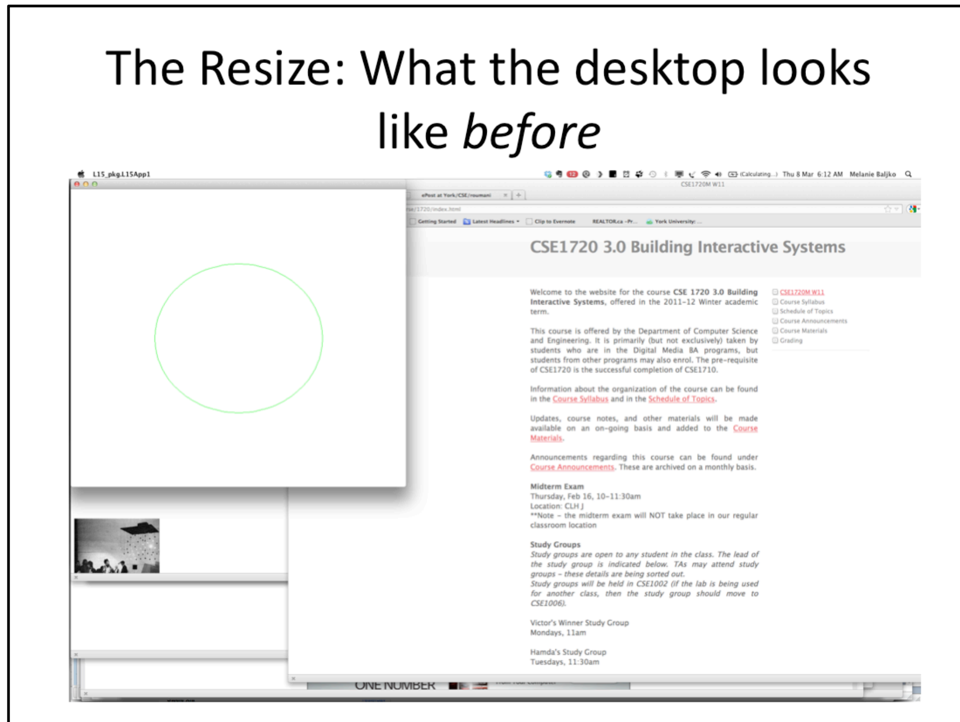
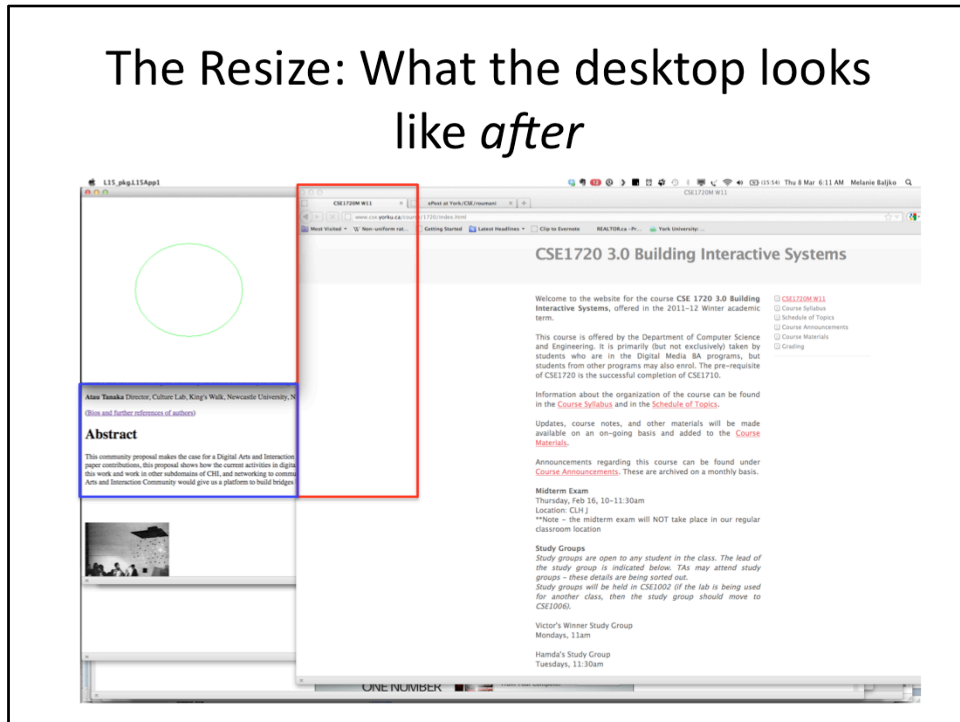
Lecture 16 Notes
Write-up of class discussion re:
worksheet question #1

