

Name: Namrata Ruchandani  
NU ID : 002125637

→ Task done in this assignment:

1. In the timer, I added a condition when preFunction and postFunction are not null. I added pause before the loop and while loop runs resume is mentioned and then lap for counting laps. For not counting post function time, I used pause. The clock was paused, but according to the question clock should be resumed. So once the loop is done, I use a resume. For returning the number of counts, I used meanLapTime.
2. In the InsertionSort, I added the code of for loop and then I put the condition of comparing and swapping with the help of swapStableConditional.
3. In the file name Assignment 2.java, I made four arrays which were reversed, random, partially, sorted. After doing insertion sort for each of them, I counted the timer. Secondly, I made a loop for creating an array with double size and with random numbers and calculated timer value.

The screenshot of main method (or part 3 of assignment) Assignment2.java code compilation and output

The screenshot displays an IDE window titled 'INFO6205 - Assignment2.java'. The left sidebar shows a project structure with various classes, including 'Assignment2'. The main editor area shows the code for 'Assignment2.java', which includes a 'main' method that creates an 'InsertionSort' sorter, initializes a list with values 7, 6, 3, 2, and a 'Timer' object. The code then performs an insertion sort on the list and prints the result. The bottom panel shows the 'Run' output, which includes the command to run the program and the resulting output, which is a list of numbers: 1.1128635346534653, 0.00987916, 0.00409375, 0.00622958, 0.0023787500000000002, 1.8208E-4, 2.3709000000000002E-4, 2.35E-4, and 0.0012183299999999999. The 'Event Log' on the right shows messages such as 'Tests passed: 10' and 'All files are up-to-date'.

```
public class Assignment2
{
    private static Object ConfigTest;

    public static void main(String[] args)
    {
        InsertionSort sorter = new InsertionSort();
        //reverse sorted
        final List<Integer> list = new ArrayList<>();
        list.add(7);
        list.add(6);
        list.add(3);
        list.add(2);
        Integer[] xs = list.toArray(new Integer[0]);
        Timer timer = new Timer();
```

Run: Assignment2 x

/Library/Java/JavaVirtualMachines/jdk1.8.0\_301.jdk/Contents/Home/bin/java ...

