**Section 22.2 JSlider**

• JSliders (p. [912](http://proquest.safaribooksonline.com/9780133813036/ch22_html#page_912)) enable you to select from a range of integer values. They can display major and minor tick marks, and labels for the tick marks (p. [912](http://proquest.safaribooksonline.com/9780133813036/ch22_html#page_912)). They also support snap-to ticks (p. [912](http://proquest.safaribooksonline.com/9780133813036/ch22_html#page_912))—positioning the thumb (p. [912](http://proquest.safaribooksonline.com/9780133813036/ch22_html#page_912)) between two tick marks snaps it to the closest tick mark.

• JSliders have either horizontal or vertical orientation. For a horizontal JSlider, the minimum value is at the extreme left and the maximum value at the extreme right. For a vertical JSlider, the minimum value is at the extreme bottom and the maximum value at the extreme top. The position of the thumb indicates the current value of the JSlider. Method getValue (p. [916](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec2_html#page_916)) of class JSlider returns the current thumb position.

• JSlider method setMajorTickSpacing () sets the spacing for tick marks on a JSlider. MethodsetPaintTicks (p. [916](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec2_html#page_916)) with a true argument indicates that the tick marks should be displayed.

• JSliders generate ChangeEvents when the user interacts with a JSlider. A ChangeListener (p. [916](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec2_html#page_916)) declares method stateChanged (p. [916](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec2_html#page_916)) that can respond to ChangeEvents.

#### Section 22.3 Understanding Windows in Java

• A window’s (p. [916](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec2_html#page_916)) events can be handled by a WindowListener (p. [917](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec2_html#page_917)), which provides seven window-event-handling methods—windowActivated, windowClosed, windowClosing, windowDeactivated,windowDeiconified, windowIconified and windowOpened.

#### Section 22.4 Using Menus with Frames

• Menus neatly organize commands in a GUI. In Swing GUIs, menus can be attached only to objects of classes with method setJMenuBar (p. [917](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec2_html#page_917)).

• A JMenuBar (p. [917](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec2_html#page_917)) is a container for menus. A JMenuItem appears in a menu. A JMenu (p. [917](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec2_html#page_917)) contains menu items and can be added to a JMenuBar or to other JMenus as submenus.

• When a menu is clicked, it expands to show its list of menu items.

• When a JCheckBoxMenuItem (p. [918](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec5_html#page_918)) is selected, a check appears to the left of the menu item. When theJCheckBoxMenuItem is selected again, the check is removed.

• In a ButtonGroup, only one JRadioButtonMenuItem (p. [918](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec5_html#page_918)) can be selected at a time.

• AbstractButton method setMnemonic (p. [923](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec7_html#page_923)) specifies the mnemonic (p. [918](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec5_html#page_918)) for a button. Mnemonic characters are normally displayed with an underline.

• A modal dialog box (p. [923](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec7_html#page_923)) does not allow access to any other window in the application until the dialog is dismissed. The dialogs displayed with class JOptionPane are modal dialogs. Class JDialog (p. [923](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec7_html#page_923)) can be used to create your own modal or nonmodal dialogs.

#### Section 22.5 JPopupMenu

• Context-sensitive pop-up menus (p. [925](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec9_html#page_925)) are created with class JPopupMenu. The pop-up trigger event occurs normally when the user presses and releases the right mouse button. MouseEvent methodisPopupTrigger (p. [928](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec5_html#page_928)) returns true if the pop-up trigger event occurred.

• JPopupMenu method show (p. [928](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec5_html#page_928)) displays a JPopupMenu. The first argument specifies the origin component, which helps determine where the JPopupMenu will appear. The last two arguments are the coordinates from the origin component’s upper-left corner, at which the JPopupMenu appears.

#### Section 22.6 Pluggable Look-and-Feel

• Class UIManager.LookAndFeelInfo (p. [929](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec6_html#page_929)) maintains information about a look-and-feel.

• UIManager (p. [929](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec6_html#page_929)) static method getInstalledLookAndFeels (p. [929](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec6_html#page_929)) returns an array ofUIManager.LookAndFeelInfo objects that describe the available look-and-feels.

• UIManager static method setLookAndFeel (p. [932](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec6_html#page_932)) changes the look-and-feel. SwingUtilities (p. [932](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec6_html#page_932))static method updateComponentTreeUI (p. [932](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec6_html#page_932)) changes the look-and-feel of every component attached to its Component argument to the new look-and-feel.

