Java Computer Programming
Bulgarian Solitaire

In this assignment, you will model the game of Bulgarian Solitaire. The game starts with 45 cards. (They need not be playing cards. Unmarked index cards work just as well.) Randomly divide them into some number of piles of random size. For example, you might start with piles of size 20, 5, 1, 9, and 10. In each round, you take one card from each pile, forming a new pile with these cards. For example, the sample starting configuration would be transformed into piles of size 19, 4, 8, 10, and 5. The solitaire is over when the piles have size 1, 2, 3, 4, 5, 6, 7, 8, and 9, in some order. (It can be shown that you always end up with such a configuration.)

In your program, produce a random starting configuration and print it. Then keep applying the solitaire step and print the result. Stop when the solitaire final configuration is reached.

Object class Pile
- Instance field – int numofcards - number of cards in this pile
- Parameterized Constructor – pass in the number of cards for this pile
- Methods – getter for number of cards
  - Void method to deduct one card

Test Class
- Create arraylist of Pile
- Randomly create piles of cards totaling 45 cards and place into array list
- Display the initial piles with counts
- Start simulation – while not done
  1. Deduct a "card" from each pile
  2. Create a new pile from the number deducted from each existing piles – place in front of arraylist
  3. Check to remove any empty piles (backwards?)
  4. Check to see if finished
- Display final list and number of iterations of simulation

Sample run:

```
13  4  6  6 10  6
 6 12  3  5  5  9  5
 7  5 11  2  4  4  8  4
 8  6  4 10  1  3  3  7  3
 9  7  5  3  9  2  2  6  2
 9  8  6  4  2  8  1  1  5  1
10  8  7  5  3  1  7  4
 8  9  7  6  4  2  6  3
 8  7  8  6  5  3  1  5  2
 9  7  6  7  5  4  2  4  1
 9  8  6  5  6  4  3  1  3
 9  8  7  5  4  5  3  2  2
 9  8  7  6  4  3  4  2  1  1
…….
 9  8  7  6  4  5  3  2  1
 9  8  7  6  5  3  4  2  1
 9  8  7  6  5  4  2  3  1
 9  8  7  6  5  4  3  1  2
 9  8  7  6  5  4  3  2  1
Number of iterations:  52
```