;
        o = o - a;
        values.add(a);
    }
    System.out.println("The start
configuration is: " + values);

while(!values.containsAll(endconfig)) {
            for(i = values.size() - 1; i >= 0;
i--) {
                if(values.get(i) != 0) {
```

```
                values.set(i,values.get(i) - 1);
                count++;
                }
                else {
                    values.remove(i);
                }

            }
            values.add(count);
            System.out.println(values);
            count = 0;
            tries++;
        }
        System.out.println("It took " + tries
+ " tries");
    }
}
```