**Further Object- Oriented Programming**

U08026: Week 9

Sorting

David Lightfoot

Department of Computing and Communication Technologies

# Practical: Sorting

# Exercise 1 - Setup

# Extract the code provided, organise your folders, and get a project set up as usual in *NetBeans*. You should now see the following files in the project:

# • The class *Sequence* manages a sequence of integers of capacity *max*. It also contains methods *insertionSort* and *quickSort*.

# • The class *SortExplore* is the main program: it initialises a sequence to random values and then sorts the sequence.

# Within *SortExplore*, try changing the value of numElements to different values (less than about 200,000 is best), watching how long it takes for each sort.

# Exercise 2

# Count the number of steps that *insertionSort* takes, in the following way:

# Add code to the *insertionSort* method to increase the value of the variable *stepCount* each time an element of the sequence is *accessed*.

# (To be clear, accessing an element of the sequence could be when an element of the array is inspected or copied, or when a value is written into a position in the array.)

# Also add a comment at each place you increase the value of the variable *stepCount*, to make it clear which access of the sequence it is that you are counting.

# Run the main program and note how many comparisons are done by the insertion sort.

# Exercise 3

# Similarly, also add statement(s) to the *quickSort* method to count the number of steps that it takes:

# Add code to the *quickSort* method to increase the value of the variable *stepCount* each time an element of the sequence is accessed.

# Also add a comment at each place you increase the value of the variable *stepCount*, to make it clear which access of the sequence it is that you are counting.

# Run the main program and note how many comparisons are done by quicksort.

# Exercise 4

# Add statements to the main program so that your sequence is sorted twice by each method – that is, for each sorting algorithm the already-sorted sequence is sorted again and the number of comparisons is counted afresh.

# Append the content of the output window to your source text for *SortExplore* in a comment /\* … \*/ and also include some brief comments on the differences / similarities between the two sorting algorithms: Which is quicker (in terms of the number of comparisons)? What difference does it make the second time round? Why?

# What you have to submit

# Please paste your S*equence.java* and *SortExplore.java* files into a *Word* document (including the contents of the output window as pasted into *Tester.java*).

# Zip your source files. Upload *Word* and *zipped* to Moodle.