Lab 4

Part -1

Understanding runtime complexity. This lab will provide you some practice on understanding the runtime complexity of programs. The code is provided below.
Note: No need to run the code.

Q1. Write down complexity in terms of Big-O for methodA, methodB and MethodC

```
import java.util.Date;
import java.util.Timer;

public class ComplexityExperiment {
    public static void main (String args []) throws InterruptedException {
       run_method_A(250);
       run_method_A(500);
       run_method_A(1000);
       run_method_A(2000);
       System.out.println();

       run_method_B(250);
       run_method_B(500);
       run_method_B(1000);
       run_method_B(2000);
       System.out.println();

       run_method_C(250);
       run_method_C(500);
       run_method_C(1000);
       run_method_C(2000);
       System.out.println();

    }

    public static void run_method_A(int n)  {
            int i = 0;
            double start, end;
      start = System.currentTimeMillis();
            methodA(n);
            end = System.currentTimeMillis() - start;
            System.out.println("methodA(n = " + n + ") time = " + end + "ms");
    }


    public static void run_method_B(int n)  {
            int i = 0, loop = 1000;
            double start, end;
```

```
  start = System.currentTimeMillis();
  for (i = 0; i < loop; i++)  {
         methodB(n);
       }
       end = System.currentTimeMillis() - start;
       System.out.println("methodE(n = " + n + ") time = " + end/loop + "ms");
}
public static void run_method_C(int n)  {
  int i = 0;
  double start, end;
  start = System.currentTimeMillis();
  methodC(n);
  end = System.currentTimeMillis() - start;
  System.out.println("methodC(n = " + n + ") time = " + end + "ms");
}

 public static void methodA (int n){

                int i = 0;
                int j = 0;
                int k = 0;
                int total = 0;
                while (i<n){
                        while (j<n){
                                while (k<n) {
                                        total++;
                                        k++;
                                }
                                k=0;
                                j++;
                        }
                        j=0;
                        i++;
                }
        }

        public static void methodB (int n){
                int i = 0;
                int j = 0;
                int total = 0;
                while (i<n){
                        while (j<n){
                                total++;
                                j++;
                        }
                        i++;
                }
        }
```

```
    public static void methodC (int n){
        int i = 0;
        int j = 0;
        int total = 0;

        j = n;
        while ((j = j/2) > 0)  {
           for (i = 0; i < 100*n; i++)
               total++;
        }
    }
}
```

Part -2

Q2. You need to implement "Bubble Sort"

You are provided with "emp.txt" file. Create array/arraylist of objects of employee class you created in Lab3. In lab3 you taken input from user. Here you need to read file having each line name and ID separated with "space". Read line in file and create employee object and store it in array. You are given 30 employees in file. **Sort them based on IDs.**

Print sorted output on console.

You can create separate output file to save this sorted output. **This is optional.**

Write down complexity in terms of Big-O of your Bubble Sort program.

Also timestamp bubble sort method and record time required to sort the file.

In the above code, there is timestamp recorded.

Please see below code for timestamp.

```
double start, end;
 start = System.currentTimeMillis();
methodA(n);
end = System.currentTimeMillis() - start;
```

You can put timestamp before and after "Method for bubble sort".

If you are not creating separate method for Bubble Sort please put start before actual bubble sort code and end after you finished code for bubble sort.