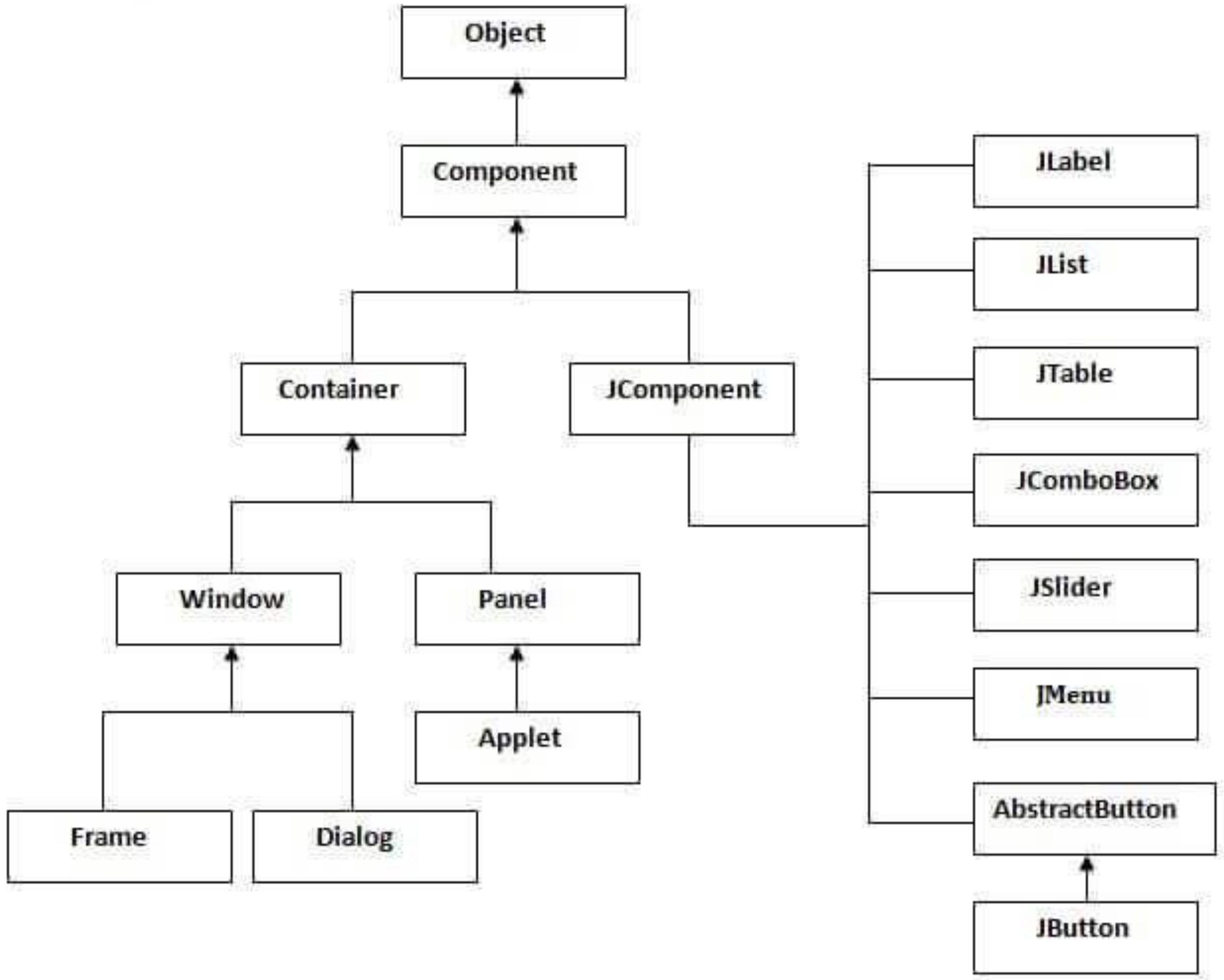
GUI Components- Part 2
Introduction to Swing

By Prof. Zarrin J

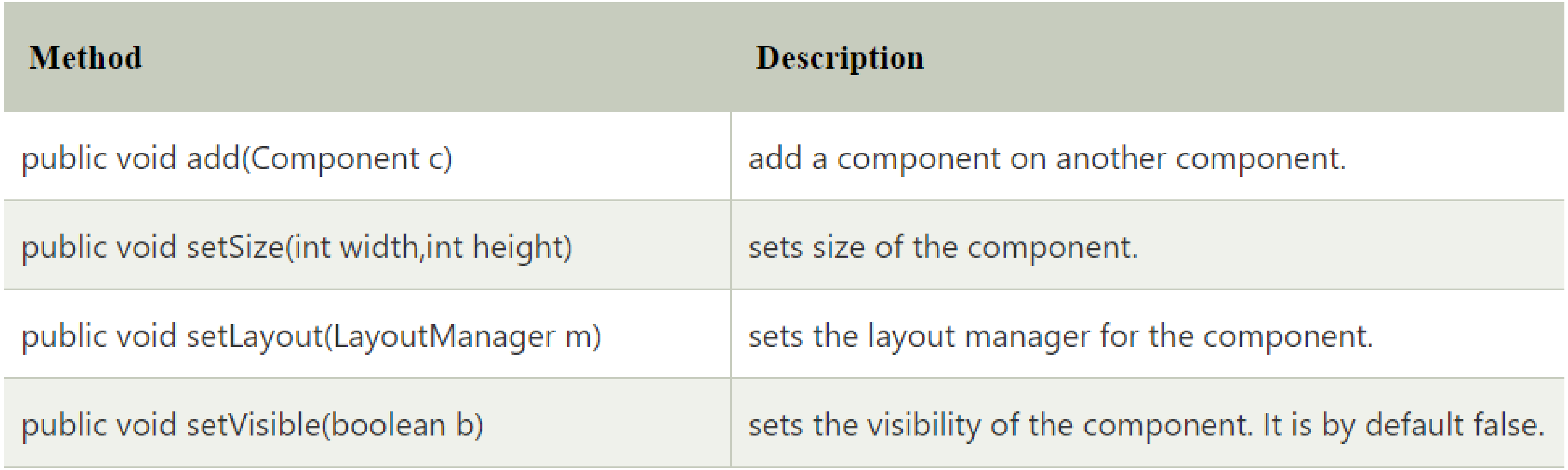
What is Java Swing ?

- Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.
- Unlike AWT, Java Swing provides platform-independent and lightweight components.
- The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Hierarchy of Java Swing Classes

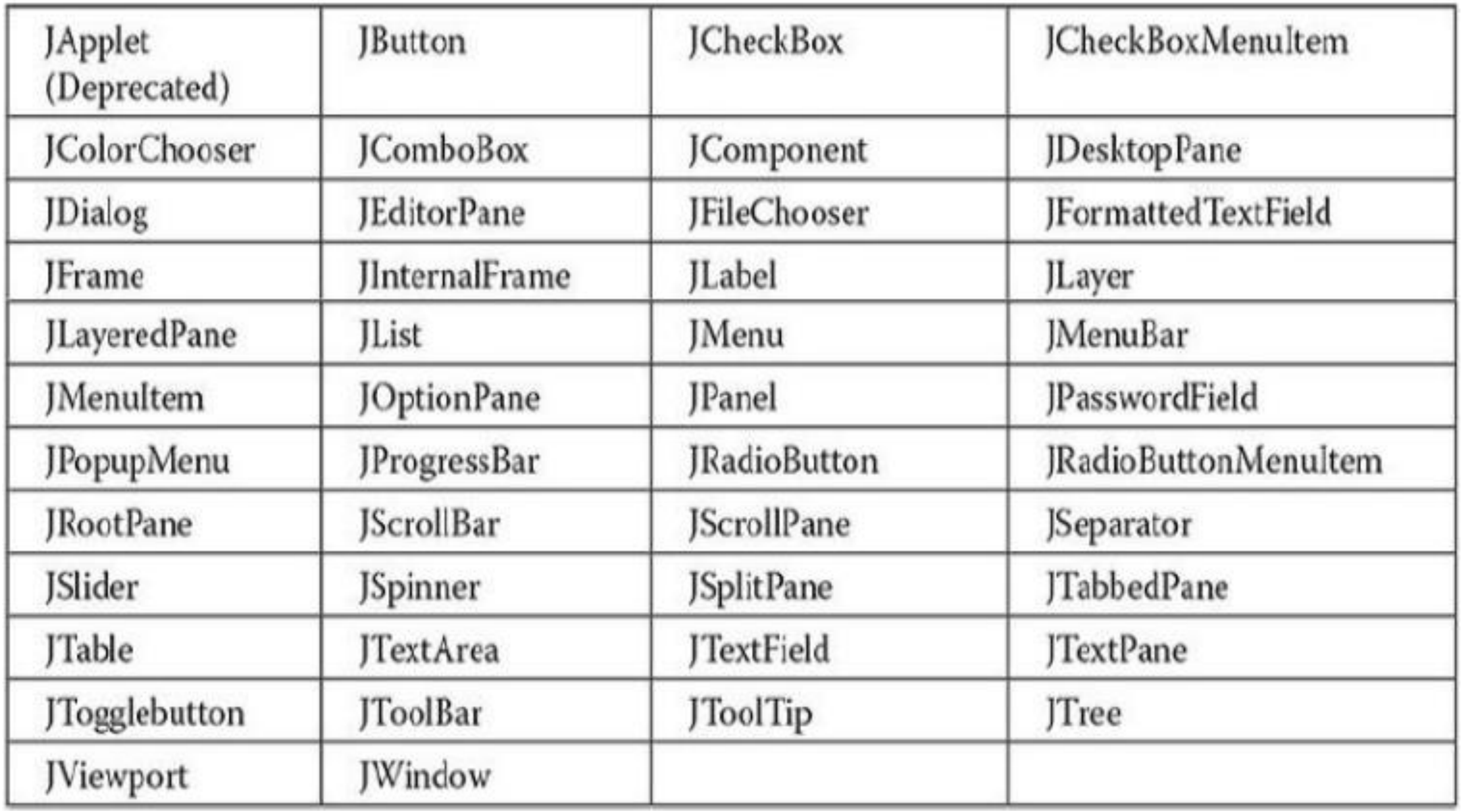


Commonly used methods of component class



Method	Description
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

Java Swing Components



JApplet (Deprecated)	JButton	JCheckBox	JCheckBoxMenuItem
JColorChooser	JComboBox	JComponent	JDesktopPane
JDialog	JEditorPane	JFileChooser	JFormattedTextField
JFrame	JInternalFrame	JLabel	JLayer
JLayeredPane	JList	JMenu	JMenuBar
JMenuItem	JOptionPane	JPanel	JPasswordField
JPopupMenu	JProgressBar	JRadioButton	JRadioButtonMenuItem
JRootPane	JScrollBar	JScrollPane	JSeparator
JSlider	JSpinner	JSplitPane	JTabbedPane
JTable	JTextArea	JTextField	JTextPane
JTogglebutton	JToolBar	JToolTip	JTree
JViewport	JWindow		

1. Top level Containers

- It inherits Component and Container of AWT.
- It cannot be contained within other containers.
- Heavyweight.
- Example: JFrame, JDialog, Japplet

2. Light weight Containers

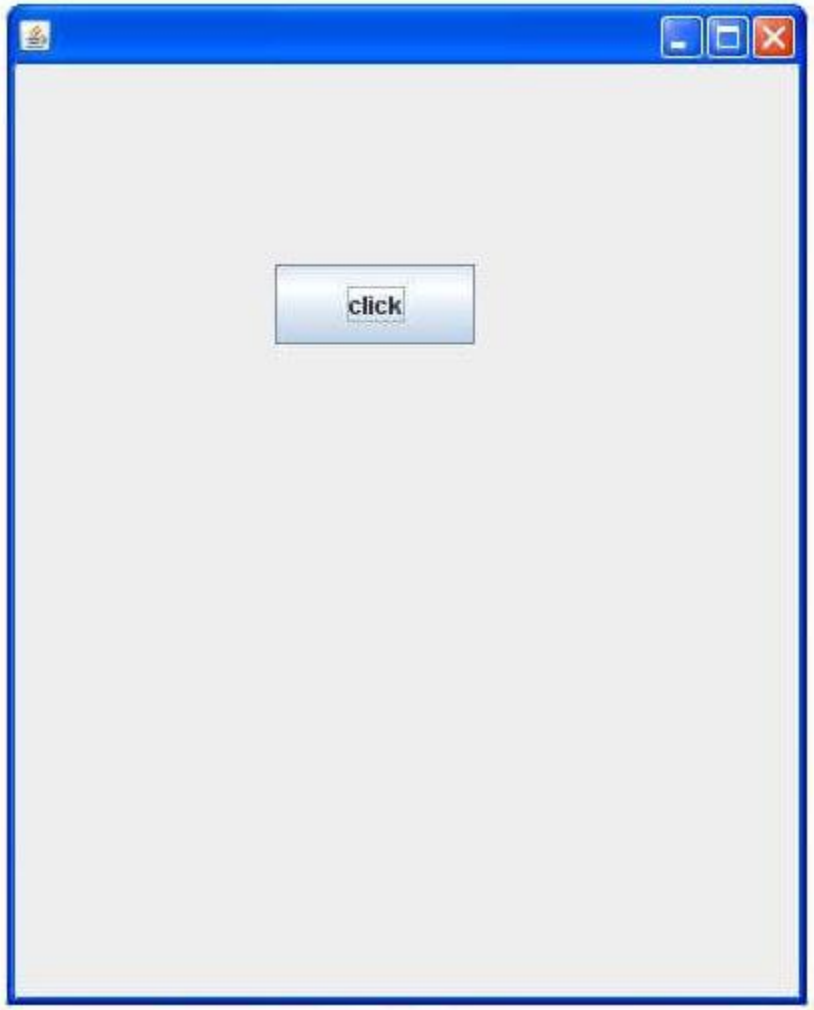
- It inherits Jcomponent class.
- It is a general purpose container.
- It can be used to organize related components together.
- Example: JPanel

JFrame

The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.

