Pemrograman Berorientasi Objek Lanjut
Lecture 4: Layout Manager

NIKO IBRAHIM, MIT
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN MARANATHA

Review

- Menu memiliki 3 komponen?
  - JMenu, JMenuItem, & JMenuBar
- Akses menu melalui keyboard?
  - Mnemonics (alt + karakter)
  - Accelerator (ctrl + karakter)
- Perbedaan Pop up Menu dengan Menu biasa?
  - Pop up menu tidak di-attach ke JMenuBar
- ScrollBars memiliki 3 display policy?
  - Needed, Always, Never
- Dialog dibuat dari JOptionPane dengan cara memanggil static methods-nya. Ada 4 variasi method ini:
  - JOptionPane.showMessageDialog
  - JOptionPane.showConfirmDialog
  - JOptionPane.showInputDialog
  - JOptionPane.showOptionDialog

Materi Hari Ini

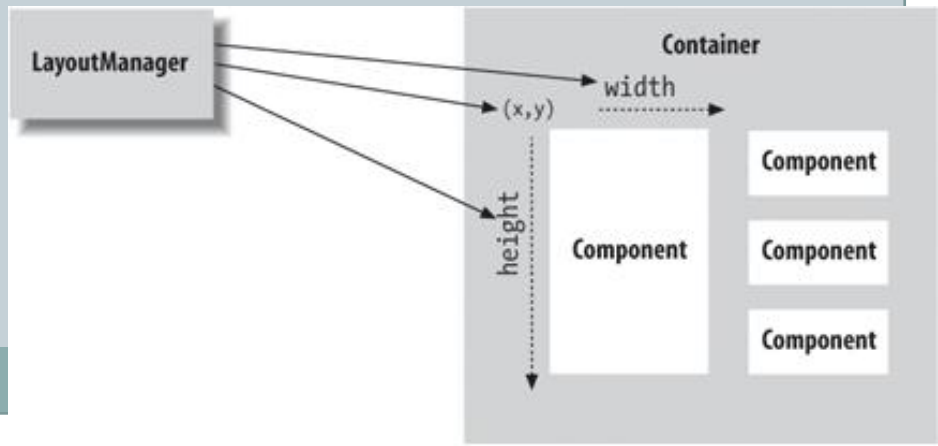
- Layout Manager
- Standard layout manager
- Latihan

Problems with Swing

- Selama ini, pada saat kita merancang GUI untuk program Java, salah satu halangannya adalah bahwa komponen yang digunakan dapat berpindah-pindah lokasi. Hal ini terjadi apabila kita me-resize frame utama.
- Komponen dapat diorganisasikan di dalam container dengan menggunakan class "layout manager".
- Kita dapat mengubah secara manual class layout yang ingin digunakan, dan setiap container dapat memiliki layout manager masing-masing.

Layout Manager

- Seperti pada gambar, layout manager bertugas menyusun komponen-komponen (button, label, checkbox, dll) di dalam suatu container (panel, frame, dll).
- Layout manager menentukan posisi dan ukuran setiap komponen di dalam container. Proses ini akan berbeda untuk setiap class layout yang digunakan.
- AWT dan Swing memiliki beberapa layout manager standar yang penggunaannya seringkali dikombinasikan sesuai situasi dan kebutuhan kita.



Standard Layout Manager

- FlowLayout
- GridLayout
- BorderLayout
- BoxLayout
- CardLayout
- GridBagLayout
- SpringLayout
- GroupLayout

Mengubah Default Layout Manager

- Setiap container memiliki default layout manager.
- Pada saat kita membuat sebuah container (misal: panel, frame, tabbed pane, split pane, dll), maka container tsb memiliki objek LayoutManager masing-masing.
- Kita dapat mengubah layout manager default tersebut dengan suatu layout yang baru dengan menggunakan method "setLayout()"
- Contoh:
  - Default layout manager untuk JFrame adalah: FlowLayout
  - Kita dapat mengubah layout tersebut dengan cara misalnya:

```
myFrame.setLayout(new BorderLayout());
```

1. FlowLayout

- FlowLayout merupakan layout manager yang simpel.
- FlowLayout menyusun komponen berdasarkan ukuran default masing-masing, dengan posisi mulai dari kiri ke kanan dan dari atas ke bawah di dalam container yang digunakan.
- FlowLayout dapat memiliki "row justification": LEFT, CENTER, atau RIGHT serta "padding" horizontal/vertical.
- Secara default, flow layout menggunakan justification CENTER. Artinya, semua komponen akan disimpan di posisi tengah-tengah.
- FlowLayout merupakan default untuk JPanel.

