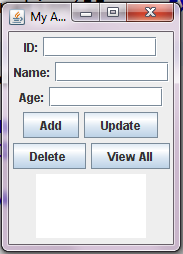
**PRATICE AND undERSTAND THE FOLLOWING EXAMPLES FIRST.**

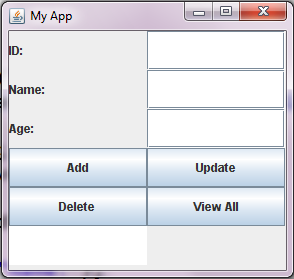
# Example of flow layout



|  |
| --- |
| import java.awt.FlowLayout;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.\*;  public class MyApp {    JTextField t1;  JTextField t2;  JTextField t3;  JTextArea ta1;  void buildApp()  {    JFrame obj=new JFrame("My App");  obj.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  obj.setVisible(true);  obj.setSize(200, 250);  obj.setLocationRelativeTo(null);    JLabel l2=new JLabel("Name:");  JLabel l1=new JLabel("ID:");  JLabel l3=new JLabel("Age:");  t1=new JTextField(10);  t2=new JTextField(10);  t3=new JTextField(10);  ta1=new JTextArea(4,10);  JButton btnAdd=new JButton("Add");  JButton btnDelete=new JButton("Delete ");  JButton btnUpdate=new JButton("Update");  JButton btnShow=new JButton("View All");    obj.add(l1);  obj.add(t1);  obj.add(l2);  obj.add(t2);  obj.add(l3);  obj.add(t3);  obj.add(btnAdd);  obj.add(btnUpdate);  obj.add(btnDelete);  obj.add(btnShow);  obj.add(ta1);  //obj.setResizable(false);  **obj.setLayout(new FlowLayout());**  }  public static void main(String[] args) {  MyApp o=new MyApp();  o.buildApp();  }  } |

# Simple Example of Grid layout

In gridLayout your controls are arranged in a grid (like table with rows and columns). In contructor you specify no. of columns and no. of rows.So, here I have replaced flowLayout with gridlaout, you can see line of code heighted in RED. My grid has 2 columns and 6 rows., and controls are arranged in those automatically in order.

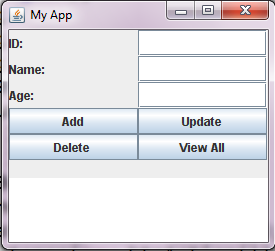


|  |
| --- |
| import java.awt.FlowLayout;  import java.awt.GridLayout;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.\*;  public class MyApp {  JTextField t1;  JTextField t2;  JTextField t3;  JTextArea ta1;  void buildApp()  {  JFrame obj=new JFrame("My App");  obj.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  obj.setVisible(true);  obj.setSize(200, 250);  obj.setLocationRelativeTo(null);  **obj.setLayout(new GridLayout(6,2));**  JLabel l2=new JLabel("Name:");  JLabel l1=new JLabel("ID:");  JLabel l3=new JLabel("Age:");  t1=new JTextField(10);  t2=new JTextField(10);  t3=new JTextField(10);  ta1=new JTextArea(4,10);  JButton btnAdd=new JButton("Add");  JButton btnDelete=new JButton("Delete ");  JButton btnUpdate=new JButton("Update");  JButton btnShow=new JButton("View All");  obj.add(l1);  obj.add(t1);  obj.add(l2);  obj.add(t2);  obj.add(l3);  obj.add(t3);  obj.add(btnAdd);  obj.add(btnUpdate);  obj.add(btnDelete);  obj.add(btnShow);  obj.add(ta1);  }  public static void main(String[] args) {  MyApp o=new MyApp();  o.buildApp();}  } |

# BorderLayout, GridLayout and JPanels

We have discussed one container today i.e. JFrame. Here is another container JPanel. I have created 2 JPanels ( added my controls to JPanles) and then added those JPanels to my JFrame. (highlighted in red). However, I have set layout of my JFrame as Border layout (highlighted in green).

|  |
| --- |
| Border Layout Notes In java.awt.BorderLayout, the container is divided into 5 zones: EAST, WEST, SOUTH, NORTH, and CENTER.   * Components are added using methodaContainer.add(acomponent, aZone), where azone is either BorderLayout.NORTH , BorderLayout.SOUTH, BorderLayout.WEST , BorderLayout.EAST or BorderLayout.CENTER. * The method aContainer.add(aComponent) without specifying the zone adds the component to the CENTER. * You need not add components to all the 5 zones. The NORTH and SOUTH components may be stretched horizontally; the EAST and WEST components may be stretched vertically; the CENTER component may stretch both horizontally and vertically to fill any space left over. |

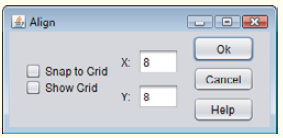


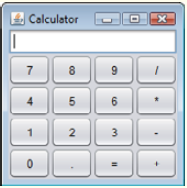
|  |
| --- |
| import java.awt.BorderLayout;  import java.awt.FlowLayout;  import java.awt.GridLayout;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.\*;  public class MyApp {  JTextField t1;  JTextField t2;  JTextField t3;  JTextArea ta1;  void buildApp()  {  JFrame obj=new JFrame("My App");  obj.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  obj.setVisible(true);  obj.setSize(200, 250);  obj.setLocationRelativeTo(null);  **JPanel p1=new JPanel(new GridLayout(5,2));**  **JPanel p2=new JPanel(new BorderLayout());**  **obj.setLayout(new BorderLayout());**  JLabel l2=new JLabel("Name:");  JLabel l1=new JLabel("ID:");  JLabel l3=new JLabel("Age:");  t1=new JTextField(10);  t2=new JTextField(10);  t3=new JTextField(10);  ta1=new JTextArea(4,10);  JButton btnAdd=new JButton("Add");  JButton btnDelete=new JButton("Delete ");  JButton btnUpdate=new JButton("Update");  JButton btnShow=new JButton("View All");  p1.add(l1);  p1.add(t1);  p1.add(l2);  p1.add(t2);  p1.add(l3);  p1.add(t3);  p1.add(btnAdd);  p1.add(btnUpdate);  p1.add(btnDelete);  p1.add(btnShow);  **obj.add(p1, BorderLayout.*NORTH*);**  p2.add(ta1,BorderLayout.*CENTER* );  **obj.add(p2, BorderLayout.*SOUTH* );**  }  public static void main(String[] args) {    MyApp o=new MyApp();  o.buildApp();  }  } |

**Problem 1:**

1. Create the following GUIs. You don’t need to provide functionality just create the interface.

**Note: don’t use drag and drop to create this gui. Do through writing code.**

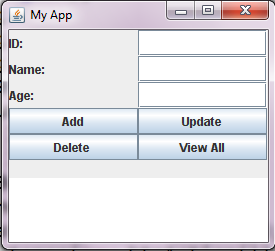






**Problem 2:**

**Perform Event handling for the following form (***code for this form is given in previous pages***)**



Handle click event for the buttons:

**>> Add Button**

On click of add button Add the record to Student table in DB.

***Hint:*** *use Textfields ‘s getText() method to get values entered in textbox*

**>> View All**

On click of view all button, display the existing students record in text area attached in the bottom of the form.

***Hint:*** *use Textfields ‘s setText() to display text in textArea*

**>> Add Button**

On click of delete button delete the record from Student table in DB, on basis of Id only.

**>> Update Button**

On click of update button update the name and gpa of Student table in DB, on basis of Id.