What’s the Point?

Tired of programs which are supposed to do things the right way? This is a program where there’s no right answer! In this program, you will be generating random pictures.

What We’ll Be Doing

Here are the key methods and techniques we’re going to practice:

• Creating a Random object
• Using some of the methods of the Random class
• Looping many times to do the same thing
• Simple drawing with StdDraw
• Exploring RGB colors
• Keeping track of variables that change through a loop

Pair up!

Remember, we’ll be doing pair programming this semester. So choose a partner (somebody you’ve never worked with in section), and find a computer.

Part 1: Pseudocode

We’re still experimenting with the group-pseudocode projects. For our first checkpoint, the entire Section will work together to write pseudocode. For this time, though, we’re going to write the pseudocode for two programs, not just one.

Program 1 - White Noise

This program will generate white noise - that is, random static on the screen. The screen will start out white, and then you will randomly choose many locations to draw black squares. The result will be a random picture, something like this one:
The program run with the command line argument: 10000

To be specific, your program should:

- Read a single integer from the command line, and store it into a variable.
- Loop a certain number of times - that argument tells you how many.
- Each time that you loop, generate two random numbers (the x and y coordinates) - and draw a square at that location.
Some things to think about when you convert this program to Java later:

- You’ll need to call `StdDraw.setScale(int, int)` to set the boundaries of the window. Set them to (0,100). (See the Project 3 spec for details.)
- When you generate the random x and y values, they should always be in the ranges (0,100), so that the squares are in the window. How would you do this?
- Do you know how to use `StdDraw` to draw a square? If not, review the Project 3 spec.

**Program 2 - RGB Randomness**

In Project 3, you use the `StdDraw.setPenColor(Color)` method to tell `StdDraw` to draw with a certain color. But there is another way you can do it: use `StdDraw.setPenColor(int, int, int)` to select any RGB value.

“RGB” simply means “red, green, blue” - it means a color represented by three integers. Each of these integers can go from 0 (none of this color) to 255 (the max of this color). So the color 255,0,0 is pure red; 0,0,255 is pure blue, and 255,255,255 is white. Of course, you can mix the colors as well - to get all of the hues of the rainbow.

In this program, you will use nested loops to draw one square at every location on the window. But every square will have a randomly chosen color, like this:

![A run of the program](image)

Write pseudocode as a Section which does the following:
- Uses nested loops to iterate through all of the combinations of x and y. Remember, each one goes from 0 to 100, inclusive.

- At every location, generate three random numbers, between 0 and 255 inclusive. Use the `StdDraw.setPenColor(int, int, int)` call to tell `StdDraw` to use this color - and then draw a single square at the current location.

Again, when you convert this program to Java, you will have some things to think about which we didn’t mention in the pseudocode - such as setting the `StdDraw` scale (use the same scale as before), creating the `Random` object, etc.

**When you’re done...**

This section ends when your Section Leader says so; you may finish all of the points above, or your Section Leader may end it early.

✓ **CHECKPOINT 1**

Everybody who has been taking part should get this checkpoint for free!

---

**Part 2: Convert it to Java**

Log onto your lab computer. Open up a text editor, and copy down all of the pseudocode that the Section has written (unless your SL comes up with a faster way, that lets you download it).

Next, open up the class homepage:

[http://www.cs.arizona.edu/classes/cs127a/spring16/](http://www.cs.arizona.edu/classes/cs127a/spring16/)

Once you’re there, open up two things:

- This section activity
- Slide deck 04 (“Loops”)
- Slide deck D (“Random”)

Now we’re ready to get started.

**Convert from Pseudocode**

Now, convert the pseudocode to Java, a few lines at a time. It’s wise to run your program every few lines (or at least compile it), so that you can find errors early.

I’d like you to convert both programs. I’d recommend doing the random-squares program first; it’s probably simpler. Once you’ve got it working, use what you’ve learned there to get the second program working as well.

✓ **CHECKPOINT 2**
Raise your hand. Your SL will come over and verify that you completed this step correctly.

Time to switch!
Switch seats! The driver should become the navigator, and vice-versa.

Program 3: Random Walk
In this program, you will simulate a “random walk” - that is, an entity which randomly moves about on the screen. To do this, you will start the entity at (0,0) and then loop forever - in every pass of the loop, you will figure out what direction it moves (to any adjacent square), and draw a new square there.

The program, after it ran for a short while

Random and Negative Numbers
In this program, you will need to generate random numbers, but sometimes you want the number to be negative. Happily, though, there’s a simple trick to make this work: generate non-negative numbers first (zero and positives), and then subtract. For instance, if you wanted to generate random numbers between -50 and 50, you would generate random numbers between 0 and 100 - and then simply subtract 50 from whatever you got.

Program Requirements
Your third program should:
• Set the StdDraw scale to some scale (what do you think is a good scale?)

• Start the “walker” in the middle of the screen (how will you keep track of the walker’s position as it moves?)

• Have an infinite loop. In each pass of the loop, generate two random numbers (for x and y) and have the walker move that distance. Then draw a square or circle at the walker’s new position. (How far do you think the walker should move? You get to choose.)

Your walker must sometimes walk forwards, and sometimes backwards - so you must generate negative random numbers!

✓ CHECKPOINT 3
Raise your hand. Your SL will come over and verify that you completed this step correctly.

The End: Clean Up

The G/S 930 lab is host to sections from several of our programming classes, and is available the rest of the time for any CS student to use to work on their programming assignments. To help you get in the habit of leaving your work space neat and clean, we’ll end each section activity with this checkpoint, which also serves as confirmation that you attended section - even if you get nothing else done, you can clean up before you go, thereby earning one checkpoint and proving you attended.

• Log out of your computer.

• Pick up your papers, writing implements, cell phones, etc.

• Push in your chairs.

✓ CHECKPOINT 4
Raise your hand. Your SL will come over and verify that you completed this step correctly.

As with so much in this course, thanks to Dr. McCann for some of the ideas - and even some of the exact text - for this Section Activity!