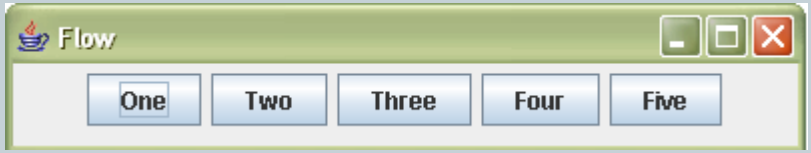
Penggunaan FlowLayout

- Contoh:

```
JPanel panel1 = new JPanel();
panel1.setLayout(new FlowLayout());
```

- Pada dasarnya, untuk aplikasi sesungguhnya, kita tidak disarankan menggunakan FlowLayout karena sifatnya yang tidak bisa memposisikan komponen dengan pasti.

Latihan 1: Flow.java




```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Flow extends JFrame {
    public Flow(  ) {
      createUserInterface();
   }
    private void createUserInterface(){
    // FlowLayout is default layout manager for a JPanel
      JPanel panel1 = new JPanel();
      panel1.setLayout(new FlowLayout()); // !! baris ini dapat dihapus !!
      panel1.add(new JButton("One"));
      panel1.add(new JButton("Two"));
      panel1.add(new JButton("Three"));
      panel1.add(new JButton("Four"));
      panel1.add(new JButton("Five"));

      this.add(panel1);
      this.setTitle("Flow");
      this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      this.setSize(400, 75);
      this.setLocation(200, 200);
      this.setVisible(true);
   }

    public static void main(String[] args) {
      Flow app = new Flow();
  }
}
```

2. GridLayout

- GridLayout menempatkan komponen dalam bentuk “rectangular grid”. Ada 3 constructor untuk GridLayout:
  - GridLayout(): membuat layout dengan satu kolom per komponen. Hanya satu baris yang digunakan.
  - GridLayout(int rows, int cols): membuat suatu layout berdasarkan jumlah baris dan kolom yang diinginkan.
  - GridLayout(int rows, int cols, int hgap, int vgap): membuat layout berdasarkan jumlah baris dan kolom yang diinginkan, serta ukuran jarak (gap) horisontal maupun vertical untuk setiap baris dan kolom tersebut.
- GridLayout menempatkan komponen dengan urutan dari kiri ke kanan dan dari atas ke bawah.
- GridLayout akan memaksa setiap komponen untuk menempati space container yang kosong serta membagi rata ukuran space tersebut.

Penggunaan GridLayout

- GridLayout paling cocok digunakan untuk menyusun komponen yang berukuran sama, misalnya 2 buah JPanel berukuran sama di dalam sebuah frame.
- Contoh:

```
frame.add(panel1);
frame.add(panel2);
frame.setLayout(new GridLayout(2, 1));
```