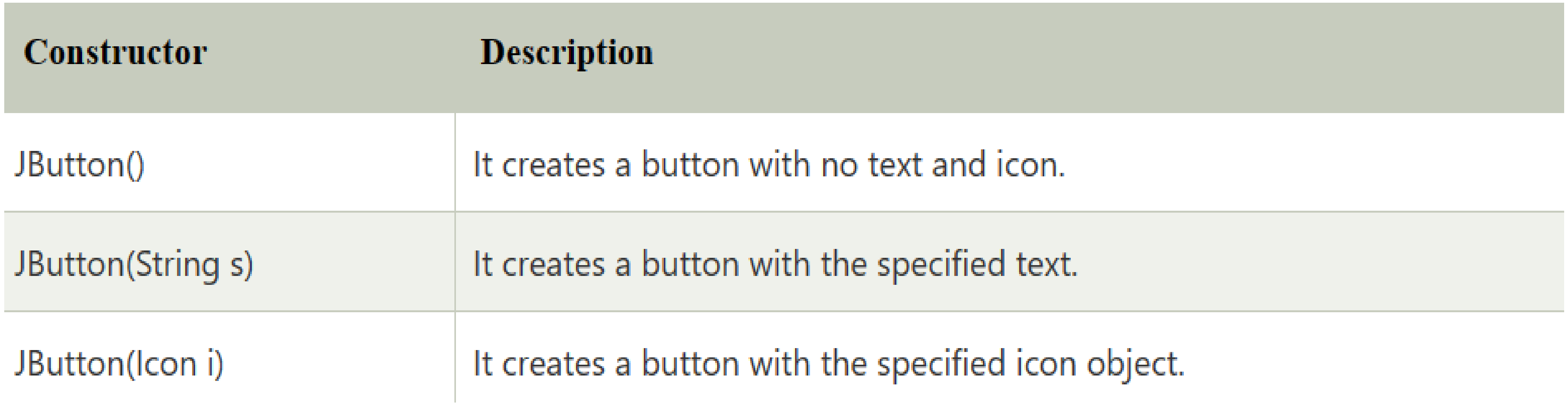
The two methods are:

1. By inheriting JFrame
2. By creating object



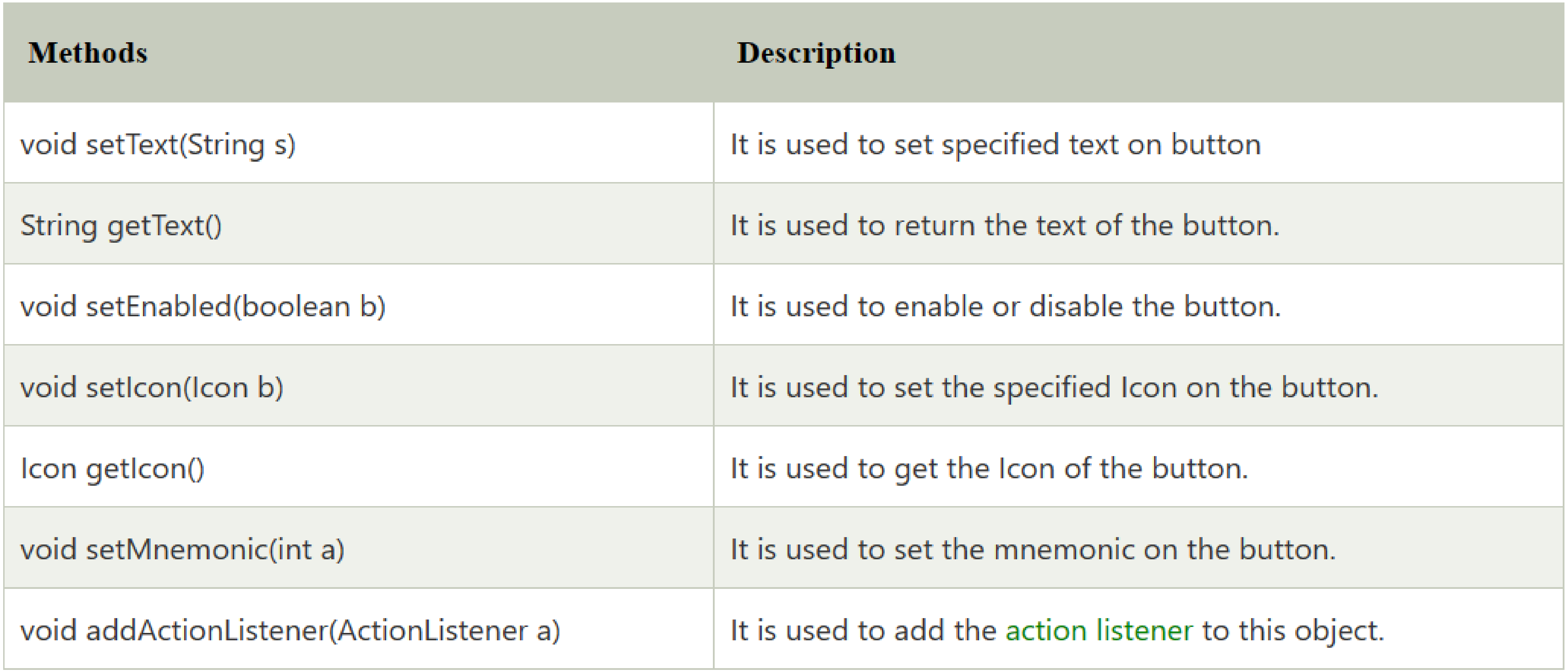
JBelow

- The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.
- Constructors of Jbutton:



Constructor	Description
JButton()	It creates a button with no text and icon.
JButton(String s)	It creates a button with the specified text.
JButton(Icon i)	It creates a button with the specified icon object.

Method of AbstractButton class used by JButton



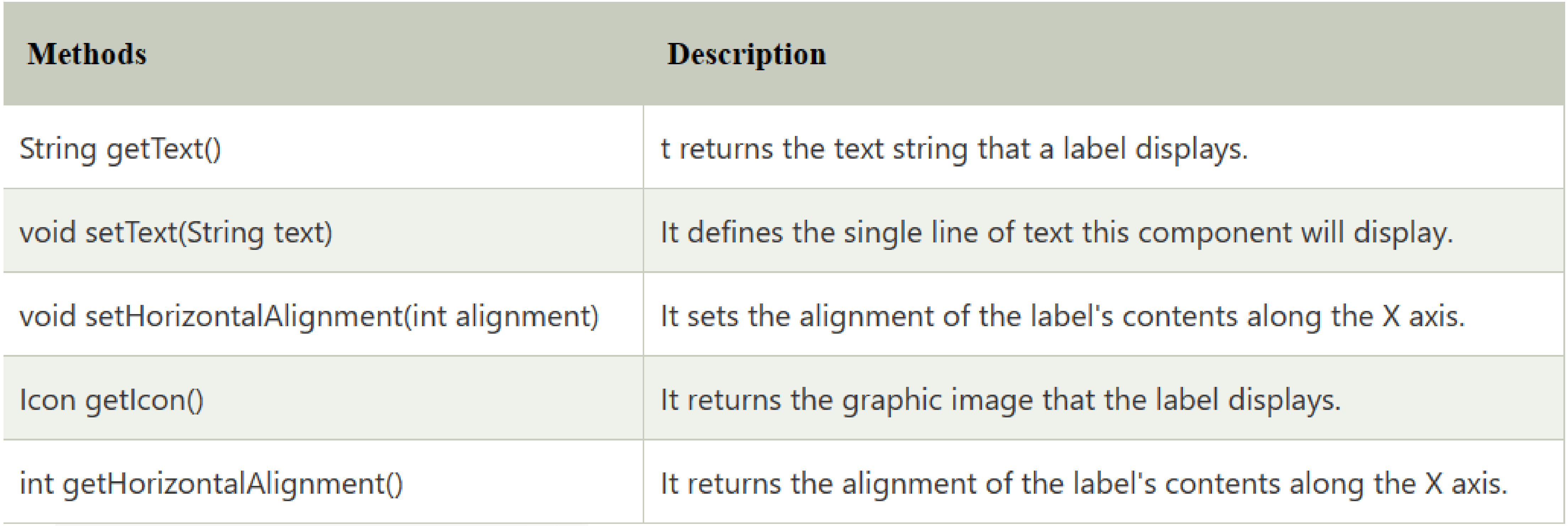
Methods	Description
void setText(String s)	It is used to set specified text on button
String getText()	It is used to return the text of the button.
void setEnabled(boolean b)	It is used to enable or disable the button.
void setIcon(Icon b)	It is used to set the specified Icon on the button.
Icon getIcon()	It is used to get the Icon of the button.
void setMnemonic(int a)	It is used to set the mnemonic on the button.
void addActionListener(ActionListener a)	It is used to add the action listener to this object.

JLabel

- The object of JLabel class is for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.
- Constructors of Jlabel:

Constructor	**Description**
JLabel()	Creates a JLabel instance with no image and with an empty string for the title.
JLabel(String s)	Creates a JLabel instance with the specified text.
JLabel(Icon i)	Creates a JLabel instance with the specified image.
JLabel(String s, Icon i, int horizontalAlignment)	Creates a JLabel instance with the specified text, image, and horizontal alignment.

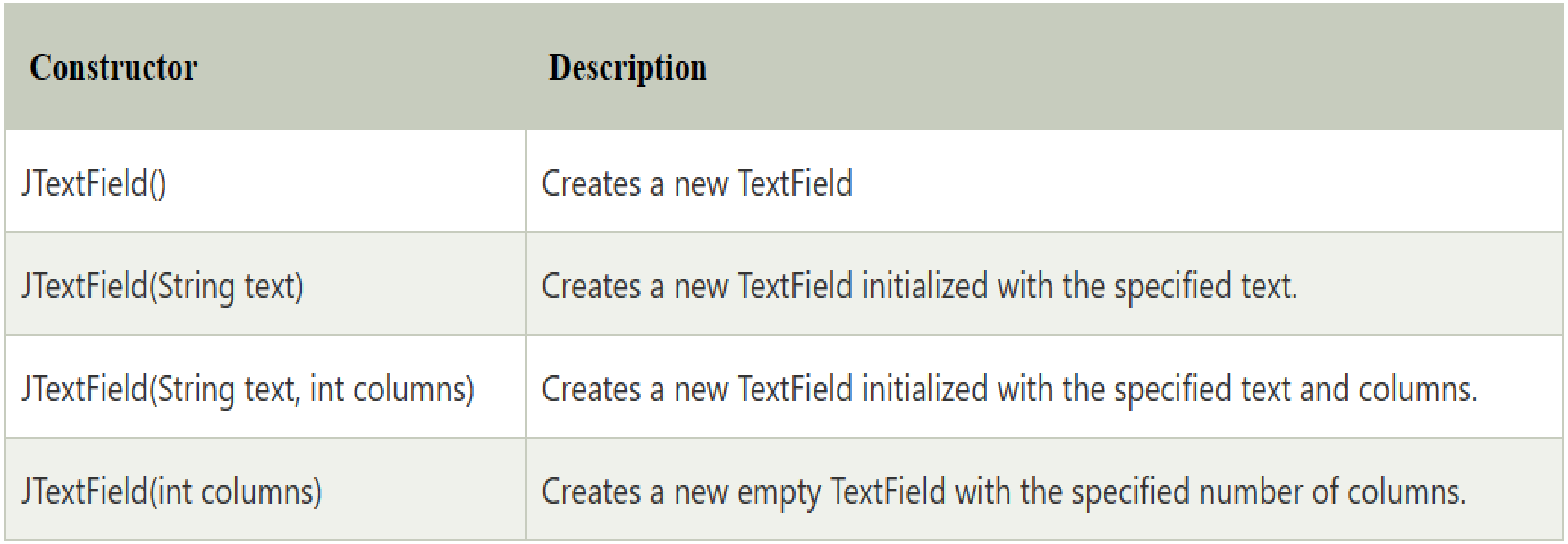
Methods Used by JLabel



Methods	Description
String getText()	t returns the text string that a label displays.
void setText(String text)	It defines the single line of text this component will display.
void setHorizontalAlignment(int alignment)	It sets the alignment of the label's contents along the X axis.
Icon getIcon()	It returns the graphic image that the label displays.
int getHorizontalAlignment()	It returns the alignment of the label's contents along the X axis.

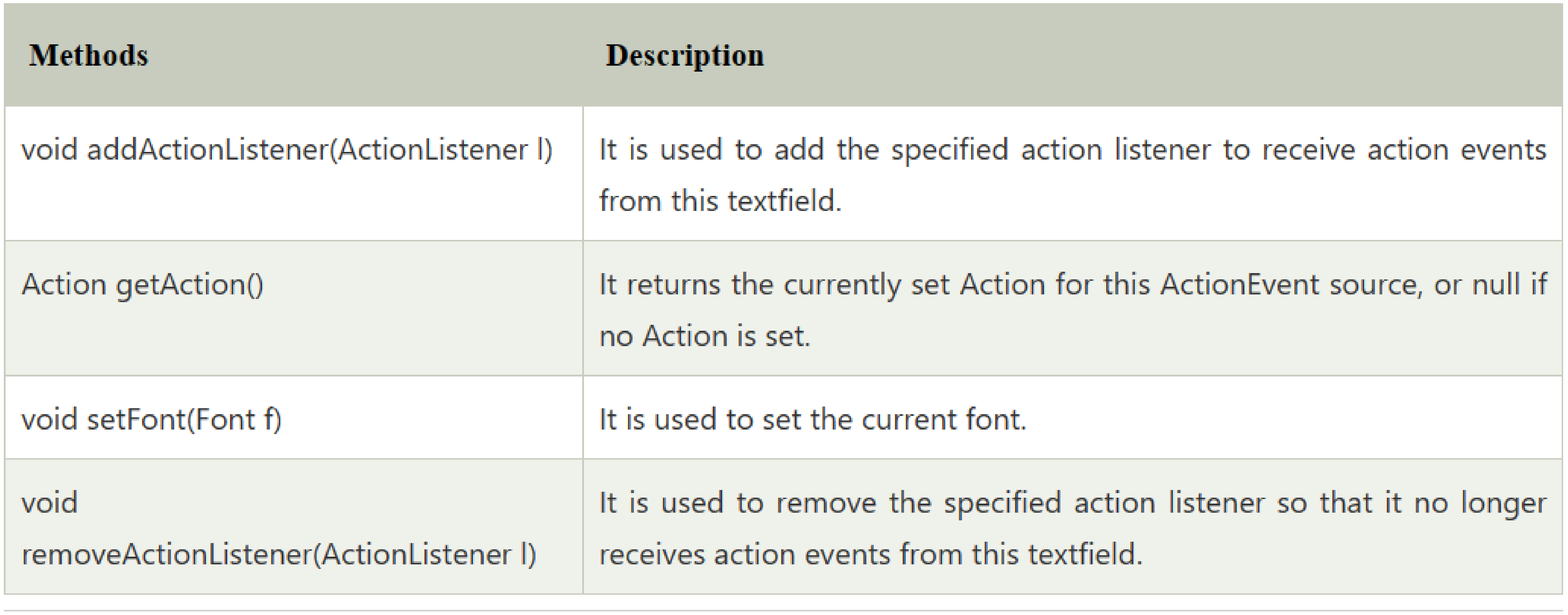
JTextField

The object of a JTextField class is a text component that allows the editing of a single line text. It inherits JTextComponent class.



Constructor	Description
JTextField()	Creates a new TextField
JTextField(String text)	Creates a new TextField initialized with the specified text.
JTextField(String text, int columns)	Creates a new TextField initialized with the specified text and columns.
JTextField(int columns)	Creates a new empty TextField with the specified number of columns.

