CS 235 - Homework 8

Deadline:
Due by 11:59 pm on Wednesday, October 28, 2015.

How to submit:
Submit your files using ~st10/235submit on nrs-projects, with a homework number of 8, by the
deadline shown above.

Purpose
To provide practice with creating and controlling Java threads, and to practice with simple graphics
(painting/drawing) in Swing.

Important notes:
• Follow the Java coding standards mentioned in previous homework handouts and discussed in class.
• Be sure to submit copies of all .jpg, .gif, or .png files used in your homework problems.
• Note that Java applications with graphical user interfaces are expected to be structured in the way
demonstrated in the in-class examples (as in ButtonTest.java)
  – ...but using appropriate additional helper methods in the JPanel subclass is fine, as is using
    additional classes if you see the opportunity. IF you use additional public classes, be sure to submit
    those as well!
• It is possible that some of your programs may be posted to the course Moodle site.

Problem 1
To provide some beginning practice with threads, modify the Week 9 Lecture example
ThreadPlay1.java into ThreadPlayMore.java, meeting the following requirements:
• add another @author line to its opening Javadoc comment, indicating that you have adapted this
• change the @version line to the date that you last modified this
• ADD something that has VISIBLE output, including the executing thread's name, to class
  MyFirstRunnable's run method.
  – (this can be as simple as printing your name along with the executing thread's name, or it can be
    more intricate -- your choice!)
• create at least one more Thread instance, with a name including some version of the number three,
  and start it running appropriately/properly, and stop it appropriately/properly before the end of the
  main method
  – this thread/these threads can be created from the class MyFirstRunnable, OR you can create an
additional class/classes implementing the Runnable interface and create your thread/threads from that/those

  (if you make your own class(es) implementing the Runnable interface, make sure each does something "visible" to show that it is running)

• You may change what the main method prints out, and how much and when it sleeps, if you like, BUT make sure one can still tell from the program output that three or more threads are running.

• Make sure you modify comments as necessary so that "fit" YOUR version of ThreadPlayMore.java.

Submit your resulting ThreadPlayMore.java.

Problem 2

Note that the class Graphics is in the package java.awt, as is its more modern subclass, Graphics2D. You can read about the many methods they include in the Java 8 API (a copy of which is linked from the public course web site, under "References").

Note that you can draw/paint an image stored in a .jpg, .png, or .gif file as well! Say I had such a file, cow.jpg, stored in the same directory as my .java (and resulting .class) files -- I could create an ImageIcon object containing the image with:

ImageIcon myImageIcon = new ImageIcon("cow.jpg");

...and I could then paint that image onto a panel using (in method paintComponent):

myImageIcon.paintIcon(this, g, desiredX, desiredY);

(and the image would be painted with its top-left corner at (desiredX, desiredY)).

To provide some beginning practice with painting/drawing in Swing, modify the Week 9 Lab example DrawPlay1.java into DrawPlayMore.java, meeting the following requirements:

• add another @author line to its opening Javadoc comment, indicating that you have adapted this

• change the @version line to the date that you last modified this

• (you may change the frame's width and height as you would like, making sure everything you draw/paint below is still visible!)

• within DrawPlayMorePanel's method paintComponent, remove everything AFTER the first statement (everything AFTER the call to super.paintComponent(g)), and replace it as follows:

  – using font and color of your choice, visibly draw/paint a string including your name on the panel

  – using a color/colors other than the default color, visibly draw/paint at least one shape that is NOT a rectangle to the panel, using at least one method from either class Graphics or class Graphics2D

  – obtain at least one reasonably small image of type .jpg or .png or .gif, and save it/them in the same directory as DrawPlayMore.java (and its resulting .class files). Then, visibly paint
it/them on the panel, in location(s) of your choice.
  – you may paint/draw any additional items that you would like, as long as the above are also visibly
    included.

Submit your resulting DrawPlayMore.java, AND ALSO all .jpg or .png or .gif files that it
uses!

Problem 3

Now, for a little more thread practice that also involves drawing/painting: consider the Week 9 Lab
example BlockThread1.java. Right now, the red filled rectangle starts in about the middle-ish of the
panel, and when the Begin button is pushed, it moves right until it goes past the right edge, then it starts
again from the left edge, repeating until the End button is pushed.

Modify the Week 9 Lab example BlockThread1.java into MoveIt.java, meeting the following
requirements:

• add another @author line to its opening Javadoc comment, indicating that you have adapted this
• change the @version line to the date that you last modified this
• (you may change the frame's width and height as you would like, making sure everything you
draw/paint below is still visible!)
• DECIDE on a shape OTHER than a red rectangle to move. (It can be an image from a file, or a shape
  other than a red rectangle, or some composite image you build from lines and shapes and strings, etc.!) 
• DECIDE on a movement OTHER THAN horizontal left-to-right (it can be right-to-left, top-to-bottom,
  bottom-to-top, diagonal, sinusoidal, wobbly, careening off the edges, going in an oval, anything BUT
  horizontal left-to-right).
• ...and modify this so that THAT shape starts THAT movement when the Begin button is pushed, and
  ends it when the End button is pushed.

Submit your resulting MoveIt.java, AND ALSO any .jpg or .png or .gif files that it uses (if it
uses any).

Problem 4

For a little more practice with threads and painting/drawing in Swing...

Create an application AnimateTwo.java that animates TWO "things".

• These two "things" should be properly painted onto a sub-panel, and they should be visibly different,
  but otherwise you can paint strings, lines, rectangles, shapes, images, some combination of these, or you
  can go further afield if you would like.
• YOU CHOOSE if both will be controlled by a single begin and a single end button (both controlled by
  a single thread),
  OR if each is controlled by its OWN begin and end button (4 buttons total) (each controlled by its OWN
  individual thread).
  – tastefully place these two or four buttons on top, bottom, left, or right, your choice;
– You are required to create your thread(s) using a class/classes implementing the `Runnable` interface.

– You may NOT use deprecated methods to control your thread(s) – you are required to use the `interrupt()` method appropriately to cause each thread's `run()` method to terminate gracefully.

– NEITHER "thing" can move horizontally left-to-right across the screen -- (they can move up or down, right-to-left instead of left-to-right, bouncing, diagonally, sinusoidally, etc. -- how they move is your choice, as long as neither is moving left-to-right horizontally.)

– As long as neither is moving horizontally left-to-right across the screen, they can move similarly or very differently.

• IF you choose to control both "things" with a single pair of begin-end buttons, then clicking the begin button should start both in their movements, and clicking end should stop both of their movements.

• IF you choose to control each "thing" with its own pair of begin-end buttons, then clicking its begin button starts JUST that one "thing" in its movement, and clicking its end button ends JUST that one "thing"'s movement.

– Note, then, that it should be possible for both items to be moving at the same time, or for either one to be moving while the other is still, or for both to be still, depending on whose buttons have been pushed.

– As in `BlockThread1.java`, pushing the Start button for a thread while that thread is running shouldn't have any effect; and, pushing the Stop button for a thread while that thread is not running shouldn't have any effect, either. But if there currently isn't a running thread for that item, then pushing its Start button should start it moving on the screen, and if there is currently a running thread for that item, then pushing its Stop button should stop its thread (and its moving on the screen).)

Submit your resulting `AnimateTwo.java`, AND ALSO any `.jpg` or `.png` or `.gif` files that it uses (if it uses any).