CS1316 Homework 2: Learning to Walk Due 30 May, Friday

Well, hopefully you already know how to walk. But now it's time to teach your Sausage Man! For this assignment you will be making your Sausage Man walk in place, as well as advancing some of the ideas from HW1. To do this, you will create a **`WalkingSausageMan`** class and you will modify your **`ColoringPicture`** class. Below are the requirements for each class and the overall homework, as well as suggestions on how to approach this homework.

Overall task:

Create a 'movie' using frame sequences that shows a Sausage Man walking in place, where the sausage man's body color changes five times during each 14-picture loop.

Getting Started:

1. Download the SausageMen.zip file from the Homework 2 resources (under the Assignments tab) on T-Square. Unzip the file, and place all 14 SausageMen pictures into your media-sources file (they must be put directly into media-sources, not within a file within media-sources).
2. If you didn't finish the colorIn() method for HW1 completely / correctly, take a look at the solution posted under Homework 2 resources on T-Square (it will be posted sometime after the HW1 grace period ends…by Monday night at the very latest).

Modifying ColoringPicture.java:

You must remove your ColoringPicture colorIn() method's dependence on java.awt.Color by instead using threshold values for red, green, and/or blue pixel values. In the real world, picture colors are not always pure, and hence, do not usually correspond to the 13 java.awt.Color colors. In homework 1, you probably found yourself doing something like this:

```
if(pixel.getColor().equals(Color.YELLOW)) {
    do something
}
```

But if the color we are referring to is not pure yellow (as is the case with the 14 Sausage Men you will be working with in the homework), then this will not work. Instead, we can check the pixel's red, green, and/or blue values and determine from these if we want to consider this pixel as yellow (or whatever color we are comparing to).
Remember that we did something similar to this in the PurplePicture class written in lecture last Wednesday. You can search for RGB charts for colors that interest you to determine what appropriate thresholds might be. For yellow, it seems that red greater than 240, green greater than 165, and blue less than 20 are appropriate (these are approximate…you can adjust your thresholds as needed to get the desired results).

Required:

Change your calls in colorIn() to be independent of the pure colors defined by java.awt.Color. You should do this by defining and using threshold values for red, green, and/or blue values for each color you want to compare to (instead of using java.awt.Color).

Suggestions:

1. Instead of using Color.YELLOW, define the red, green, and blue thresholds for yellow, and use these in your

   ```
   if(pixel.getColor().equals(Color.YELLOW)) {
       do something
   }
   ```

   block.
2. The statement where you set your pixel to be Color.RED if it is currently yellow should remain the same (ie, it should still set the pixel to Color.RED).

Creating WalkingSausageMan.java

Now it's time to actually put all the Sausage Man pictures together in a Frame Sequence, so that we can get our movie of a walking Sausage Man.

Required:

Create a 'movie' using frame sequences that shows a Sausage Man walking in place, where the sausage man's body color changes five times during each 14-picture loop

Suggestions:

1. Cut and paste the StripedSwan class from last Wednesday's lecture into your new WalkingSausageMan class. Change all occurrences of StripedSwan in your new WalkingSausageMan class to WalkingSausageMan.
2. Make the FrameSequence variable 'frames' static.
3. Within the else block in the main method, add the lines

   ```
   frames.show();
   frames.replay(60);
   ```

   after your renderAnimation call.
4. Remove the

   ```
   frames.show();
   ```

   call from the renderAnimation(int times) method.
5. Make your constructor empty…it doesn't need to take in anything or do anything.
6. Now all that you need to work with is the renderScene method. Unlike in the StripedSwan example, we do not need to write any additional methods or use methods from any other classes (although you can if you want).

a. Think about the logic necessary to change the color five times during each 14-picture loop. How do we decide when colorIn() should be called on a Sausage Man picture? (Hint: You might find the % useful)
b. Also, we must consider how to specify the name of a particular Sausage Man picture given only int t. (Hints: Remember there are 14 different Sausage Man pictures. You will need string concatenation, and you might find the % useful here as well).

Reminders:

- If you're code does not compile, you will receive a zero for the assignment
- Make sure to turn in the .java file, NOT the .java~ or the .class files
- When you turn in, *NEVER* use the Save as Draft feature. You can (and should) submit multiple times.
- You must comment your code (you will lose points if you do not)
- You will lose points if you do not put your name, T-Square log-in, and a collaboration statement at the top of all files.

Files to turn in

- **`ColoringPicture.java`**
- **`WalkingSausageMan.java`**

Extra Credit

- Write another method in WalkingSausageMan, called **renderCoolScene(int t)**, that returns a ColoringPicture and produces a neat animation when you comment out the call to renderScene in the main method, and replace it with a call to renderCoolScene. Points will be awarded based on how cool your animation is and difficult your renderScene method is.