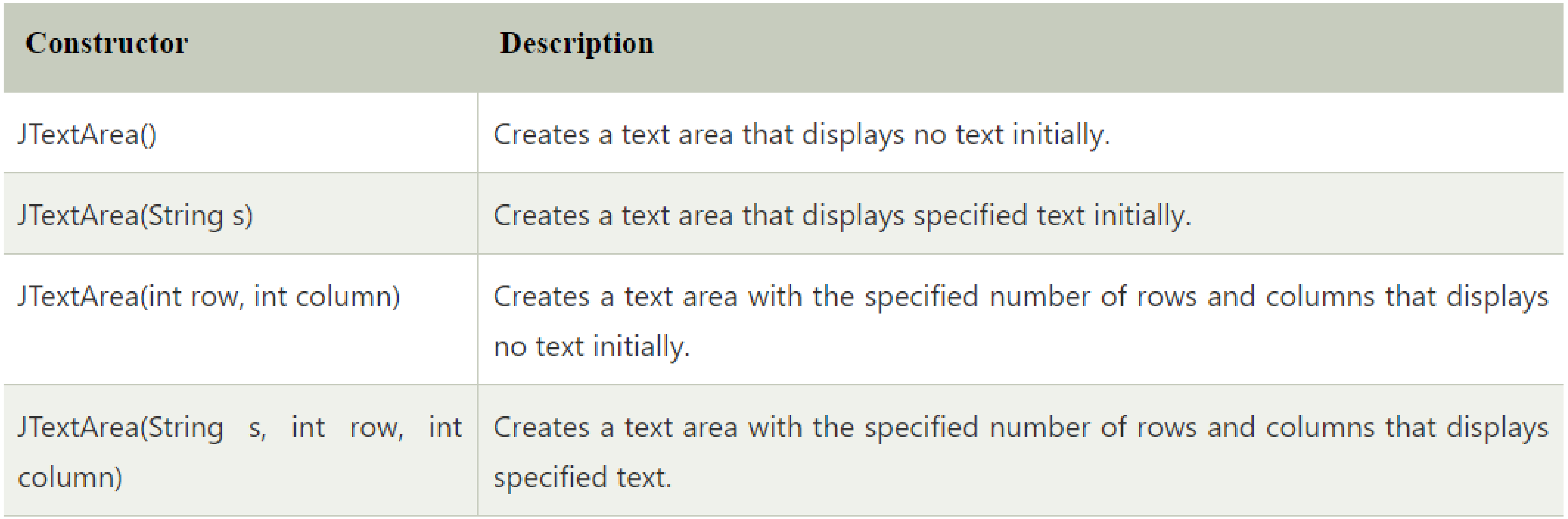
Methods Used by JTextField



Methods	Description
void addActionListener(ActionListener l)	It is used to add the specified action listener to receive action events from this textfield.
Action getAction()	It returns the currently set Action for this ActionEvent source, or null if no Action is set.
void setFont(Font f)	It is used to set the current font.
void removeActionListener(ActionListener l)	It is used to remove the specified action listener so that it no longer receives action events from this textfield.

JTextArea

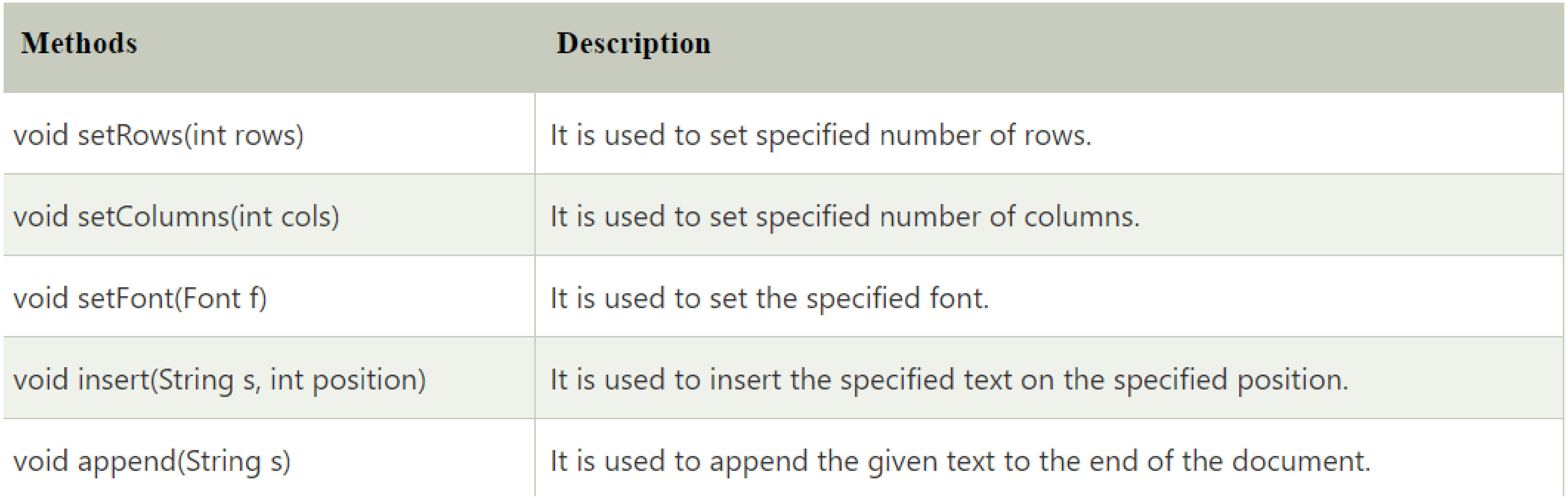
- The object of a JTextArea class is a multi line region that displays text. It allows the editing of multiple line text. It inherits JTextComponent class
- Commonly used Constructors:



Constructor	Description
JTextArea()	Creates a text area that displays no text initially.
JTextArea(String s)	Creates a text area that displays specified text initially.
JTextArea(int row, int column)	Creates a text area with the specified number of rows and columns that displays no text initially.
JTextArea(String s, int row, int column)	Creates a text area with the specified number of rows and columns that displays specified text.

Methods Used by JTextArea

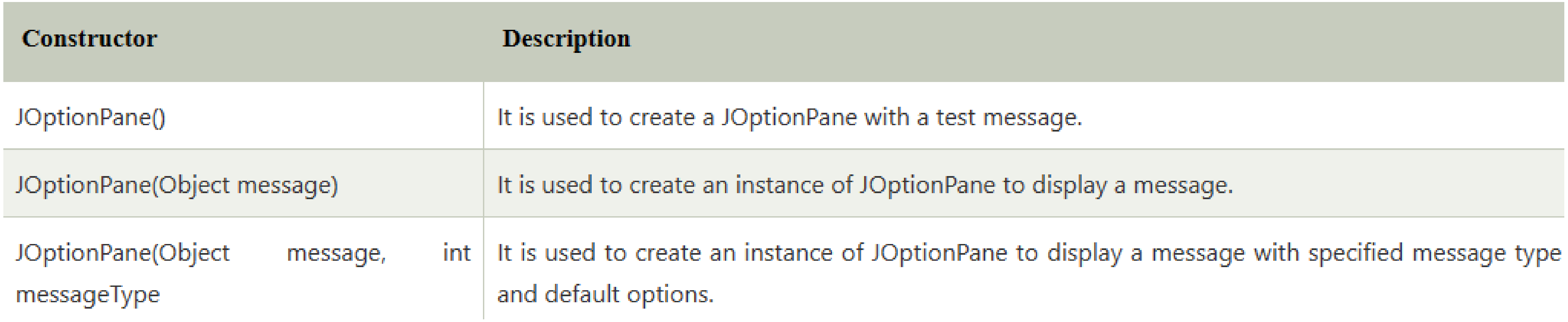
- Commonly used methods:



Methods	Description
void setRows(int rows)	It is used to set specified number of rows.
void setColumns(int cols)	It is used to set specified number of columns.
void setFont(Font f)	It is used to set the specified font.
void insert(String s, int position)	It is used to insert the specified text on the specified position.
void append(String s)	It is used to append the given text to the end of the document.

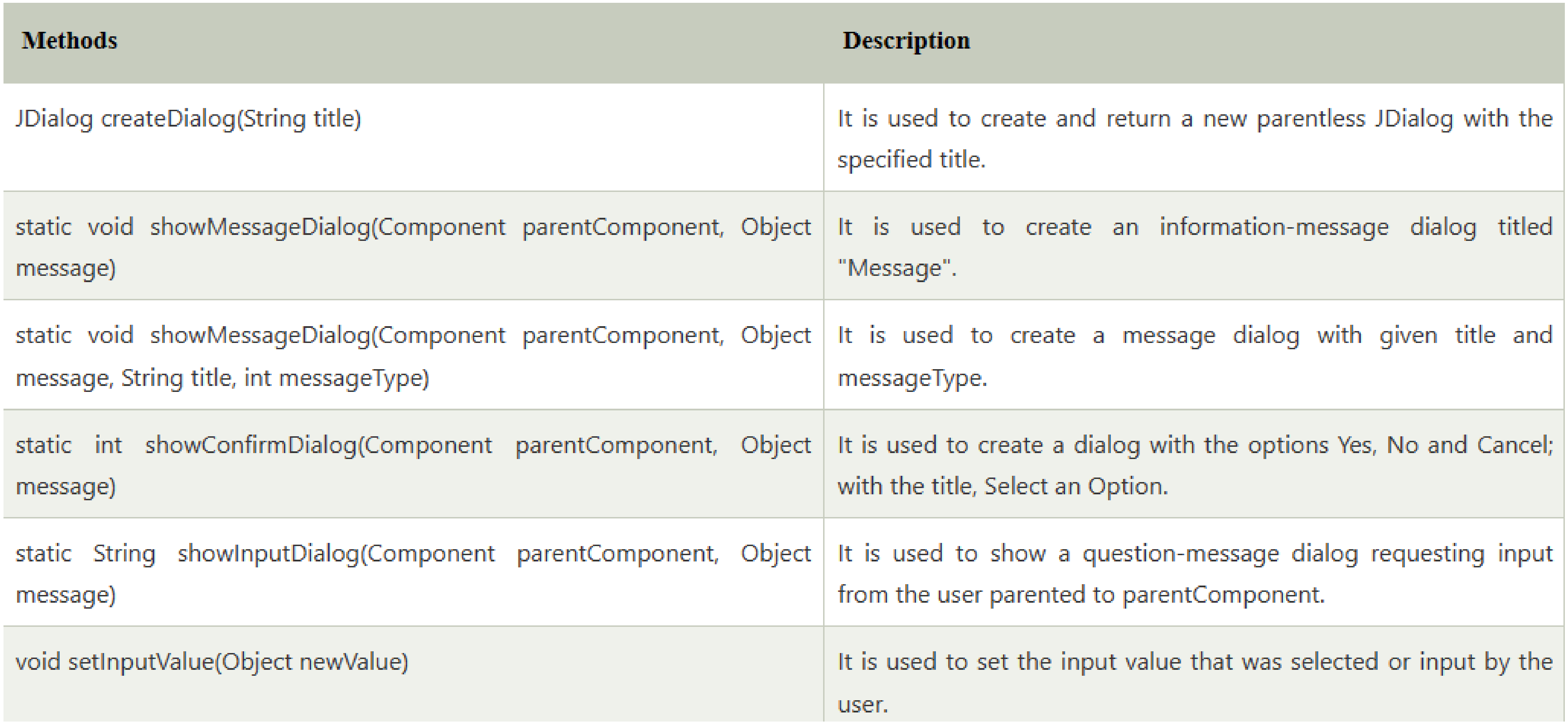
JOptionPane

- It is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user. The JOptionPane class inherits JComponent class.
- Constructors of JOptionPane are:



Constructor	Description
JOptionPane()	It is used to create a JOptionPane with a test message.
JOptionPane(Object message)	It is used to create an instance of JOptionPane to display a message.
JOptionPane(Object message, int messageType	It is used to create an instance of JOptionPane to display a message with specified message type and default options.

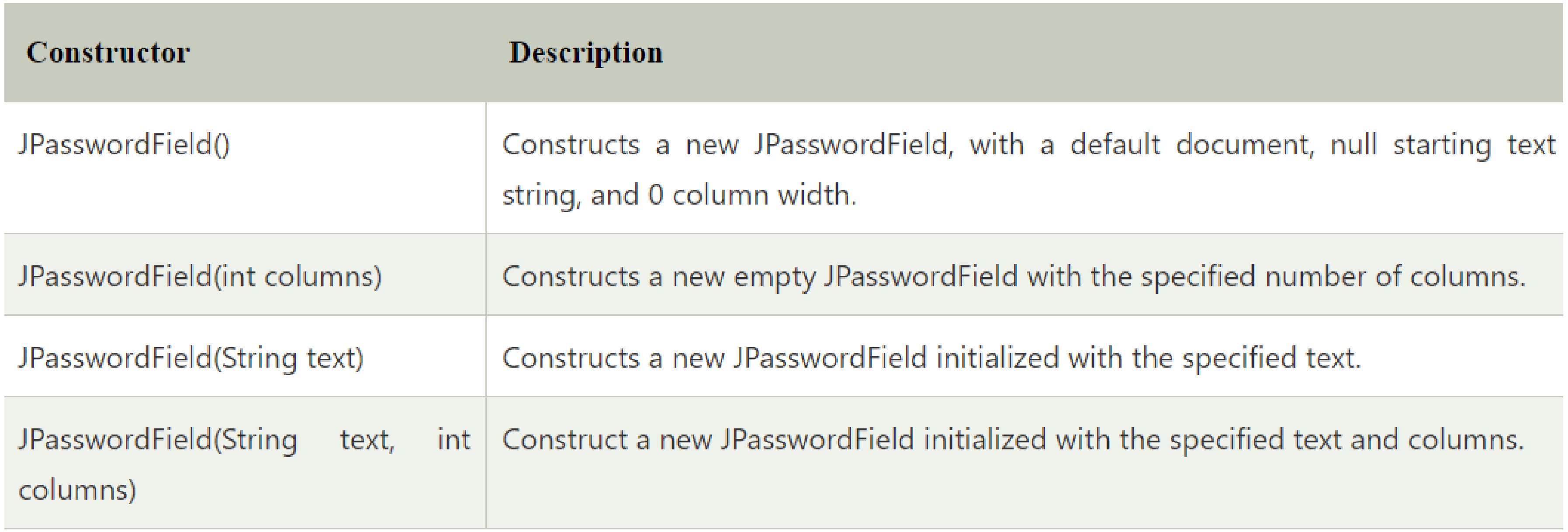
Methods Used by JOptionPane



Methods	Description
JDialog createDialog(String title)	It is used to create and return a new parentless JDialog with the specified title.
static void showMessageDialog(Component parentComponent, Object message)	It is used to create an information-message dialog titled "Message".
static void showMessageDialog(Component parentComponent, Object message, String title, int messageType)	It is used to create a message dialog with given title and messageType.
static int showConfirmDialog(Component parentComponent, Object message)	It is used to create a dialog with the options Yes, No and Cancel; with the title, Select an Option.
static String showInputDialog(Component parentComponent, Object message)	It is used to show a question-message dialog requesting input from the user parented to parentComponent.
void setInputValue(Object newValue)	It is used to set the input value that was selected or input by the user.

