CS 145: Homework 1

Sprite

1 Description

Your primary goals in this homework are to learn about bundling up code into methods and using objects given to you by someone else. You will do this in the context of telling a story using images that shift, rotate, and scale.

You will write one class for this homework: `Story`. Place it in a package named `hw1`.

2 Keyframe animation

Keyframe animation is a method of animating in which the animator explicitly defines the position of an object at only a handful of frames of the animation. In between any two of these so-called *keyframes*, it is the computer who positions the object. It does so by blending the position information registered at the two surrounding keyframes. For example, an animator places a ball at the left of the screen at frame 0. At frame 100, the animator places it at the right. The animation has two keyframes. When the computer draws frame 10, it positions the ball 10% of the way to the right of the screen, because frame 10 is 10% of the way between the keyframes. At frame 50, it positions it in the middle.

In this homework, we provide a class for animating a 2-D image. Animators call animated 2-D images *sprites*, so our class is named `Sprite`. You can do three things to a `Sprite`: rotate it about its center, scale it about its bottom middle pixel, and move it on the x- and y-axes. See its documentation (linked in the Files section below) for more details.

3 Story

Your `Story` class scripts the animation by using some classes we provide. It must meet these requirements:

- It must have a `main` method that creates a `SpriteAnimator` object (only one), creates `Sprite` objects and positions them at various keyframes, and plays the animation. See the documentation for the `SpriteAnimator` class to see how a `SpriteAnimator` object can be created and how you can trigger the animation.
- Your animation must contain at least three different `Sprite`s.
- Your animation must be broken down into at least three methods. You may take one of two approaches. The first choice is to animate each sprite in its own `private` method. For example, you may have a moon orbit through the sky. So, write a method, perhaps named `animateMoon`, that accepts an `SpriteAnimator` parameter. Call this method

from `main`, passing it the `SpriteAnimator` object you made in `main`.) This method handles all moon-related code, keeping `main` halfway readable. The second choice is to break your story up into coherent chapter-methods, like `animateWaltzScene` and `animateGlassSlipper`.

- To animate a sprite, you must create or locate an image file, drop it in your `hw1` package, create a `Sprite` object, position it, and then register it with your `SpriteAnimator` object. Here's an example of ten-second swirly animation using an `SpriteAnimator` object named `movie`:

```
Sprite vortex = new Sprite("vortex.png", 0);

// Adjust sprite for keyframe at time 0
vortex.move(10, 10);
movie.register(vortex, 0.0);

// Adjust sprite for keyframe at time 10
vortex.rotate(1440.0);
movie.register(vortex, 10.0);
```

  The image files (e.g., `vortex.png`, not provided) must be placed in your `hw1` package.

- Your animation must have at least ten keyframes overall. Each sprite must have at least two keyframes in order to be drawn.

- Each instance of a `Sprite` (created by calling `new Sprite(...)` is completely independent of all other instances. Tweaking one will not change the others, even if they are made with the same image file. (See the chickens in the example video.)

The `SpriteAnimator` and `Sprite` classes are provided to you in the SpecChecker JAR file. Just add the file to your build path and they'll be usable in your code. As with `Scanner`, you'll need to import them.

You are encouraged to share your animation by capturing it with software like Camtasia and uploading it to YouTube. Share a link to it on Piazza with tag `#hw1video`. The owner of the video with the most likes will be handsomely rewarded.

4 Other expectations

We need to be able to read your code. Some of your grade will be based on stylistic matters—not just a functioning final result.

- Variable and method names start with a lowercase letter and have capitalized internal words, e.g., `kittenCount`.

- Code should be documented. Explain what you are doing in brief statements.
- Code should be indented properly. Hit Control-Shift-F inside your Java files to let Eclipse clean up your code.

5 Files

- SpecChecker: twodee.org/teaching/cs145/2012A/homework/speccheck_hw1.jar
- `Sprite` API: twodee.org/teaching/cs145/2012A/homework/Sprite.html
- `SpriteAnimator` API: twodee.org/teaching/cs145/2012A/homework/SpriteAnimator.html
- Example: youtu.be/rhe0aQ4U33s

6 Submission

This is a full homework. The SpecChecker will only see to it that you named your classes properly, placed them in the right package, and formed your methods with the right arguments and return types. It will not grade, but it will package up your code for submission. Please follow the directions from preassignment 1 closely in order to submit your work.