COMP1102/8702 – Practical Class 10

Aims and Objectives

The idea of the practical is to experiment with sorting both by observing the process as displayed by an application and by making modifications to that application to change the way sorting is performed or to output related information.

Background

The starting point for the practical is an Application, however the structure and function of the Application is not of importance to the practical. It demonstrates and implements a bubble sort algorithm.

Getting Started

Start *IntelliJ* and open the project "Practical10" (download it from FLO).

Run the program.

Once the application window has appeared and the application started, click on the *Sort* button. The application will display each step in sorting 6 numbers. Each time a value is copied from one place to another it will appear with a green background momentarily, the destination will then be displayed with a blue background.

The values to be sorted can be generated automatically by clicking on the *Reset* button, cleared by clicking the *Clear* button or entered manually.

Task 1

Modify the method `bubbleSort`, in `Sort.java`, so that the values are sorted into descending order (largest to smallest), rather than ascending order.

Note that the numbers to be sorted are stored as strings so they must be converted to integers before they can be compared (the code already does this).

Also, the values must be swapped using the method `assign` (it not only assigns the values but performs other functions).

Hint: only one character in the method needs to be modified to complete the task!

Checkpoint 47

Have the program source code and output marked by a demonstrator

Task 2

1. Modify the method `bubbleSort` so that it calculates and outputs, in the application's status label which appears at the bottom of the application's window, the number of swaps performed during sorting.

To count the number of swaps you will need to define an integer variable and initialise it to 0. It should be increased by one each time a swap is performed.

The output should be generated by modifying the following line which appears at the end of the method `bubbleSort`:

```
showStatus("Sort complete");
```

Compile and run the application. After clicking the Sort button and once sorting is complete a line of the following form should appear at the bottom of the application's window:

```
Sorting complete, number of swaps = 7
```

To enter your own list to sort, click *Clear* and then type integers into each empty box. The following list should cause the above status line output to be produced:

```
3 4 9 17 1 2
```

Checkpoint 48

Have the program source code and output marked by a demonstrator

Task 3

This task involves recognising the methods and objects which are being used to achieve certain program behaviours. In particular, it is not an exercise in understanding how applications are constructed – this is not necessary to complete the task.

1. The version of bubble sort implemented in the application traverses the entire list during each step of the sort.
   This is not necessary since after the first traversal the smallest item will be at the end of the list and after the second traversal, the second smallest item will be in its correct position (2nd last) and so on.

   Modify the method `bubbleSort` so that it does not perform unnecessary comparisons (the *for* loop should terminate earlier on each successive step).

Checkpoint 49

Have the program source code and output marked by a demonstrator

Task 4

This task involves recognising the methods and objects which are being used to achieve certain program behaviours.

1. Modify the application so that clicking the *Reset* button, after sorting is complete, will restore the values in the list to their original values, rather than generating a new set of randomly chosen vales.

The original values should be stored in an array of Strings (you will need to declare this). A value in a list can be retrieved with the expression:

```
items[i].getText();
```

and restored with:

```
items[i].setText(...);
```

where ... is the value to be restored (a string).

Checkpoint 50

Have the program source code and output marked by a demonstrator

Task 5 (Extension Practice)

This task involves recognising the methods and objects which are being used to achieve certain program behaviours.

1. Modify the application so that it performs a "selection sort" or "bubble sort". The *Clear* button's label should be changed to *Toggle Sort* and the action performed when it is clicked should be changed to toggle the type of sort performed.

   Note: The status label should indicate the number of swaps for either sort.

The *Sort* button's label should change to indicate the current type of sort ("Selection" or "Bubble").

A button's label can be changed with its `setLabel` method. e.g.

```
mybutton.setLabel("Wombat");
```