Graphics Programming

Introduction

- GOAL: Build the Indexer Client
- Event-driven vs. Sequential programs
- Terminology
  - Top-level windows are called “frame windows”
    - Implemented by JFrame class in Java
    - Create subclass of JFrame to implement custom frame window for a particular application
  - Elements inside a frame window are called “components”
    - Buttons, Text boxes, Menus, Tables, etc.
    - Implemented by the JComponent class in Java, and its various subclasses (JButton, JTextField, JMenu, JTable, etc.)

EventQueue.invokeLater

- In Swing, all user interface operations must occur on the "UI thread"
  - All components should be created on the UI thread
  - All method calls on UI components should happen on the UI thread
- EventQueue.invokeLater runs the specified code on the UI thread
- The main method for Swing programs should call EventQueue.invokeLater to create the UI
- The main thread exits immediately after calling EventQueue.invokeLater, but the UI thread keeps the program running
- EXAMPLE: Empty Frame

JFrame

- Use JFrame class to create top-level windows
- setTitle method sets the window's title
- setDefaultCloseOperation method specifies what should happen when the user clicks the window's close icon
- setLocation method sets the window's location on the desktop
- setVisible method shows or hides the window
- setSize method sets the window's size
- EXAMPLE: Empty Frame

JFrame

- add method adds a new subcomponent to the window
  - Creates a parent/child relationship between the frame and its subcomponents (i.e., makes a tree)
- pack method sets the window's size according to the preferred size and layout of the window's subcomponents
- EXAMPLE: Simple Frame

JComponent

- User interface components are subclasses of JComponent that provide custom drawing and event handling functionality
- Built-in components are implemented in classes such as JButton, JTextField, JMenu, JTable, etc.

- getSize and setSize methods get and set component's width and height
  - getWidth and getHeight methods return component's width and height individually
- setBackground method is used to set the component's background color
- setPreferredSize, setMinimumSize, setMaximumSize methods are used to express the components preferred, min, and max sizes
- paintComponent method draws the contents of the component
- Graphics2D class is used to perform drawing operations in a component
- EXAMPLE: Drawing

Drawing

- Color
  - Red, Green, Blue, Alpha (transparency) components
  - new Color(210, 180, 140, 192)
  - Graphics.setColor method sets the current drawing color

- Drawing origin is the component's top-left corner. X values increase as you move right. Y values increase as you move down.

- Representing points
  - Point2D superclass
    - Point2D.Float and Point2D.Double subclasses (nested inside Point2D)
  - Point2D.Double pt = new Point2D.Double(x, y)

Drawing Rectangles

- Rectangles
  - Graphics.drawRect(x, y, width, height)
  - Graphics.fillRect(x, y, width, height)
  - OR
  - Rectangle2D superclass
    - Rectangle2D.Float and Rectangle2D.Double subclasses (nested inside Rectangle2D)
  - Rectangle2D rect = new Rectangle2D.Double(x, y, w, h)
  - Graphics2D.draw(rect)
  - Graphics2D.fill(rect)
  - EXAMPLE: Drawing

Drawing Ellipses

- Ellipses
  - Graphics.drawOval(x, y, width, height)
  - Graphics.fillOval(x, y, width, height)
  - OR
  - Ellipse2D superclass
    - Ellipse2D.Float and Ellipse2D.Double subclasses (nested inside Ellipse2D)
  - Ellipse2D ellipse = new Ellipse2D.Double(x, y, w, h)
  - Graphics2D.draw(ellipse)
  - Graphics2D.fill(ellipse)

Drawing Lines

- Lines
  - Graphics.drawLine(x, y, width, height)
  - OR
  - Line2D superclass
    - Line2D.Float and Line2D.Double subclasses (nested inside Line2D)
  - Line2D line = new Line2D.Double(x1, y1, x2, y2)
  - Graphics2D.setStroke method sets the line thickness and style
    - g2d.setStroke(new BasicStroke(5)); // line 5 pixels wide
  - Graphics2D.draw(line)
  - EXAMPLE: Drawing

Drawing Text

- Font class represents fonts
- Font font = new Font(name, style, size)
  - Font font = new Font(“SansSerif”, Font.PLAIN, 72);
- Graphics.setFont method sets the current font
  - g2d.setFont(font);
- Graphics2D.drawString(string, x, y)
  - (x, y) is location of text’s baseline
  - g2d.drawString(“Hi There”, 100, 200);
- EXAMPLE: Drawing

Drawing Text

- Calculating text metrics (width, height, etc.)
  - FontRenderContext context = g2d.getFontRenderContext();
  - Rectangle2D bounds = font.getStringBounds(message, context);
  - double stringWidth = bounds.getWidth();
  - double stringHeight = bounds.getHeight();
  - double ascent = -bounds.getY();
- If you need descent and leading
  - LineMetrics metrics = font.getLineMetrics(message, context);
  - float descent = metrics.getDescent();
  - float leading = metrics.getLeading();

Drawing Images

- The BufferedImage class can be used to store and manipulate images in memory. You can:
  - Load an existing image into a BufferedImage
  - Modify an image by changing the pixel values in a BufferedImage
  - Create a new image by creating an empty BufferedImage and modifying its pixel values
  - Save a BufferedImage to a file

Drawing Images

- The ImageIO class can be used to load images from disk or the web, and to save images to disk
- Load image from disk:
  - String filename = "...";
  - Image image = ImageIO.read(new File(filename));
  - EXAMPLE: Drawing
- Load image from URL:
  - String urlname = "...";
  - Image image = ImageIO.read(new URL(urlname));

Drawing Images

- Graphics.drawImage(image, destX1, destY1, destX2, destY2, srcX1, srcY1, srcX2, srcY2)
  - "dest" is the destination rectangle where the image should be drawn in the component
  - "src" is the source rectangle in the image to be drawn
    - Can be only part of the image
  - "dest" and "src" do not have to be the same size. The drawImage method will scale the source image to fit in the destination rectangle.
    - This is one way to scale an image
  - EXAMPLE: Drawing