

CSC108H lab – week 12

This document contains the instructions for the week 12 CSC108H lab.

1 Objectives

1. Work with arrays and sorting algorithms.

2 Printing an array

Create a class called `Sorting`. Add a `static` method called `print` to class `Sorting` that has an `int[]` as a parameter and prints the array so that it looks like a real array, including the square brackets. Here is an example of output for an array with three integers: `[1, 2, 3]`. Test it from the Interactions Pane.

3 Printing a String array

Switch navigator and driver.

Add a `static` method called `print` to class `Sorting` that has a `String[]` as a parameter and prints the array so that it looks like a real array, including the square brackets. Here is an example of output for an array with three integers: `["Cows", "eat", "grass"]`. Test it from the Interactions Pane.

4 Comparing two String arrays

Switch navigator and driver.

Add a `static` method called `equal` to class `Sorting` that has two `String[]` parameters and returns true if they have the same `Strings` in the same order, and false otherwise. Use `String`'s `equals` method to compare, of course. You can assume that no element in the arrays is `null`.

5 Backwards selection sort

Switch navigator and driver.

In class you saw a selection sort that behaved like this:

```
for each index i in the range 0 .. list.length - 1
  select the smallest item in list[i .. list.length - 1] and swap it with list[i]
```

Add a `static` method called `backwardsSelectionSort` to class `Sorting` that has an `int[]` as a parameter and sorts starting at the other end of the array and looking backwards through the array for the largest element.

```
for each index i in the range list.length - 1 .. 0  <-- Start at the right end
  find the largest item in list[i .. 0] and swap it with list[i]
```

6 Selection sort with Strings

Switch navigator and driver.

Add a `static` method called `backwardsSelectionSort` to class `Sorting` that has a `String[]` as a parameter and sorts in alphabetical order starting at the other end of the array and looking backwards through the array for the largest element. Your algorithm should be identical to the previous section, except it should work using `String`'s `compareTo` method. You can assume that no element of the array is `null`.

7 Backwards insertion sort

Switch navigator and driver.

In class you saw an insertion sort that behaved like this:

```
for each index i in the range 0 .. list.length - 1
  insert list[i] where it belongs in list[0 .. i]
```

Add a `static` method called `backwardsInsertionSort` that has an `String[]` as a parameter and sorts in alphabetical order starting at the other end of the array:

```
for each index i in the range list.length - 1 .. 0  <-- Start at the right end
  insert list[i] where it belongs in list[i .. list.length - 1]
```

8 Testing

Switch navigator and driver.

Write a JUnit test class that tests your `String[]` sorting methods, using your `equal` method from part 4. Test only one array per method. As always, interesting test cases are 0, 1, and many, but for this you should test even and odd length arrays as well.