

# CSE 1325 OBJECT-ORIENTED PROGRAMMING

## Exam #2 --# HANDOUT #-- Fall 2022

Note: "extends" in this document includes multi-generational inheritance: Superclass members are also available to the subclass. Square brackets [ ] denote optional parameters via overloaded members.

## Superclasses

### Component

#### **Rectangle getBounds()**

Gets the bounds of this component in the form of a Rectangle object.

#### **void repaint()**

Repaints this component at the earliest opportunity.

### Container extends Component

#### **Component add(Component comp, [int index])**

Appends the specified component at the given position or end of the container.

#### **void add(Component comp, Object constraints, [int index])**

Adds the specified component with constraints at the given position or end of the container.

#### **void setLayout(LayoutManager manager)**

Sets the LayoutManager.

### JComponent extends Container

#### **void paintComponent(Graphics g)**

Calls the UI delegate's paint method, if the UI delegate is non-null.

#### **void setBackground(Color bg)**

Sets the background color of this component.

#### **void setBorder(Border border)**

Sets the border of this component.

#### **void setEnabled(boolean enabled)**

Sets whether or not this component is enabled.

#### **void setForeground(Color fg)**

Sets the foreground color of this component.

#### **void setToolTipText(String text)**

Registers the text to display in a tool tip.

#### **void setVisible(boolean aFlag)**

Makes the component visible or invisible.

## Window extends Container

`void setSize(int width, int height)`

## Primary Containers

### JFrame extends Window

**JFrame(String title)**

**Container getContentPane()**

Returns the contentPane object for this frame.

**void setDefaultCloseOperation(int operation)**

Sets the operation that will happen by default when the user initiates a "close" on this frame (DISPOSE\_ON\_CLOSE, DO\_NOTHING\_ON\_CLOSE, EXIT\_ON\_CLOSE, HIDE\_ON\_CLOSE).

**void setJMenuBar(JMenuBar menubar)**

Sets the menubar for this frame.

### JDialog extends Window

**JDialog(Frame owner, String title)**

**void setDefaultCloseOperation(int operation)**

Sets the operation that will happen by default when the user initiates a "close" on this dialog (DISPOSE\_ON\_CLOSE, DO\_NOTHING\_ON\_CLOSE, EXIT\_ON\_CLOSE, HIDE\_ON\_CLOSE).

### JPanel extends JComponent

**JPanel()**

## Layout Managers

### BorderLayout

**BorderLayout([int hgap, int vgap])**

Constraints: PAGE\_START, PAGE\_END, LINE\_START, LINE\_END, CENTER

### BoxLayout

**BoxLayout(Container target, int axis)**

Axis: LINE\_AXIS, PAGE\_AXIS, X\_AXIS, Y\_AXIS

## FlowLayout

**FlowLayout([int align, [int hgap, int vgap]])**

Align: CENTER, LEFT, RIGHT, LEADNG, TRAILING

## GridLayout

**GridLayout([int rows, int cols, [int hgap, int vgap]])**

## GridBagLayout

**GridBagLayout()**

**void setConstraints(Component comp, GridBagConstraints constraints)**

Sets the constraints for the specified component in this layout.

## GridBagConstraints

**GridBagConstraints()**

**int anchor**

Determines where, within the display area, to place the component: PAGE\_START, PAGE\_END, LINE\_START, LINE\_END, FIRST\_LINE\_START, FIRST\_LINE\_END, LAST\_LINE\_START, LAST\_LINE\_END

**int fill**

Determines how to resize the component: NONE, HORIZONTAL, VERTICAL, BOTH

**int gridheight, gridwidth**

Specifies the number of cells in a row and column for the component's display area. REMAINDER means use all remaining cells. RELATIVE means use all remaining cells except one.

**Insets insets**

Specifies the external padding of the component, the minimum amount of space between the component and the edges of its display area. The constructor for Insets is `Insets(int top, int left, int bottom, int right)`

**int ipadx, ipady**

Specifies the internal padding of the component, how much space to add to the minimum width of the component.

**double gridx, gridy**

Specifies the cell containing the leading edge of the component's display area, where the first cell in a row has `gridx=0` and in a column has `gridy=0`. RELATIVE specifies that the component be placed just below the component that was added to the container just before this component was added.

