```
/*********************************************************
//  This is an implementation of quicksort in Java.
//
//  It wants a filename followed by a number of elements
//  which must be comparable by a simple < = or >
//  So, it is called like this:
//      java QSort /path/to/file number
//
//  Jordan Thomas
//  February 2007
/*********************************************************/

import java.io.*;

public class QSort
{

        public static long count = 0;
        public static int[] x;
        //public static long depth = 0;
        //depth was used when I was finding out all about putting
        //too many calls on the stack, which was less than fun.
        //
        //This was eventually solved by calling java like so:
        //  java -Xss300M QSort file number

        //pass in how many numbers and a filename
        public static void main(String[] args)
        {

                //have to use an Integer AND and int because
                //java will not let me dereference an Integer
                Integer temp = Integer.parseInt(args[1]);
                int length = (temp).intValue();
                String file = args[0];
                readFromDisk(file, length);     //load up our array with all our numbers

                System.out.println("*******************************");
                System.out.println("QuickSort of " + length + " numbers:");
                System.out.println("_____________________________________");
                System.out.println("Started sort at  :  " + Time.getDate() + "   ||");
//              try{
                        QSort(0, length - 1);
//              }
//              catch(StackOverflowError SOE)
//              {
                        //remnances of a fun time.  **shudder**
//                      System.out.println("depth is" + depth);
//              }
                //make sure the file was indeed sorted correctly
//              for(int i = 0; i < length; i++)
//              {System.out.println(x[i]);}

                System.out.println("Finished sort at :  " + Time.getDate() + "   ||");
                System.out.println("_____________________________________");
                System.out.println("comparisons:  " + count );
                System.out.println("\n\n");

                //added so user has the option to put the now sorted numbers in a file
                if(args.length >= 3)
                {
                        WriteFile.write(x, length, args[2]);
                }
        }

//      recursive quicksort algorithm follows.
//  Basically, it picks a pivot (first element in range)
```

```
//  and then proceeds to put it where it belongs.
//  This happens recursively on each half of the array
//  past the placed pivot and the result is a sorted array.
public  static void QSort(int a, int b)
{

                int left = a;   //left pointer
                int right = b;  //right pointer
                if (left >= right)
                {               //we don't want to sort with pointers moving the wrong direction,
                                //as it would cause an infinite loop, and this is when we are done.
                        return;
                }
                int pivot = x[left];
                while (left < right)
                {//partition
                        count++;
                        while (left < b && x[left] < pivot)
                        {
                                count++;
                                left++;  //move the left pointer to the right if the value we are checking
                                        // (x[left]) is less than the pivot value mid
                        }
                        while (right > a && x[right] >= pivot)
                        {
                                count++;
                                right--;  //similarly, we move the right pointer left if the item it points
                                          //to is greater than our pivot because this value belongs to the
                                          //right of the pivot after sorting.
                        }
                        if (left < right)
                        {
                                //after the two while loops above, we have a situation in which left is not less
                                //than the pivot and right is not greater than the pivot.
                                //We need to swap these values if the left and right pointers still have not met.
                                int temp =x[left];
                                x[left] = x[right];
                                x[right] = temp;
                        }
                }
                if (right < left)
                {
                        //swap left and right
                        int temp = right;
                        right = left;
                        left = temp;
                }
                QSort(a, left);
                QSort(left + 1, b);

                //printStudents();
}

        //simply reads from the file and parses an int out and puts it into our
        //array for sorting
        public static void readFromDisk(String filename, int length)
        {
                try
                {
                FileReader fr = new FileReader(filename);//Reads the file name
                BufferedReader br = new BufferedReader(fr);//Checks the file.
                String line = br.readLine();//Reads first line of text.

                x = new int[length];
                int i = 0;
                Integer readFromLine = Integer.parseInt(line);
```

```
                while(line != null)//while there is something on the line.
                {
                        x[i] = readFromLine.intValue();
                        line = br.readLine();
//                      System.out.println(line);
                        i++;
                        readFromLine = Integer.parseInt(line);
                }
                br.close();//Closes the buffer.
                }
                catch(FileNotFoundException fnfe)
                {
                        System.out.println("bad file name");
                }
                catch(IOException ioe)
                {
                        System.out.println("Input / Output Exception found:\n\r"+ioe.getMessage());
                }
                catch(NumberFormatException nfe)
                {
//                      System.out.println("done\n");
                }

        }
}
```