Lab 8: Gnome Sort

```
Location: CourseWeb -> Labs/Recitations -> Lab 8: Gnome Sort
Download the following files:
  1. SortingFrame.java
  2. VisualSortingComponent.java
```

Introduction to Gnome Sort[1]

Gnome sort is a sorting algorithm which is similar to insertion sort, except that moving an element to its proper place is accomplished by a series of swaps, as in bubble sort. It is conceptually simple, requiring no nested loops. The running time is $\mathbf{O(n^2)}$, but tends towards $\mathbf{O(n)}$ if the list is initially almost sorted. In practice the algorithm can run as fast as insertion sort. The average runtime is $\mathbf{O(n^2)}$.

The algorithm always finds the first place where two adjacent elements are in the wrong order, and swaps them. It takes advantage of the fact that performing a swap can introduce a new out-of-order adjacent pair only right before or after the two swapped elements. It does not assume that elements forward of the current position are sorted, so it only needs to check the position directly before the swapped elements.

Simply speaking, first we compare the first pair (let's call them left elemenet and right element), if the left element is less than or equal to the right element, move on the the next pair. If the left element of a pair is greater than the right element, we swap them. But after we swap, nothing guarantee that the previous pair is in order. So, we have to move back to the previous pair and perform the same process all over again.

What to Do?

For this lab, you are going to implement the Gnome Sort algorithm to sort arrays of integers with visualizations. First, simply run `SortingFrame.java` and you will see a visualization of a sorting algorithm called Bubble sort. The `main` method of `SortingFrame.java` is shown below:

```
public static void main(String[] args) throws InterruptedException
{
        JFrame frame = new JFrame();

        int[] data = randomIntArray(40);
        VisualSortingComponent vsc = new VisualSortingComponent(data);
        frame.setTitle("Sorting Visualization");
        frame.setSize(500,500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.add(vsc);
        frame.setVisible(true);
```

[1] Modified from Wikipedia

```
        Thread.sleep(5000);

        bubbleSort(data,vsc);
        //gnomeSort(data,vsc);
}
```

Note that the method `bubbleSort()` requires two arguments, `data` which is an array of integer to be sorted, and `vsc` of type `VisualSortingComponent`. `vsc` is used for sorting visualization. To see the sorting visualization, every time you swap two values in the array, you should call `vsc.repaint();` follows by `Thread.sleep(100);` as shown in the method `bubbleSort()` below:

```
public static void bubbleSort(int[] data, VisualSortingComponent vsc)
       throws InterruptedException
{
        int size = data.length;

        for(int i = 0; i < size; i++)
        {
                for(int j = 0; j < size - 1; j++)
                {
                        if(data[j] > data[j + 1])
                        {
                                int temp = data[j];
                                data[j] = data[j + 1];
                                data[j + 1] = temp;
                                vsc.repaint();
                                Thread.sleep(100);
                        }
                }
        }
}
```

Your job is to develop the method `gnomeSort()` (marked `TO DO`). **DO NOT** modify the signature of the method `gnomeSort()`. To see the visualization of your Gnome Sort algorithm, modify the last two lines in the `main` method as shown below:

```
        //bubbleSort(data,vsc);
        gnomeSort(data,vsc);
```

Test Class

There is no test class for this lab. We will test by looking at the animation.

Due Date and Submission

For the due date, please check the lab in the CourseWeb. Submit your `SortingFrame.java` to the CourseWeb under this lab by the due date. **No late submission will be accepted**.