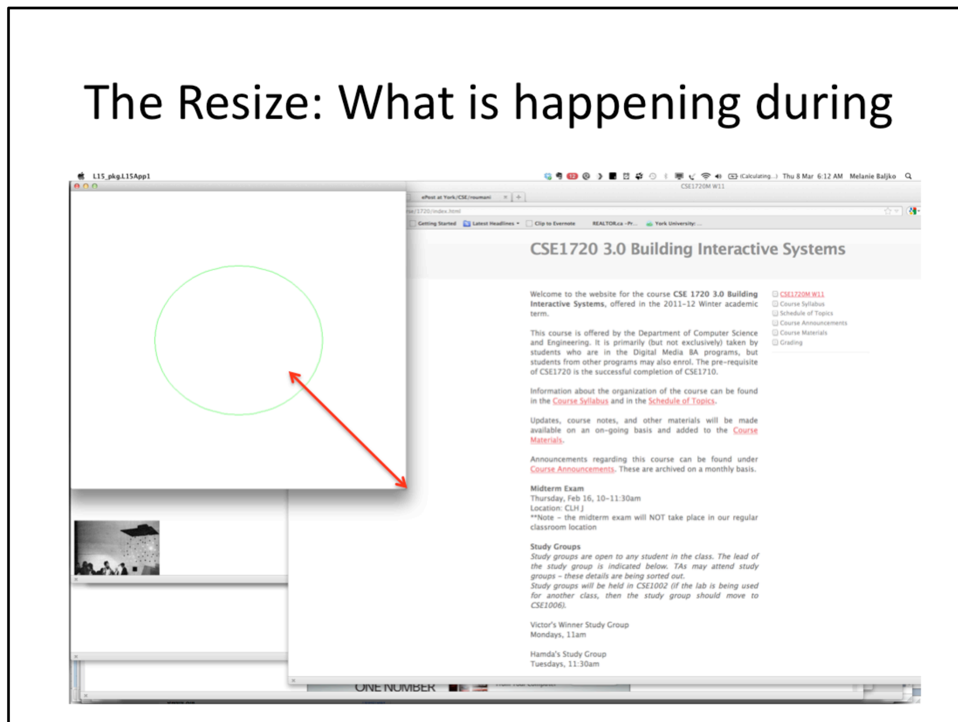
Consider the following scenario:
The app L15App1 is launched. The user resizes the window. Describe the sequence of actions that happen next…



Recall, the app that is running is the VM. The VM in turn is running the main method of the class (L15App1) that was given to it as a run-time parameter.
That main method, as part of its operations, used the services of JFrame. A JFrame object, when constructed and setVisible, invokes the services of the Window Manager (WM).
The WM allocates a window on the desktop for use by L15App1.
This L15App1 app window is *occluding* two firefox windows…



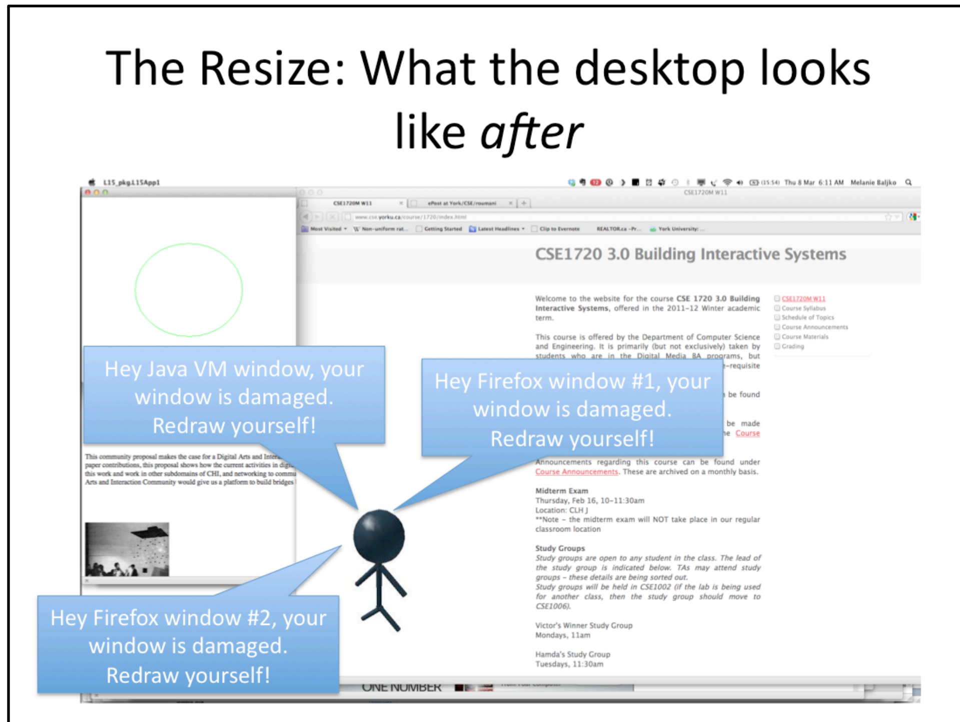
The window with the L15App1 app has been reduced – the rectangles indicate the regions that had been previously occluded. Each region is contained within a window.



