// Kelvin Kellner

// Unit 4 Test - Sorting Algorithms

// Mrs. Cooper

// April 17th, 2019

import java.util.Random;

public class Unit4SortingTest {

public static void main(String[] args) {

// Lotto 649!

// LOTTERY VALUES

int min = 1;

int max = 49;

int amount = 200;

// Generate an array of 200 random numbers from 1-49

int[] nums = fillRandoms(amount, min, max);

int[] tally = new int[max-min+1]; // The count for each lottery value

int[] value = new int[tally.length]; // Each lottery value in order

// Count the number of occurrences for each value from our 200 random numbers

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

tally[nums[i]-min]++;

// Fill Value array with the numbers 1-49

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

value[i]=min+i;

// Sort the value and tally arrays from the greatest to lowest order of the values in the tally array

sortBy(tally, value);

// Print the tally of each value in order

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

{

//if(tally[i]>0) // Re-enable this line to hide numbers that did not appear at all (0 times)

System.out.println(value[i] + " - " + tally[i] + " times");

}

} // Close Main Method

// Sort By Method

// Sorts two arrays based on the order of the first array.

// Sort arrays A and B in the sorted order of array A.

public static void sortBy(int[] basis, int[] linked)

{

// Call the overloaded method for the range of the whole array

sortBy(basis, linked, 0, basis.length-1);

} // Close First Sort By Method (Of Two)

public static void sortBy(int[] basis, int[] linked, int x, int y)

{

// USES QUICK SORT!

if(y>x)

{

// Note: we could add some extra code here to select the median of 3 randoms numbers within our range to be our pivot,

// but since the numbers we are working with are randomly generated either way, this is unnecessary, as we know our array is not already sorted (or reverse sorted).

int p = y; // Pivot

int s=x-1; // Smallest

for(int i=x;i<y;i++)

{

if(basis[i]>basis[p]) // a > b: greatest->lowest, a < b: lowest->greatest

// NOTE: swapping the > or < sign above is the only difference between incremental or reverse sorting.

// An "increasing" boolean could easily be added to make this method more reusable, however this is not necessary for this particular program.

{

// Swap the values in both arrays together

swap(basis, i, s+1);

swap(linked, i, s+1);

s++;

}

}

// Swap the pivot to the middle of both partitions (for both arrays)

swap(basis, s+1, p);

swap(linked, s+1, p);

// Repeat this process on both partitions around the pivot

sortBy(basis,linked,x,s);

sortBy(basis,linked,s+2,y);

}

} // Close Second Sort By Method

// Swap Method

// Swaps two values in an array of integers.

// Swap values B and C in array A.

public static void swap(int[] a, int x, int y)

{

int temp = a[x];

a[x] = a[y];

a[y] = temp;

} // Close Swap Method

// Fill Randoms Method

// Returns an array of A integers from B to C.

public static int[] fillRandoms(int amount, int min, int max)

{

Random random = new Random();

int[] rand = new int[amount];

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

rand[i] = random.nextInt(max-min+1)+min;

return rand;

} // Close Fill Randoms Method

// Print Method

// Prints all the values of an array in one line (most useful for bug testing)

public static void print(int[] a)

{

if(a.length>0)

{

String line = "";

for(int i=0;i<a.length-1;i++)

line+=a[i] + " ";

line+=a[a.length-1];

System.out.println(line);

}

} // Close Print Method

} // End Class