Exercise Description: `CircularQuote`

Booksite web exercise 4.3.2 Write a class `CircularQuote` that mimics the `Quote` class, but uses a circularly-linked list instead of a null-terminated linked list. Its API will be:

```
public CircularQuote()               // constructor - create an empty quote
public void addWord(String w)        // add the word w to the end of the quote
public String toString()             // string representation of the quote
public int count()                   // number of words in the quote
public String circularGetKth(int k)  // the kth word in the quote (k=1 is first
                                     // word. loops around if needed)
```

This exercise will give you practice with `do {} while ()` loops. One such example loop, in the `toString()` method, is already completed for you. Use it again in the constructor and in `count()`.

```java
public class CircularQuote {
   // the first card in the circular linked list
   private Card start;

   // helper linked-list data type
   private class Card {
      private String word;
      private Card next;

      public Card(String word) {
         this.word = word;
         this.next = null;
      }
   }

   // constructor - create an empty quote
   public CircularQuote() {
                    // no card intitially
   }

   // add the word w to the end of the quote
   public void addWord(String w) {
       Card newCard = 

       // degenerate case for empty quote, w is the first word
       if (              ) {
                             // save the card with the new word
          start.next = start; // make it circular
       }

       // otherwise, traverse list until card points to last word
       else {
          // find the current last word
          Card card = start;
          do {

          } while (                    );

          // insert new word






       }
   }

   // string representation of the entire quote
   public String toString(){
      String result = "";
      if (start == null) // special case
         return result;

      Card card = start;
      do {
         result = result + card.word + " "; // build string
         card = card.next; // traverse list
      } while (card != start);
      return result;

      // note! using a plain while loop would normally require separate
      // logic for the 1-node and the (>1)-node case
   }

   // number of words in the quote
   public int count() {









   }

    // the kth word in the quote (where k = 1 is the first word)
    public String circularGetKth(int k) {
       Card card = start;
       for (int j = 1; j < k; j++) {
          card = card.next;
       }
       return card.word;
    }

   // test client
   public static void main(String[] args) {
      CircularQuote q = new CircularQuote();
      StdOut.println(q.count() + ": " + q);

      q.addWord("A");
      StdOut.println(q.count() + ": " + q);

      q.addWord("rose");
      StdOut.println(q.count() + ": " + q);
      StdOut.println("Second word: " + q.circularGetKth(2)); // rose

      q.addWord("is");
      StdOut.println(q.count() + ": " + q);
      StdOut.println("Tenth word: " + q.circularGetKth(10)); // A

      q.addWord("a");
      StdOut.println(q.count() + ": " + q);
      StdOut.println("Seventh word: " + q.circularGetKth(7)); // is

      q.addWord("rose.");
      StdOut.println(q.count() + ": " + q);
      StdOut.println("First word: " + q.circularGetKth(1)); // A
   }
}
```