Continue

Home Page > Creating a GUI with JFC/Swing > Laying Out Components Within a Container How to Use GridLayout Here's a snapshot of an application that uses a GridLayout. You can run GridLayoutDemo using JavaTM Web Start. Its code is in GridLayoutDemo.java. A GridLayout places components in a grid of cells. Each component takes all the available space within its cell, and each cell is exactly the same size. If you resize the GridLayoutDemo window, you'll see that the GridLayout changes the cell size so that the cells are as large as possible, given the space available to the container. Below is the code that creates the GridLayout and the components it manages. You can find the whole program in GridLayoutDemo.java. pane.setLayout(new GridLayout(0,2)); pane.add(new JButton("Button 1")); pane.add(new JButton("Button 2")); pane.add(new JButton("Button 3")); pane.add(new JButton("Long-Named Button 4")); pane.add(new JButton("5")); The constructor tells the GridLayout class to create an instance that has two columns and as many rows as necessary. [PENDING: This section will be converted to present its information in an API table.] The GridLayout class has two constructors: public GridLayout(int rows, int columns) public GridLayout(int rows, int columns, int horizontalGap, int verticalGap) At least one of the rows and columns arguments must be nonzero; the rows argument has precedence over the columns argument. The horizontalGap and verticalGap arguments to the second constructor allow you to specify the number of pixels between cells. If you don't specify gaps, their values default to zero. The following table lists some of the examples that use grid layout. In Swing, in order to arrange components in a form, dialog box etc. in user friendly manner layout manager is found to be very useful. There are several layout managers. GridLayout is such layout manager.1. IntroductionGridLayout actually forms a grid like arrangement of cells. This is just like one excel spreadsheet where each cell is of same size. If the parent window is resized, the grids are also resized, keeping the same aspect ratio. Components are required to be added in each cell.2. Technologies UsedJava (jdk 1.6.x or higher will be fine)Eclipse (Galileo or higher version is required)3. OverviewLike other layout manager classes, java.awt.GridLayout class implements java.awt.LayoutManager.At the time GridLayout object is created, number of rows and number of columns must be mentioned. For example GridLayout layout = new GridLayout(2,3). Here 2 refers to number of rows and 3 refers to number of columns. The grid mentioned above will have 6 cells in it, 3 in each row.While adding components in cells, if the cell no is not specified, the components will be added to the cells starting from upper leftmost cell to lower rightmost cell, in this direction i.e. addition will start from the leftmost cell of the topmost row moving rightwards, and then comes down to the next row (if available) and gets filled up in the same way.If component is required to be added in a specific cell, then row number and column number is required to be specified while adding the component. For example 0,0 cell number refers to the leftmost cell of the first i.e. topmost row.4. Description of GridLayout feature in the exampleIn the example a GridLayout of 2 rows and 3 columns are created. In all except 5th cell, JEditPane is attached. In the 5th cell, one JSplitPane component is added. In the right side of the JSplitPane component, one JList component is added which shows cell numbers. In the right side of the JSplitPane, one JFileChooser is added to show only .txt files in system drive.JTextPane and JSplitPane components added to a 2X3 grid layout.Once a text file is chosen, keeping 1st Box option selected from the JList, after clicking the Open button of the JFileChooser component, content of the text file appears in cell 1.Content of .txt file appears in cell-1 of the grid.Once a text file is chosen, keeping 6th Box option selected from the JList, after clicking the Open button of the JFileChooser component, content of the text file appears in cell 6.Content of the .txt file appears in cell-6 of the grid.If a file is opened, without selecting any option in JList, error message is generated.Error message shows that no option is selected from the list.5. Description of GridLayout feature in the source codeHere first of all one JPanel object is created for GridLayout of dimension 2X3 and then components are added to the grid in turn. To the JFileChooser object, ActionListner is assigned which is taking care of the click to Open button of the JFileChooser component. As per the choice of the JList component, the content of the file is shown in the corresponding cell of the grid.SwingGridLayoutExampleFrame.javapackage com.javacodegeeks.example.swing.layoutmanager.gridlayout; import java.awt.GridLayout; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.io.BufferedReader; import java.io.File; import java.io.FileNotFoundException; import java.io.FileReader; import java.io.IOException; import javax.swing.JButton; import javax.swing.JEditorPane; import javax.swing.JFileChooser; import javax.swing.JFrame; import javax.swing.JList; import javax.swing.JOptionPane; import javax.swing.JPanel; import javax.swing.JScrollPane; import javax.swing.JSplitPane; import javax.swing.JTextField; import javax.swing.filechooser.FileNameExtensionFilter; public class SwingGridLayoutExampleFrame extends JFrame{ /** * */ private static final long serialVersionUID = 8008949174170019398L; public SwingGridLayoutExampleFrame(){ JPanel panel = new JPanel(new GridLayout(2,3)); final JEditorPane textField1 = new JEditorPane(); JScrollPane scrollPane1 = new JScrollPane(textField1); panel.add(scrollPane1,0,0); final JEditorPane textField2 = new JEditorPane(); JScrollPane scrollPane2 = new JScrollPane(textField2); panel.add(scrollPane2,0,1); final JEditorPane textField3 = new JEditorPane(); JScrollPane scrollPane3 = new JScrollPane(textField3); panel.add(scrollPane3); final JEditorPane textField4 = new JEditorPane(); JScrollPane scrollPane4 = new JScrollPane(textField4); panel.add(scrollPane4); final JEditorPane textField5 = new JEditorPane(); JSplitPane splitPane = new JSplitPane(); splitPane.setOrientation(JSplitPane.HORIZONTAL_SPLIT); splitPane.setDividerLocation(50); final JList list = new JList(new String[]{"1st Box","2nd Box","3rd Box","4th Box","6th Box"}); splitPane.setLeftComponent(list); final JFileChooser fileChooser = new JFileChooser(); fileChooser.addActionListener(new ActionListener() { @Override public void actionPerformed(ActionEvent evt) { if (evt.getActionCommand().equals(javax.swing.JFileChooser.APPROVE_SELECTION)) { String selected = list.getSelectedValue() != null ? list.getSelectedValue().toString() : null; if(selected == null) JOptionPane.showMessageDialog(fileChooser, "You must choose one item from the list to proceed."); else{ File file = fileChooser.getSelectedFile(); if(file.getName().endsWith(".txt")){ char[] content = readFile(file); if(content == null){ JOptionPane.showMessageDialog(fileChooser, "File size too large."); }else if(content.length == 0){ JOptionPane.showMessageDialog(fileChooser, "Empty file."); }else{ switch(selected.charAt(0)){ case '1': textField1.setText(new String(content)); break; case '2': textField2.setText(new String(content)); break; case '3': textField3.setText(new String(content)); break; case '4': textField4.setText(new String(content)); break; case '6': textField5.setText(new String(content)); break; } } } } } }); FileNameExtensionFilter filter = new FileNameExtensionFilter("TEXT FILES", "txt", "text"); fileChooser.setFileFilter(filter); splitPane.setRightComponent(fileChooser); panel.add(splitPane); JScrollPane scrollPane5 = new JScrollPane(textField5); panel.add(scrollPane5); add(panel); pack(); } private char[] readFile(File inputFile){ BufferedReader inputReader = null; char[] content = null; long availableHeap = Runtime.getRuntime().freeMemory(); long fileSize = inputFile.length(); try { if(fileSize