Java Chapter 14 Part 1

* Applet Basics, Drawing Shapes
* CIS 255 • Shelby-Hoover Campus

Applets vs. Applications

* Applications are standalone programs that run on the local machine
* Applets are usually stored on a web server (embedded in a Web page)
* Java applets allow more application-like functionality on a Web page than HTML alone
* Limitations on Applets:
  + No interaction with local files
  + No execution of local programs
  + No access to OS procedures on the local machine
  + No access to system information, user identification
  + No network connections beyond the host server
  + Applet-launched windows contain warnings indicating the source of the window is an applet, not a local program

HTML

* Hypertext Markup Language (HTML) is a markup language (not a programming language) used to indicate the contents of a web page
* The textbook covers some of the basic tags:
  + The entire document is placed between <html> and </html>
  + The document contains two sections, head and body, with opening and closing tags for each
  + Within the block for head, you can specify the text to display in the title bar in a title block
  + Within the block for body, you can insert normal paragraphs using p blocks and headings of different sizes using h1 (largest) through h6 (smallest) blocks

Writing an Applet Class

* Applet classes extend JApplet instead of JFrame
* Classes that inherit from JApplet do not require a constructor: they contain a public void method named init with the necessary setup operations (excluding setTitle, setSize, setDefaultCloseOperation, pack, and setVisible)
* You do not need to instantiate the class (there is no main method); when you use a tag to embed the class within the page, it automatically instantiates the class
* Example: SimpleApplet.java (Code Listing 14-6)

Embedding an Applet

* When embedding an applet, the body of the page contains a tag referencing the class file (ending in .class) for the applet
* The old, deprecated tag for embedding an applet (works in most browsers) is applet:  
    
  <applet code="ClassName.class" width="width" height="height">Message for browsers without Java</applet>
* The new, recommended tag (still doesn’t work in some browsers) is object:  
    
  <object codetype="application/java" classid="java:ClassName.class" width="width" height="height">Message for browsers without Java</object>
* An example using applet: SimpleApplet.html (Code Listing 14-7)

Running an Applet

* To run an applet embedded within a Web page without launching a browser, use the command-line tool appletviewer with the Web page’s file name:  
    
  appletviewer FileName.html
* TextPad also has a Java tool named “Run Java Applet” that does not require the creation of a Web page (it automatically creates a temporary page)

Applets and Events

* An applet handles events in the same manner an application
  + An applet class can contain a private listener class to respond to an event
  + A new event listener object can be registered to a GUI component within an applet
* The applet can contain several panels, and these panels can be built using private helper methods

Applet Event Example

* TempConverter.java (Code Listing 14-8) defines an applet that allows the user to convert a temperature from Fahrenheit to Celsius
  + Three panels in a GridLayout with three rows, one column
  + Each temperature panel is aligned to the right using FlowLayout
  + Two text fields: one of them is set to read-only using the method setEditable(false) (it displays the results)
  + The applet labels each text field within the helper method for the panel (local variables instead of class members)
  + A private inner listener class implements the ActionListener interface and responds to the user’s click on the button in the third panel to calculate and display the converted temperature in the read-only text field
* The applet is embedded in the HTML document TempConverter.html (Code Listing 14-9)

Applets without Swing

* An applet that targets computers that may not have a Swing-compatible Java plug-in (rare) may use AWT classes instead
* For the most part, AWT class names are the same as Swing class names without the “J”:
  + Applet (JApplet)
  + Frame (JFrame)
  + Panel (JPanel)
  + Button (JButton)
  + Label (JLabel)
  + TextField (JTextField)
  + Checkbox (JCheckBox)
* An additional import statement may be required: import java.applet.Applet;

Shapes

* Standard GUI components (buttons, text fields, labels) are placed in an applet via the init method (or a helper method)
* Shapes are drawn in an applet using a separate method, paint
  + Inherited from JApplet, JFrame, Applet, or Frame
  + Must be overridden (285)
  + The parameter is an object of the Graphics class
  + The method calls the inherited version of the method via the super call (285) before actually invoking any of the graphics methods on the Graphics object

