// Kelvin Kellner

// Mrs. Cooper

// Unit One Test - Better Methods & Method Overloading

// 21 February 2019

/\*

Instructions:

1) There are 3 methods you need to improve/make more efficient:

createRandArray, findQuantity, orderArray. You MAY WISH add to or modify

printIntArray or create other similar overload methods for that for testing but

it is not required.

2) You are to improve these methods in reusability and efficiency.

Make them more generalized to use in a wider variety of situations

3) You are also to create a new method also called findQuantity that finds the

quantity of matching strings in a given string array

4) 4++ Challenge: Can you create a new method also called orderArray that will put

four integer parameters (instead of three) in an ordered array? For highest marks you

should do this more efficiently than just using a lot more "else if" statements.

5) Comment your code/changes as appropriate and discussed in class.

\*/

import java.util.\*;

public class KelvinKellnerUnit1Quiz2

{

public static void main(String[]args)

{

int [] a = createRandArray(20, 1, 10); // Added parameters for array size and range of random integers (min and max in no particular order)

printArray(a); // method overloading allows us to not need to say "printIntArray". since the array is an int, it will run the integer "printArray" automatically

/\*

Find Quantity Edits: We may not always wish to prompt the user for a number to search,

sometimes our computer may want to run it's own searches without notifying the user.

For this reason, we use the lookFor integer as a parameter, and ask the user within

our main method. Additionally, we may not always want to print out the count to

the Console, so we simply return the count from the method and print it in Main.

\*/

Scanner input = new Scanner(System.in);

System.out.println("\nWhat number should I search for?");

int lookFor = input.nextInt();

int count = findQuantity(a, lookFor); // look for the number of occurrences of the number the user wishes to search for

System.out.println("There are " + count + " occurrences of " + lookFor + " in the array.");

input.close(); // close scanner

/\*

Order Array Edits: rather than limiting this method to sorting a NEW array of 3 ints,

I used a simple sorting algorithm to reorder any integer array given as a parameter,

regardless of length. This is more convenient, because we may not always have 3 values,

nor might we always want to create a new array.

\*/

int[] myOrderedArray = {5, 2, 18, 21, 3, 57, 4, 12, 99, 4}; // change these values, try it out!

orderArray(myOrderedArray);

System.out.println("\n\n"); // good spacing ;)

printArray(myOrderedArray); // let's print the array too, just so it has a purpose

// Just to be extra, I added an orderArray overload that alphabetically orders a String array, the old fashioned way (no Arrays.sort(), hehe)

System.out.println("\n\n"); // more spacing!

String[] myOrderedStrings = {"Jacob", "Kelvin", "Cameron", "Brian", "Lauren", "Soy", "Sara", "Ela", "Harvir"}; // change these values, try it out!

orderArray(myOrderedStrings);

printArray(myOrderedStrings);

}

//method to print an integer array

public static void printArray(int[] myArray)

{

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

System.out.println(myArray[i]);

}

//method to print a string array (method overloading!)

public static void printArray(String[] myArray)

{

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

System.out.println(myArray[i]);

}

//method to create an array of 20 random integers from 1 to 10

// EDIT: method to create an array of x random integers from y/z to z/y ;)

public static int[] createRandArray(int size, int range1, int range2)

{

// new variables to define the minimum and maximum range of random numbers

int min = 0;

int max = 0;

// find which input is the max and which is the min

if(range1>range2)

{

max=range1;

min=range2;

}

else

{

max=range2;

min=range1;

}

int[] newArray = new int[size]; // use the size given as a parameter

for (int i=0; i<newArray.length; i++) // use array length so no errors!

{

newArray[i]= (int)(Math.random()\*max)+min; // use max and min to give a number within the given range

}

return newArray; // return our new array

}

//method to prompt user for value and search given array for number of occurences

public static int findQuantity(int[] toSearch, int lookFor)

{

int count = 0;

for (int i=0; i<20; i++)

{

if(toSearch[i]==lookFor)

count++;

}

return count;

}

// METHOD OVERLOADING: findQuantity for a String array as well

public static int findQuantity(String[] toSearch, int lookFor)

{

int count = 0;

for (int i=0; i<20; i++)

{

if(toSearch[i].equals(lookFor)) // use array.equals(String) instead

count++;

}

return count;

}

//method to sort the order of an integer array

public static void orderArray(int[] array)

{

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

{

for(int j=array.length-1;j>i;j--)

{

if(array[i]>array[j])

moveValue(array, i, j);

}

}

}

// PLACE IN ARRAY: method used to move a value in an array (x) from a previous index (y) into a new index (z) and shift the rest of the array accordingly

public static void moveValue(int[] array, int oldIndex, int newIndex)

{

int value = array[oldIndex]; // store the value we want to move (since we are going to overwrite it)

for(int i=oldIndex; i<newIndex; i++) // shift all of the values back one position up until our new position (overwriting our original position's value)

array[i]=array[i+1];

array[newIndex]=value; // restore the value and place it into it's new position

}

//method to sort the order of a string array alphabetically (oh yes, I'm going there)

public static void orderArray(String[] array)

{

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

{

for(int j=array.length-1;j>i;j--)

{

if(array[i].length()>0) // blank Strings will be placed on the front of the array in else statement

{

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

{

if(k<array[j].length()) // prevent comparing a character to a character that doesn't exist

{

if(array[i].charAt(k)>array[j].charAt(k)) // if the value of the nth character (starting from the first character) is later in the alphabet, then move our String forward to that position

{

moveValue(array, i, j);

k=array[i].length(); // end the loop, since the String has already been moved

}

else if(array[i].charAt(k)!=array[j].charAt(k)) // if they are not the same character, don't bother to keep comparing the Strings. we know to move on to the next String. we only want to check if the first k values are the same, in case the Strings are similer (e.g. if we input James and Jason, we want to keep searching up until the 'm' and 's')

{

k=array[i].length(); // end loop, the String should not be moved here

}

}

}

}

else

moveValue(array, i, 0); // move to the very beginning of the array if String is blank

}

}

}

//PLACE IN ARRAY (STRING EDITION): method used to move a value in an array (x) from a previous index (y) into a new index (z) and shift the rest of the array accordingly

public static void moveValue(String[] array, int oldIndex, int newIndex)

{

String value = array[oldIndex]; // store the value we want to move (since we are going to overwrite it)

for(int i=oldIndex; i<newIndex; i++) // shift all of the values back one position up until our new position (overwriting our original position's value)

array[i]=array[i+1];

array[newIndex]=value; // restore the value and place it into it's new position

}

}//end class