CSCI-513
Lab 1

Objective

In this lab you will import a file containing population statistics for countries per year, parse each row to extract the population count, store it into an array of "long" integers, and sort it using three different sorting strategies. I provide standard sorting algorithms in Appendix A, but you will need to modify these to accommodate "long" instead of "int".

The skills you practice in this assignment will help you with your first homework assignment – namely (1) importing a file, (2) parsing a string using tokenizer, and (3) using the strategy pattern.

Steps

1. Clone Lab1 from GitHub.

2. Import the code into Eclipse. Make sure you also import the WorldPopulation.csv file. If you successfully import it under the src package you can reference it as "src\\WorldPopulation.csv"

3. **Create three strategy java classes** based on **bubble** sort, **selection** sort, and **insertion** sort. The classes must implement the SortStrategy interface.

4. Add sort timers to each class. The following code may help you:

```
// Before starting the sort routine
long startTime = System.currentTimeMillis();

// After sorting
long endTime = System.currentTimeMillis();

// Compute the time difference in a method called getSortTime();
long totalTime = endTime - startTime;
```

   Notice that the SortStrategy interface requires you to distribute this code across two methods entitled: getSortTime();

5. **Follow instructions in the WorldPopulation.java file**. (Note that you need to read the population figures in as long integers. Your tasks include:
   a. In ReadPopulationFile(String filename): Write code to read in the WorldPopulation.csv file. Note that you need to read the file in line by line, and for each line you read you need to use StringTokenizer to split the line and to extract the population.
   b. Set default strategy in the constructor.
   c. Delegate sorting to the SortStrategy object in sortPopulation()

d. Experiment with various combinations of reading in the file and sorting using different strategies. Try to answer the following questions?
    i. Which sort strategy is fastest?
    ii. What happens with each strategy if you sort a previously sorted file.

6. Add functionality so that each time you run the sort routine it **prints the name of the algorithm** used and **the time in milliseconds** to perform the sort.

7. Commit to GitHub--Lab1 by Sunday night. Note: This does not count as homework – but all students are expected to complete the lab and it will count as the %5 of your grade for attendance.

Appendix: Sort Routines

This code was retrieved from the internet (https://www.cs.cmu.edu/~adamchik/15-121/lectures/Sorting%20Algorithms/sorting.html). It sorts ints – so you will need to modify it to sort long types. You may also wish to consider a more meaningful variable name than "ar"!!! and modify the methods to return the sorted list of long numbers.

Selection Sort:

```
void selectionSort(int[] ar){
  for (int i = 0; i ‹ ar.length-1; i++)
  {
    int min = i;
    for (int j = i+1; j ‹ ar.length; j++)
        if (ar[j] ‹ ar[min]) min = j;
    int temp = ar[i];
    ar[i] = ar[min];
    ar[min] = temp;
  }
}
```

Insertion Sort:

```
void insertionSort(int[] ar)
{
  for (int i=1; i ‹ ar.length; i++)
  {
    int index = ar[i]; int j = i;
    while (j > 0 && ar[j-1] > index)
    {
       ar[j] = ar[j-1];
       j--;
    }
    ar[j] = index;
  }
}
```

Bubble Sort

```
void bubbleSort(int ar[])
{
  for (int i = (ar.length - 1); i >= 0; i--)
  {
    for (int j = 1; j ≤ i; j++)
    {
      if (ar[j-1] > ar[j])
      {
          int temp = ar[j-1];
          ar[j-1] = ar[j];
          ar[j] = temp;
      }
    }
  }
}
```