Joao Paulo Dos Santos Ferreira

Fundamentals of Programing II – CSIT 112\_04

Professor Dajin Wang

April 28, 2018

**Programing Project 4/23/2018**

**1. PP 10.4 -** Rewrite the Sorting class so that both sorting algorithms put the values in descending order. Create a driver class with a main method to exercise the modifications.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Sorting.java Author: Lewis/Loftus

//

// Demonstrates the selection sort and insertion sort algorithms.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **class** Sorting

{

//-----------------------------------------------------------------

// Sorts the specified array of objects using the selection

// sort algorithm.

//-----------------------------------------------------------------

**public** **static** **void** selectionSort (Comparable[] list)

{

**int** max;

Comparable temp;

**for** (**int** index = 0; index < list.length-1; index++)

{

max = index;

**for** (**int** scan = index+1; scan < list.length; scan++)

**if** (list[scan].compareTo(list[max]) > 0)

max = scan;

// Swap the values

temp = list[max];

list[max] = list[index];

list[index] = temp;

}

}

//-----------------------------------------------------------------

// Sorts the specified array of objects using the insertion

// sort algorithm.

//-----------------------------------------------------------------

**public** **static** **void** insertionSort (Comparable[] list)

{

**for** (**int** index = 1; index < list.length; index++)

{

Comparable key = list[index];

**int** position = index;

// Shift smaller values to the right

**while** (position > 0 && key.compareTo(list[position-1]) > 0)

{

list[position] = list[position-1];

position--;

}

list[position] = key;

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Contact.java Author: Lewis/Loftus

//

// Represents a phone contact.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **class** Contact **implements** Comparable

{

**private** String firstName, lastName, phone;

//-----------------------------------------------------------------

// Constructor: Sets up this contact with the specified data.

//-----------------------------------------------------------------

**public** Contact (String first, String last, String telephone)

{

firstName = first;

lastName = last;

phone = telephone;

}

//-----------------------------------------------------------------

// Returns a description of this contact as a string.

//-----------------------------------------------------------------

**public** String toString ()

{

**return** lastName + ", " + firstName + "\t" + phone;

}

//-----------------------------------------------------------------

// Returns a description of this contact as a string.

//-----------------------------------------------------------------

**public** **boolean** equals (Object other)

{

**return** (lastName.equals(((Contact)other).getLastName()) &&

firstName.equals(((Contact)other).getFirstName()));

}

//-----------------------------------------------------------------

// Uses both last and first names to determine ordering.

//-----------------------------------------------------------------

**public** **int** compareTo (Object other)

{

**int** result;

String otherFirst = ((Contact)other).getFirstName();

String otherLast = ((Contact)other).getLastName();

**if** (lastName.equals(otherLast))

result = firstName.compareTo(otherFirst);

**else**

result = lastName.compareTo(otherLast);

**return** result;

}

//-----------------------------------------------------------------

// First name accessor.

//-----------------------------------------------------------------

**public** String getFirstName ()

{

**return** firstName;

}

//-----------------------------------------------------------------

// Last name accessor.

//-----------------------------------------------------------------

**public** String getLastName ()

{

**return** lastName;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//PhoneList.java Author: Lewis/Loftus

//

//Driver for testing a sorting algorithm.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **class** PhoneList

{

//-----------------------------------------------------------------

// Creates an array of Contact objects, sorts them, then prints

// them.

//-----------------------------------------------------------------

**public** **static** **void** main (String[] args)

{

Contact[] friends = **new** Contact[8];

friends[0] = **new** Contact ("John", "Smith", "610-555-7384");

friends[1] = **new** Contact ("Sarah", "Barnes", "215-555-3827");

friends[2] = **new** Contact ("Mark", "Riley", "733-555-2969");

friends[3] = **new** Contact ("Laura", "Getz", "663-555-3984");

friends[4] = **new** Contact ("Larry", "Smith", "464-555-3489");

friends[5] = **new** Contact ("Frank", "Phelps", "322-555-2284");

friends[6] = **new** Contact ("Mario", "Guzman", "804-555-9066");

friends[7] = **new** Contact ("Marsha", "Grant", "243-555-2837");

System.***out***.println("Selection Sorting: ");

Sorting.*selectionSort*(friends);

**for** (Contact friend : friends)

System.***out***.println (friend);

System.***out***.println("\nInsertion Sorting: ");

Sorting.*insertionSort*(friends);

**for** (Contact friend : friends)

System.***out***.println (friend);

}

}

**2.** Write a new main method (modify PhoneList.java) that generates and sorts an array of 8 Integers, instead of Contacts.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// PhoneList.java Author: Lewis/Loftus

//

// Driver for testing a sorting algorithm.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**import** java.util.Random;

**public** **class** PhoneList2

{

//-----------------------------------------------------------------

// Creates an array of Contact objects, sorts them, then prints

// them.

//-----------------------------------------------------------------

**public** **static** **void** main (String[] args)

{

// Creates an array of integers and sorts them

// in descending order

Integer[] numbers = **new** Integer[8];

Random rand = **new** Random();

**for** (**int** i = 0; i < 8; i++)

{

numbers[i] = rand.nextInt(100);

}

System.***out***.println("Insertion Sorting: ");

Sorting.*insertionSort*(numbers);

**for** (Integer number : numbers)

System.***out***.println (number);

System.***out***.println("\nSelection Sorting: ");

Sorting.*selectionSort*(numbers);

**for** (Integer number : numbers)

System.***out***.println (number);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Sorting.java Author: Lewis/Loftus

//

// Demonstrates the selection sort and insertion sort algorithms.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **class** Sorting

{

//-----------------------------------------------------------------

// Sorts the specified array of objects using the selection

// sort algorithm.

//-----------------------------------------------------------------

**public** **static** **void** selectionSort (Comparable[] list)

{

**int** max;

Comparable temp;

**for** (**int** index = 0; index < list.length-1; index++)

{

max = index;

**for** (**int** scan = index+1; scan < list.length; scan++)

**if** (list[scan].compareTo(list[max]) > 0)

max = scan;

// Swap the values

temp = list[max];

list[max] = list[index];

list[index] = temp;

}

}

//-----------------------------------------------------------------

// Sorts the specified array of objects using the insertion

// sort algorithm.

//-----------------------------------------------------------------

**public** **static** **void** insertionSort (Comparable[] list)

{

**for** (**int** index = 1; index < list.length; index++)

{

Comparable key = list[index];

**int** position = index;

// Shift smaller values to the right

**while** (position > 0 && key.compareTo(list[position-1]) > 0)

{

list[position] = list[position-1];

position--;

}

list[position] = key;

}

}

}