```
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;
	}

	// Creates an array for the ending configuration so it can be used to check when the
Bulgarian Solitaire game is over
	ArrayList<Integer>endingConfig = new ArrayList<Integer>();
	int sum = 0;
	int p = 1;
	while (sum != numCards) {
		sum += p;
		endingConfig.add(p);
		p++;
	}
	System.out.println("The ending configuration should look like: " + endingConfig);

	// Generates a random number of piles
	Random rand = new Random();
	int piles = rand.nextInt(numCards)+1;
	System.out.print("\nPiles: " + piles);
	System.out.println();

	// Generates random pile sizes for each of the piles generated (creates the starting
configuration)
	int total = 0;
	int num = 0;
	ArrayList<Integer>cardOrder = new ArrayList<Integer>();
	for(int i = 0; i<piles; i++) {
		if(i==piles-1) {
			num = numCards-total;
		}
		else{
			num = rand.nextInt(numCards-(piles-i)-total+1)+1; //creates random
number of cards for each pile
		}
		//System.out.print(num +" ");
		if(num!=0) {
			cardOrder.add(0,num);
		}
		total += num; //updates the total for the random number generator
```

```
        }
        System.out.println("The starting configuration is: " + cardOrder);

        //algorithm
        int stop = 0;
        while(stop == 0) {
                if(cardOrder.containsAll(endingConfig)) {
                        stop = 1;
                }
                else{
                        int size = cardOrder.size();
                        int count = 0;
                        for(int k = 0; k<size ;k++) {
                                int h = cardOrder.get(k);
                                cardOrder.set(k, h-1);
                                if(cardOrder.get(k) == 0) {
                                        cardOrder.remove(k);
                                        k--;
                                }
                                size = cardOrder.size();
                                count++;
                        }
                        cardOrder.add(count);
                        System.out.println(cardOrder);
                }
        }
}
```