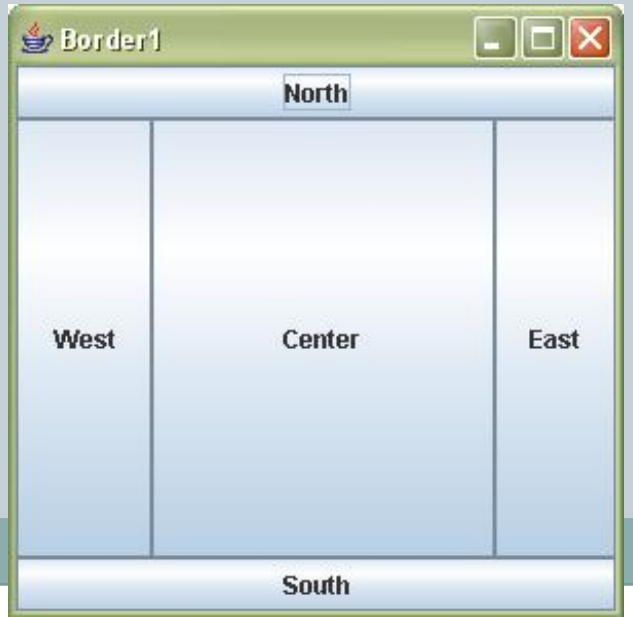
- Kita dapat men-set jumlah baris dan kolom dengan angka 0. Artinya, kita tidak mempedulikan berapa banyak komponen yang akan masuk ke dalam dimensi layout manager tersebut.
  - Contoh: GridLayout(2,0)
  - Artinya: kita membuat layout manager dengan 2 baris dan unlimited number untuk kolomnya.
  - Apa yang terjadi kalau kita memiliki 10 komponen?

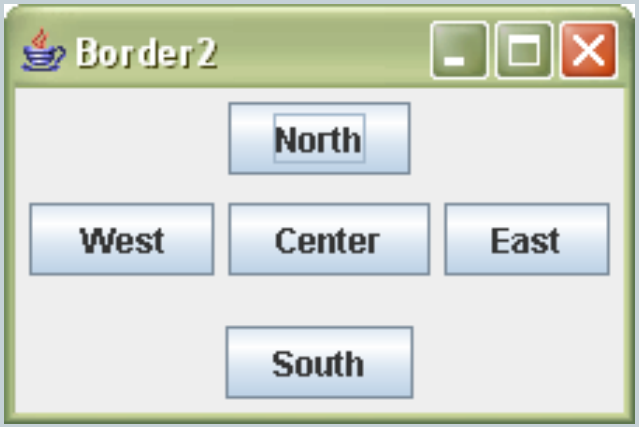
Latihan 2: Grid.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
 public class Grid extends JFrame {
    public Grid(  ) {
      createUserInterface();
   }
    private void createUserInterface(){
      JPanel panel1 = new JPanel();
      panel1.setLayout(new GridLayout(3, 2));
      panel1.add(new JButton("One"));
      panel1.add(new JButton("Two"));
      panel1.add(new JButton("Three"));
      panel1.add(new JButton("Four"));
      panel1.add(new JButton("Five"));

      this.add(panel1);
      this.setTitle("Grid");
      this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      this.setSize(200, 200);
      this.setLocation(200, 200);
      this.setVisible(true);
   }
    public static void main(String[] args) {
      Grid app = new Grid();
   }
}
```

3. BorderLayout

- BorderLayout menyusun komponen berdasarkan lokasi geografis: NORTH, SOUTH, EAST, WEST, and CENTER.
- Secara optional, kita dapat juga memberikan padding di antara komponen.
- BorderLayout merupakan layout default untuk JWindow dan JFrame.
- Karena setiap komponen diasosiasikan dengan suatu arah geografis, akibatnya layout ini hanya dapat menangani maksimal 5 komponen.





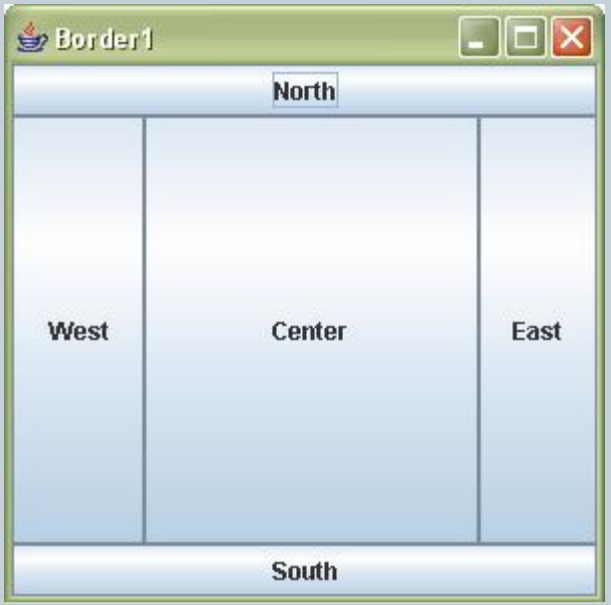
Penggunaan BorderLayout

- Pada saat menambahkan suatu komponen kepada container yang memiliki border layout, kita harus menentukan secara bersamaan komponen-nya dan posisi-nya.
- Contoh:

```
frame.setLayout(new BorderLayout(  ));
frame.add(new JButton("Button1"), BorderLayout.NORTH );
frame.add(new JButton("Button2"), BorderLayout.SOUTH );
```

Latihan 3: BorderLayoutApp1.java

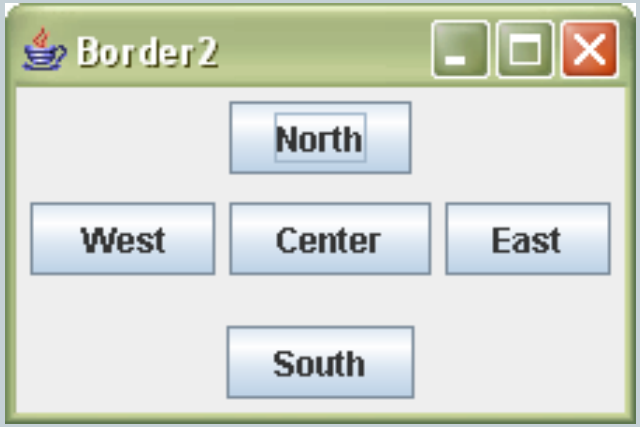
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
 public class BorderLayoutApp1 extends JFrame {

    public BorderLayoutApp1(  ) {
      createUserInterface();
   }

    private void createUserInterface(){
      this.setTitle("Border1");
      this.setSize(300, 300);
      this.setLocation(200, 200);
      this.setLayout(new BorderLayout(  ));
      this.add(new JButton("North"), BorderLayout.NORTH );
      this.add(new JButton("South"), BorderLayout.SOUTH );
      this.add(new JButton("East"), BorderLayout.EAST );
      this.add(new JButton("West"), BorderLayout.WEST );
      this.add(new JButton("Center"), BorderLayout.CENTER );
      this.setVisible(true);
   }

    public static void main(String[] args) {
      BorderLayoutApp1 app = new BorderLayoutApp1();
      app.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
   }
}
```



Latihan 4: BorderLayoutApp2.java

if we don't want BorderLayout messing with the sizes of our components

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
 public class BorderLayoutApp2 extends JFrame {
    public BorderLayoutApp2(  ) {
      createUserInterface();
  }

    private void createUserInterface(){
      this.setTitle("Border2");
      this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      this.setSize(225, 150);
      this.setLocation(200, 200);
      this.setVisible(true);
      this.setLayout(new BorderLayout(  ));

      JPanel p = new JPanel(  );
      p.add(new JButton("North"));
      this.add(p, BorderLayout.NORTH);

      p = new JPanel(  );
      p.add(new JButton("South"));
      this.add(p, BorderLayout.SOUTH);

      p = new JPanel(  );
      p.add(new JButton("East"));
      this.add(p, BorderLayout.EAST);
```



continue

```
  p = new JPanel(  );
  p.add(new JButton("West"));
  this.add(p, BorderLayout.WEST);

  p = new JPanel(  );
  p.add(new JButton("Center"));
  this.add(p, BorderLayout.CENTER);
}