JPasswordField

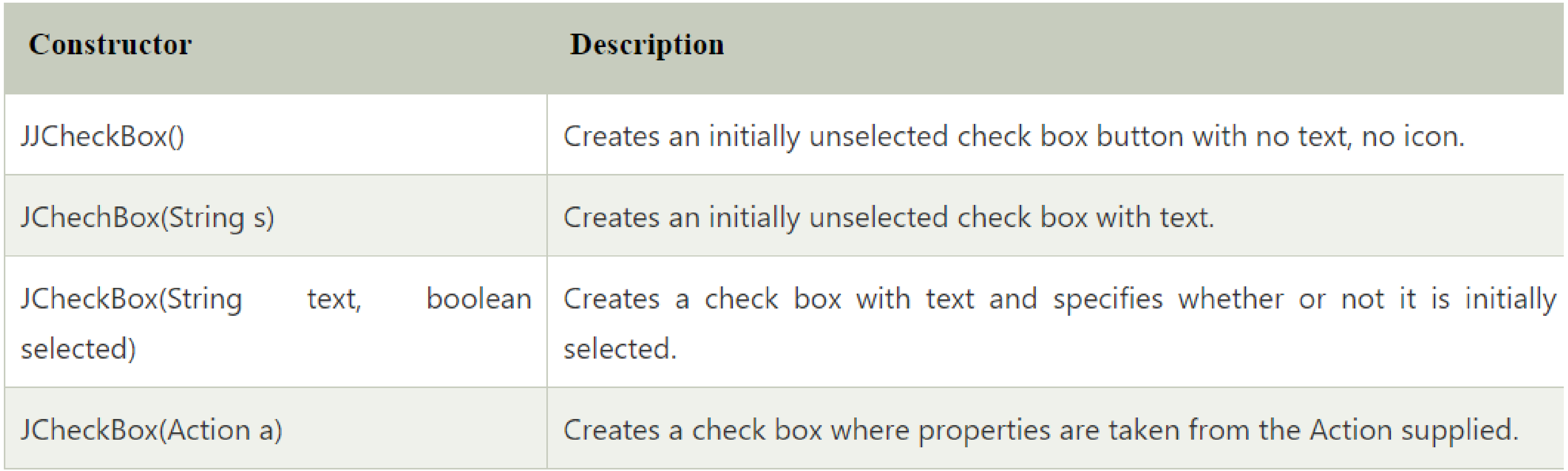
- The object of a JPasswordField class is a text component specialized for password entry. It allows the editing of a single line of text. It inherits JTextField class.
- Constructors:



Constructor	Description
JPasswordField()	Constructs a new JPasswordField, with a default document, null starting text string, and 0 column width.
JPasswordField(int columns)	Constructs a new empty JPasswordField with the specified number of columns.
JPasswordField(String text)	Constructs a new JPasswordField initialized with the specified text.
JPasswordField(String text, int columns)	Construct a new JPasswordField initialized with the specified text and columns.

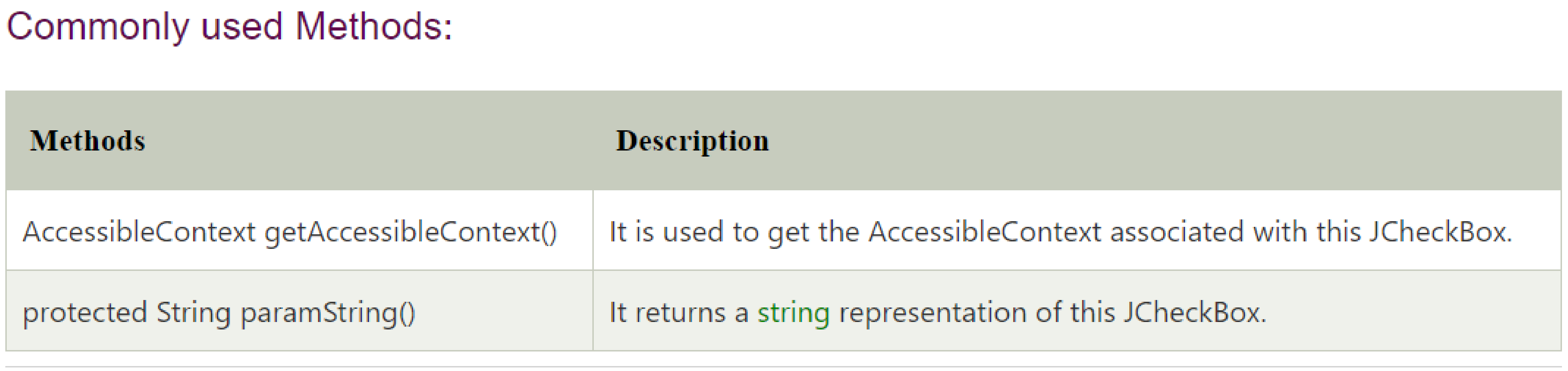
JCheckBox

- The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on ".It inherits JToggleButton class.
- Constructors:



Constructor	Description
JJCheckBox()	Creates an initially unselected check box button with no text, no icon.
JChechBox(String s)	Creates an initially unselected check box with text.
JCheckBox(String text, boolean selected)	Creates a check box with text and specifies whether or not it is initially selected.
JCheckBox(Action a)	Creates a check box where properties are taken from the Action supplied.

Methods Used by JCheckBox

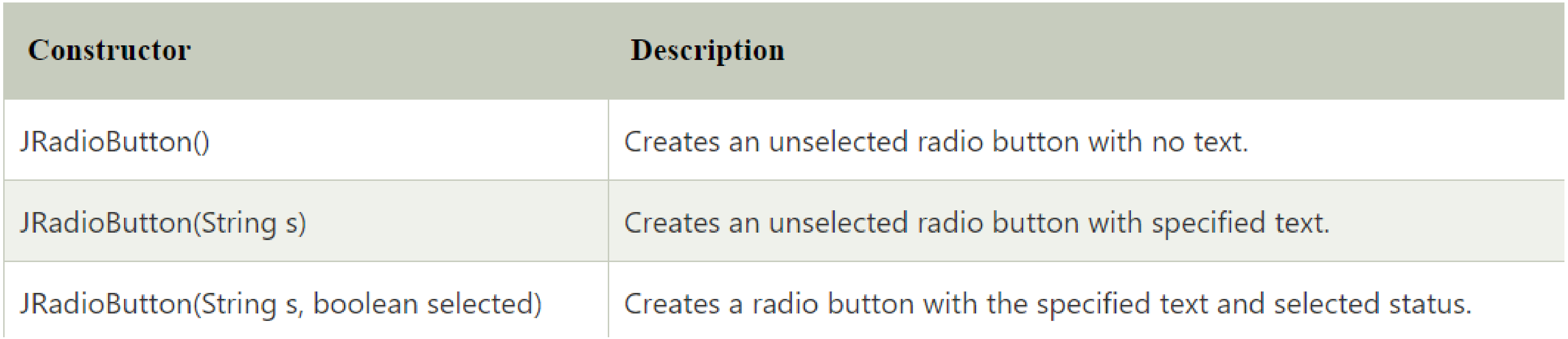


Commonly used Methods:

Methods	Description
AccessibleContext getAccessibleContext()	It is used to get the AccessibleContext associated with this JCheckBox.
protected String paramString()	It returns a string representation of this JCheckBox.

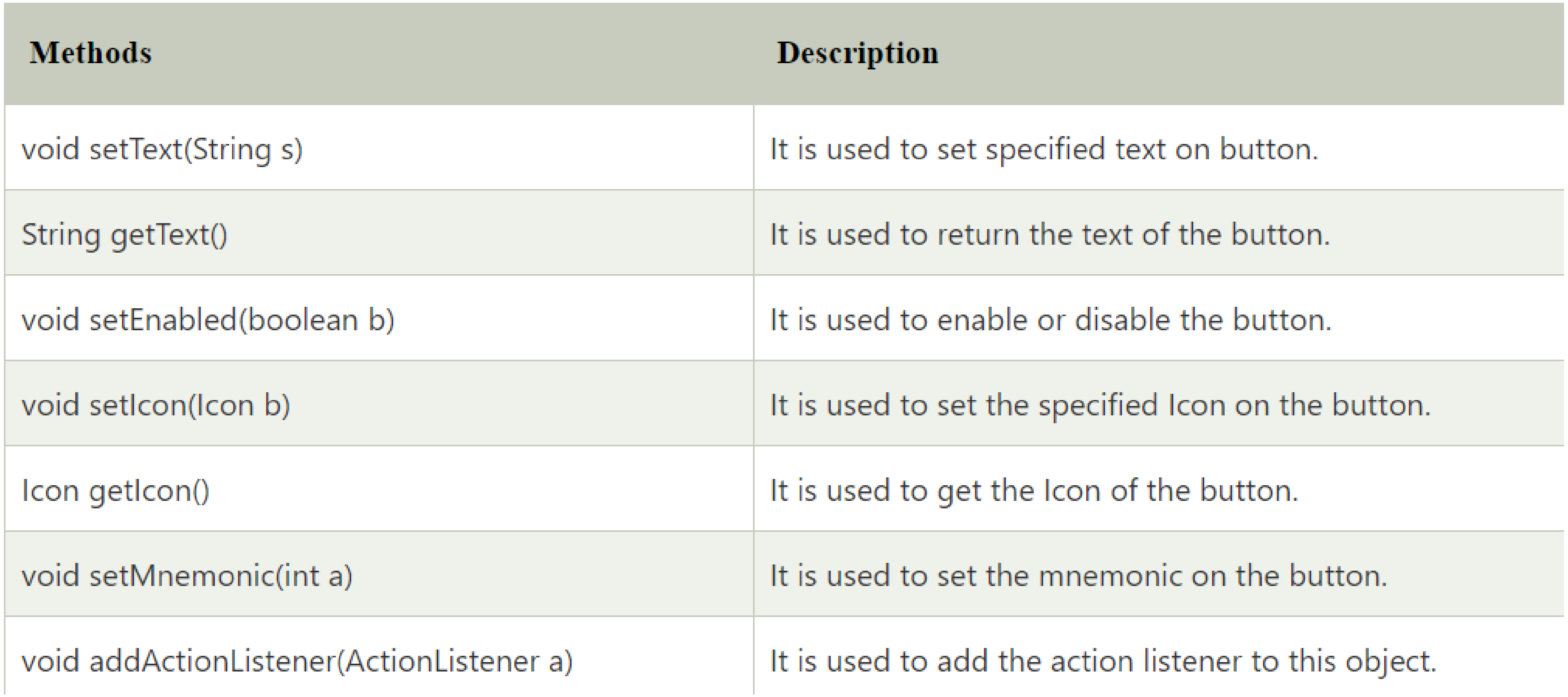
JRadioButton

- The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz.
- It should be added in ButtonGroup to select one radio button only.



Constructor	Description
JRadioButton()	Creates an unselected radio button with no text.
JRadioButton(String s)	Creates an unselected radio button with specified text.
JRadioButton(String s, boolean selected)	Creates a radio button with the specified text and selected status.

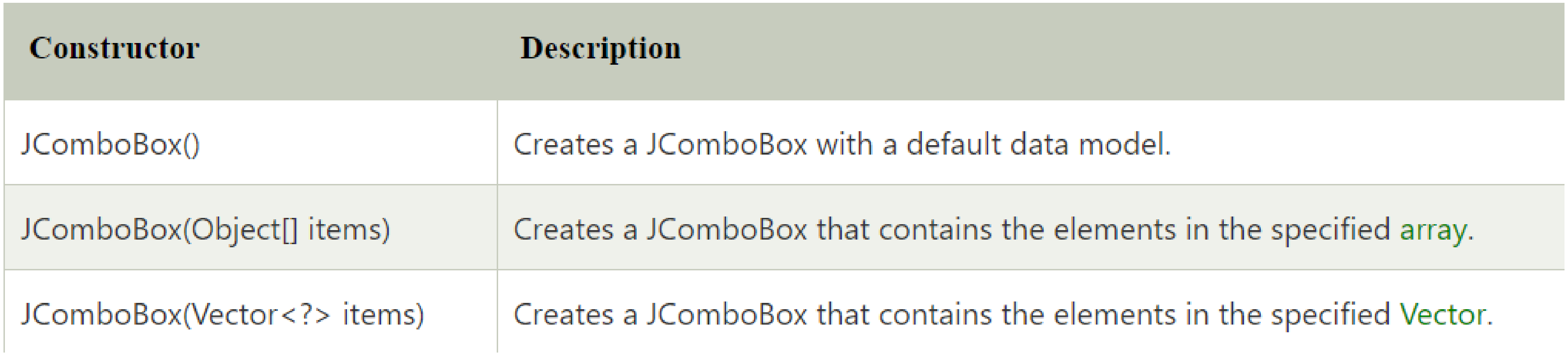
Methods Used by JRadioButton



Methods	Description
void setText(String s)	It is used to set specified text on button.
String getText()	It is used to return the text of the button.
void setEnabled(boolean b)	It is used to enable or disable the button.
void setIcon(Icon b)	It is used to set the specified Icon on the button.
Icon getIcon()	It is used to get the Icon of the button.
void setMnemonic(int a)	It is used to set the mnemonic on the button.
void addActionListener(ActionListener a)	It is used to add the action listener to this object.

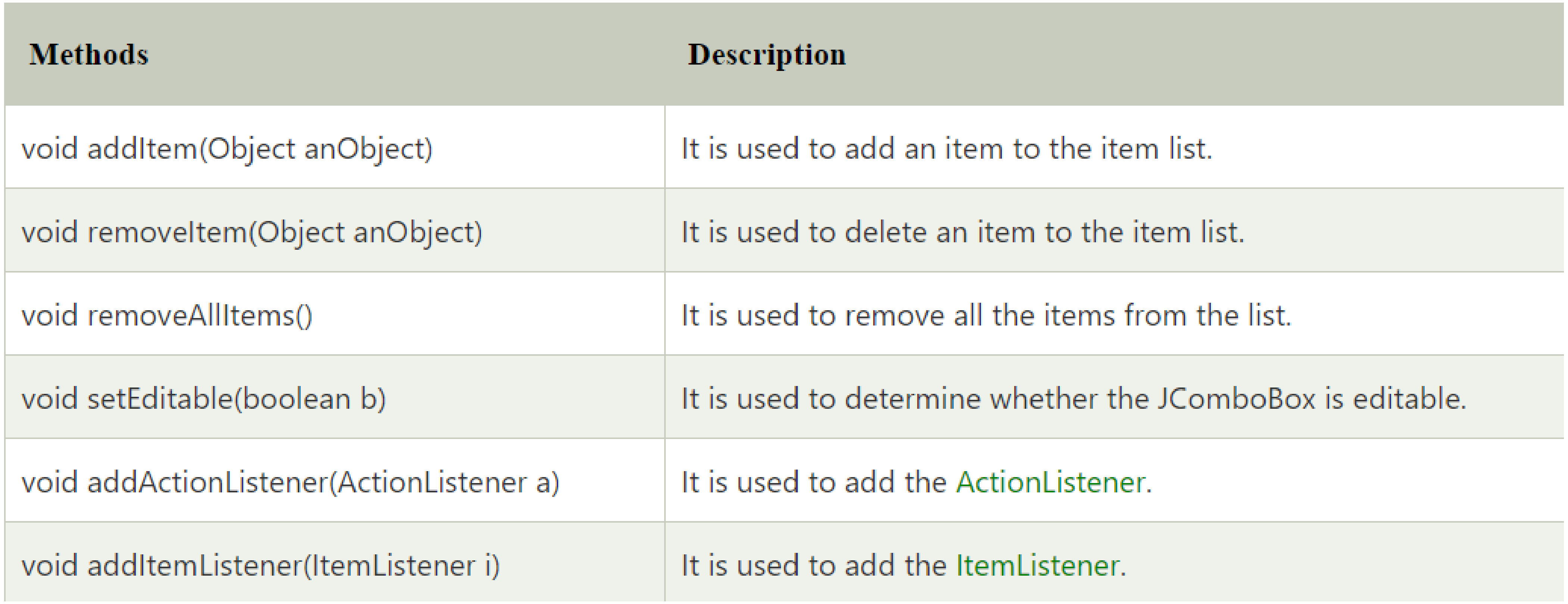
JComboBox

- The object of Choice class is used to show popup menu of choices. Choice selected by user is shown on the top of a menu. It inherits JComponent class.



