Programming Assignment 9 (100 Points)

Due: 11:59pm Wednesday, November 23

Start early … Start often!

README (10 points)

You are required to provide a text file named **README,** NOT Readme.txt, README.pdf, or README.doc, with your assignment in your pa9 directory. There should be no file extension after the file name "**README**". Your README should include the following sections:

Program Descriptions (4 points): Provide a high level description of what each of your programs do and how you can interact with them. Make this explanation such that your grandpa or uncle or someone you know who has no programming experience can understand what these programs do and how to use them. Do not assume your reader is a computer science major.

Short Response (6 points):
First, you will need to copy readFile from the public directory to answer the following questions.

```
$ mkdir ~/pa9
$ cp ~/../public/readFile ~/pa9
```

<u>Unix Questions:</u>

1. Using the "cat" command and readFile, how would you display the contents of readFile with each of its lines numbers?
2. Using the "cat" command and readFile, how would you display the contents of readFile showing the tabs and spaces within the file?
3. Without using the "cat" command, what other unix command will write readFile to stdout with line numbers?

<u>Vim Questions:</u>

4. In vim, how would you display the contents of readFile with line numbers?
5. In vim, how would you display the contents of readFile showing the spaces and tabs within this file?
6. In vim, how would you replace only the first instance of "fun" with "amazing" in each line? (*Hint*: go through the Vim tutorials provided in the "Useful links" page of the course site.)

<u>Power Questions:</u> (see section Program 2: Power.java below)

STYLE (20 points)

Please see previous programming assignments; especially PA2's writeup.
You will be specifically graded on commenting, file headers, class and method headers, meaningful variable names, judicious use of blank lines, not having more than 80 characters on a line, perfect indentation, no magic numbers/hard-coded numbers other than -1, 0, +1, using setters and getters, etc (**see the style section of PA2's writeup**).

CORRECTNESS (70 points)

Begin by first creating your pa9 directory and copying over the starter code using the following commands. You will need Power.java for Program 2.

```
$ mkdir ~/pa9
$ cp ~/../public/Power.java ~/pa9
```

Program 1: MultipleTurtles.java

For this assignment, you will modify and extend your PA1 CS11TurtleGraphics programming assignment. We will NOT be using Acme.jar or objectdraw.jar for this assignment; it is just like your PA1 with a few additions. You will create multiple turtles (in separate threads) to draw the letters and numbers in parallel.

A primer on Java threading can be found here: http://www.javaworld.com/javaworld/jw-06-2012/120626-modern-threading.html

Setup:

Copy your pa1 single-threaded turtle graphics program to your new pa9 directory.

```
$ cp ~/pa1/CS11TurtleGraphics.java ~/pa9/MultipleTurtles.java
$ cp ~/../public/turtleClasses.jar ~/pa9
```

Make sure you change all instances of CS11TurtleGraphics to MultipleTurtles: change the class name, filename (you just did this with the cp above), constructor name, and any other places CS11TurtleGraphics shows up in your program (Hint: README Q6 should help with this).

You will add (implement) the Runnable interface to your class.

You will need to modify the constructor to take the character to be drawn, the x and y coordinates where the character is to be drawn, and a delay for creating the animation.

```
public MultipleTurtles(World w, char ch, int x, int y, int delay)
```

Remember, the call to the superclass (Turtle) constructor needs to be the first statement in the constructor:

```
public MultipleTurtles(World w, char ch, int x, int y, int delay) {
  super(w, delay);
  //…
}
```

The delay is to help see the animation. Set the delay to half a second:

```
private final static int DELAY = 500; // Half sec delay for the animation
```

You will set the pen width and color in the constructor. Most importantly, you must add:

```
new Thread(this).start();
```

as the **<u>last line</u>** of your MultipleTurtles constructor. The Thread start() method will call the run() method for this Thread. **Do not call run() directly**!