public static void main(String[] args) {
   BorderLayoutApp2 app = new BorderLayoutApp2();
       }
}
```

4. BoxLayout

- Layout manager yang telah kita bahas sebelumnya merupakan bagian dari package java.awt.
- Javax.swing memiliki beberapa tambahan layout manager lagi, salah satunya adalah: BoxLayout.
- Layout manager ini sangat berguna untuk membuat toolbars sederhana atau vertical button bars.
- Cara kerjanya sangat sederhana yaitu menempatkan komponen dalam satu baris atau satu kolom.

Penggunaan BoxLayout

- Untuk mempermudah penggunaan BoxLayout, Swing menyediakan sebuah kelas yang bernama Box yaitu sebuah container yang secara otomatis memiliki BoxLayout manager.
- Box memiliki beberapa methods yang akan mempermudah kita dalam menggunakan BoxLayout manager, yaitu:
  - `createHorizontalBox()` → untuk membuat box horizontal
    - `createHorizontalGlue()` → untuk merekatkan komponen
    - `createHorizontalStrut(int n)` → untuk memberi jarak antar komponen
  - `createVerticalBox()` → untuk membuat box vertical
    - `createVerticalGlue()` → untuk merekatkan komponen
    - `createVerticalStrut(int n)` → untuk memberi jarak antar komponen

Latihan 5: Boxer.java

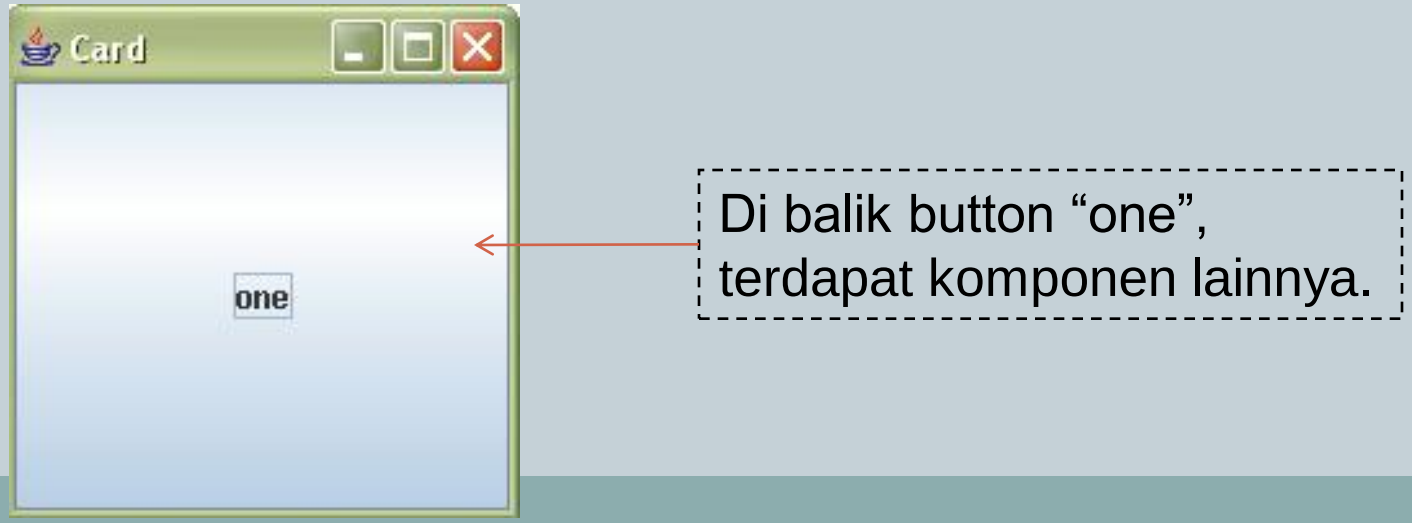
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Boxer extends JFrame {
  public Boxer(){
    createUserInterface();
 }
  private void createUserInterface(){
    Container box = Box.createHorizontalBox(  );
    box.add(Box.createHorizontalGlue(  ));
    box.add(new JButton("In the"));
    box.add(Box.createHorizontalGlue(  ));
    box.add(new JButton("clearing"));
    box.add(Box.createHorizontalStrut(10));
    box.add(new JButton("stands"));
    box.add(Box.createHorizontalStrut(10));
    box.add(new JButton("a"));
    box.add(Box.createHorizontalGlue(  ));
    box.add(new JButton("boxer"));
    box.add(Box.createHorizontalGlue(  ));

    this.add(box);
    this.setTitle("Boxer");
    this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    this.setSize(250, 250);
    this.setLocation(200, 200);
    this.pack(  );
    this.setVisible(true);
 }
```

continue

```
public static void main(String[] args) {
        Boxer app = new Boxer();
    }
}
```

5. CardLayout

- CardLayout merupakan layout manager yang mampu menciptakan efek "tumpukan" komponen.
- Artinya, layout ini tidak memposisikan komponen di lokasi-lokasi tertentu di dalam kontainer, melainkan menampilkannya satu demi satu.



