NAME: **login ID:**
precept:

COS 126 Midterm 1 Programming Exam, Fall 2010

This part of the exam is like a mini-programming assignment. You will create a program, compile it, create test data for it and run it on your laptop. Debug it as needed. This exam is open book, open browser. You may use code from your assignments or code found on the COS126 website. When you are done, submit your program via the course website using the submit link for Precept Exam 1 on the Assignments page.

Grading. Your program will be graded on correctness, clarity (including comments), design, and efficiency. You will lose a substantial number of points if your program does not compile or if it crashes on typical inputs.

Print your name, login ID, and precept number on this page (*now*), and write out and sign the Honor Code pledge before turning in this paper. Note: It is a violation of the Honor Code to discuss this midterm exam question with anyone until after everyone in the class has taken the exam. You have 50 minutes to complete the test.

"I pledge my honor that I have not violated the Honor Code during this examination."

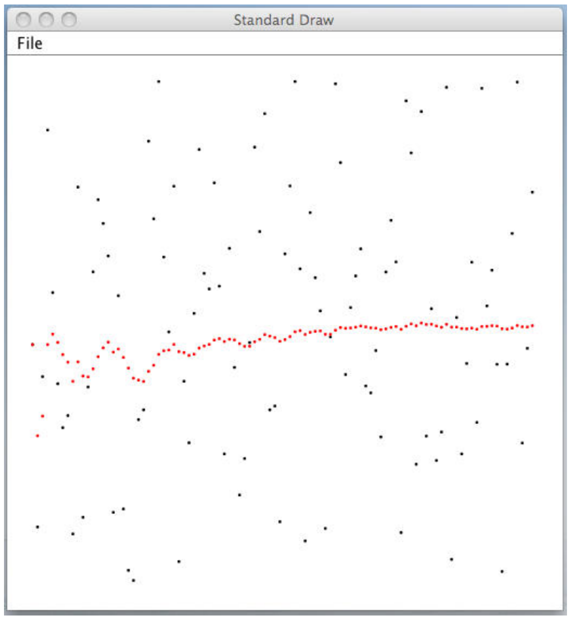
Signature

1	/20
2	/10

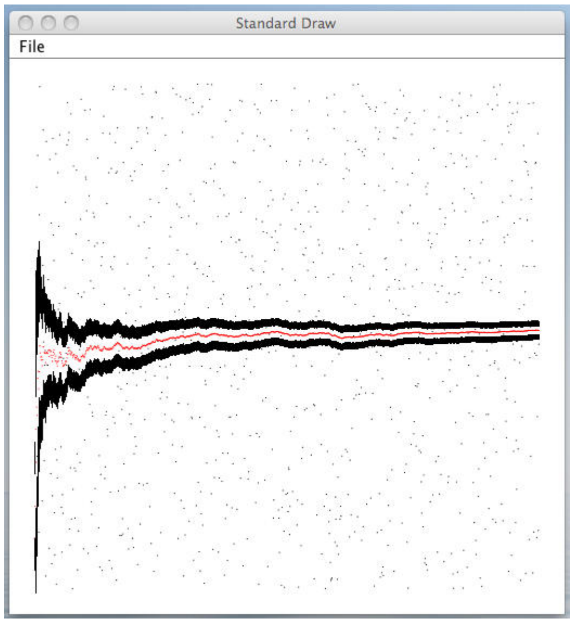
October 21, 2010

Your goal is to write programs that produce pictures like these, which are known as *Tukey plots*. In Part 1, you write the code to draw the dots (shown on the left); in Part 2 you write the code to draw the lines(shown on the right).

Part 1: `java PlotMean 100`



Part 2: `java PlotTukey 1000`



Part 1. Write a Java program `PlotMean.java` that takes an `int` value `N` from the command line and uses `StdDraw` to draw the running average of `N` random `double` values (plus a plot of the data itself). Use `Math.random()` to produce `N` random numbers between 0 and 1, saving them in an array `a[]`. Use `StdDraw.setXscale(0, N-1)` to change the scale of the *x* coordinate. Then, for each `i` between `0` and `N-1`:

- plot a black point at (`i`, `a[i]`).
- plot a red point at (`i`, `mean`) where `mean` is sample mean of the first `i+1` points

To compute the sample mean for the first `i+1` values, compute their sum and divide by `i+1`. For example, if `a[]` is {`.1, .2, .3`} the sample mean for `i = 2` is `(.1+.2+.3)/3 = .2`.

You will need to have `StdDraw.java` or `StdDraw.class` in your directory in order to compile and test your program. Use `StdDraw.setPenColor(StdDraw.RED)` to change the pen color to red and `StdDraw.setPenColor(StdDraw.BLACK)` to change it back to black. You may use `StdDraw.setPenRadius(.5/N)` to better see the plotted points.

Submit your program `PlotMean.java` via the link on the Assignments page.

Part 2. Using `PlotMean.java` as a starting point, develop a program `PlotTukey.java` that draws a full Tukey plot.

First, write a static method with the following signature:

```
    public static double mean(double[] a, int n)
```

The purpose of this method is to encapsulate the computation of the *sample mean* of `a[0..n]` as described in Part 1. Modify your code in Part 1 to use this method to compute the mean that you will plot in red.

Next, write a static method with the following signature:

```
    public static double sigma(double[] a, int n)
```

The purpose of this method is to compute the *standard error of the mean* of `a[0..n]` which is defined as follows: take the *sample variance,* divide by the size of the sample (`n+1`, in this case), and take the square root. To compute the sample variance, compute the sum of the squares of the differences between the sample mean and each of the values and divide by n. For example, when `a[]` is {`.1, .2, .3`} the sample mean for `i = 2` is `.2`, the sample variance is `(.2-.1)*(.2-.1)+(.2-.2)*(.2-.2)+(.2-.3)*(.2-.3))/2 = .01`, and the standard error of the mean is the square root of `.01/3 = .057735...`

Finally, have your `main` plot as you did in Part 1 and add the code needed to plot two lines of length equal to `sigma(a, i)`, the sample standard deviation of the average, spaced a distance equal to `sigma(a, i)`, both above and below each red dot. The result should be a Tukey plot, like the one shown at right on the previous page.

Note that `StdDraw` will allow you to plot `Infinity` (which is the result when you divide a number by zero). Please make sure that your program plots only valid numbers, not `Infinity`.

Submit your program `PlotTukey.java` via the link on the Assignments page.

Be sure to click the *Check All Submitted Files* button to verify your submission.