Assignment 5

Empirical Study	30 pts.
Quicksort for Date Objects	40 pts.
Number Analyzer	30 pts.

TOTAL: 100 pts.

General Requirements

- *All files you are turning in must be signed. Please add your name to the source code in comments. Unsigned work automatically loses **5 pts**.*
- *Add comments to the source code you are writing:*
  - *Describe the purpose of every variable*
  - *Explain the algorithm you are using for solution*
  - *Add proper comments for all methods. Include @param, @return, and @throws tags*
- *Uncommented work loses up to **25% of the grade.***
- *Archive the entire project using ZIP or RAR utility. Turn it in into the digital drop box.*

Empirical Study of Sorting Algorithms Complexity

The goal of this project is to compare efficiency of three sorting algorithms by measuring the time it takes for them to sort exactly same array of integers.

- Find and copy into your code implementations of three classic sorting algorithms: Quicksort, Bubble sort, and Selection sort. All 3 can be found in Chapter 17 folder of the text book source code posted in Course Info and Resources module of the course.
- In main() write code that creates 3 identical arrays of size 1000000 filled with random numbers in the range [1 – 1000000]
- Use code from RecoursiveFibonacciTimer.java from Chapter 17 folder to measure time it took each of the sorting algorithms to sort the array. Each sorting method must have the same set of numbers to sort, so use one of the 3 arrays you created for each method.
- When printing out time, convert it into minutes and seconds format.
- In the beginning of your file with the solution, in comments, record the timing results you got when running the sorting methods – for me to see.
- Finally, find out how much time it takes Quicksort to sort 100000000 random integers on your computer. Record that result in the beginning of the source code file too.

Name your solution EmpiricalStudy.java

Quicksort for Date Objects

Start with the provided class called Date that can be found next to this assignment.

- Add an override of toString() method to the class that prints out date in the form "December 25, 2016".
- Make Date class implement Comparable interface (https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html). For comparison use an obvious rule: the earlier date is smaller than the later one.
- Modify Quicksort implementation in such a way that it becomes a generic method and can work with objects that implement Comparable interface. Study source code samples from Chapter 17 folder to find a good sample code for this task.
- Use loop and random number generator to create an array of 100 dates in the time range from January 1 2018 to December 31 2018.
- Use Quicksort generic method to sort the dates in the array.
- Print out the original array and the array after it has been sorted.

Name your project DateQuicksort.java, keep the name for Date.java class the same.

Number Analyzer

Write a generic class NumberAnalyzer with a type parameter constrained to the Number class or any subclass of Number. The constructor should accept an array of such objects. The class should have methods that return the highest and lowest values in the array, the total of the elements, and the average value of all elements. Test the class in main() with a couple of arrays of different base types.
Guidelines:

- Do not create a deep copy of the array passed to constructor. You will not be able to do so because the base type is generic. The array will not be modified, so create a shallow copy instead.
- Any arithmetic operations with values of type Number are impossible. Before doing any comparisons or arithmetic operations you will need to convert Number object into a double value by using method doubleValue(). Example:

```
Number a;
// initialization code
double k = a.doubleValue();
```

Name source files in your solution NumberAnalyzerDemo.java and NumberAnalyzer.java