Programming Assignment 9 (PA9) - Multiple Turtles and Power

Due Date: Wednesday, November 28 @ 11:59 pm

<table>
<thead>
<tr>
<th>Assignment Overview</th>
<th>Grading</th>
<th>Gathering Starter Files</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple Turtles</td>
<td>README File</td>
<td>Extra Credit: Early Turnin</td>
<td>Turnin Summary</td>
</tr>
</tbody>
</table>

**Assignment Overview**

For the first part of the assignment, you will write a command line application that will examine the efficiency of two different recursive algorithms to calculate the power of a base and exponent.

For the second part of the assignment, you will modify and extend your PA1 CSE11_TurtleGraphics programming assignment. We will NOT be using Acme.jar or objectdraw.jar for this assignment; it is just like your PA1 with some modifications. You will create multiple turtles (in separate threads) to draw letters and numbers with seven-segment displays in parallel.

![Multiple Turtles and Power](image-url)

**Grading**

- **README: 10 points** - See README Requirements [here](http://cseweb.ucsd.edu/~ricko/CSE11READMEGuidelines.pdf) and questions below
- **Style: 20 points** - See Style Requirements [here](http://cseweb.ucsd.edu/~ricko/CSE11StyleGuidelines.pdf)
- **Correctness: 70 points**
- **Extra Credit: 5 points** - View Extra Credit section for more information.

**NOTE:** If what you turn in does not compile, you will receive 0 points for this assignment.

**Gathering Starter Files**

You will need to create a new directory named pa9 and go into that directory. The $ represents your command prompt. What you type in is in **bold**.
Copy the starter files from the public directory:

```bash
$ cp ~/../public/turtleClasses.jar .
$ cp ~/../public/PA9StarterCode/Power.java .
$ cp ~/../public/PA9StarterCode/funFile .
```

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

```bash
$ cp ~/pa1/CSE11_TurtleGraphics.java ~/pa9/MultipleTurtles.java
```

Starter files provided:

<table>
<thead>
<tr>
<th>turtleClasses.jar</th>
<th>Power.java</th>
<th>funFile</th>
</tr>
</thead>
</table>

**Power**

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

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:**

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

**Power2 Recurrence Relation:**

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

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

1. No arguments
   $ java Power
   Usage: java Power BASE EXPONENT
   $

2. Too many arguments
   $ java Power 2 19 5
   Usage: java Power BASE EXPONENT
   $

3. Invalid Base
   $ java Power abc 123
   BASE must be a valid floating point value
   $

4. Invalid Exponent
   $ java Power 2 abc
   EXPONENT must be a valid integer value >= 0
   $

5. Valid arguments
   $ java Power 2 5
   power1(2.000000, 5) = 3.200000e+01  # of multiplies = 5
   power2(2.000000, 5) = 3.200000e+01  # of multiplies = 4
   $

6. Valid arguments
   $ java Power 3 4
   power1(3.000000, 4) = 8.100000e+01  # of multiplies = 4
   power2(3.000000, 4) = 8.100000e+01  # of multiplies = 3
   $

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

---

**MultipleTurtles**

For this assignment, you will modify your PA1 CSE11_TurtleGraphics 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 letters and numbers using seven-segment displays in parallel.

Make sure you change all instances of CSE11_TurtleGraphics to MultipleTurtles: change the class name, filename (you just did this with the cp above), constructor name, and any other places CSE11_TurtleGraphics shows up in your program.

