/\*\*

A selective sort algorithm that uses recursion to

sort an array of low to high numbers.

@author Pj Kim

\*\*/

import java.util.Random;

public class SelectiveRecursion

{

public static void main(String[] args)

{

int[] array = new int[10];

Random randomObj = new Random(); //fills each element of the 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();

Sort(array, 0);

}

/\*\*

This method will sort the values in the array recursively

and then print them out.

\*\*/

public static void Sort(int[] array, int j)

{

int minSize = 1000;

int location = -1;

int temp;

if (j == array.length - 1) //Base case, when method scans last element of array.

{

System.out.print(array[j] + " "); //array is already sorted, so last element is just printed.

}

else

{

location = findMin(array, j, minSize, location); //Calls another method to find the location of the smallest number.

minSize = array[location];

//Puts smallest value into the first unsorted element.

temp = array[j];

array[j] = minSize;

array[location] = temp;

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

j++;

Sort (array, j);

}

}

/\*\*

A method that will find the location of the smallest value and return it.

\*\*/

public static int findMin(int[] array, int j, int minSize, int location)

{

if (j == array.length - 1) //Base case, when it scans the last element in the array.

{

if (array[j] < minSize)

{

location = j;

}

return location;

}

if (array[j] < minSize)

{

minSize = array[j];

location = w;

}

j++;

return findMin(array, j, minSize, location);

}

}