Every time you create a new MultipleTurtles object, you will be creating a new turtle in a new Thread and starting this Thread (turtle graphics object) to draw a specific character at a specific location. If you want to learn more about Java's Thread class and Runnable interface, go to:
http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html and
http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html .

You will need to add a run() method to your MultipleTurtles class. It should look something like this:

```
public void run() {
  switch(ch) {
    case 'C': this.drawC(x,y); break;
    case 'S': this.drawS(x,y); break;

    // … and so on.
    // You may need to tweak the draw methods depending
    // on how you did it in the first assignment.
  }
}
```

Again, you **do not call run() directly -** it is automatically called from the Thread start() method after the Thread begins.

In main() you will want to create a new MultipleTurtles object (a new turtle drawing a character in a separate thread) for each letter or number that needs to be drawn, so they can all execute simultaneously in parallel.

```
new MultipleTurtles(w, 'C', x, y, DELAY); // This turtle draws a C char
new MultipleTurtles(w, 'S', x, y, DELAY); // This turtle draws a S char
// … and so on.
```

Result:
When you run your program, you will see all the turtles drawing all the characters at the same time in parallel. Note: it is ok for some turtles to finish drawing before others (since some letters will be more complicated than others). The important part is that all the turtles start drawing at the same time. Here are some screenshots (your's should spell out your cs11f course-specific account name):