**double weightx, weighty**

Specifies how to distribute extra horizontal and vertical space.

## JOptionPane

**messageType (icon):** PLAIN\_MESSAGE, INFORMATION\_MESSAGE, QUESTION\_MESSAGE, WARNING\_MESSAGE, ERROR\_MESSAGE

**optionType:** YES\_NO\_CANCEL\_OPTION, YES\_NO\_OPTION, OK\_CANCEL\_OPTION

**responseType:** YES\_OPTION, NO\_OPTION, CANCEL\_OPTION, OK\_OPTION, CLOSED\_OPTION

**static int showConfirmDialog(Component parentComponent, Object message, [String title, int optionType, [int messageType, [Icon icon]]])**

Brings up dialog with icon, where number of choices is determined by optionType parameter.

**static String showInputDialog(Component parentComponent, Object message[, Object initialSelectionValue])**

Shows question-message dialog requesting input from user and parented to parentComponent.

**static String showInputDialog(Component parentComponent, Object message, String title, int messageType)**

Shows dialog requesting input from user with dialog having title and messageType.

**static Object showInputDialog(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object initialSelectionValue)**

Prompts user for input in a blocking dialog where all options can be specified.

**static void showMessageDialog(Component parentComponent, Object message, [String title, int messageType, [Icon icon]])**

Brings up dialog displaying a message, specifying all parameters.

**static int showOptionDialog(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon, Object[] options, Object initialValue)**

Brings up dialog with specified icon, where initial choice is determined by initialValue and number of choices is determined by optionType.

## JFileChooser

**JFileChooser([String currentDirectoryPath])**

**void addChoosableFileFilter(FileFilter filter)**

Adds a filter to the list of user choosable file filters.

**File getSelectedFile()**

Returns the selected file.

**void setFileFilter(FileFilter filter)**

Sets the current file filter.

**int showOpenDialog(Component parent)**

Pops up an "Open File" file chooser dialog.

**int showSaveDialog(Component parent)**

Pops up a "Save File" file chooser dialog.

## FileNameExtensionFilter extends FileFilter

**FileNameExtensionFilter(String description, String... extensions)**

# Main Window Widgets (all extend JComponent)

## JMenuBar

**JMenuBar()**

**JMenu add(JMenu c)**

Appends the specified menu to the end of the menu bar.

## JMenu

**JMenu(String s)**

**JMenuItem add(JMenuItem menuItem)**

Appends a menu item to the end of this menu.

## JMenuItem

**JMenuItem(String text, [Icon icon])**

**void addActionListener(ActionListener l)**

Adds an ActionListener to the widget.

## JToolBar

**JToolBar()**

**void addSeparator([Dimension size])**

Appends a separator of a specified size to the end of the tool bar.

## JButton

See Other Widgets

# Other Widgets (all extend JComponent)

## AbstractButton

**void addActionListener(ActionListener l)**

Adds an ActionListener to the widget.

**boolean isSelected()**

Returns the state of the button.

**void setSelected(boolean b)**

Sets the state of the button.

**void setIcon(Icon defaultIcon)**

Sets the button's default (unselected) icon.

**void setSelectedIcon(Icon selectedIcon)**

Sets the selected icon for the button.

## JButton extends AbstractButton

**JButton([String text], [Icon icon])**

## JCheckBox extends AbstractButton

**JCheckBox([String text], [Icon icon], [boolean selected])**

## JComboBox<E>

**JComboBox(E[] items)**

**void addActionListener(ActionListener l)**

Adds an ActionListener to the widget.

**void addItem(E item)**

Adds an item to the item list.

**E getItemAt(int index)**

Returns the list item at the specified index.

**int getItemCount()**

Returns the number of items in the list.

**int getSelectedIndex()**

Returns the first item in the list that matches the given item.

**Object getSelectedItem()**

Returns the current selected item.

**void setEditable(boolean aFlag)**

Determines whether the JComboBox field is editable.

**void setSelectedIndex(int anIndex)**

Selects the item at index anIndex.

**void setSelectedItem(Object anObject)**

Sets the selected item in the combo box display area to the object in the argument.

