- Name(s) of all authors: Ishita Sarraf, Havin Lim
- Assignment name: Week 8 Labs
- Assignment due date: October 31th 2022
- Written/online sources used: None
- Help obtained: Mentor, Professor
- “I/we confirm that the above list of sources is complete AND that I/we have not talked to anyone else (e.g., CSC 207 students) about the solution to this problem.”

Lab 22: Merge Sort

MergeSorterTester.java

```
package sorting;

import java.io.File;

public class MergeSorterTester {

public static void main(String[] args) {
Integer [] numbers = {24, 13, 26, 1, 2, 27, 38, 15};
MergeSorter.mergeSort(numbers);

System.out.println(isCorrectlyOrdered(numbers));

System.out.println("Sorting an array of type: "+ numbers.getClass().getName());
System.out.println("Sorted Array:");
for (Integer i: numbers)
System.out.print(i+" ");

try {
```

```
Scanner sc = new Scanner(new File("eight-cousins.txt.tokenized"));
ArrayList<String> words = new ArrayList<String>();
while (sc.hasNextLine()) {
words.add(sc.nextLine());
}
sc.close();

String [] w = new String [words.size()];
words.toArray(w);

MergeSorter.mergeSort(w);



System.out.println("Sorting an array of type: "+ w.getClass().getName());
System.out.println("Sorted Array:");



for (String i: w)
System.out.println(i);
System.out.println(isCorrectlyOrdered(w));

}
catch(FileNotFoundException e)
{
System.out.println(e.getMessage());
}
```

```
}

public static <AnyType extends Comparable<? super AnyType>> boolean isCorrectlyOrdered(AnyType []
a) {

for (int i = 0; i < a.length-1; i++) {
if (a[i].compareTo(a[i+1]) > 0)
return false;
}
return true;

}

}
```

ComparableSwitch.java

```
package sorting;

public class ComparableSwitch extends Switch implements Comparable<Switch> {

@Override
public int compareTo(Switch arg0) {
if(arg0.report() == OFF) {
return 1;
}
return 0;
}
```

```
}
```

JUnit Tests

```
package sorting;

import static org.junit.Assert.assertEquals;

class ComparableSwitchTester {

@Test
void test() {

ComparableSwitch[] array = new ComparableSwitch[10000];
Random rmd = new Random();
for (int i = 0; i < 10; i++){
array[i] = new ComparableSwitch();
array[i].turn(rmd.nextBoolean());
}

boolean [] sorted = new boolean[array.length];
for (int i = 0; i < array.length; i++) {
sorted[i] = array[i].report();
System.out.println(sorted[i]);
}

}

@Test
```

```
void test2()
{
ComparableSwitch[] array = new ComparableSwitch[10000];
Random rmd = new Random();
for (int i = 0; i < 10000; i++){
array[i] = new ComparableSwitch();
array[i].turn(rmd.nextBoolean());
}

boolean [] sorted = new boolean[array.length];
for (int i = 0; i < array.length; i++) {
sorted[i] = array[i].report();
System.out.println(sorted[i]);
}



}
}
```

Lab 23: Quick Sort

1> We think both Comparisons and Movements are relevant operations but based on the values for NlogN we think, Comparisons is more relevant.

	A	B	C	D	E	F
1	Name/Categories	Tokens	Swaps	Movements	Comparisons	N log N
2	little-men	129,670	614,530	1,843,590	2,693,560	2202378.2
3	eight-cousins	91,021	410,599	1,231,797	1,792,085	1499471.93
4	rose-in-bloom	113,866	527,164	1,581,492	2,325,994	1912604.64
5						
6						

Lab 24: Inner Classes

7> We expected to get output:

1 of 5

2 of 5

3 of 5

4 of 5

5 of 5

6 of 10

7 of 10

8 of 10

9 of 10

10 of 10

We got the same output. This is because once we have created the object for Greetable, the value of num is initialized once to 0 and then prints upto 5 on the first call and then continues from 5 upto 10 for the second call.

10>

```
package lab24;

import java.io.PrintWriter;

/**

* A simple class that creates an anonymous greeter that references a field in

* the enclosing class at the time is built and at the time it greets.

*/

public class SampleGreetable6 implements Greetable

{

 int i = 0;


 @Override

 public Greeter greeter()

 {
```

```
   return new Greeter()

    {
      int num = ++i;
      int greet = 0;
      @Override
      public void greet(PrintWriter pen)
      {
           greet++;
       pen.println("Greeting "+ greet+ " from greeter "+num + " of " + i);
      } // greet(PrintWriter)
    }; // new Greeter
  } // greeter()
} // interface SampleGreetable6
```

Static Classes:

```
public static class ComparatorPriority implements Comparator<Tasks>{

@Override
public int compare(Tasks arg0, Tasks arg1) {
if (arg0.priority < arg1.priority)
return -1;
return 0;
}

}

public static class ComparatorTime implements Comparator<Tasks>{
```

```
@Override
public int compare(Tasks arg0, Tasks arg1) {
if (arg0.timeTaken < arg1.timeTaken)
return 1;
return -1;
}
}



}
```

Anonymous Class:

```
PriorityQueue<Tasks> p3 = new PriorityQueue<Tasks>(new Comparator<Tasks>(){
@Override
public int compare(Tasks arg0, Tasks arg1) {
if (arg0.days < arg1.days)
return 1;
return -1;
}
}
);

for (int i = 0; i < 5; i++)
{
//System.out.println(p1.peek());
p3.add(p1.remove());
}
System.out.println("Top element in Queue by days: "+p3.peek());
```

Result:

Top element in Queue by Priority: Time to complete: 170, Priority: 2, Days due: 6, Description: Watch movie

Time to complete: 170, Priority: 2, Days due: 6, Description: Watch movie

Time to complete: 119, Priority: 9, Days due: 2, Description: Singing

Time to complete: 22, Priority: 9, Days due: 9, Description: Do CS homework

Time to complete: 200, Priority: 3, Days due: 10, Description: Do CS homework

Time to complete: 98, Priority: 2, Days due: 3, Description: Dancing

Top element in Queue by Time: Time to complete: 200, Priority: 3, Days due: 10, Description: Do CS homework

Top element in Queue by days: Time to complete: 200, Priority: 3, Days due: 10, Description: Do CS homework