Executing the paint Method

* A program doesn’t call paint directly
* Java automatically invokes the method when the window is displayed for the first time
* The method is called again whenever a window that overlaps the window is moved
* If a program must trigger the paint method at another time (e.g., when an event occurs), it calls the void method repaint()
  + repaint() clears the area of any graphics drawn (refilling the area with its background color)
  + It then triggers another call to paint

Color and Font

* Before calling one of the methods to draw a shape (or a free-flowing string of text), set the color and / or font of the Graphics object
* The method setColor(Color\_arg) causes the Graphics object to use that color for the next methods invoked on that object
* The method getColor() returns the current color being used by the Graphics object
* The method setFont(Font\_arg) sets up the Graphics object to use that font for the next call to draw a string of text
  + The Font argument should be a new Font object
  + The first argument to the Font constructor is the font name; the second argument is a constant (or sum of constants) to set the style (Font.PLAIN, Font.BOLD, Font.ITALIC, or Font.BOLD + Font.ITALIC); and the third constant is the font size in points
  + Example: new Font("Serif", Font.BOLD, 18);

Lines and Hollow Shapes

* Several methods use the color of the Graphics object as the line color of a shape
* The (x, y) coordinates of a shape are the number of pixels from the upper left-hand corner (higher values of x go to the right, whereas higher values of y go down)
* drawLine(x1, y1, x2, y2)   
  draws a line from the point (x1, y1) to the point (x2, y2)
* drawRect(x, y, width, height)   
  draws a hollow rectangle starting at (x, y) and with the specified width and height in pixels
* drawOval(x, y, width, height)   
  draws a hollow oval circumscribed by the rectangle staring at (x, y) and with the specified width and height in pixels (the vertices of the oval are at the midpoints of the edges of the rectangle)
* drawArc(x, y, width, height, startAngle, arcAngle)   
  draws a hollow arc that is part of the oval circumscribed by the rectangle starting at (x, y) and with the specified width and height; the arc starts at the startAngle and extends for arcAngle degrees (0 degrees is at 3 o’clock; the degrees increase counterclockwise)

Solid Shapes

* Each draw method for a hollow shape (not a line) has a corresponding fill method that uses the color of the Graphics object as the fill color of a shape
* fillRect(x, y, width, height)   
  draws a solid rectangle starting at (x, y) and with the specified width and height in pixels
* fillOval(x, y, width, height)   
  draws a solid oval circumscribed by the rectangle staring at (x, y) and with the specified width and height in pixels
* fillArc(x, y, width, height, startAngle, arcAngle)   
  draws a solid arc (wedge / slice) that is part of the oval circumscribed by the rectangle starting at (x, y) and with the specified width and height; the arc starts at the startAngle and extends for arcAngle degrees

Polygons and Text

* drawPolygon(xPoints, yPoints, numPoints) accepts two arrays holding the x and y coordinates of each point and the number of points stored in the arrays and draws a hollow polygon composed of the lines between the points (it automatically connects the last point to the first point)
* fillPolygon(xPoints, yPoints, numPoints) draws a solid polygon in a fashion similar to drawPolygon
* drawstring(str, x, y) draws the String argument str in a position with the bottom left corner of the text at (x, y)

Shape Examples

* LineDemo.java (Code Listing 14-11) draws a red line and a blue line forming an X in an applet with a white background
* RectangleDemo.java (Code Listing 14-12) draws a hollow rectangle with a black line and a solid rectangle with a red fill
* OvalDemo.java (Code Listing 14-13) draws a hollow oval with a black line and a solid oval with a green fill
* ArcDemo.java (Code Listing 14-14) draws two hollow arcs with black and green lines and two solid arcs with red and blue fills
* PolygonDemo.java (Code Listing 14-15) draws a solid red octagon based on the eight vertices in the two arrays (parallel arrays: the x and y coordinate for a given vertex are at the same subscript in the two arrays)