Constructor	Description
JComboBox()	Creates a JComboBox with a default data model.
JComboBox(Object[] items)	Creates a JComboBox that contains the elements in the specified array.
JComboBox(Vector<?> items)	Creates a JComboBox that contains the elements in the specified Vector.

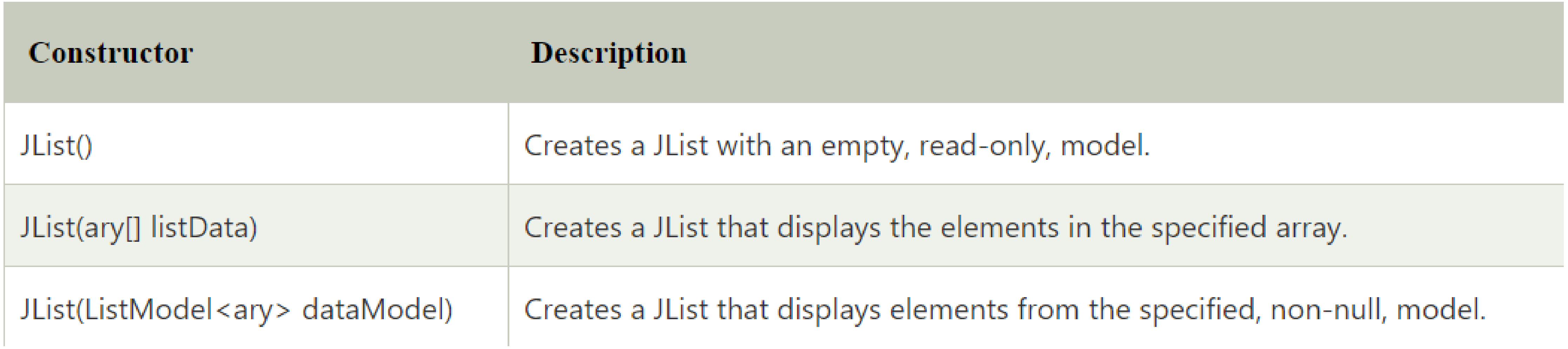
Methods Used by JComboBox



Methods	Description
void addItem(Object anObject)	It is used to add an item to the item list.
void removeItem(Object anObject)	It is used to delete an item to the item list.
void removeAllItems()	It is used to remove all the items from the list.
void setEditable(boolean b)	It is used to determine whether the JComboBox is editable.
void addActionListener(ActionListener a)	It is used to add the ActionListener.
void addItemListener(ItemListener i)	It is used to add the ItemListener.

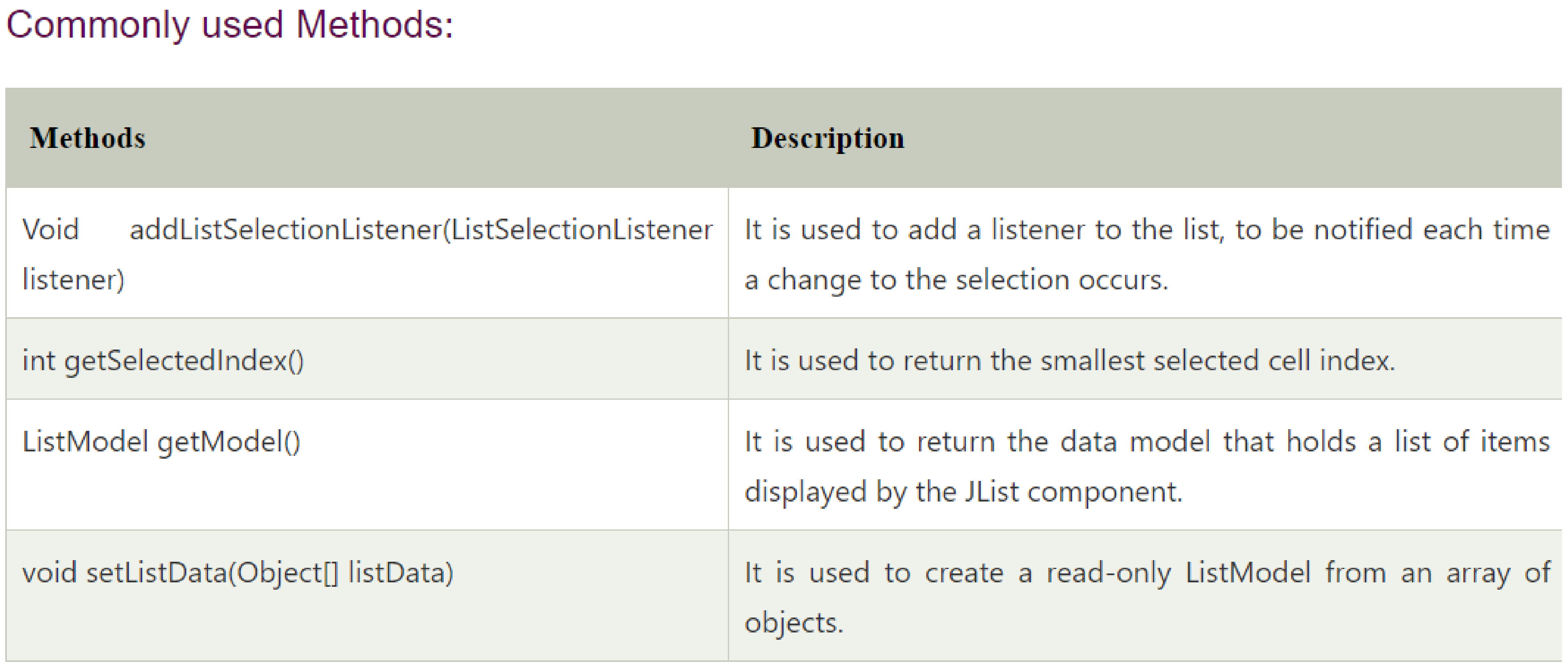
JList

- The object of JList class represents a list of text items. The list of text items can be set up so that the user can choose either one item or multiple items. It inherits JComponent class.



Constructor	Description
JList()	Creates a JList with an empty, read-only, model.
JList(ary[] listData)	Creates a JList that displays the elements in the specified array.
JList(ListModel<ary> dataModel)	Creates a JList that displays elements from the specified, non-null, model.

Methods Used by JList



Commonly used Methods:

Methods	Description
Void addListSelectionListener(ListSelectionListener listener)	It is used to add a listener to the list, to be notified each time a change to the selection occurs.
int getSelectedIndex()	It is used to return the smallest selected cell index.
ListModel getModel()	It is used to return the data model that holds a list of items displayed by the JList component.
void setListData(Object[] listData)	It is used to create a read-only ListModel from an array of objects.

Multiple-Selection List

- Multiple selection list enables the user to select many items from a JList.
- There are three types of selection modes for a JList in Java
- ListSelectionModel.SINGLE_SELECTION: Only one list index can be selected at a time.
- ListSelectionModel.SINGLE_INTERVAL_SELECTION: Only one contiguous interval can be selected at a time.
- ListSelectionModel.MULTIPLE_INTERVAL_SELECTION: In this mode, there is no restriction on what can be selected. This is a default mode.

Methods Used by JMenuBar

- The JMenuBar class is used to display menubar on the window or frame. It may have several menus.

- The object of JMenu class is a pull down menu component which is displayed from the menu bar. It inherits the JMenuItem class.

- The object of JMenuItem class adds a simple labeled menu item. The items used in a menu must belong to the JMenuItem or any of its subclass.

Layout Managers in Java

- The LayoutManagers are used to arrange components in a particular manner.
- The Java LayoutManagers facilitates us to control the positioning and size of the components in GUI forms.
- LayoutManager is an interface that is implemented by all the classes of layout managers. There are the following classes that represent the layout managers.

AWT has five layout managers:

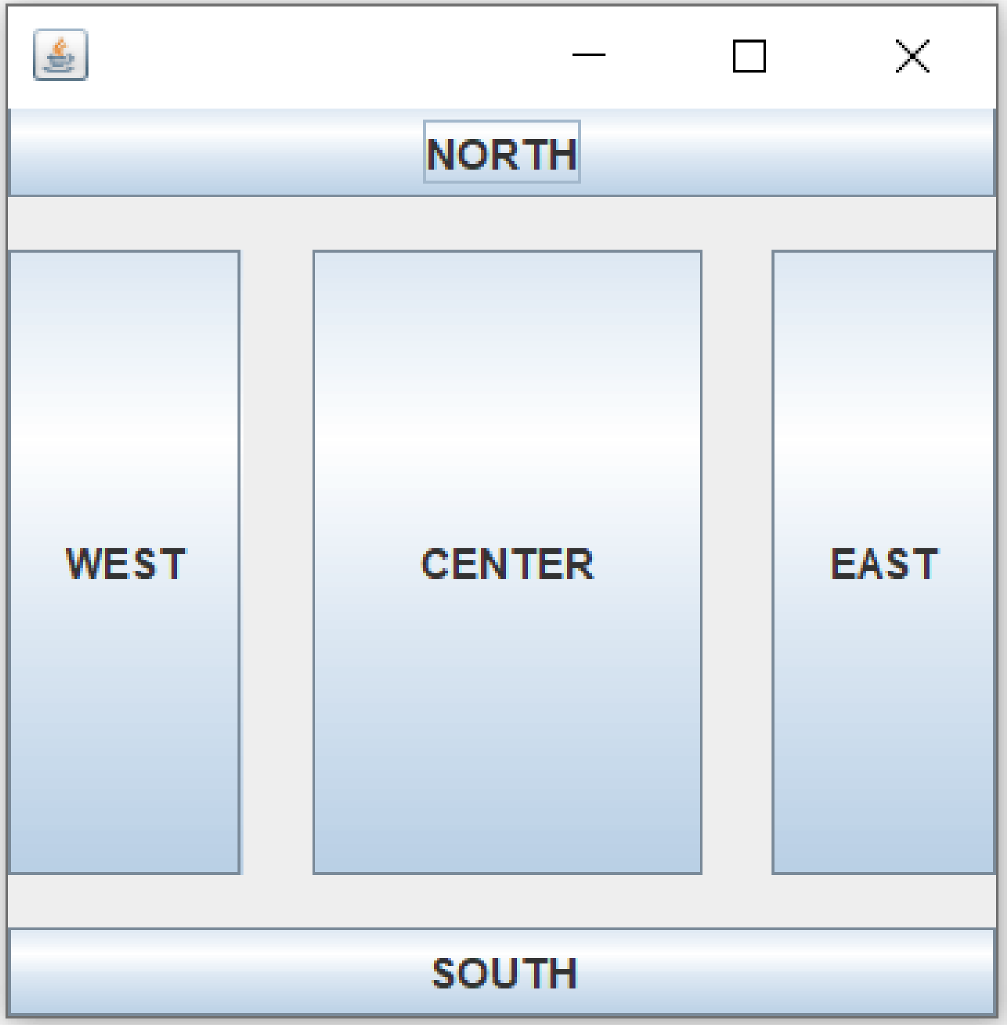
- java.awt.BorderLayout
- java.awt.FlowLayout
- java.awt.GridLayout
- java.awt.CardLayout
- java.awt.GridBagLayout

Swing added the following layout managers:

- javax.swing.BoxLayout
- javax.swing.GroupLayout
- javax.swing.ScrollPaneLayout
- javax.swing.SpringLayout etc.

Border Layout

- The BorderLayout is used to arrange the components in five regions: north, south, east, west, and center.
- Each region (area) may contain one component only. It is the default layout of a frame or window. The BorderLayout provides five constants for each region:
- public static final int NORTH
- public static final int SOUTH
- public static final int EAST
- public static final int WEST
- public static final int CENTER


NORTH
WEST
CENTER
EAST
SOUTH

FlowLayout

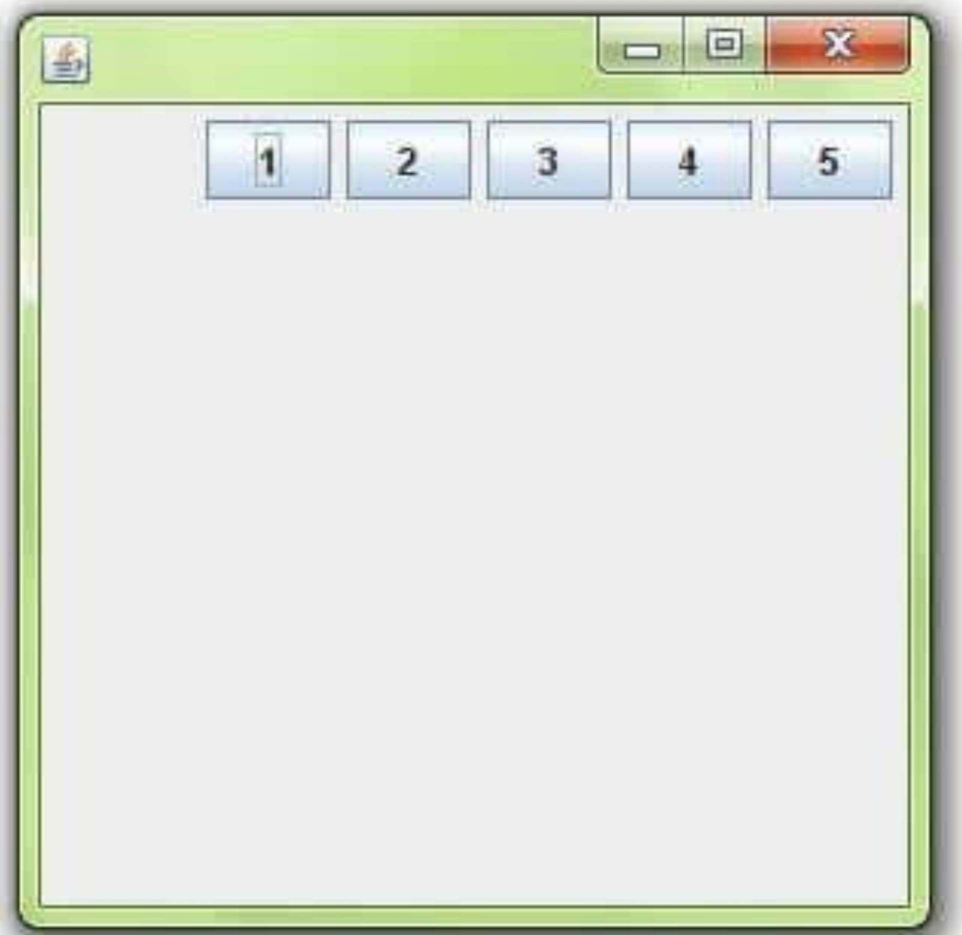
- The Java FlowLayout class is used to arrange the components in a line, one after another (in a flow). It is the default layout of the applet or panel.

