CSC 383, Sections 401 and 410
Fall, 2011
Assignment 4
Due 11:59pm CT, Wednesday, September 28th

Requirements. You are to write a class called Order that contains a set of generic static methods for computing and returning the k^{th} smallest item in a collection, where $k \geq 0$. For example, if k is 0, the method should return smallest item in the collection; if k is 4, it should return the fifth smallest item.

Each method will be generic in the least restrictive way. I will explain this in class and provide a pseudocoded algorithm. I will also provide a test program.

Your class will contain three methods:

1. getByOrder: This will be a public method with the signature:

   ```
   public static <T extends Comparable<? super T>>
   T getByOrder(Collection<T> collection, int k)
   ```

   This method, when called with a collection as its first argument and an integer k as its second, returns the k^{th} item in the natural ordering of the elements in the collection.

2. getByOrderOnArrayList: This will be a private method with the signature:

   ```
   private static <T extends Comparable<? super T>>
   T getByOrderOnArrayList(ArrayList<T> arrayList, int k)
   ```

   This method, when called with an array list as its first argument and an integer k as its second, returns the k^{th} item in the natural ordering of the elements in the array list.

3. partitionArrayList: This will be a private method with the signature:

   ```
   public static <T extends Comparable<? super T>>
   int partitionArrayList(ArrayList<T> arrayList)
   ```

   This method, when called with an array list, selects a pivot element in it, partitions the array list so that elements less than the pivot are in the left part of the array list, elements greater than the pivot are in the right part, and the pivot is between them. It returns the position of the pivot in the modified array list.

Pseudocode for each method

```
getByOrder(collection, k) {
  if k is not in the correct range {
    throw a no such element exception
  }
  copy the collection to an array list
  call getByOrderOnArrayList(array list, k)
}

getByOrderOnArrayList(array list, k) {
  pivot position = partitionArrayList(array list)
  if pivot position == k {
    return element at position k of array list
  }
  else if pivot position < k {
    create a sublist from the elements in array list
      from pivot position + 1 to the end
    return getByOrderOnArrayList(sublist,
                                 k-pivot position-1)
  }
  else { // pivot position > k
    create a sublist from the elements in array list
      from 0 to pivot position - 1
    return getByOrderOnArrayList(sublist, k)
  }
}

partitionArrayList(array list) {
  set pivot to last element in the array list
  remove the last element from the array list
  create an empty array list called lesser values
  create an empty array list called greater values
  for each element in the array list {
    if the element is less than the pivot {
      add it to lesser values
    }
    else {
      add it to greater values
    }
  }
  set pivot position to the size of the lesser values list
  clear the original array list
  add all of the elements in lesser values to array list
  add the pivot to the array list
  add all of the elements in greater values to array list
```

```
  return the pivot position
}
```

Submit a source file called `Order.java`.

Grading rubric: This assignment is worth 30 points, with points assigned as follows:

- Variable and method names descriptive and mnemonic (4 points)
- Comment block with information specified (4 points)
- The test program, using your Order class, runs correctly to completion and prints correct output (20 points)
- Code is properly indented (2 points)

Beyond the above rubric, 2 points will be deducted for each missed requirement. If I say to do something a certain way, do it that way.