```java
class recursiveLinkedLists2
{

   static class Node
   {
      String data;
      Node   next;

      Node(String data, Node next)
      {
         this.data = data;
         this.next = next;
      }
   }


   static void printList(Node p)
   {
      if (p != null)
      {
          System.out.println(p.data);
          printList(p.next);
      }
   }


   static Node copy(Node p)
   {
      if(p == null)
         return null;
      else
         return new Node(p.data, copy(p.next));
   }

//////////////////////////////////////////////////////////////////////
// Appending two lists

   /*
    * Appending two lists is a simple way of creating
    * a single list from two. This function adds the
    * second list to the end of the first list
    */

   static Node append(Node p, Node q)
   {
      if(p == null)
         return q;

      else
      {
         p.next = append(p.next, q);
         return p;
      }
   }


//////////////////////////////////////////////////////////////////////
// "Shuffle-Merging" two lists

   /*
    * Here is a more complex function to combine two
    * lists; it simply zips up two lists, taking a
    * node from one, then from the other. The first
    * list in the original call now points to the
    * new list.
    */

   static Node shuffle(Node p, Node q)
   {
      if(p == null)
         return q;

      else if(q == null)
         return p;

      else
      {
         p.next = shuffle(q, p.next);     // Note how we exchange p and q here
         return p;
      }
   }


////////////////////////////////////////////////////////////////////
// Merging two sorted lists

   /*
    * Here is another more complex function that
    * combines two lists; this one merges nodes
    * from two sorted lists, preserving their order.
    */

   static Node merge(Node p, Node q)
   {
      if(p == null)
         return q;

      else if(q == null)
         return p;

      else if(p.data.compareTo(q.data) < 0)
      {
         p.next = merge(p.next, q);
         return p;
      }
      else
      {
         q.next = merge(p, q.next);
         return q;
      }
   }

////////////////////////////////////////////////////////////////////



   static public void main(String[] args)
   {
      Node p, q, output;


      // create a new linked-list of fruit:
      Node fruit =
            new Node("apple",
                new Node("banana",
                   new Node("cherries",
                      new Node("fig",
                         new Node("grapes", null)
                      )
                   )
                )
             );


      // create another linked-list, this time of animals:
      Node animals =
            new Node("aardvark",
                new Node("bat",
                   new Node("cat",
                      new Node("dragon",
                         new Node("elephant", null)
                      )
                   )
                )
             );


      System.out.println("\nAppend Animals into Fruit:");
      p = copy(fruit);
      q = copy(animals);
      output = append(p, q);
      printList(output);

      System.out.println("\nShuffling Animals into Fruit:");
      p = copy(fruit);
      q = copy(animals);
      output = shuffle(p, q);
      printList(output);

      System.out.println("\nMerging Animals into Fruit:");
      p = copy(fruit);
      q = copy(animals);
      output = merge(p, q);
      printList(output);

      System.out.println("\nDone!");
   }


}
```