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.

```java
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:

```java
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:

```java
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:

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

as the **last line** 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](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](http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html).

**Parse command line arguments:**

The first three arguments will be the three lines of text the turtles will draw. If there are more than 3 arguments, the rest will be ignored. If there are 0-3 (inclusive) arguments, the first argument will become the first line, the second argument will become the second line, and the third argument will become the third line. If there are not enough arguments for three lines, then the rest will be spaces.

Examples:

```
$ java -cp turtleClasses.jar:. MultipleTurtles "UCSD"
```

The first line will be “UCSD” and second and third line will be spaces.
$ java -cp turtleClasses.jar:. MultipleTurtles
All three lines will be spaces.

$ java -cp turtleClasses.jar:. MultipleTurtles "hello" "" "world"

Each line the turtles draw can have at most 10 characters. For arguments less than 10 characters, fill with spaces. For arguments longer than 10 characters, only display the first 10 characters.

For characters entered that are not numbers or valid letters (letters with angles like K, M, Q, V, W, X, or Z are not valid), print a dash "-" instead. Use lowercase letters for O, N, and U.

Create turtles:
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. Think about how you can do this efficiently (hint: loops).

Draw the segments:
Instead of defining a method for each character to draw, you MUST use an array of boolean arrays, where each boolean array represents one character. This way, you don’t need to write many different methods for each character (so make sure to delete all of your drawC, drawS, drawE, drawLetter methods that draw out all the segments for a single letter). You can write a method that takes in the boolean array and draws the character based on the boolean array.

```java
private static final boolean OFF = false;
private static final boolean ON = true;
private boolean[][][] segments = {
    /* 0 [0] */ { ON, ON, ON, ON, ON, ON, OFF },
    /* 1 [1] */ { OFF, ON, ON, OFF, OFF, OFF, OFF },
    ... 
};
```
Each element in the boolean array represents whether that segment should be ON/OFF. The element at index 0 means segment A is ON/OFF and so on. Take 0 as an example, segments A-F are all ON and segment G is OFF.

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

```java
public void run() {
    /* Check what the character is and draw the corresponding character.*/
}
```

Set the turtle to invisible after it finishes drawing!

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

Result:
When you run your program, you will see all the turtles drawing all the seven-segment displays at the same time in parallel. Here are some screenshots:

At start up with input string: “Do you study” “programming” “everyday?”

When a few segments have been drawn.

When all the segments are drawn.

When you run the program with empty input string “”.

Compiling and Running:

```bash
$ javac -cp turtleClasses.jar:. MultipleTurtles.java
```

This will not have any letters drawn, just spaces:

```bash
$ java -cp turtleClasses.jar:. MultipleTurtles
```
This should display "ABC" on the first line, "DEF" on the second line, and "123" on the third line:

```
$ java -cp turtleClasses.jar:. MultipleTurtles "ABC" "DEF" "123"
```

Be sure to test your program with many different inputs to ensure it works properly.

**README File**

Remember to follow all of the guidelines outlined in the README Guidelines. If you did the extra credit, write a program description for it in the README file as well.

**Questions to Answer in your README:**

1. Using the “cat” command and funFile, how would you display the contents of funFile with each of its lines numbered?
2. Using the "cat" command and funFile, how would you display the contents of funFile showing the tabs and spaces within the file? How many tabs and how many spaces are in the third line of this file?
3. Without using the "cat" command, what other unix command will write funFile to stdout with line numbers?
4. If you double the value of n (say from 256 to 512), how does this affect the number of multiplications for Power1?
5. If you double the value of n (say from 256 to 512), how does this affect the number of multiplications for Power2?
6. What should you do when you see two classmates you don't know well comparing code and sharing answers in lab?

**Extra Credit: Early Turnin**

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

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

**Turnin Summary**

See the turnin instructions [here](#). Your file names must match the below *exactly*.

**Due Date:** Wednesday night, November 28 @ 11:59 pm

**Files Required for Turnin:**

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

If there is anything in these procedures which needs clarifying, please feel free to ask any tutor, the instructor, or post on the Piazza Discussion Board.
NO EXCUSES!
NO EXTENSIONS!
NO EXCEPTIONS!
NO LATE ASSIGNMENTS ACCEPTED!
DO NOT EMAIL US YOUR ASSIGNMENT!

Start Early, Finish Early, and Have Fun!