Penggunaan CardLayout

- Penggunaan CardLayout biasanya untuk membuat panel yang bersifat custom-tabbed.
- Namun, sebenarnya kita dapat membuat panel tersebut dengan menggunakan komponen JTabbedPane.
- Untuk mempraktekkan cara kerja dan efek "tumpukan" dari CardLayout ini, kita perlu mempelajari terlebih dahulu mengenai "event-driven programming" seperti yang ada di Latihan 5.

Latihan 5: Card.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Card extends JPanel {
   CardLayout cards = new CardLayout( );

   public Card( ) {
      setLayout(cards);
      ActionListener listener = new ActionListener( ) {
         public void actionPerformed(ActionEvent e) {
            cards.next(Card.this);
         }
      };
      JButton button;
      button = new JButton("one");
      button.addActionListener(listener);
      add(button, "one");
      button = new JButton("two");
      button.addActionListener(listener);
      add(button, "two");
      button = new JButton("three");
      button.addActionListener(listener);
      add(button, "three");
   }

   public static void main(String[] args) {
      JFrame frame = new JFrame("Card");
      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      frame.setSize(200, 200);
      frame.setLocation(200, 200);
      frame.setContentPane(new Card( ));
      frame.setVisible(true);
   }
}
```

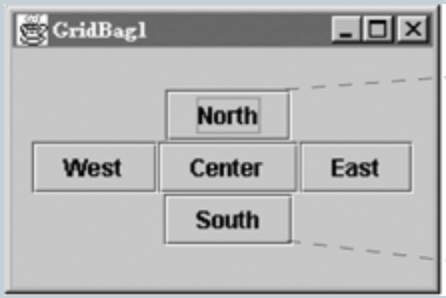
6. GridBagLayout

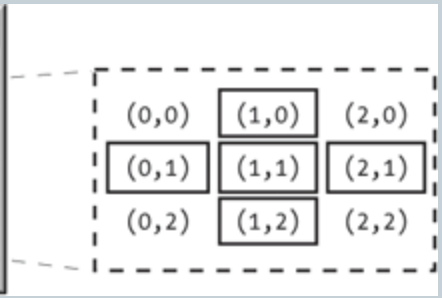
- GridBagLayout merupakan layaout manager yang sangat fleksibel.
- Layout ini memampukan kita untuk memposisikan komponen relatif terhadap komponen lainnya berdasarkan constraint tertentu.
- Dengan menggunakan GridBagLayout, kita dapat menciptakan layout apapun juga, tanpa batas.
- Komponen disusun pada koordinat tertentu pada sebuah grid yang disebut "logical coordinate".
- Logical coordinate berarti bahwa koordinat suatu komponen ditentukan oleh sekumpulan komponen lainnya.
- Baris dan kolom dari grid tersebut bersifat "stretch" yang bergantung pada size dan constraint yang dimilikinya.

Penggunaan GridBag

- Walaupun fleksibel, pembuatan GridBag ini terkadang sangat membingungkan karena kita harus mengatur berbagai size dan constraint dari setiap komponen yang digunakan.
- Sebenarnya, penggunaan GridBagLayout ini jauh lebih mudah apabila kita menggunakan tools yang mendukung WYSIWYG GUI builder. (contohnya: NetBeans IDE)

Latihan 6: GridBag1.java
A Simple GridBag layout: Grid Coordinate