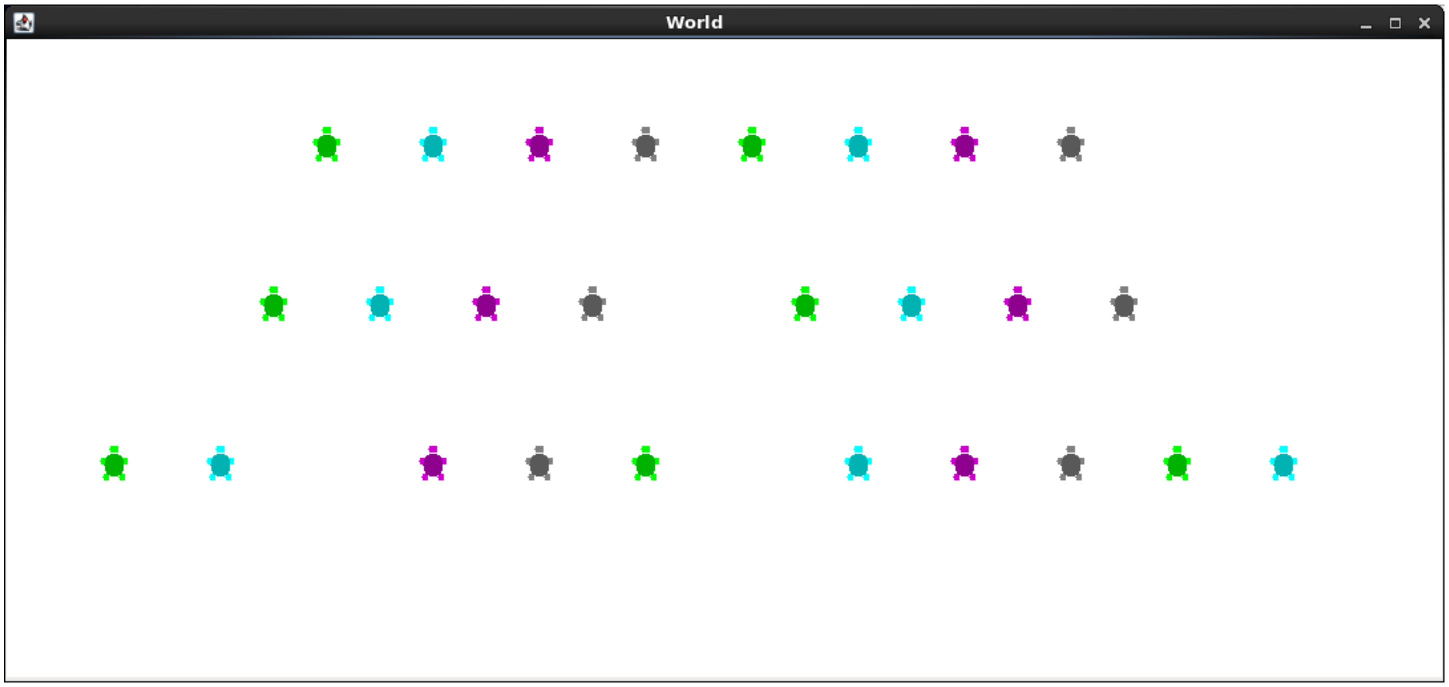
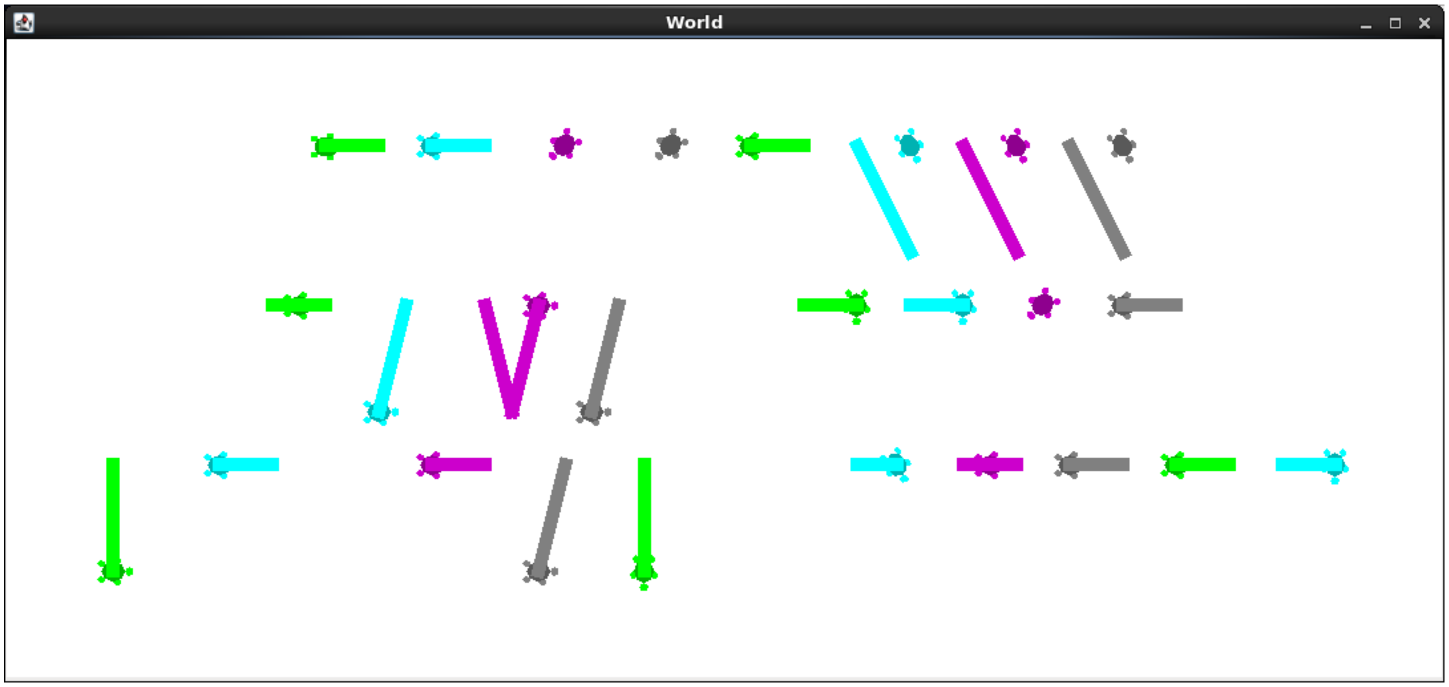