## **JLabel**

**JLabel([String text], [Icon icon], [int horizontalAlignment])**

horizontalAlignment: SwingConstants.LEFT, CENTER, RIGHT, LEADING, TRAILING

**String getText()**

Returns the text string that the label displays.

**Icon getIcon()**

Returns the graphic image (glyph, icon) that the label displays.

**void setIcon(Icon icon)**

Defines the icon this component will display.

**void setText(String text)**

Defines the single line of text this component will display.

## **JProgressBar**

**JProgressBar([int orient], [int min, int max])**

orient: SwingConstants.VERTICAL, HORIZONTAL

**double getPercentComplete()**

Returns the percent complete for the progress bar.

**void setString(String s)**

Sets the value of the progress string.

**void setValue(int n)**

Sets the progress bar's current value to n.

## **JRadioButton extends AbstractButton**

**JRadioButton(String text, Icon icon, boolean selected)**

## **JSlider**

**JSlider([int orientation], [int min, int max], [int value])**

**void addChangeListener(ChangeListener l)**

Adds a ChangeListener to the slider.

**int getValue()**

Returns the slider's current value.

**void setValue(int n)**

Sets the slider's current value to n.

## JSpinner

**JSpinner(SpinnerModel model)**

**void addActionListener(ActionListener l)**

Adds an ActionListener to the widget.

**Object getValue()**

Returns the current value of the model, typically this value is displayed by the editor.

**void setValue(Object value)**

Changes current value of the model, typically this value is displayed by the editor.

## SpinnerNumberModel implements SpinnerModel

**SpinnerNumberModel(int value, int minimum, int maximum, int stepSize)**

**SpinnerNumberModel(double value, double minimum, double maximum, double stepSize)**

## JTextField

**JTextField([String text], [int columns])**

**void addActionListener(ActionListener l)**

Adds an ActionListener to the widget.

**String getText()**

Returns the text contained in this TextComponent.

**void setText(String t)**

Sets the text of this TextComponent to the specified text.

## JToggleButton extends AbstractButton

**JToggleButton([String text], [Icon icon], [boolean selected])**



## Other Classes

### Graphics2D extends Graphics

#### **Graphics create()**

Creates a new Graphics object that is a copy of this Graphics object.

#### **boolean drawImage(Image img, int x, int y, ImageObserver observer)**

Draws as much of the specified image as is currently available.

#### **void drawLine(int x1, int y1, int x2, int y2)**

Draws a line, using the current color, between the points (x1, y1) and (x2, y2) in this graphics context's coordinate system.

#### **void drawRect(int x, int y, int width, int height)**

Draws the outline of the specified rectangle.

#### **void drawString(String str, int x, int y)**

Draws the text given by the specified string, using this graphics context's current font and color.

#### **void setBackground(Color color)**

Sets the background color for the Graphics2D context.

#### **void setColor(Color c)**

Sets this graphics context's current color to the specified color.

#### **void setFont(Font font)**

Sets this graphics context's font to the specified font.

#### **void translate(int x, int y)**

Translates the origin of the graphics context to the point (x, y) in the current coordinate system.

## Color

**Color(float r, float g, float b, [float a])**

**Color(int r, int g, int b, [int a])**

**Color(int rgb)**

#### **int getRGB()**

Returns the RGB value representing the color in the default sRGB ColorModel.

Color BLACK, BLUE, CYAN, DARK\_GRAY, GRAY, GREEN, LIGHT\_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW

## Dimension

**Dimension(int width, int height)**

#### **int height, width**

Number of pixels in each direction

## ImageIcon implements Icon

**ImageIcon([String filename, [String description]])**

## Interface ActionListener

**void actionPerformed(ActionEvent e)**

Invoked when an action occurs.

## ActionEvent

**String getActionCommand()**

Returns the command string associated with this action.

**int getModifiers()**

Returns the bitwise OR of the modifier keys held down during this action event (ALT\_MASK, CTRL\_MASK, META\_MASK, SHIFT\_MASK).

## Interface ChangeListener

**void stateChanged(ChangeEvent e)**

Invoked when the target of the listener has changed its state.

## ChangeEvent

**Object getSource()**

The object on which the Event initially occurred.

## Rectangle

**Rectangle([int x, int y], int width, int height)**

**int height, width**

Number of pixels in each direction

**int x, y**

Location on the Cartesian plane