```
import java.awt.*;
 import java.awt.event.*;
 import javax.swing.*;

  public class GridBag1 extends JFrame {
    private GridBagConstraints constraints;
     public GridBag1(  ) {
       createUserInterface();
    }
     private void createUserInterface(){
       constraints = new GridBagConstraints(  );
       this.setLayout(new GridBagLayout(  ));
       int x, y;  // for clarity
       this.addGB(new JButton("North"),  x = 1, y = 0);
       this.addGB(new JButton("West"),   x = 0, y = 1);
       this.addGB(new JButton("Center"), x = 1, y = 1);
       this.addGB(new JButton("East"),   x = 2, y = 1);
       this.addGB(new JButton("South"),  x = 1, y = 2);

       this.setTitle("GridBag1");
       this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
       this.setSize(225, 150);
       this.setLocation(200, 200);
       this.setVisible(true);
    }
     private void addGB(Component component, int x, int y) {
       constraints.gridx = x;
       constraints.gridy = y;
       this.add(component, constraints);
    }
    public static void main(String[] args) {
       GridBag1 app = new GridBag1();
    }
 }
```

Latihan 7: GridBag2.java

Using fill constraint: Controls whether the component expands to fill the allotted space

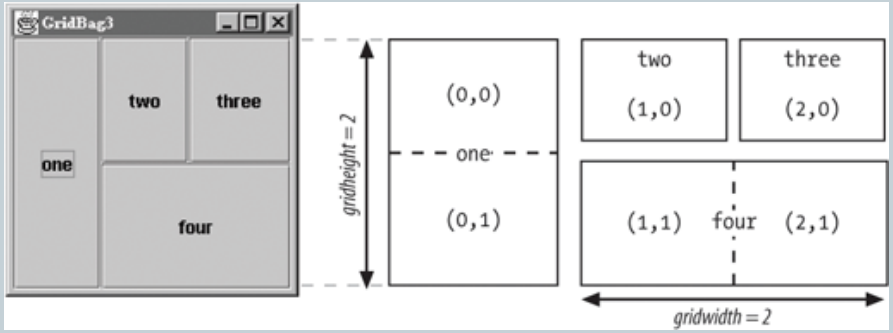
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
 public class GridBag2 extends JFrame {
   private GridBagConstraints constraints;
    public GridBag2(  ) {
      createUserInterface();
   }
    private void createUserInterface(){
      constraints = new GridBagConstraints(
      this.setLayout(new GridBagLayout(  ));
      constraints.weightx = 1.0;
      constraints.weighty = 1.0;
      constraints.fill = GridBagConstraints.BOTH;
      int x, y;  // for clarity
      this.addGB(new JButton("North"),   x = 1, y = 0);
      this.addGB(new JButton("West"),    x = 0, y = 1);
      this.addGB(new JButton("Center"),  x = 1, y = 1);
      this.addGB(new JButton("East"),    x = 2, y = 1);
      this.addGB(new JButton("South"),   x = 1, y = 2);

      this.setTitle("GridBag1");
      this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      this.setSize(225, 150);
      this.setLocation(200, 200);
      this.setVisible(true);
   }
```

continue

```
private void addGB(Component component, int x,
int y) {
          constraints.gridx = x;
          constraints.gridy = y;
          this.add(component, constraints);
       }
        public static void main(String[] args) {
          GridBag2 app = new GridBag2();
       }
   }
```

Latihan 8: GridBag3.java
Spanning Rows and Columns

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
 public class GridBag3 extends JFrame {
   GridBagConstraints constraints = new GridBagConstraints(
);
    public GridBag3( ) {
     createUserInterface();
    }
    private void createUserInterface(){
     this.setLayout(new GridBagLayout( ));
     constraints.weightx = 1.0;
     constraints.weighty = 1.0;
     constraints.fill = GridBagConstraints.BOTH;
     int x, y;  // for clarity
     constraints.gridheight = 2; // span two rows
     addGB(new JButton("one"),   x = 0, y = 0);
     constraints.gridheight = 1; // set it back
     addGB(new JButton("two"),   x = 1, y = 0);
     addGB(new JButton("three"), x = 2, y = 0);
     constraints.gridwidth = 2; // span two columns
     addGB(new JButton("four"),  x = 1, y = 1);
     constraints.gridwidth = 1; // set it back
     this.setTitle("GridBag3");
     this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE
);
     this.setSize(200, 200);
     this.setLocation(200, 200);
     this.setVisible(true);
    }
```



continue

