Leaf Project

Directions: You will be completing this program on your own – using your notes, prior projects, and online resources. Limit the number of times you ask the teacher for help – try to debug your program on your own. As a suggestion, make sure that you complete each step fully (meaning the program compiles correctly) before moving on to the next step. This means that you complete step one before you move onto step two and you complete step two before your move onto step three, etc. Mrs. Moffat will not help you with your program if you complete these steps out of sequence.

Leaf Class
complete this class first…

1. Create four **fields** for an individual Leaf:
    a. An integer (numLeaves) shared by all leaves to keep track of how many total leaves have been built.
    b. An integer (timer) to represent how long before the leaf will begin falling.
    c. An integer (dx) to represent how far left/right the left will move.
    d. A boolean (fallingRight) representing if the leaf is currently falling left or falling right.

2. In the constructor for Leaf, add the following:
    a. Set the image to a random picture (Leaf1.png, Leaf2.png, Leaf3.png)
        i. DO THIS ***WITHOUT*** USING AN if-STATEMENT
    b. Set the rotation of the Leaf to a random number [0, 360).
    c. Set the timer to a random number [100, 1000).
    d. Set the fallingRight variable to true 50% of the time and false the rest of the time.
    e. Increase the numLeaves static field by 1.

3. In the act method for Leaf, add the following (in this order!):
    a. Subtract one from the timer variable.
    b. Check if the timer is negative or zero. If so, do the rest of the steps below:
        i. Set the location of the leaf to:
            1. (old x-coordiante + dx/2, old$y$-coordinate + 5 - $|dx|/4$
                a. HINT: $|dx|$ = Math.abs(dx)
        ii. Check if the fallingRight variable is true:
            1. If so, add one to dx
            2. Check if dx is greater than 20:
                a. If so, set fallingRight to false
        iii. If the fallingRight variable was false (else):
            1. If so, subtract one to dx
            2. Check is dx is less than -20:
                a. If so, set fallingRight to true
        iv. Check if the y-coordinate is past the value of 600:
            1. If so, subtract one from the numleaves static field and remove the leaf from the screen.

4. Find the two commented out methods in Leaf… uncomment these out.

Tree Class

*to be completed **after** the Leaf Class is working…*

1. In the addLeaves method, an amount of leaves to add is coming in as a parameter. The intention of this method is to add a very specific amount of leaves onto the screen.
    a. Create a *for* loop that will loop <amount> number of times.
    b. Inside the loop's body, do the following:
        i. Create a Leaf object.
        ii. Create a random x integer from [100, 900).
        iii. Create a random y integer from [0, 400).
        iv. Add the Leaf object at (x, y).

2. In the increaseLeavesTo method, an amount of leaves to increase the number of leaves to comes in as a parameter. The intention of this method is to add enough leaves to the screen that the total leaves on the screen is equal to the amount value that came in.
    a. Create a *while* loop which checks the Leaf's static method getNumLeaves and sees if it is less than the amount variable.
    b. Inside the loop's body, follow the same steps as were done in the addLeaves' *for* loop.

3. In the Tree's constructor, under the super line of code, call the static method called reset which was written in the Leaf class.

Everything working?
No syntax errors?!
Ready to check?!!

1. On your world screen (with the picture of the tree and the pile of leaves) right click and select "void addLeaves(int amount)".
    a. Write "200" and press "Enter"
    b. Select the "Run" option…
    c. Does it work?
        i. Awesome! You just wrote an entire program on your own.

Want a challenge?

OPTIONAL:

Do you see how the same four lines of code are being done in both loops for #1 and #2 in the Tree Class? If code is being repeated multiple times, then it would be best if we made a method for it instead. Following what other methods loop like, try to write a method called addRandomLeaf which takes in no parameters and returns no information. The method should do the same four tasks that was inside both loops. After the method is written, erase the four lines of code in both loops and replace those with a call to the addRandomLeaf method.