COMP 2215: WEEK – 5

LABWORK 2

Download Following Files from Sakai website and add them to an empty Project in NetBeans / Eclipse / IDEA

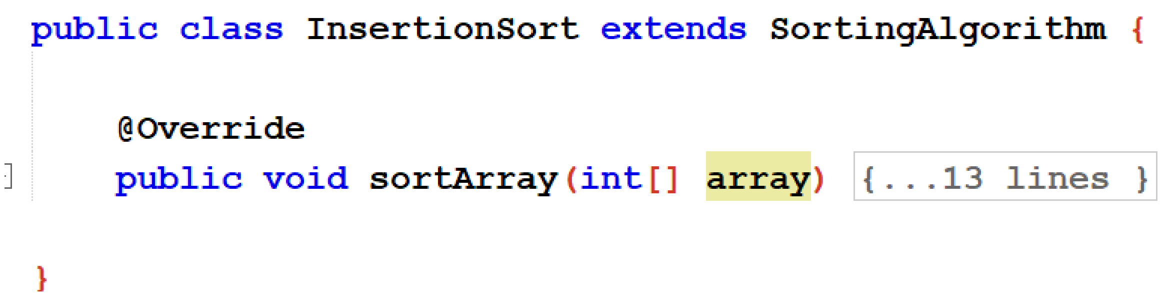
- SortingAlgorithm.java
- Main.java

Purpose of this week is to implement different sorting algorithms and compare their runtimes in relation to each other. We're going to take advantage of Inheritance and Dynamic Polymorphism (also known as Method Overriding) in Java.

SortingAlgorithm is an abstract class which is to have 5 Children classes or concrete classes. These are:

1. **Bubble Sort**
2. **Insertion Sort**
3. **Selection Sort**
4. **Merge Sort**
5. **Quick Sort**

Your first task is to implement each of these Sorting algorithms as Child Classes. To do that, you need to extend SortingAlgorithm and implement sortArray method.



```java
public class InsertionSort extends SortingAlgorithm {

    @Override
    public void sortArray(int[] array) {...13 lines }

}
```

Note that we actually don't need to know how these sorting strategies work, because it's unrelated to the programming course. You can copy-paste Java code from the following links and make changes you feel are necessary.

You can find sample implementations on the following links:

1. https://www.geeksforgeeks.org/bubble-sort/
2. https://www.geeksforgeeks.org/insertion-sort/
3. https://www.geeksforgeeks.org/selection-sort/
4. https://www.geeksforgeeks.org/merge-sort/
5. https://www.geeksforgeeks.org/quick-sort/

Now that you've completed all the strategies, your second task is to complete Main.java class. Here you need to

1. create an instance from each of the Child Classes. Example is given below.

```java
SortingAlgorithm[] algorithms = new SortingAlgorithm[5];
algorithms[0] = new BubbleSort();
algorithms[1] = new InsertionSort();
algorithms[2] = new SelectionSort();
algorithms[3] = new MergeSort();
algorithms[4] = new QuickSort();
```

2. Create 10 different randomized Integer array for measuring algorithms performance. (You can do this by calling createRandomizedArray inside a for loop.) The reason we create 10 different random arrays instead of 1 is to measure average runtime for each algorithm and avoid random bias.

3. To Compare algorithms, we need to feed them the same array. Obviously if we only create one array, after the first algorithm, array will be already sorted. This might give unfair advantage to other algorithms. So, to ensure fairness, we need distinct copies of the same newly generated random int array. To do that, call CopyArray for each of these algorithms.

4. Call testAlgorithm to save runtime for each of the algorithms using copied array. Note that you need to call this 10 times for each algorithm, so we can calculate average.

Below is a sample code which codes the steps 2 - 3 – 4.

```java
for (int k = 0; k < 10; k++) {
    int[] originalArray = createRandomizedArray(k * 100, 100);
    String original = algorithms[0].printArray(originalArray);
    System.out.println("Original Array:" + original);
    for (SortingAlgorithm algorithm : algorithms) {
        int[] copy = copyArray(originalArray);
        algorithm.testAlgorithm(copy);
    }
}
```

5. After all tests are done, call findAverage() for each of the algorithms. You can obtain class name by getClass(). getSimpleName(). If the object(algorithm) is a BubbleSort object, this will return String “BubbleSort”.

```java
for (SortingAlgorithm algorithm : algorithms) {
    System.out.println("Average Run time for "
            +algorithm.getClass().getSimpleName()
            +" :"+algorithm.findAverage()+"ms");
}
```

6. Run your code and observe the results.

7. Explain how using Runtime Polymorphism (Method Overrides) and Inheritance played a part in this lab work and what was the main advantage of using Java instead of C language while coding this benchmark application for sorting algorithms? Do you think there are disadvantages using Object oriented approach compared to C? If so, what are they?

Submission:

Upload following files into Sakai.

1. BubbleSort.java
2. InsertionSort.java
3. SelectionSort.java
4. MergeSort.java
5. QuickSort.java
6. Main.java
7. A txt / docx / pdf document that answers Task 7.