```
void addGB(Component component, int x, int y) {
     constraints.gridx = x;
     constraints.gridy = y;
     this.add(component, constraints);
   }

   public static void main(String[] args) {
     GridBag3 app = new GridBag3();
   }
 }
```

Latihan 9: GridBag4.java
Using weight to control component size

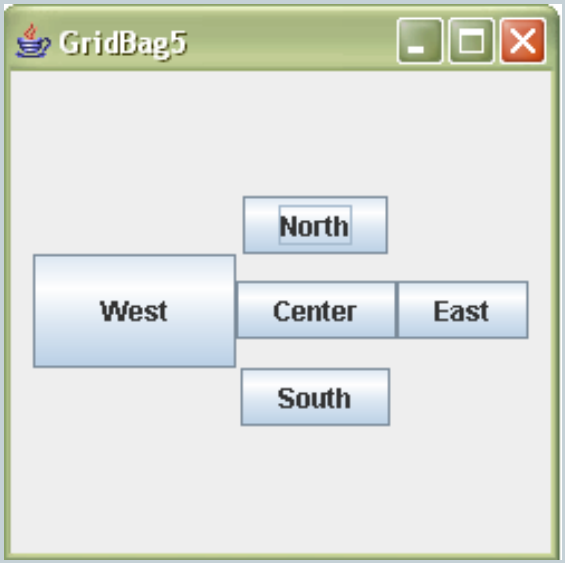
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class GridBag4 extends JFrame {
 GridBagConstraints constraints = new GridBagConstraints(  );
  public GridBag4(  ) {
   createUserInterface();
 }
  private void createUserInterface(){
    this.setLayout(new GridBagLayout(  ));
    constraints.fill = GridBagConstraints.BOTH;
    constraints.weighty = 1.0;
    int x, y; // for clarity
    constraints.weightx = 0.1;
    addGB(new JButton("one"),   x = 0, y = 0);
    constraints.weightx = 0.5;
    addGB(new JButton("two"),   ++x,   y);
    constraints.weightx = 1.0;
    addGB(new JButton("three"), ++x,   y);
    this.setTitle("GridBag4");
    this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    this.setSize(300, 100);
    this.setLocation(200, 200);
    this.setVisible(true);
 }
```

continue

```
void addGB(Component component, int x,
int y) {
        constraints.gridx = x;
        constraints.gridy = y;
        this.add(component, constraints);
    }
     public static void main(String[] args) {
        GridBag4 app = new GridBag4();
    }
  }
```

Latihan 10: GridBag5.java
Using padding and insets in a layout

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
 public class GridBag5 extends JFrame {
   GridBagConstraints constraints = new GridBagConstraints(  );
    public GridBag5(  ) {
      createUserInterface();
   }
    private void createUserInterface(){
      this.setLayout(new GridBagLayout(  ));
      int x, y;  // for clarity
      addGB(new JButton("North"),   x = 1, y = 0);
      constraints.ipadx = 25;  // add padding
      constraints.ipady = 25;
      addGB(new JButton("West"),    x = 0, y = 1);
      constraints.ipadx = 0;   // remove padding
      constraints.ipady = 0;
      addGB(new JButton("Center"), x = 1, y = 1);
      addGB(new JButton("East"),    x = 2, y = 1);
      addGB(new JButton("South"),   x = 1, y = 2);

      this.setTitle("GridBag5");
      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      this.setSize(250, 250);
      this.setLocation(200, 200);
      this.setVisible(true);
   }
```




```
void addGB(Component component, int x, int y) {
        constraints.gridx = x;
        constraints.gridy = y;
        this.add(component, constraints);
    }

     public static void main(String[] args) {
       GridBag5 app = new GridBag5();
    }
 }
```

Last but not least: GroupLayout

- Lihat contoh program:
  - Group.java
  - AlumniForm.java
- GroupLayout dapat kita buat dengan cara mengetik manual seperti membuat layout lainnya.
- Namun, keunggulan GroupLayout adalah dapat dibuat dengan mudah apabila menggunakan IDE seperti NetBeans.

Summary

- Congratulations!, you've learnt how to manage component's layout in Java GUI application!