Fields of FlowLayout class:

- public static final int LEFT
- public static final int RIGHT
- public static final int CENTER
- public static final int LEADING
- public static final int TRAILING

Constructors of FlowLayout class:

- FlowLayout(): creates a flow layout with centered alignment and a default 5 unit horizontal and vertical gap.
- FlowLayout(int align): creates a flow layout with the given alignment and a default 5 unit horizontal and vertical gap.
- FlowLayout(int align, int hgap, int vgap): creates a flow layout with the given alignment and the given horizontal and vertical gap.

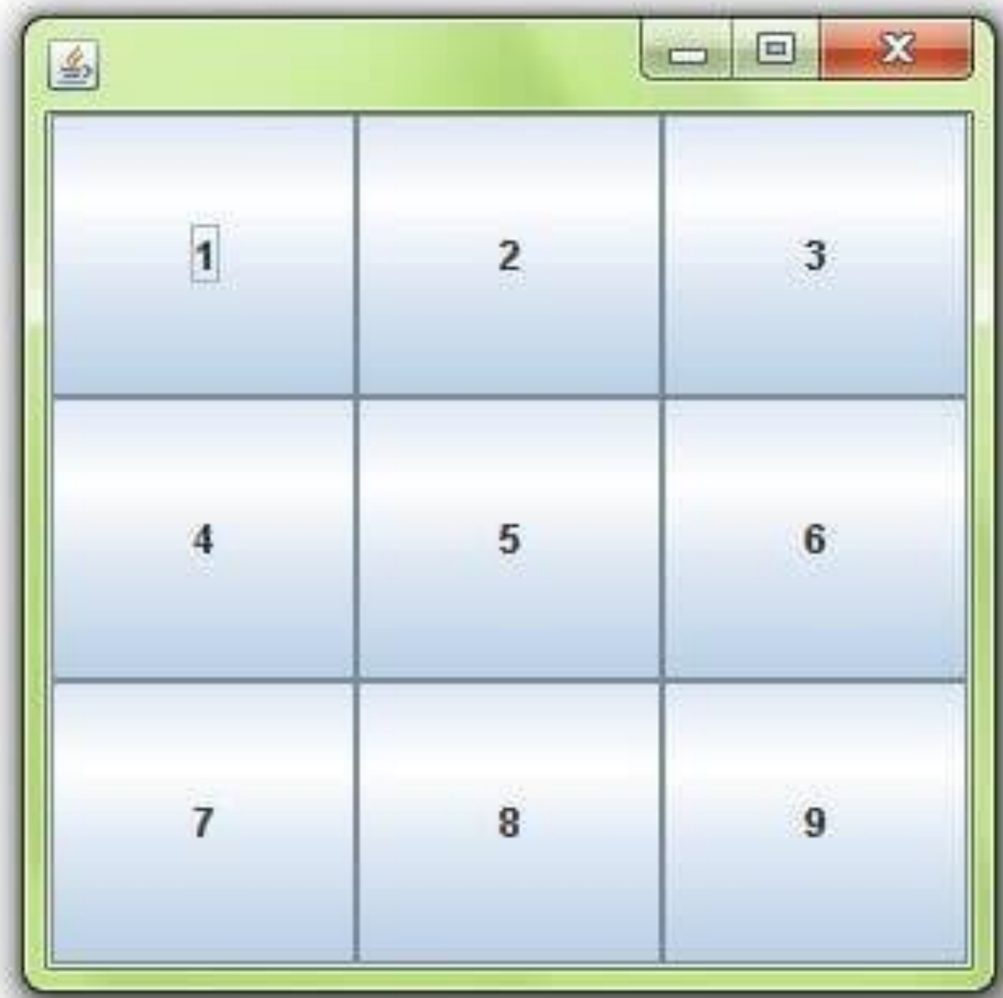

1
2
3
4
5

Grid Layout

- The Java GridLayout class is used to arrange the components in a rectangular grid. One component is displayed in each rectangle.

Constructors of GridLayout class:

- GridLayout(): creates a grid layout with one column per component in a row.
- GridLayout(int rows, int columns): creates a grid layout with the given rows and columns but no gaps between the components.
- GridLayout(int rows, int columns, int hgap, int vgap): creates a grid layout with the given rows and columns along with given horizontal and vertical gaps.


1
2
3
4
5
6
7
8
9

CardLayout

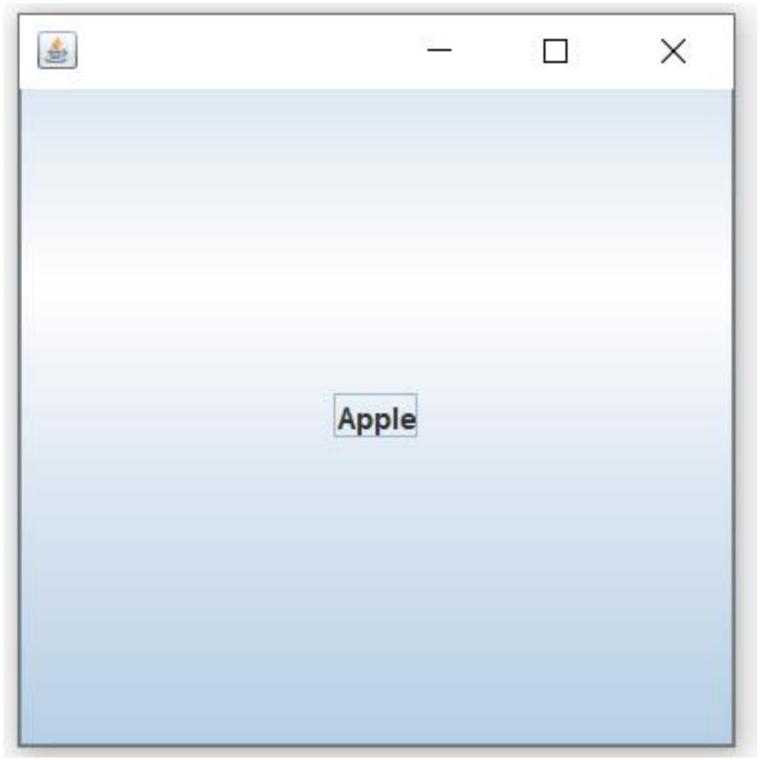
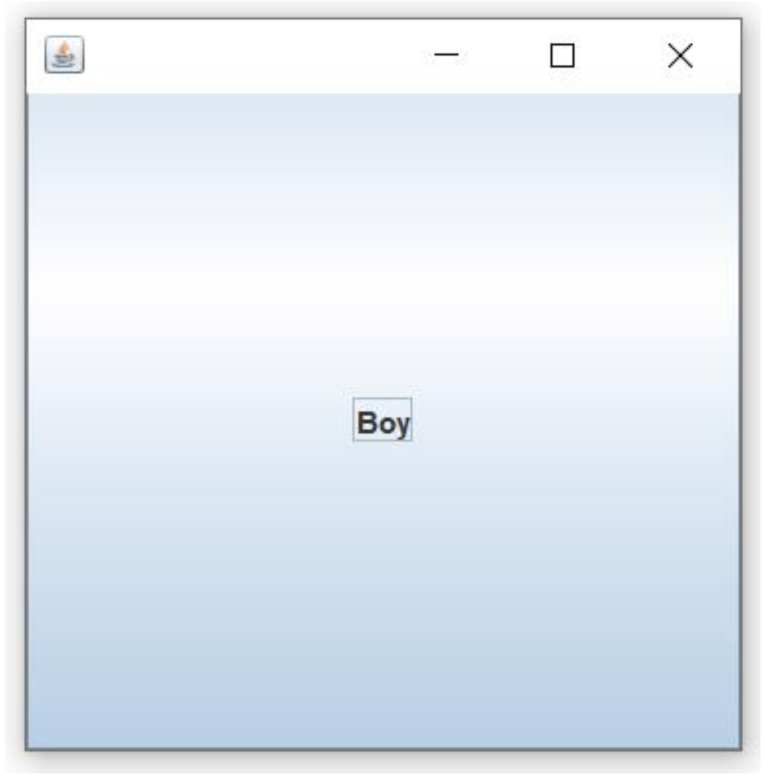
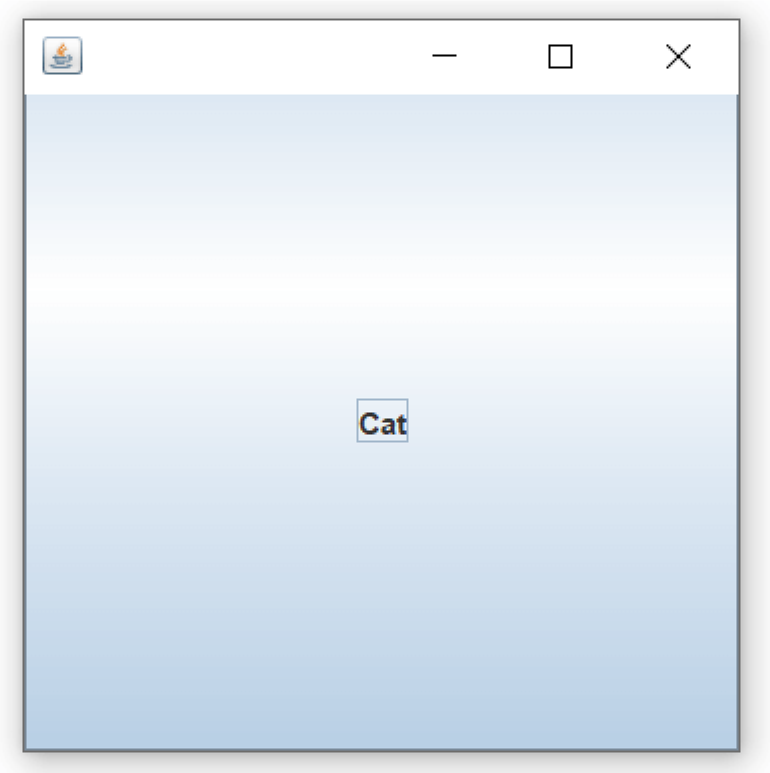
- The Java CardLayout class manages the components in such a manner that only one component is visible at a time. It treats each component as a card that is why it is known as CardLayout.

Constructors of CardLayout Class:

- CardLayout(): creates a card layout with zero horizontal and vertical gap.
- CardLayout(int hgap, int vgap): creates a card layout with the given horizontal and vertical gap.

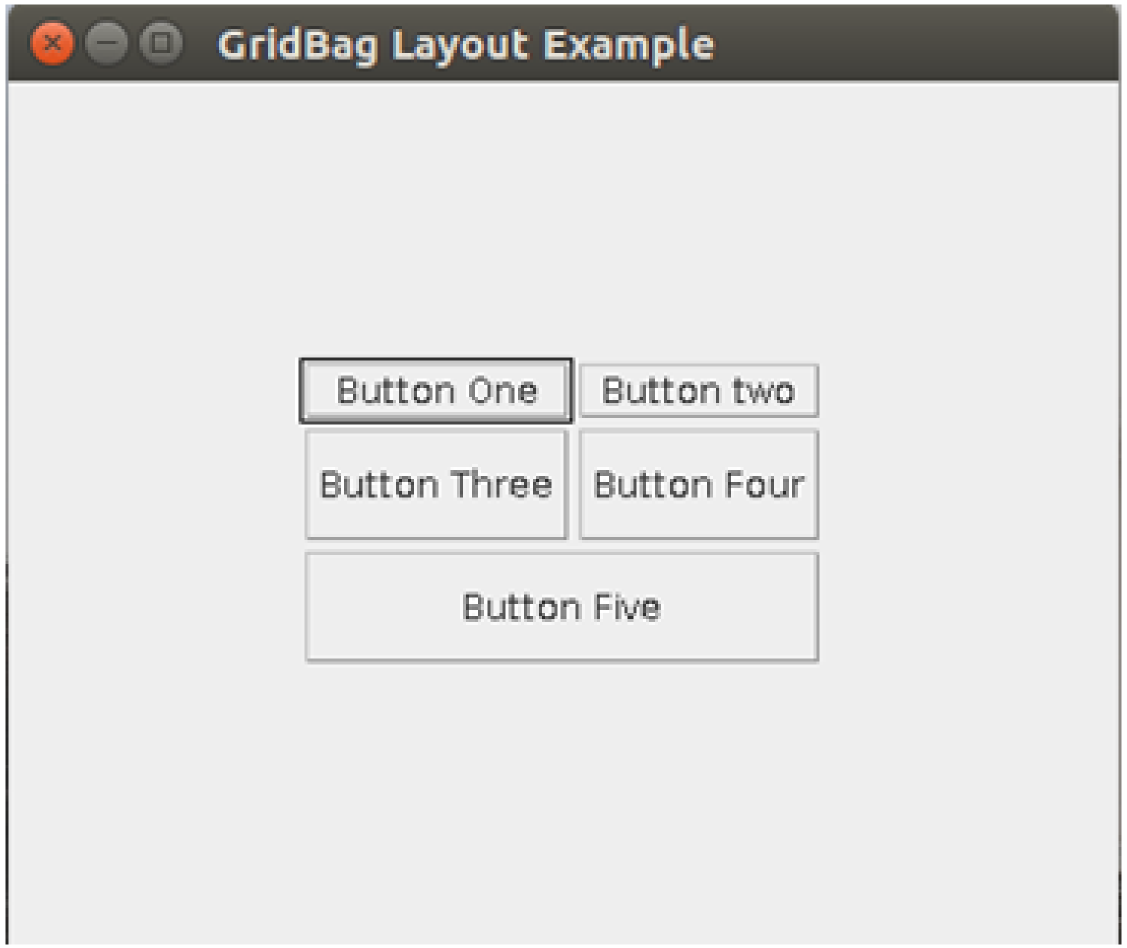
Methods of CardLayout

- public void next(Container parent): is used to flip to the next card of the given container.
- public void previous(Container parent): is used to flip to the previous card of the given container.
- public void first(Container parent): is used to flip to the first card of the given container.
- public void last(Container parent): is used to flip to the last card of the given container.
- public void show(Container parent, String name): is used to flip to the specified card with the given name.