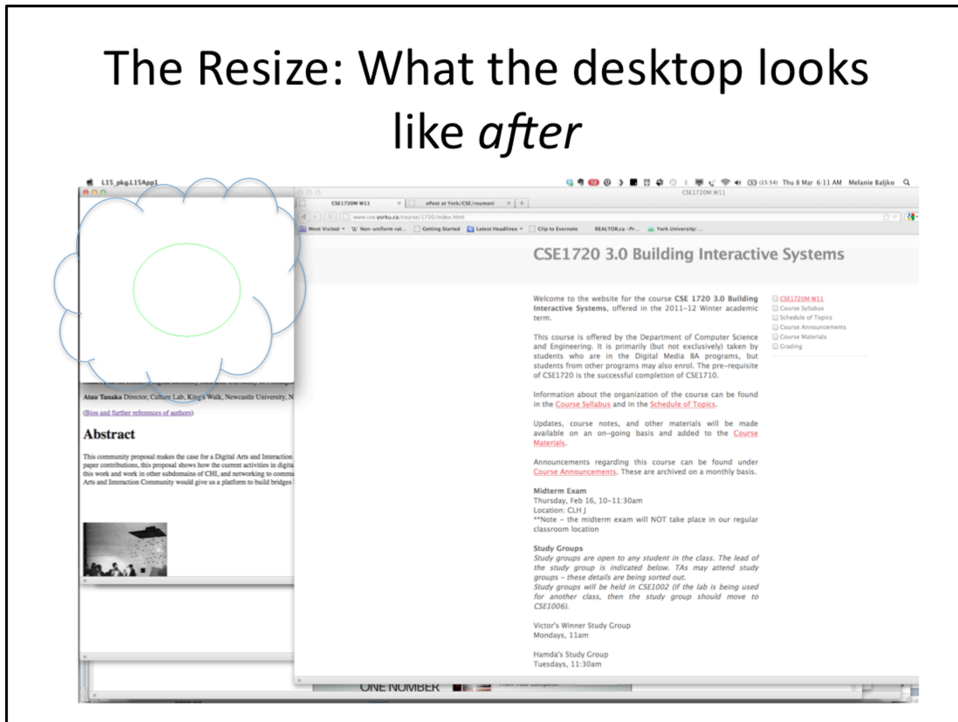
As the user resizes the L15App1 window, there is a “mark-as-damaged” process running that is checking to see which windows become *damaged* (i.e., they need to be redrawn).

As the window is resized, the L15App1 window is marked as being *damaged*. As well, the other Firefox windows become un-occluded, and they are also marked as *damaged*.

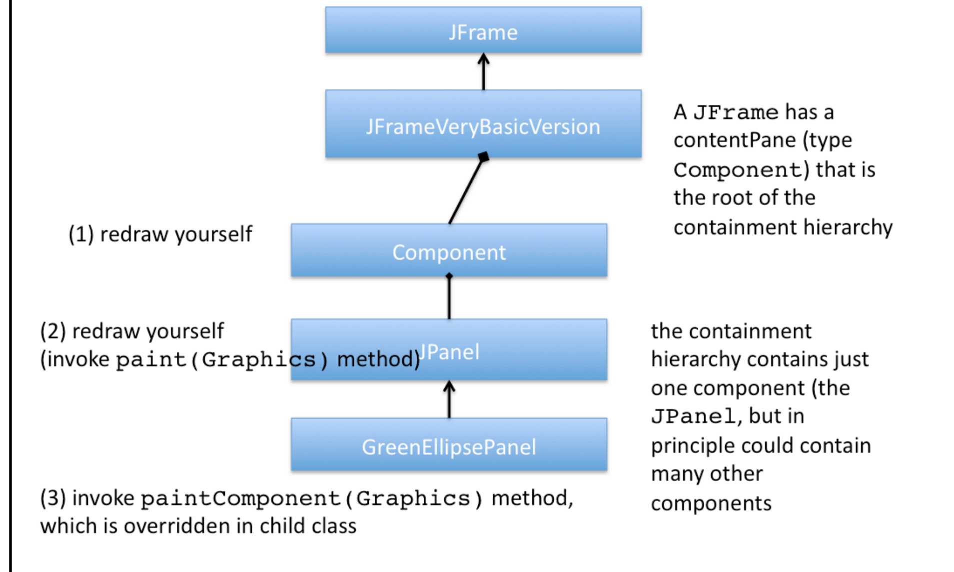
The WM has another “redraw-damaged-windows” process that is running, checking for damaged windows. (These processes are interleaved and coordinated)