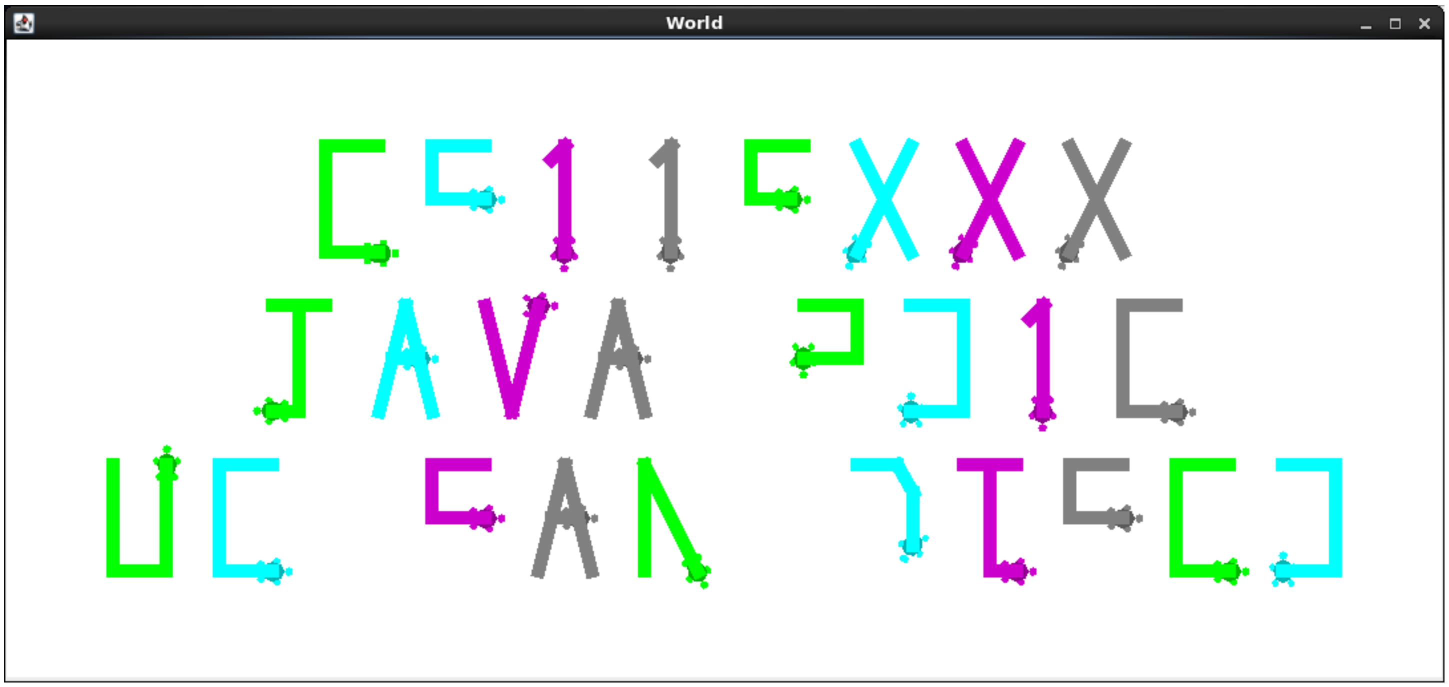