Apple

Boy

Cat

GridBagLayout

- GridBagLayout class is used to align components vertically, horizontally or along their baseline.
- Each GridBagLayout object maintains a dynamic, rectangular grid of cells.
- Each component occupies one or more cells known as its display area.
- Each component associates an instance of GridBagConstraints.
- With the help of the constraints object, we arrange the component's display area on the grid.
- The GridBagLayout manages each component's minimum and preferred sizes in order to determine the component's size.
- GridBagLayout components are also arranged in the rectangular grid but can have many different sizes and can occupy multiple rows or columns

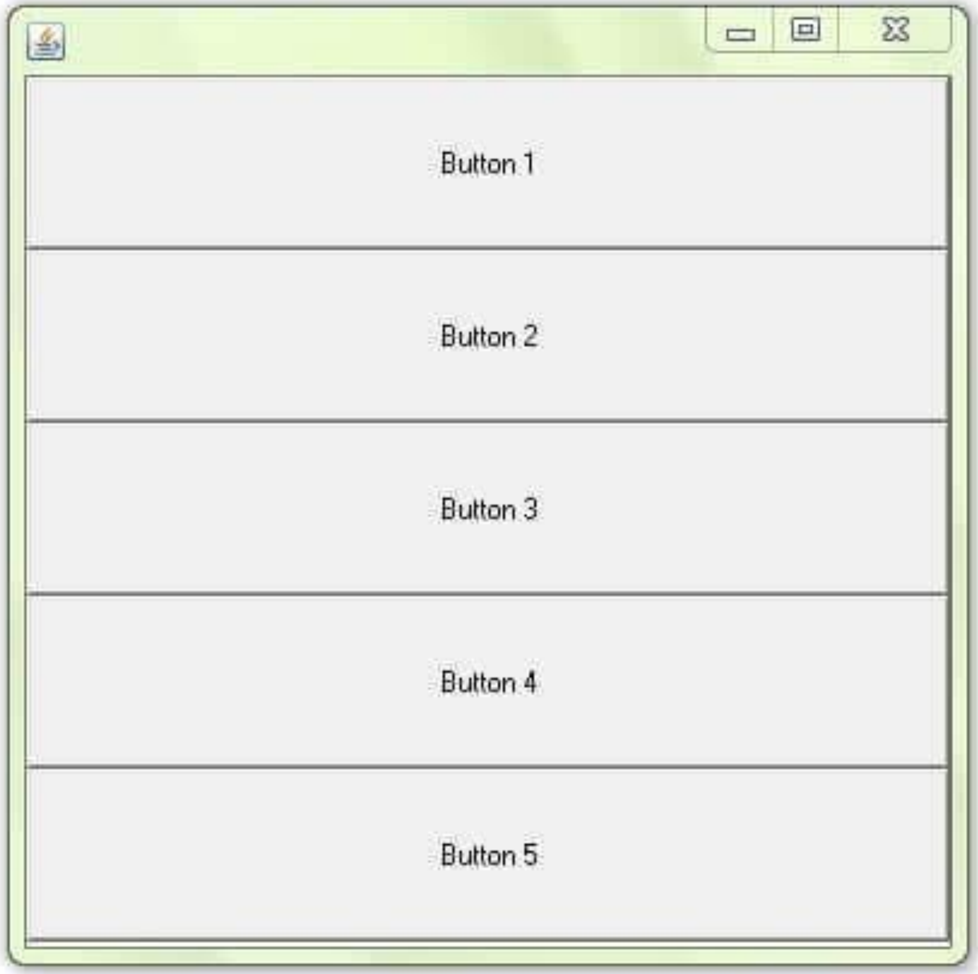

GridBag Layout Example
Button One
Button two
Button Three
Button Four
Button Five

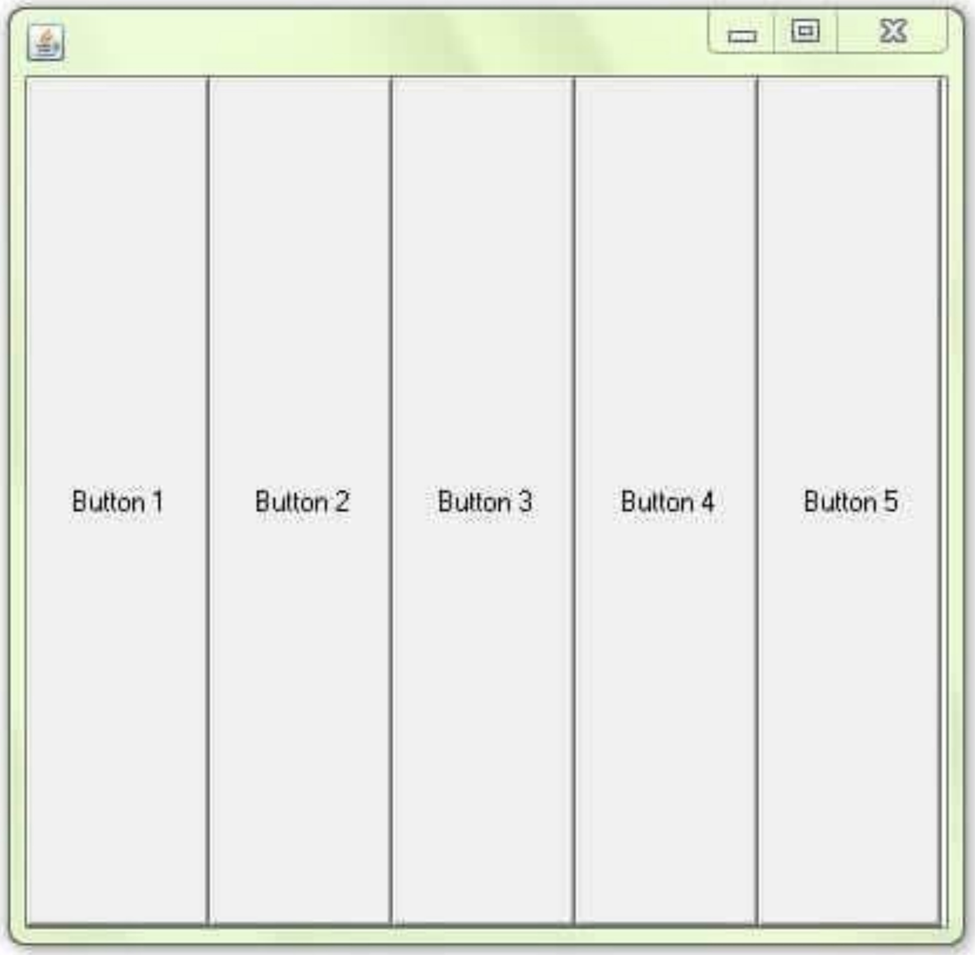
Methods of GridBagLayout

- addLayoutComponent(Component comp, Object constraints)-It adds specified component to the layout, using the specified constraints object.
- ArrangeGrid(Container parent)- This method is obsolete and supplied for backwards compatibility
- getConstraints(Component comp)- It is for getting the constraints for the specified component.
- getLayoutAlignmentX(Container parent) - It returns the alignment along the x axis.
- getLayoutAlignmentY(Container parent) - It returns the alignment along the y axis.

BoxLayout

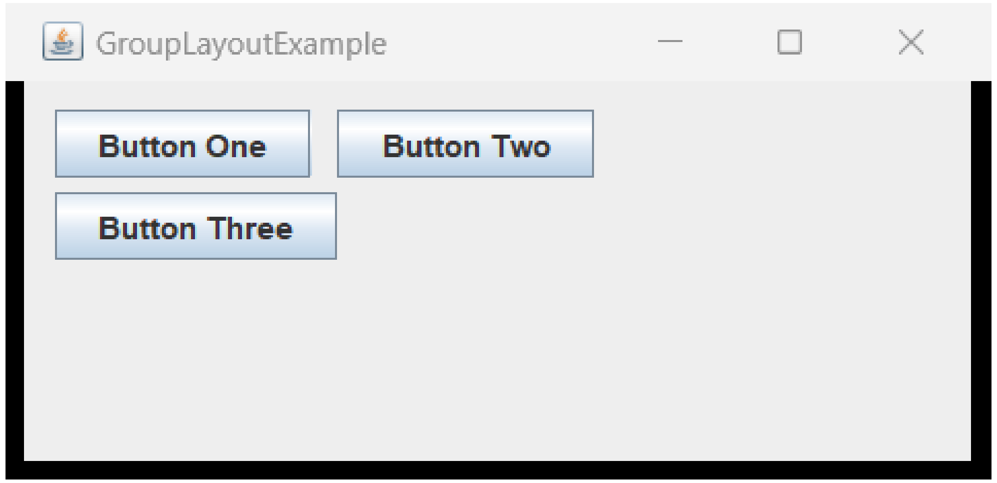
- The Java BoxLayout class is used to arrange the components either vertically or horizontally. For this purpose, the BoxLayout class provides four constants. They are as follows:
- static final int X_AXIS: Alignment of the components are horizontal from left to right.
- static final int Y_AXIS: Alignment of the components are vertical from top to bottom.
- static final int LINE_AXIS: Alignment of the components is similar to the way words are aligned in a line, which is based on the ComponentOrientation property of the container.
- static final int PAGE_AXIS: Alignment of the components is similar to the way text lines are put on a page, which is based on the ComponentOrientation property of the container.


Button 1
Button 2
Button 3
Button 4
Button 5


Button 1
Button 2
Button 3
Button 4
Button 5

GroupLayout

- GroupLayout groups its components and places them in a Container hierarchically. The grouping is done by instances of the Group class.
- Group is an abstract class, and two concrete classes which implement this Group class are SequentialGroup and ParallelGroup.
- SequentialGroup positions its child sequentially one after another whereas ParallelGroup aligns its child on top of each other.
- The GroupLayout class provides methods such as createParallelGroup() and createSequentialGroup() to create groups.
- GroupLayout treats each axis independently. There is a group representing the horizontal axis, and a group representing the vertical axis.
- Each component must exist in both a horizontal and vertical group, otherwise an IllegalStateException is thrown during layout or when the minimum, preferred, or maximum size is requested.


GroupLayoutExample
Button One
Button Two
Button Three

Methods of GroupLayout

- addLayoutComponent(Component component, Object constraints)-It notify that a Component has been added to the parent container.
- createParallelGroup() - It creates and returns a ParallelGroup with an alignment of Alignment.LEADING
- createParallelGroup(GroupLayout.Alignment alignment, boolean resizable) It creates and returns a ParallelGroup with the specified alignment and resize behavior.
- createSequentialGroup() It creates and returns a SequentialGroup.

ScrollPaneLayout

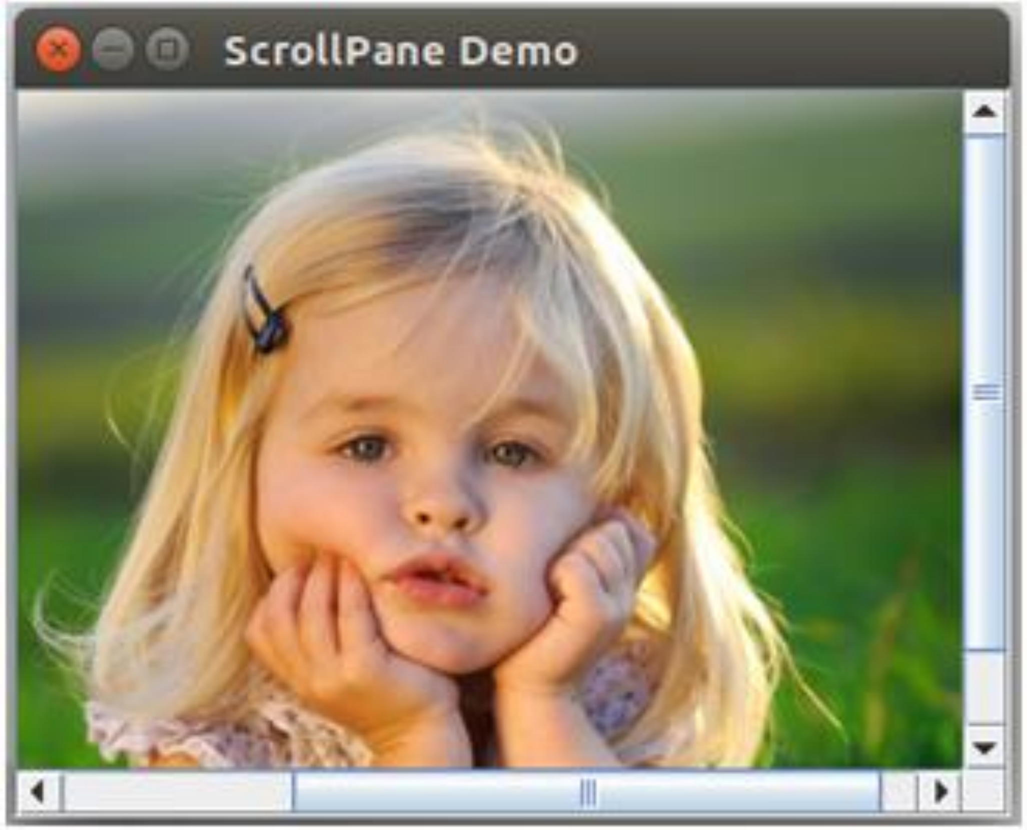
- The layout manager is used by JScrollPane. JScrollPaneLayout is responsible for nine components: a viewport, two scrollbars, a row header, a column header, and four "corner" components.

Constructor:

- ScrollPaneLayout(): The parameterless constructor is used to create a new ScrollPanelLayout.

Methods of ScrollPaneLayout

- addLayoutComponent(String s, Component c) - It adds the specified component to the layout.
- getColumnHeader()-It returns the JViewport object that is the column header.
- getCorner(String key)-It returns the Component at the specified corner.
- getHorizontalScrollBar()- It returns the JScrollBar object that handles horizontal scrolling.
- getViewport() - It returns the JViewport object that displays the scrollable contents.

