**Selection Sort**

1. Create a method selectionSort that uses a selection sort to arrange an array of double values in ascending order. Create a main method in the class to test the method.

| /\*  \* Program name: SelectionSort.java  \*  \* By: Lucas Chow (Last edited: 2022-10-3)  \*  \* ICS4U1: Selection Sort  \*  \* This method uses selection sort: loops through array, remembers lowest item, swapping that with value in position i  \*  \* It is o(n^2) REGARDLESS of sorting, with minimized swapping. The program also tests the method  \*  \*/  public class SelectionSort{  /\*  \* void selectionSort(int[] inputArr)  \*  \* int[] inputArr - the inputed array  \*  \* This method uses selection sort to sort the inputed array.  \* Selection sort loops through the array, (o(n^2)), and for every value, finds the biggest  \* value in array of the sub-array preceding it  \*  \*  \*/  public static void selectionSort(int[] inputArr)  {  int maxValue;  int indexOfMax;  int temp;    for (int i = 0; i < inputArr.length-1; i++) {  maxValue = inputArr[i];  indexOfMax = i;  for (int x = i; x < inputArr.length; x++)  {  if (inputArr[x]<maxValue)  {  maxValue = inputArr[x];  indexOfMax = x;  }  }    //swap  temp = inputArr[i];  inputArr[i] = inputArr[indexOfMax];  inputArr[indexOfMax] = temp;    }    }  public static void main(String[] args)  {    //creating test array, where array is sorted backwards (approximately, not completely opposite)  int[] testArr = {1,2,3,43,6,3,6,99,101,103,102};    //precondition of program  System.out.print("Array before: {");  for (int i = 0; i < testArr.length; i++)  {  System.out.print(testArr[i]+",");  }  System.out.println("}");    //performing insertion sort  selectionSort(testArr);    //post-condition of the program  System.out.print("Array After: {");  for (int i = 0; i < testArr.length; i++)  {  System.out.print(testArr[i]+",");  }  System.out.println("}");  }  } |
| --- |

1. If a selection sort were to be used to sort the data shown below in alphabetical order, show the data after each pass of the sort.

| Renee | Brien | Vincent | Doris | Scarlett |
| --- | --- | --- | --- | --- |

| Pre-Condition:  Renee Brien Vincent Doris Scarlett  First Run  Brien Renee Vincent Doris Scarlett  Second Run  Brien Doris Vincent Renee Scarlett  Third Run  Brien Doris Renee Vincent Scarlett  Fourth Run  Brien Doris Renee Scarlett Vincent  (I made the program stop on the second last element, as it implies the array is sorted) |
| --- |

1. Write a program that asks user to input 10 random strings then sorts the list using selection sort and outputs the sorted list

| /\*  \* Program name: SequentialSortRandomStrings.java  \*  \* By: Lucas Chow (Last edited: 2022-)  \*  \* ICS4U1: SequentialSort  \*  \* This program finds creates 10 random strings, then sorts the strings alphabetically using selection sort  \*  \*  \*/  public class SelectionSortRandomStrings{  /\*  \* void alphabeticalSelectionSort(String[] inputArr)  \*  \* String[] inputArr -> the inputed array  \*  \* This method returns the string array sorted alphabetically  \*  \*  \*/  public static void alphabeticalSelectionSort(String[] inputArr)  {    String maxValue;  int indexOfMax;  String temp;    for (int i = 0; i < inputArr.length-1; i++)  {  maxValue = inputArr[i];  indexOfMax = i;  for (int x = i; x < inputArr.length; x++)  {  if (inputArr[x].compareToIgnoreCase(maxValue) < 0)  {  maxValue = inputArr[x];  indexOfMax = x;  }  }    //swap  temp = inputArr[i];  inputArr[i] = inputArr[indexOfMax];  inputArr[indexOfMax] = temp;  }  }  public static void main(String[] args)  {  //creating random strings  String chars = "abcdefghijklmnopqrstuvwxyz";  final int STRING\_ARRAY\_LENGTH = 10;  String[] stringArray = new String[STRING\_ARRAY\_LENGTH];    //creating random words  for (int i = 0; i < STRING\_ARRAY\_LENGTH; i++)  {  stringArray[i] = "";  for (int x = 0; x < Math.random()\*10+1; x++)  {  stringArray[i] += chars.charAt((int)(Math.random()\*26));  }  }    //precondition of program  System.out.print("Array before: {");  for (int i = 0; i < stringArray.length; i++)  {  System.out.print(stringArray[i]+",");  }  System.out.println("}");    //performing insertion sort  alphabeticalSelectionSort(stringArray);    //post-condition of the program  System.out.print("Array After: {");  for (int i = 0; i < stringArray.length; i++)  {  System.out.print(stringArray[i]+",");  }  System.out.println("}");  }  } |
| --- |

.

1. In our version of selection sort, if the largest item is already at right location, the algorithm still swaps that value with itself, even though it is not necessary.
   1. How could the algorithm be changed to avoid this unnecessary swapping?

This unnecessary swapping can easily be prevented by

stopping short the algorithm at i < inputArr.length-1

* 1. Why might it better to leave the algorithm as it is?

The main reason I see of leaving this algorithm be is just the fact that this

extraneous case is insignificant for our purposes, and logically it makes

more intuitive sense to have it not consider this case. Modifications to the

selection sort, etc will be more complication. The method is inherently a

strictly o(n^2) algorithm, it’s plain unnecessary.

1. Sometimes we are only interested in knowing the values that would occupy one end of the list if the list were sorted. As an example, we may want to know the scores of only the top ten competitors in a contest. Modify the selection sort so that instead of sorting the entire array, it puts the *k* largest values in order in the last *k* positions in the array. The value of *k* should be a parameter of the method.

| public static void modifiedSelectionSort(int[] inputArr, int k)  {  int maxValue;  int indexOfMax;  int temp;    for (int i = inputArr.length-1; i > inputArr.length-1-k; i--)  {  maxValue = inputArr[i];  indexOfMax = i;  for (int x = i; x > -1; x--)  {  if (inputArr[x]<maxValue)  {  maxValue = inputArr[x];  indexOfMax = x;  }  }    //swap  temp = inputArr[i];  inputArr[i] = inputArr[indexOfMax];  inputArr[indexOfMax] = temp;    }    } |
| --- |