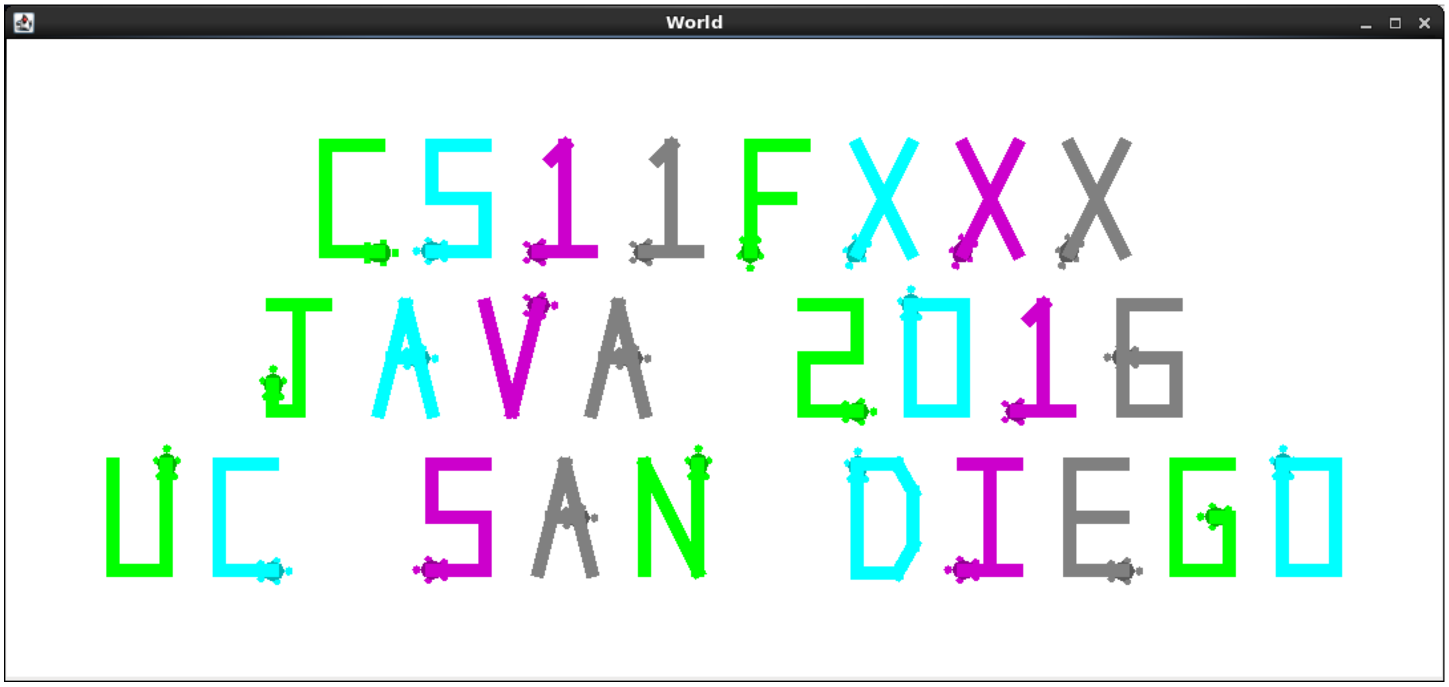
Image of what the program starts with



In the process of drawing



Nearly done drawing. Note that it's ok if some turtles finish drawing before others.



Completed.

IMPORTANT: the feature for centering the text horizontally and vertically is <u>required</u>. If you did not do this for PA1 as extra credit, you must complete this feature for PA9. The turtle colors DO NOT have to match the writeup, it's best if you use your PA1 Extra Credit to do this assignment. There is no extra credit offered for including any extra decoration like in PA1 but it is still highly encouraged.

Running:
Multi-Threaded Turtle Graphics (application): You should already know how to run this program/application. (Hint: you must include turtleClasses.jar in the classpath).

Program 2: Power.java

You must use recursion and implement the methods listed below or you will lose all 20 points for this program. Also, do not edit the main() method in Power.java as it will negatively affect your grade (you may edit style at your own risk).