1.1128635346534653

0.00987916

0.00409375

0.00622958

0.0023787500000000002

1.8208E-4

2.3709000000000002E-4

2.35E-4

0.0012183299999999999

7:27 PM Tests passed: 10

7:27 PM Tests passed: 10

7:28 PM All files are up-to-date

7:28 PM Tests passed: 2

7:28 PM Tests passed: 2

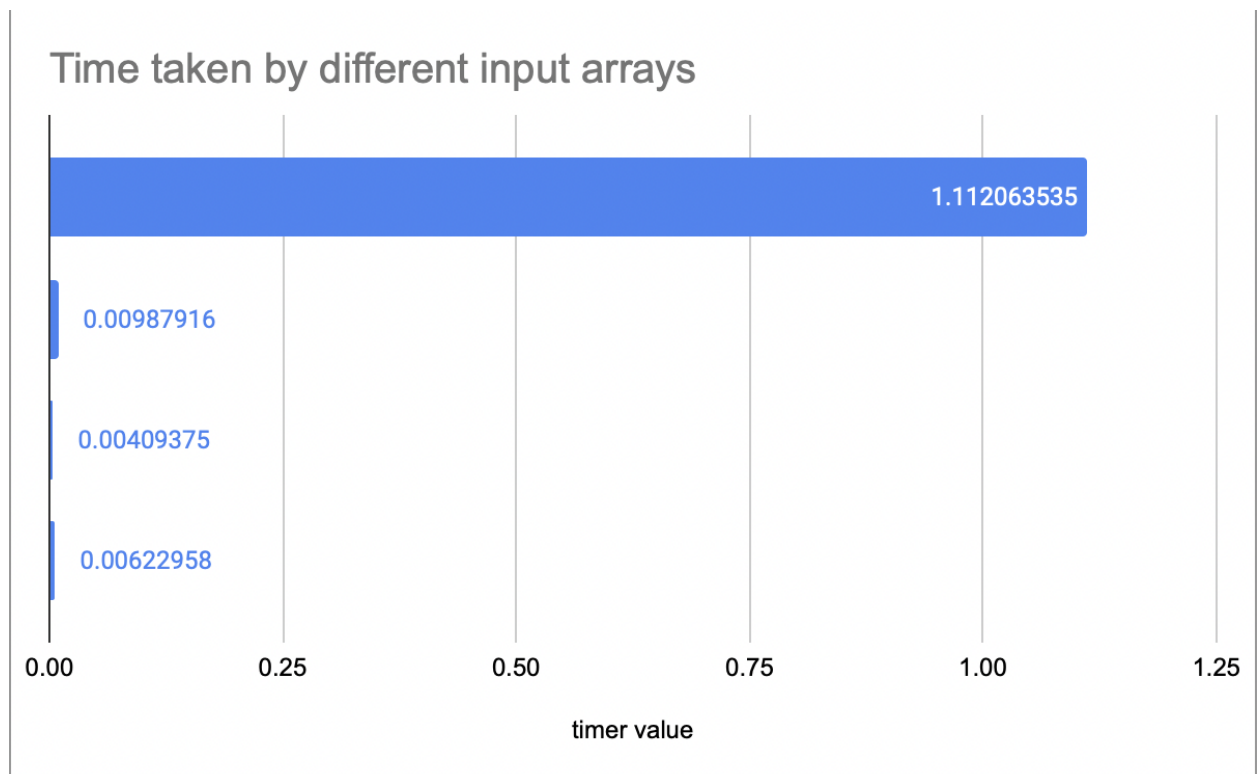
7:28 PM All files are up-to-date

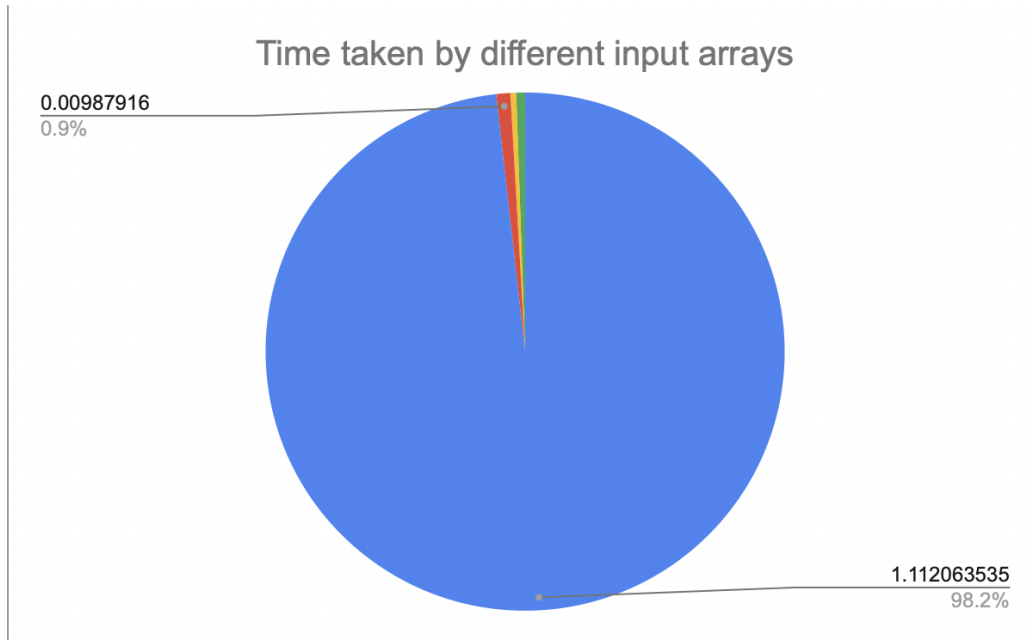
The link for file Assignment2.java :

<https://github.com/Namrata2108/INFO6205/blob/Fall2021/src/main/java/edu/neu/coe/info6205/Assignment2.java>

- Conclusion about time taken by various kinds of input arrays. The maximum time was taken by a reverse sorted array. Time taken by partially ordered and sorted arrays are quite small and similar. With the help of charts below, we can say that reverse sort took the longest time and partially ordered/sorted array took least time.

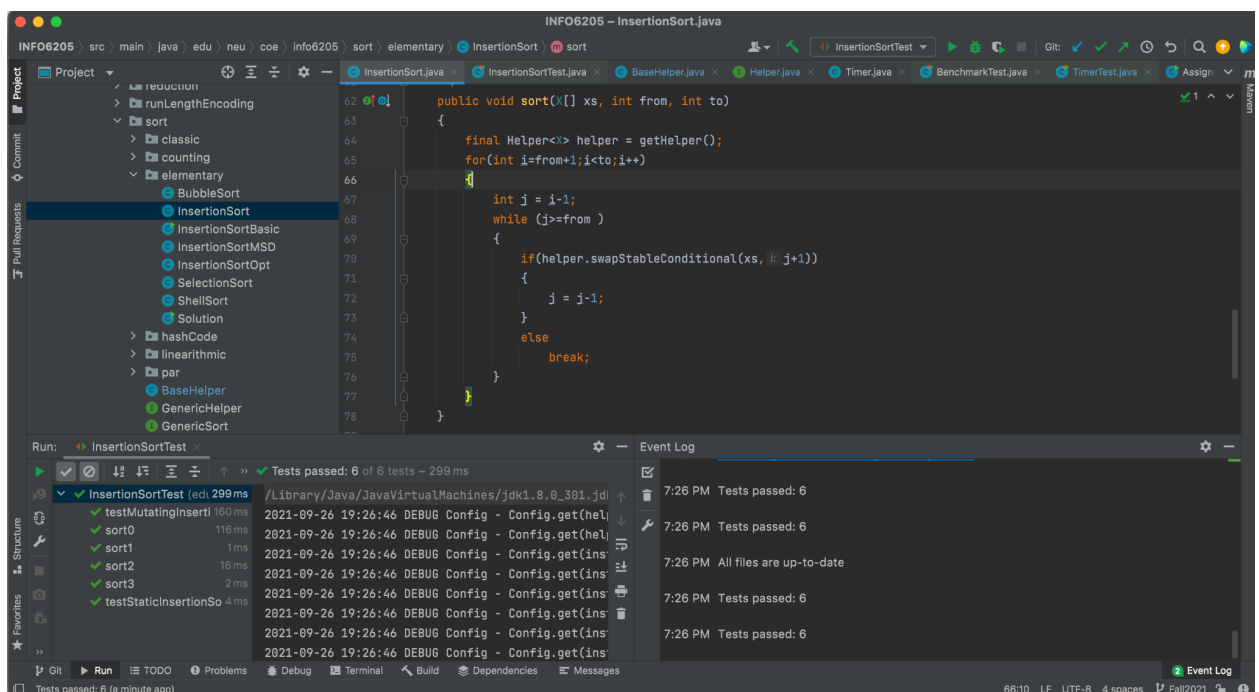
Input Array	timer value
reverse	1.112063535
random	0.00987916
partially	0.00409375
sorted	0.00622958