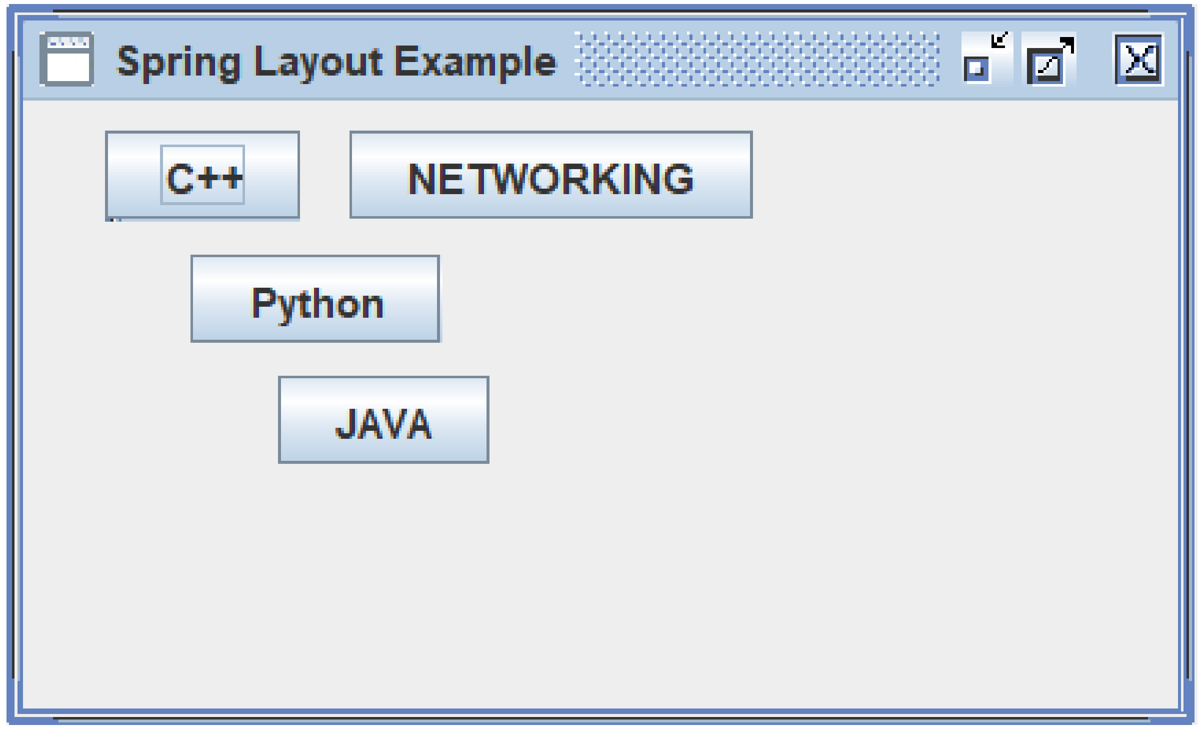

ScrollPane Demo

SpringLayout

- A SpringLayout arranges the children of its associated container according to a set of constraints. Constraints are nothing but horizontal and vertical distance between two-component edges.
- Each edge position is dependent on the position of the other edge. If a constraint is added to create a new edge, than the previous binding is discarded. SpringLayout doesn't automatically set the location of the components it manages.
- **Constructor**
- SpringLayout(): The default constructor of the class is used to instantiate the SpringLayout class.

Methods of SpringLayout

- getConstraint(String edgeName, Component c) - It returns the spring controlling the distance between the specified edge of the component and the top or left edge of its parent.
- maximumLayoutSize(Container parent) - It is used to calculates the maximum size dimensions for the specified container, given the components it contains.
- minimumLayoutSize(Container parent) - It is used to calculates the minimum size dimensions for the specified container, given the components it contains.
- preferredLayoutSize(Container parent)- It is used to calculates the preferred size dimensions for the specified container, given the components it contains.


Spring Layout Example
C++
NETWORKING
Python
JAVA

JLayeredPane

- JLayeredPane is a Swing component that enables us to add components to a different layer.
- It adds depth to swing container. It is used to provide a third dimension for positioning component and divide the depth-range into several different layers.

Layers of JLayeredPane

- Here are some layers of JLayeredPane as explained in detail:

- DEFAULT_LAYER: This is the standard and bottommost layer, where most components are inserted.
- PALETTE_LAYER: This layer of JLayeredPane sits over the default layer and useful for floating toolbars and palettes.
- MODAL_LAYER: This layer of JLayeredPane is used for model dialogs and shown above the palette layer.
- POPUP_LAYER: This layer of JLayeredPane is used to show popup windows above the modal layer.
- DRAG_LAYER: A component is shown in this layer (above the popup layer) while dragging it. When the dragging is finished, the component is shown in its original layer.

JDesktopPane

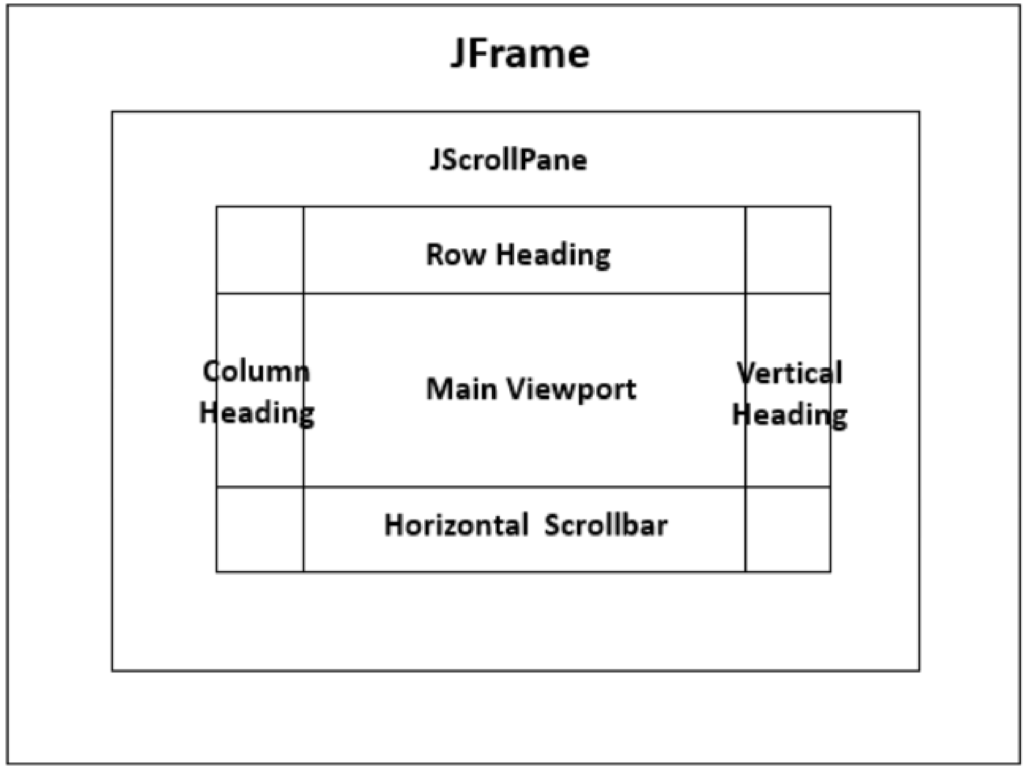
- It used for generating multi-documented interfaces, which can hold many windows or application pages.
- It is achieved by making use of the contentPane and JInternalFrame from the main window and internal window respectively.
- **Constructor:**
- It has only one constructor JDesktopPane() that helps in creating a new JDesktopPane.

Methods Used by JDesktopPane

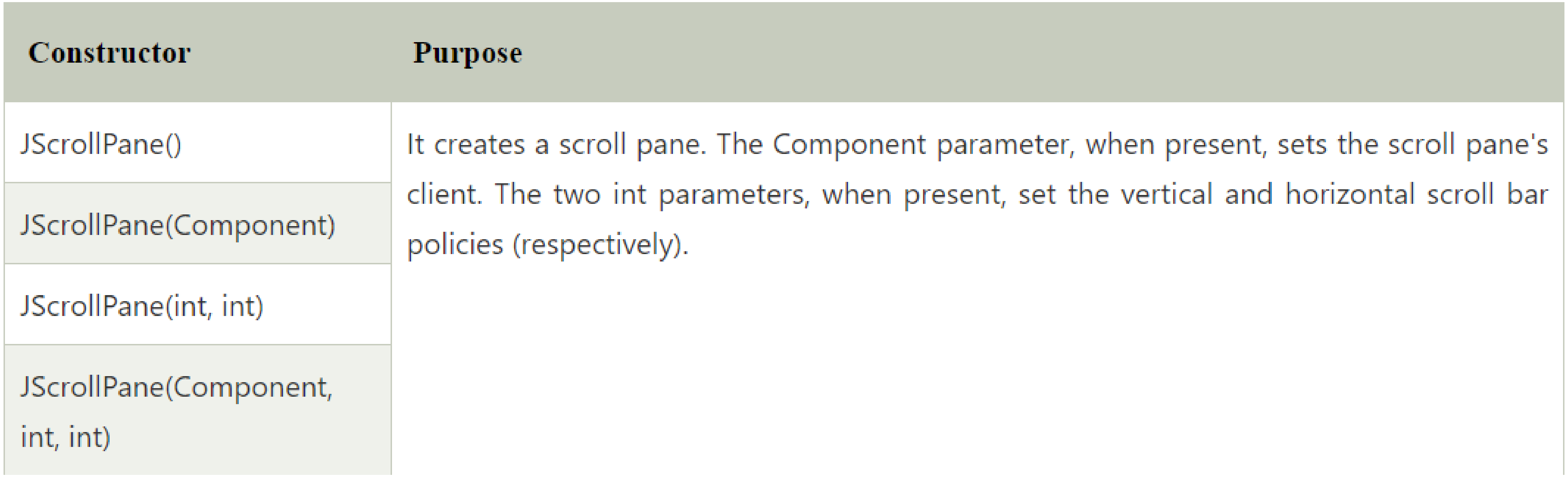
- getAllFrames(): JInternalFrames that is presently displayed in the desktop will be returned.
- getAllFramesInLayer(int l): JInternalFrames that is presently displayed in the desktop on the specified layer l will be returned.
- remove(int i): Indexed component i will be removed from this pane.
- removeAll(): Each and every component from the container will be removed.
- getDragMode(): The style of dragging will be returned which is currently used by the desktop pane.
- getSelectedFrame(): JInternalFrame will be returned which is currently active in the desktop pane. If no active JInternalFrame is present, then null will be returned.

JScrollPane

- JScrollPane is used to give a scrollable view to your component. When the screen size is small or limited, we can use a scroll pane to showcase a large component or a component whose size changes dynamically.
- we need to use a scroll pane for the following two conditions:
- To display a large component.
- To display a dynamic size changeable component.

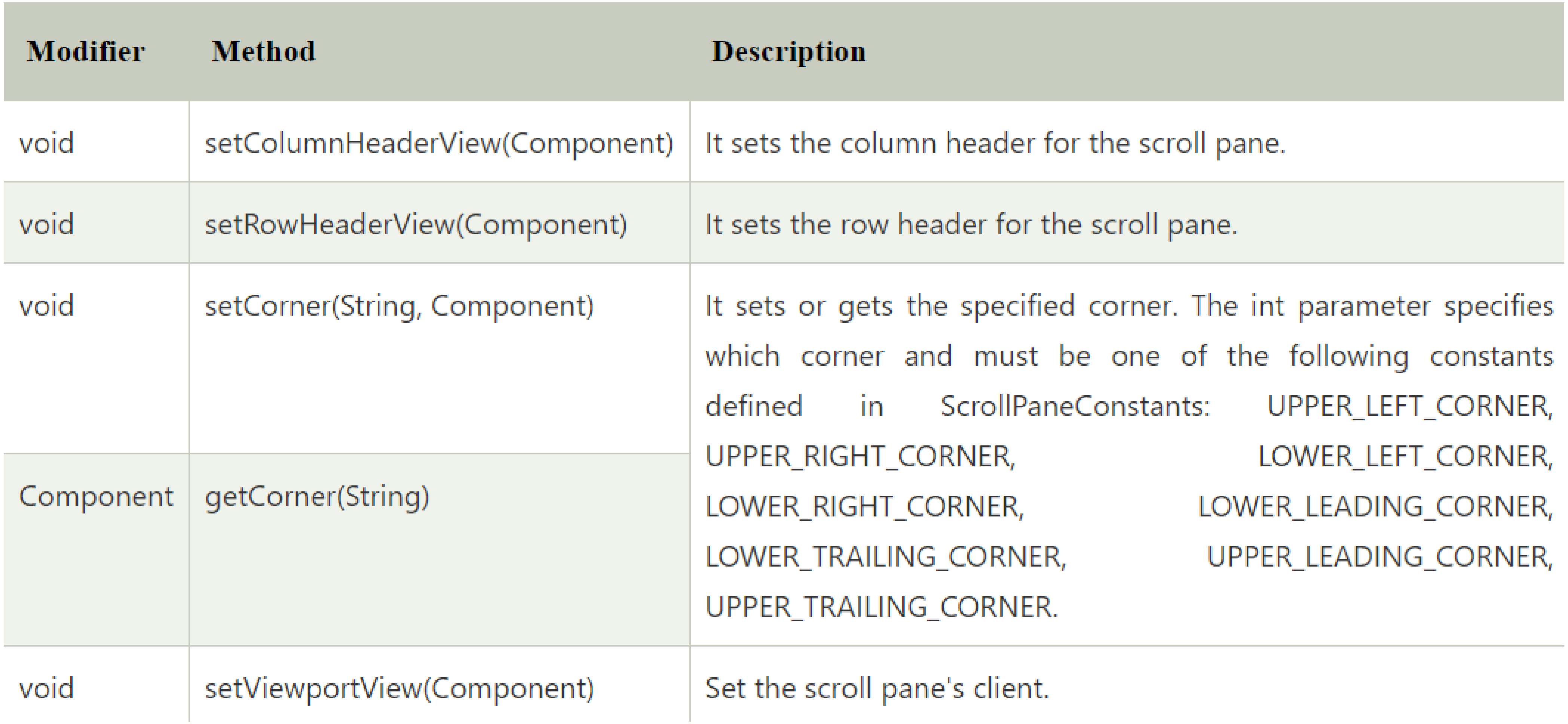


Constructors Used by JScrollPane



Constructor	Purpose
JScrollPane()	It creates a scroll pane. The Component parameter, when present, sets the scroll pane's client. The two int parameters, when present, set the vertical and horizontal scroll bar policies (respectively).
JScrollPane(Component)	
JScrollPane(int, int)	
JScrollPane(Component, int, int)	

Methods Used by JScrollPane



Modifier	Method	Description
void	setColumnHeaderView(Component)	It sets the column header for the scroll pane.
void	setRowHeaderView(Component)	It sets the row header for the scroll pane.
void	setCorner(String, Component)	It sets or gets the specified corner. The int parameter specifies which corner and must be one of the following constants defined in ScrollPaneConstants: UPPER_LEFT_CORNER, UPPER_RIGHT_CORNER, LOWER_LEFT_CORNER, LOWER_RIGHT_CORNER, LOWER_LEADING_CORNER, LOWER_TRAILING_CORNER, UPPER_LEADING_CORNER, UPPER_TRAILING_CORNER.
Component	getCorner(String)	
void	setViewportView(Component)	Set the scroll pane's client.

Difference Between AWT and Swing

No.	Java AWT	Java Swing
1	AWT components are platform-dependent.	AWT components are platform-dependent.
2	AWT components are heavyweight	Swing components are lightweight.
3	AWT doesn't support pluggable look and feel.	Swing supports pluggable look and feel
4	AWT provides less components than Swing.	Swing provides more powerful components such as tables, lists, scrollpanes, colorchooser, tabbedpane etc
5	AWT doesn't follows MVC(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing follows MVC.

Thank You ！