Assignment 1

Tools & Techniques for Large Scale Data Analytics

NUI Galway, Academic year 2020/2021, Semester 2

- ***Submission deadline (strict): Monday, 1st March, 23:59. Late submission only with a medical or counsellor's certificate.***
- *Put all your source code files (and other answers, if any) into a single .zip archive with name "YourName_Assignment1.zip" and submit via Blackboard by the deadline.*
- *Include all source code files (files with name ending .java) required to compile and run your code.*
- *Unless specified otherwise in the question, use only plain Java for this assignment (no external libraries).*
- *All submissions will be checked for plagiarism.*
- ***Use Java comments to explain your source code. Missing or insufficient comments lead to mark deductions.***

You are given the following Java code for a sorting algorithm:

```java
int[] array;

public void sort(int[] array) { // array length must be a power of 2
       this.array = array;
       sort(0, array.length);
}

private void sort(int low, int n) {

   if (n > 1) {
        int mid = n >> 1;

        sort(low, mid);
        sort(low + mid, mid);

        combine(low, n, 1);
   }
}

private void combine(int low, int n, int st) {

    int m = st << 1;

    if (m < n) {
         combine(low, n, m);
         combine(low + st, n, m);

         for (int i = low + st; i + st < low + n; i += m)
            compareAndSwap(i, i + st);

    } else
        compareAndSwap(low, low + st);
}

private void compareAndSwap(int i, int j) {
    if (array[i] > array[j])
        swap(i, j);
}

private void swap(int i, int j) {
   int h = array[i];
   array[i] = array[j];
   array[j] = h;
}
```

PTO

Remark: *For this assignment and future assignments, the Java API documentation might be useful.*

Q1. Put the provided sorting code into a class and add a `main`-method which uses it to sort some array of `int` numbers (with at least 16 elements) and prints both the original array and the resulting sorted array.
Observe that the length of the array needs to be a power of 2, otherwise this sorting algorithm fails.

[10 marks]

Q2. Modify the provided sorting code so that it is able to sort arrays of *objects* of arbitrary classes (instead of numbers of primitive type `int`), making use of `java.util.Comparator<…>` instances ("comparators") for comparing array elements with each other.

Furthermore, provide a `main`-method which creates or obtains a comparator for objects of type `String` and another comparator for objects of type `Double`, and uses the modified sorting code to sort 1) some array of strings (in lexicographic order) and 2) some array of numbers of type `Double`. These arrays should each contain at least 16 elements and your `main`-method should again print both the original and the respective sorted arrays.
(But of course, the modified sorting code should be generic, so that it can sort arrays of objects of any class, not only `String` and `Double` arrays.)

Do not change the sorting algorithm otherwise and do not call any other sorting routine (so you need to keep your modifications of the provided code to the minimum required for doing this question, and you also cannot call any other sorting method such as Arrays.sort or Collections.sort).

Do not use any Lambda expressions in your code for this question (but see next question).

[30 marks]

Q3. Modify your code for Q2 so that the comparator for strings and the comparator for `Double` numbers are provided as *Lambda expressions*. Hint: this requires only very little coding.
As before, do not change the sorting algorithm otherwise and do not call any other sorting routine.

[20 marks]

Q4. Modify the original code provided on the first page of this assignment sheet so that it makes sensible use of *parallel* computing (using multithreading) in the body of method `sort(int low, int n)`. Do not change the algorithm otherwise.

Hint: look at the two `sort()` calls inside the `if`-statement…

Use only the "traditional" (Java ≤ 7) approach to multithreading for this question (that is, using class `Thread` explicitly); do not use, e.g., parallel Java 8 Streams.

It is not required that your parallelized code runs actually faster than the original code (no need to run any benchmarks) - but it should make use of concurrent threads in a sensible way.

Also, provide again a `main`-method which tests your code with some example arrays and prints the respective original and the sorted array.

As before, do not change the sorting algorithm otherwise and do not call any other sorting routine.

[40 marks]