/\*\*

A quick sort algorithm that uses recursion to

sort an array from low to high numbers.

@author Pj Kim

\*\*/

import java.util.Random;

public class QuickRecursion

{

public static void main(String[] args)

{

int[] array = new int[10];

Random randomObj = new Random();

//fills each index of array with random numbers.

for (int i = 0; i < array.length; i++)

{

array[i] = randomObj.nextInt(1000);

System.out.print(array[i] + " ");

}

System.out.println();

quickSort(array, 0, array.length-1);

//prints out the sorted array

for (int j = 0; j < array.length; j++)

{

System.out.print(array[j] + " ");

}

}

public static void quickSort(int[] array, int leftPointer, int rightPointer)

{

if (leftPointer < rightPointer) //base case, if leftPointer equals rightPointer, sorting is done

{

int pivotPoint = sort(array, leftPointer, rightPointer); //sorts the array and returns pivot position

quickSort(array, leftPointer, pivotPoint - 1); //left side of array is sorted

quickSort(array, pivotPoint + 1, rightPointer); //right side of array is sorted

}

}

public static int sort(int[] array, int leftPointer, int rightPointer) //sorts the array according to the pointers and a pivot

{

int pivot = array[rightPointer];

int index = leftPointer - 1;

index = swap(array, rightPointer, leftPointer, index, pivot); //sorts the array about the pivot and returns the index.

//swap the first element greater than the pivot with the pivot.

array[rightPointer] = array[index + 1];

array[index + 1] = pivot;

return index + 1; //return the location of the pivot

}

//Performs checks and swaps until the array has been sorted about the pivot.

public static int swap(int[] array, int rightPointer, int i, int index, int pivot)

{

if (array[i] <= pivot) //Swap elements if they are less than the pivot.

{

index++;

int temp = array[index];

array[index] = array[i];

array[i] = temp;

}

i++;

if (i != rightPointer) //when i < rightPointer, more checking and swapping needs to be performed

{

index = swap(array, rightPointer, i, index, pivot);

}

return index;

}

}