#### Section 22.7 JDesktopPane and JInternalFrame

• Many of today’s applications use a multiple-document interface (MDI; p. [933](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec7_html#page_933)) to manage several open documents that are being processed in parallel. Swing’s JDesktopPane (p. [933](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec7_html#page_933)) and JInternalFrame (p.[933](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec7_html#page_933)) classes provide support for creating multiple-document interfaces.

#### Section 22.8 JTabbedPane

• A JTabbedPane (p. [936](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec7_html#page_936)) arranges GUI components into layers, of which only one is visible at a time. Users access each layer by clicking its tab.

#### Section 22.9 BoxLayout Layout Manager

• BoxLayout )p. [939](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec9_html#page_939)) arranges GUI components left-to-right or top-to-bottom in a container.

• Class Box represents a container with BoxLayout as its default layout manager and provides staticmethods to create a Box with a horizontal or vertical BoxLayout.

#### Section 22.10 GridBagLayout Layout Manager

• GridBagLayout (p. [942](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec12_html#page_942)) is similar to GridLayout, but each component size can vary.

• A GridBagConstraints object (p. [943](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec10_html#page_943)) specifies how a component is placed in a GridBagLayout.

### Self-Review Exercises

[**22.1**](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec14_html#ch22ans1) Fill in the blanks in each of the following statements:

a) The \_\_\_\_\_\_\_\_\_ class is used to create a menu object.

b) The \_\_\_\_\_\_\_\_\_ method of class JMenu places a separator bar in a menu.

c) JSlider events are handled by the \_\_\_\_\_\_\_\_\_ method of interface \_\_\_\_\_\_\_\_\_.

d) The GridBagConstraints instance variable \_\_\_\_\_\_\_\_\_ is set to CENTER by default.

[**22.2**](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec14_html#ch22ans2) State whether each of the following is true or false. If false, explain why.

a) When the programmer creates a JFrame, a minimum of one menu must be created and added to theJFrame.

b) The variable fill belongs to the GridBagLayout class.

c) Drawing on a GUI component is performed with respect to the (0, 0) upper-left corner coordinate of the component.

d) The default layout for a Box is BoxLayout.

[**22.3**](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec14_html#ch22ans3) Find the error(s) in each of the following and explain how to correct the error(s).

a) JMenubar b;

b) mySlider = JSlider(1000, 222, 100, 450);

c) gbc.fill = GridBagConstraints.NORTHWEST; // set fill

d)

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p954pro01a)

// override to paint on a customized Swing component  
public void paintcomponent(Graphics g)  
{  
   g.drawString("HELLO", 50, 50);  
}

e)

[**Click here to view code image**](http://proquest.safaribooksonline.com/9780133813036/app06_html#p954pro02a)

// create a JFrame and display it  
JFrame f = new JFrame("A Window");  
f.setVisible(true);

### Answers to Self-Review Exercises

[**22.1**](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec13_html#ch22que1)

a) JMenu.

b) addSeparator.

c) stateChanged, ChangeListener.

d) anchor.

[**22.2**](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec13_html#ch22que2)

a) False. A JFrame does not require any menus.

b) False. The variable fill belongs to the GridBagConstraints class.

c) True.

d) True.

[**22.3**](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec13_html#ch22que3)

a) JMenubar should be JMenuBar.

b) The first argument to the constructor should be SwingConstants.HORIZONTAL orSwingConstants.VERTICAL, and the keyword new must be used after the = operator. Also, the minimum value should be less than the maximum and the initial value should be in range.

c) The constant should be either BOTH, HORIZONTAL, VERTICAL or NONE.

d) paintcomponent should be paintComponent, and the method should call super.paintComponent(g) as its first statement.

e) The JFrame’s setSize method must also be called to establish the size of the window.

### Exercises

**22.4 (Fill-in-the-Blanks)** Fill in the blanks in each of the following statements:

a) A JMenuItem that is a JMenu is called a(n) \_\_\_\_\_\_\_\_\_.

b) Method \_\_\_\_\_\_\_\_\_ attaches a JMenuBar to a JFrame.

c) Container class \_\_\_\_\_\_\_\_\_ has a default BoxLayout.

d) A(n) \_\_\_\_\_\_\_\_\_ manages a set of child windows declared with class JInternalFrame.

**22.5 (True or False)** State whether each of the following is true or false. If false, explain why.

a) Menus require a JMenuBar object so they can be attached to a JFrame.

b) BoxLayout is the default layout manager for a JFrame.

c) JApplets can contain menus.

