Java Tick 3*

In this exercise you will write a program to produce animated GIFs showing successive generations of a world in Conway's Game of Life. Your first task is to complete the implementation of `OutputAnimatedGif` displayed at the end of this exercise by providing an implementation for the body of the `makeFrame` method. The method must draw the current state of the world into a `BufferedImage`. You can create a new `BufferedImage` as follows:

```
BufferedImage image = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
```

where `w` and `h` are `int` variables containing the desired width and height of `image`. To draw onto `image` you need to create a graphics context. When you have finished using the graphics context, you must call the `dispose` method to clean up system resources. Here is an example usage pattern:

```
Graphics g = image.getGraphics(); //create a new graphics context
 // call methods on "g" here to draw onto the image
g.dispose(); // free up resources used by the graphics context
```

The drawing routines for the graphics context are available in the Java library documentation. You may find the methods `setColor`, `fillRect` and `drawLine` helpful. Once you have written your implementation of `makeFrame` you can use `OutputAnimatedGif` to produce an animated GIF file as follows:

- Create an instance of `OutputAnimatedGif` by calling the constructor and supplying the filename of the animated GIF as a Java String.
- Call the `addFrame` method to add a specific generation of the world to the animated GIF; you need to call `addFrame` each time you wish to add a new generation.
- Call the `close` method to write the animation to the file.

Your next task is to write a Java class called `AnimatedLife` which accepts a format string, an integer describing the number of generations to include, and a filename for the output. For example,

```
crsid@machine~> java uk.ac.cam.your-crsid.tick3star.AnimatedLife \
 "Glider:Richard Guy (1970):20:20:1:1:010 001 111" 10 output.gif
```

should draw the first 10 generations of a glider to the file named `output.gif`. You may find it helpful to use `PatternLife.java` as the basis for writing `AnimatedLife`. Please include a sample output from your program in a file called `competition.gif`, and the actual pattern you gave your program on the command line to generate your competition GIF entry, saved in a plain text file `competition.txt`; the best submission will appear on the course website.

Construct a jar file named `crsid-tick3star.jar` containing your code and competition entry and email it as an attachment to `ticks1a-java@cl.cam.ac.uk`. Your jar file should contain:

```
META-INF/
META-INF/MANIFEST.MF
uk/ac/cam/your-crsid/tick3star/competition.gif
uk/ac/cam/your-crsid/tick3star/competition.txt
uk/ac/cam/your-crsid/tick3star/AnimatedLife.java
uk/ac/cam/your-crsid/tick3star/AnimatedLife.class
uk/ac/cam/your-crsid/tick3star/Pattern.java
uk/ac/cam/your-crsid/tick3star/Pattern.class
uk/ac/cam/your-crsid/tick3star/OutputAnimatedGif.java
uk/ac/cam/your-crsid/tick3star/OutputAnimatedGif.class
```

You should set the entry point of the jar file so that the following works:

```
crsid@machine:~> java -jar crsid-tick3star.jar \
"Glider:Richard Guy (1970):20:20:1:1:010 001 111" 10 output.gif
```

```
package uk.ac.cam.your-crsid.tick3star;

// Tell the compiler where to find the additional classes used in this file
import java.awt.*;
import java.awt.image.*;
import java.io.*;
import javax.imageio.*;
import javax.imageio.stream.*;
import javax.imageio.metadata.*;

public class OutputAnimatedGif {

 private FileImageOutputStream output;
 private ImageWriter writer;

 public OutputAnimatedGif(String file) throws IOException {
  this.output = new FileImageOutputStream(new File(file));
  this.writer = ImageIO.getImageWritersByMIMEType("image/gif").next();
  this.writer.setOutput(output);
  this.writer.prepareWriteSequence(null);
 }

 private BufferedImage makeFrame(boolean[][] world) {
  //TODO: complete this method
 }

 public void addFrame(boolean[][] world) throws IOException {
  BufferedImage image = makeFrame(world);
  try {
   IIOMetadataNode node = new IIOMetadataNode("javax_imageio_gif_image_1.0");
   IIOMetadataNode extension = new IIOMetadataNode("GraphicControlExtension");
   extension.setAttribute("disposalMethod", "none");
   extension.setAttribute("userInputFlag", "FALSE");
   extension.setAttribute("transparentColorFlag", "FALSE");
   extension.setAttribute("delayTime", "1");
   extension.setAttribute("transparentColorIndex", "255");
   node.appendChild(extension);
   IIOMetadataNode appExtensions = new IIOMetadataNode("ApplicationExtensions");
   IIOMetadataNode appExtension = new IIOMetadataNode("ApplicationExtension");
   appExtension.setAttribute("applicationID", "NETSCAPE");
   appExtension.setAttribute("authenticationCode", "2.0");
   byte[] b = "\u0021\u00ff\u000bNETSCAPE2.0\u0003\u0001\u0000\u0000\u0000".getBytes();
   appExtension.setUserObject(b);
   appExtensions.appendChild(appExtension);
   node.appendChild(appExtensions);

   IIOMetadata metadata;
   metadata = writer.getDefaultImageMetadata(new ImageTypeSpecifier(image), null);
   metadata.mergeTree("javax_imageio_gif_image_1.0", node);

   IIOImage t = new IIOImage(image, null, metadata);
   writer.writeToSequence(t, null);
  }
  catch (IIOInvalidTreeException e) {
   throw new IOException(e);
  }
 }

 public void close() throws IOException {
  writer.endWriteSequence();
 }
}
```