When the “redraw-damaged-windows” WM process finds that there are damaged windows, it delegates to each of the windows: REDRAW yourself!!
More specifically, the “redraw-damaged-windows” WM process tells the ROOT of each window’s containment hierarchy: REDRAW yourself!!!

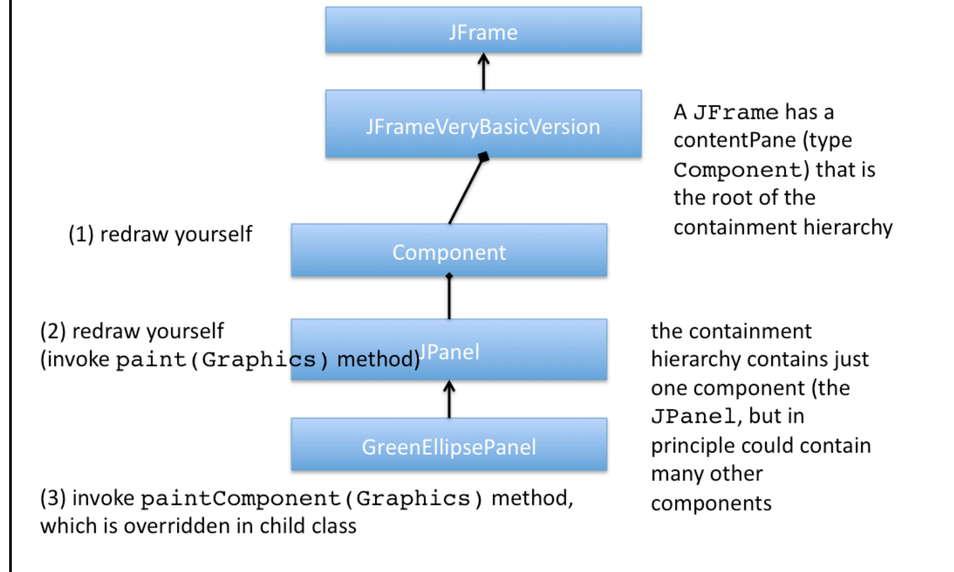


Each of the windows that has been asked to be redrawn itself does so. We will consider the case of the L15App1 here (and set aside the case of the Firefox windows for now)



The content pane is told: (1) "You need to redraw yourself. Here is the Graphics2D object that you can use to specify how you want to be drawn" (remember, the WM will use the graphics pipeline to actually render the graphics primitives).
The content pane delegates to each of its child components. So it says to the JPanel object (thePanel) (the only child component in this case) – (2) "You need to redraw yourself. Here is the Graphics2D object that you can use to specify how you want to be drawn". This happens specifically by the JPanel's paint(Graphics) method being invoked.

The JPanel object's paint(Graphics) method (which lives in the class definition for JPanel) in turn, (3) calls the object's paintComponent(Graphics) method.



The paintComponent(Graphics) method is overridden in the child class GreenEllipsePanel. The body of the paintComponent(Graphics) method takes the Graphics object and casts it to Graphics2D

- it is a historical quirk that the API specifies the method parameter as Graphics; we exploit background knowledge that the runtime type will always be Graphics2D, which is the next-generation version of Graphics); We do this because the API for Graphics2D is much richer than the API for Graphics; we can draw and accomplish much more

-the body of the painComponent(Graphics) method next specifies *how* the component should be drawn. When it is done (the body of the method terminates), the now-modified Graphics2D object is taken and, through a chain of actions, is caused to be rendered on the screen. This is how the L15App1 window, when resized, gets re-rendered.