- I pushed the commands in the git and attached the link for the repository.  
<https://github.com/Namrata2108/INFO6205/tree/Fall2021/src/main/java/edu/neu/coe/info6205>  
But as I'm new to git commands, I'm attaching a zip file of my code also.

- Below is the screenshot of successful test cases run  
◆ Insertion Sort Test cases



## ◆ Timer Test cases

The screenshot shows an IDE with the `Timer.java` file open. The code defines a `repeat` method that takes a number of iterations `n`, a `Supplier`, a `Function`, and a `UnaryOperator`. It uses a `logger` to trace the execution and includes a `pause` method. The test results pane shows that 10 tests passed in 2 seconds and 399 milliseconds. The event log shows that all files are up-to-date and tests passed at 7:26 PM.

```

public <T, U> double repeat(int n, Supplier<T> supplier, Function<T, U> function, UnaryOperator<T> pr...
{
    logger.trace("repeat: with " + n + " runs");
    pause();
    //long start=getClock();
    // TO BE IMPLEMENTED: note that the timer is running when this method is called and should still be running when
    for(int i=0;i<n;i++)
    {
        //if(i!=0) if timer is just started so why to resume it
        //resume();
        if(preFunction!=null)
        {
            preFunction.apply(supplier.get());
        }
        resume();
        U temp = function.apply(supplier.get());
        lap();
        pause(); //doing this so that post should not be timed
    }
}

```

Run: TimerTest x Tests passed: 10 of 10 tests - 2sec 399 ms

- testPauseAndLapR 749 ms
- testPauseAndLapR 315 ms
- testLap 206 ms
- testPause 210 ms
- testStop 106 ms
- testMillisecs 106 ms
- testRepeat1 242 ms
- testRepeat2 246 ms
- testRepeat3 614 ms
- testPauseAndLap 105 ms

Process finished with exit code 0

Event Log

- 7:26 PM All files are up-to-date
- 7:26 PM Tests passed: 6
- 7:26 PM Tests passed: 6
- 7:27 PM All files are up-to-date
- 7:27 PM Tests passed: 10
- 7:27 PM Tests passed: 10

## ◆ Benchmark\_Timer Test cases

The screenshot shows an IDE with the `Benchmark_Timer.java` file open. The code defines a `repeat` method that takes a number of iterations `n`, a `Supplier`, a `Function`, and a `UnaryOperator`. It uses a `logger` to trace the execution and includes a `pause` method. The test results pane shows that 2 tests passed in 1 second and 606 milliseconds. The event log shows that all files are up-to-date and tests passed at 7:27 PM.

```

public <T, U> double repeat(int n, Supplier<T> supplier, Function<T, U> function, UnaryOperator<T> pr...
{
    logger.trace("repeat: with " + n + " runs");
    pause();
    //long start=getClock();
    // TO BE IMPLEMENTED: note that the timer is running when this method is called and should still be running when
    for(int i=0;i<n;i++)
    {
        //if(i!=0) if timer is just started so why to resume it
        //resume();
        if(preFunction!=null)
        {
            preFunction.apply(supplier.get());
        }
        resume();
        U temp = function.apply(supplier.get());
        lap();
        pause(); //doing this so that post should not be timed
    }
}

```

Run: BenchmarkTest x Tests passed: 2 of 2 tests - 1sec 606 ms

- testWaitPeriod 1 sec 606 ms
- getWarmupRuns 0 ms

Process finished with exit code 0

Event Log

- 7:27 PM All files are up-to-date
- 7:27 PM Tests passed: 10
- 7:27 PM Tests passed: 10
- 7:28 PM All files are up-to-date
- 7:28 PM Tests passed: 2
- 7:28 PM Tests passed: 2