Assignment 5
Hand-in date: Monday 05-10-2015 15:00h

5.1 Initialize an array of Integers, using Java's shorthand notation. The size of the array is **5** and it contains the following elements: **10**, **20**, **30**, **40**, **50**.

5.2 Initialize an array of Integers. The size of the array is **100** and it contains the numbers **1 – 100**.

5.3 Write a method that prints out the content (all of the elements) of an array in the following format:

X has index i in the array.

Where X corresponds to the element (Integer) located at index I in the array. You can test your method on the arrays from the previous exercises.

5.4 Initialize a two-dimensional array of **25*25** random double values between **50,0** and **100,0**.

5.5 Write a method **min(double[] arr)** that takes a two dimensional array of doubles as parameter and returns the smallest element in the array. Use the array from Exercise 5.4 to test your method (You have to use the method from exercise 5.3 in order to know the elements in the array, if you wish to test it).

5.6 Write a method **isSorted(int[] arr)** that returns **true** if the array (arr) is already sorted in increasing order, and **false** if it is not.

5.7 Write a method **sort(int[] arr)** that takes an integer array and uses the **isSorted** method from exercise 5.6 to determine if the array is sorted in increasing order or not. If it is already sorted, return the array, if it is not, the method should sort the array before returning it. In this exercise it is fine to use the **Arrays.sort** method to sort the array.

5.8 Write a method **merge(int[] arr1, int[] arr2)** that returns the joined array containing **arr1**'s elements followed by **arr2**'s elements.

5.9 (OPTIONAL) Write a program that generates **1.000** random integers between (and including) **0 – 9** and displays the count of each number (How many times each number has been generated).

In this exercise you are NOT allowed to use variables holding counts for each number; instead the trick is to use an array. (Yes, that was a hint).

5.10 (OPTIONAL) Consider the **SelectionSort** algorithm on page 269 and 270 in the book. When sorting the array of 9 integers as shown on page 269, the

Assignment 5
Hand-in date: Monday 05-10-2015 15:00h

algorithm iterates through the array every time it looks for the smallest element. For each of these iterations, a specific amount of indexes are checked for the smallest element before it is found.

Imagine that we were to count every time an index is visited. In the example on page 269, 20 indexes are visited by the algorithm before the array is sorted, but I claim that in the worst case, the algorithm would have to visit 28 array indexes. What is the worst case array of 9 integers (**1 – 9**), this algorithm could get?