Iterators and Java2D Graphics

3-11-2010
Let's look at solutions to the interclass problem.

Making a linked list in C.
- Don't forget C.
- Pointers in C and Java.

Do you have any questions about the assignment?

Do you have any questions about the reading?
■ Direct access on linked lists is very inefficient. How then should we walk through a list with outside code? Remember that the outside code doesn't have access to the nodes so it can't use the style of loop we have been doing internally.

■ The concept of an Iterator is something that abstracts the process of walking through all of the elements in a container. Iterators can not only be efficient, they also make code more flexible because they don't depend on the implementation details of the containers.
An iterator basically needs to encapsulate the information and functionality we would put into a standard method of going through a container.
With this in mind, what do we need to put in an iterator for an array based list?
What would we put in an iterator for a linked list?
You can do lots of things with the standard GUI elements in Swing. We've been able to set up quite a bit of a GUI using that. However, no GUI can predict everything that you will want to do and we want to be able to add custom drawing to our application.

For this we will rely on the Java2D library. Java2D was added about the same time Swing was and it is fundamentally based on the Graphics2D class. There is also a Graphics class that provides more basic custom graphics capabilities. Graphics2D inherits from Graphics so it can do all the same things and more.
There are three steps to making a component class that we can do custom drawing to.

- Make a new class and have it inherit from JComponent or a subtype of it. We'll use JPanel.
- Override the paintComponent method in your class.
- Draw with the Graphics object that was passed into the paintComponent method.

Let's look a bit at the Graphics2D class to see what some of the possibilities might be for what we can draw.

Now we can do these steps in our program to make a central panel we can draw to.
There are several things that we can set on the Graphics2D object that are used when we draw things. Here are some:

- **Paint** – could be a color, but there are also gradients and textures
- **Stroke** – determines how lines are drawn
- **Font** – how you want text to appear
- **Transform** – AffineTransform allows translate, rotate, scale, or shear
- **Composite** – how colors combine when you draw over old stuff
- **Clip** – where your drawings will appear
- **Render hints** – other things like antialiasing
Of course, Graphics2D objects aren't limited to just drawing on components.
The Image class (and it's subtype BufferedImage) will let you get Graphics objects that you can draw to and what you draw will be on the image.
We'll typically do this even if we are drawing to a component to implement buffering which reduces flicker.
Let's play with our panel some to experiment with the drawing options.
What do you think we should try adding to our drawing program given that we now know how to draw?

Interclass Problem – Write a simple drawing program similar to Paint. Use buttons for selecting at least rectangles, ellipses, and lines. Use the mouse to draw things. Have color options with JColorChooser.