Write a command line application that will examine the efficiency of two different recursive algorithms to calculate the power of a base and exponent (these values are passed in as command line arguments). The main() method has already been written for you (do not modify it--see note above). Your task will be to implement the two different recursive power formulas (power1() and power2() -- see the starter code) given their recurrence relations. In other words, you need to translate the recurrence relations into code. You will also need to keep track of how many multiplications were performed during the calculations, so we can compare the efficiency of the two algorithms.

In both the power functions, x represents the base, and n represents the exponent. Compute the powers by converting the following recurrence relations into code. The result should be that power1(x, n) = x^n and power2(x, n) = x^n. You must use recursion. You can assume that n >= 0 and x != 0.

Keep a count of the number of multiplications for both recursive versions of Power with the already defined variables power1MultCnt and power2MultCnt. main() prints out the computed power values and the number of multiplications for both recursive power functions for you.

Power1 Recurrence Relation:

$$\text{Power1}(x,n) = \begin{cases} 1 & \text{if } n = 0 \\ x * \text{Power1}(x, n-1) & \text{if } n > 0 \end{cases}$$

Power2 Recurrence Relation:

$$\text{Power2}(x,n) = \begin{cases} 1 & \text{if } n = 0 \\ \text{Power2}(x, n/2) * \text{Power2}(x, n/2) & \text{if } n \text{ is even} \\ x * \text{Power2}(x, n-1) & \text{if } n \text{ is odd} \end{cases}$$

Note that for Power2, you should only compute Power2(x, n/2) once, and then simply multiply that result with itself (vs. recursively calling Power2(x, n/2) twice).

Once you have successfully implemented these methods, answer the following questions in your README file.

7. If you double the value of n (say from 256 to 512), how does this affect the number of multiplications for Power1?
8. If you double the value of n (say from 256 to 512), how does this affect the number of multiplications for Power2?

Sample Output: (user input shown in **BOLD**)

1. No arguments

```
[cs11fxyz@ieng6-201]:pa9$ java Power
Usage: java Power BASE EXPONENT
[cs11fxyz@ieng6-201]:pa9$
```

2. Too many arguments

```
[cs11fxyz@ieng6-201]:pa9$ java Power 2 19 5
Usage: java Power BASE EXPONENT
[cs11fxyz@ieng6-201]:pa9$
```

3. Valid arguments

```
[cs11fxyz@ieng6-201]:pa9$ java Power 2 5
power1(2.000000, 5) = 3.200000e+01  # of multiplies = 5
power2(2.000000, 5) = 3.200000e+01  # of multiplies = 4

[cs11fxyz@ieng6-201]:pa9$
```

4. Valid arguments

```
[cs11fxyz@ieng6-201]:pa9$ java Power 3 4
power1(3.000000, 4) = 8.100000e+01  # of multiplies = 4
power2(3.000000, 4) = 8.100000e+01  # of multiplies = 3

[cs11fxyz@ieng6-201]:pa9$
```

Note: If your output does not match the sample output EXACTLY, NO partial credit will be given for test cases.

EXTRA CREDIT (2 points): Early Turnin

[2 Points] Turn in your assignment 48 hours before regular due date and time
[1 Point] Turn in your assignment 24 hours before regular due date and time

Note: Early Turnin is one or the other, not both.

Turnin

Turnin

To turnin your code, navigate to your home directory and run the following command:
> **cse11turnin pa9**

You may turn in your programming assignment as many times as you like - each turnin overwrites the previous turnin. The last submission you turn in before the deadline is the one that we will collect and grade.

Verify

To verify a previously turned in assignment,
> **cse11verify pa9**

If you are unsure your program has been turned in, use the verify command. **We will not take any late files you forgot to turn in.** Verify will help you check which files you have successfully submitted. It is your responsibility to make sure you properly turned in your assignment.

Files to be collected:

- README
- MultipleTurtles.java
- Power.java
- turtleClasses.jar

NO LATE ASSIGNMENTS ACCEPTED.
DO NOT EMAIL US YOUR ASSIGNMENT!
And above all … Have Fun!