**22.6 (Find the Code Errors)** Find the error(s) in each of the following. Explain how to correct the error(s).

a) x.add(new JMenuItem("Submenu Color")); // create submenu

b) container.setLayout(new GridbagLayout());

**22.7****(Display a Circle and Its Attributes)** Write a program that displays a circle of random size and calculates and displays the area, radius, diameter and circumference. Use the following equations: diameter = 2 × radius, area = π × radius2, circumference = 2 × π × radius. Use the constant Math.PI for pi (π). All drawing should be done on a subclass of JPanel, and the results of the calculations should be displayed in a read-only JTextArea.

**22.8 (Using a** ***JSlider*)** Enhance the program in [Exercise 22.7](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec15_html#ch22que7) by allowing the user to alter the radius with aJSlider. The program should work for all radii in the range from 100 to 200. As the radius changes, the diameter, area and circumference should be updated and displayed. The initial radius should be 150. Use the equations from [Exercise 22.7](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec15_html#ch22que7). All drawing should be done on a subclass of JPanel, and the results of the calculations should be displayed in a read-only JTextArea.

**22.9 (Varying** ***weightx*** **and** ***weighty*)** Explore the effects of varying the weightx and weighty values of the program in [Fig. 22.21](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec26_html#ch22fig21). What happens when a slot has a nonzero weight but is not allowed to fill the whole area (i.e., the fill value is not BOTH)?

**22.10 (Synchronizing a** ***JSlider*** **and a** ***JTextField*)** Write a program that uses the paintComponent method to draw the current value of a JSlider on a subclass of JPanel. In addition, provide a JTextField where a specific value can be entered. The JTextField should display the current value of the JSlider at all times. Changing the value in the JTextField should also update the JSlider. A JLabel should be used to identify the JTextField. The JSlider methods setValue and getValue should be used. [Note: The setValue method is a public method that does not return a value and takes one integer argument, the JSlider value, which determines the position of the thumb.]

**22.11 (Creating a Color Chooser)** Declare a subclass of JPanel called MyColorChooser that provides threeJSlider objects and three JTextField objects. Each JSlider represents the values from 0 to 255 for the red, green and blue parts of a color. Use these values as the arguments to the Color constructor to create a new Color object. Display the current value of each JSlider in the corresponding JTextField. When the user changes the value of the JSlider, the JTextField should be changed accordingly. Use your new GUI component as part of an application that displays the current Color value by drawing a filled rectangle.

**22.12 (Creating a Color Chooser: Modification)** Modify the MyColorChooser class of [Exercise 22.11](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec15_html#ch22que11) to allow the user to enter an integer value into a JTextField to set the red, green or blue value. When the user presses Enter in the JTextField, the corresponding JSlider should be set to the appropriate value.

**22.13 (Creating a Color Chooser: Modification)** Modify the application in [Exercise 22.12](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec15_html#ch22que12) to draw the current color as a rectangle on an instance of a subclass of JPanel which provides its own paintComponentmethod to draw the rectangle and provides set methods to set the red, green and blue values for the current color. When any set method is invoked, the drawing panel should automatically repaint itself.

**22.14 (Drawing Application)** Modify the application in [Exercise 22.13](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec15_html#ch22que13) to allow the user to drag the mouse across the drawing panel (a subclass of JPanel) to draw a shape in the current color. Enable the user to choose what shape to draw.

**22.15 (Drawing Application Modification)** Modify the application in [Exercise 22.14](http://proquest.safaribooksonline.com/9780133813036/ch22lev1sec15_html#ch22que14) to allow the user to terminate the application by clicking the close box on the window that is displayed and by selecting Exitfrom a File menu. Use the techniques shown in [Fig. 22.5](http://proquest.safaribooksonline.com/9780133813036/ch22lev2sec6_html#ch22fig05).

**22.16 (Complete Drawing Application)** Using the techniques developed in this chapter and [Chapter 12](http://proquest.safaribooksonline.com/9780133813036/ch12_html#ch12), create a complete drawing application. The program should use the GUI components from [Chapter 12](http://proquest.safaribooksonline.com/9780133813036/ch12_html#ch12) and this chapter to enable the user to select the shape, color and fill characteristics. Each shape should be stored in an array of MyShape objects, where MyShape is the superclass in your hierarchy of shape classes. Use a JDesktopPane and JInternalFrames to allow the user to create multiple separate drawings in separate child windows. Create the user interface as a separate child window containing all the GUI components that allow the user to determine the characteristics of the shape to be drawn. The user can then click in anyJInternalFrame to draw the shape.