SUMMARY

In this lab you will create a new class, and its associated objects and methods. The class will represent some aspects of the climate in NYC. People today worry about climate change trends, and are affected by rapid changes in weather. Analyzing climate measurements and alerting us to anomalies is an important use for computers.

In this lab, you will refactor the stock price plotting software we developed live in the last lecture to plot the average temperature in one spot in NYC during the last year, and alert us to daily changes of 10 degrees (Fahrenheit) or more.

Each instance of your new class, `Temps`, will represent the information to plot the temperature and alert us to big daily change.

PRE-LAB

You should have reviewed your notes and/or the web material from Lecture 22, Monday, Nov. 19, and begun to read chapter 11 in G&E’s book.

IN-LAB

STOCK

In this lab you will create a class to be the blueprint for an object representing a history of average temperature measurements in Central Park, NYC. The object will contain an array of type `double`. This array will represent the average temperatures for each of the days of the last year. The array will be big enough to hold at least one thousand temperatures. In addition, there will be an integer in the object to keep track of the number of temperatures recorded in the array. You will also write a main method that uses an instance of your `Temps` class.

You will do this by refactoring the `StockPricePlotter` class that was live coded on Monday, Nov. 19. Begin by copying the original and the data files into a fresh directory for Lab11. Get them from the Web outside of lab; inside of lab get them by:

```
  cd CSI201 #or other name for your CSI201 folder/dir.
  # begins a comment in Unix shell languages.
  mkdir Lab11
  cd Lab11
  cp /usr/local/depts/cs/geintro/Fal12Lab11/* /usr/local/depts/cs/geintro/drjava
```

VERIFY IT COMPILLES AND RUNS BEFORE YOU GO ON!!!
**FIELDS**

Your new class should be named `Temps`. This class should specify THREE (3) fields:

- An array of `doubles` referred to by `dailyTemps` (change the name from `dailyPrices`)
- An `int` named `numberOfDaysWithData`
- A double named `lastTChange`

**CONSTRUCTORS**

(We will cover constructors officially in the lectures, and after that, you will make constructors in future labs.)

**METHODS**

Add the following methods in your `Temps` class:

```java
void addTemp(double temp)
```

Refactor: Move the code

```java
dailyPrices[arrayIndex] = currPrice;
arrayIndex = arrayIndex + 1;
this.numberOfDaysWithData++;
```

from the `readAndPlot` to your `addTemp` method. Of course, you must REPLACE the now out-of-scope variable `currPrice` with the parameter variable so your code records (in an element of the array) the temperature stored in the parameter `temp`. And, you must use the appropriate field of the class (`numberOfDaysWithData`) instead of `arrayIndex` to locate WHERE to store the latest added temperature.

Finally, add functionality:

**UPDATE the field `lastTChange`** according to the specification of what `lastTChange` is supposed to mean.

```java
int numTemps()
```

This method should return the current number of temperatures in the array for this stock.

```java
double lastTempChange()```

If there are two or more temperatures in the array, it will return the difference between the last two: (previous – last). If there are fewer than 2 items in the array, the method should return zero. It should use the value stored in `lastTChange`.

```java
void checkChangeAndPrint()```

Check the big change condition (see the summary) for the last added trade, and, if the average temperature changed by 10 degrees or more in one day, up or down, PRINT a scary message of your choice!

**Followup:** Also, make it plot in RED the bar representing the temperature that it changed TO, for each time it changed by 10% or more, up or down.