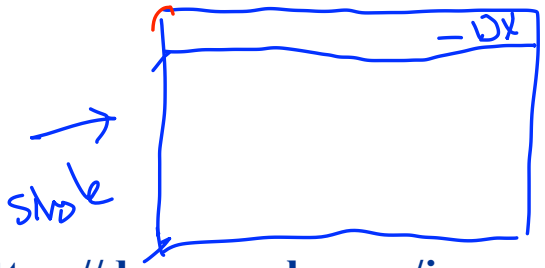
layout in Java Swing

the way programmers arrange their visual components is called layout.

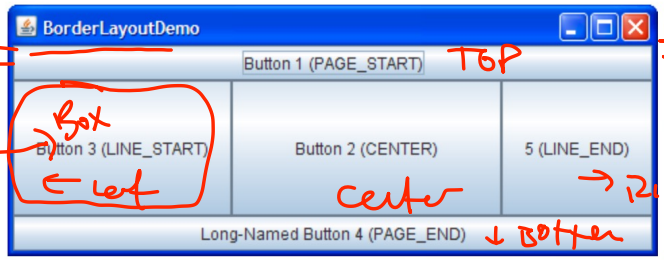
⇒ flow layout
grid layout
null layout

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



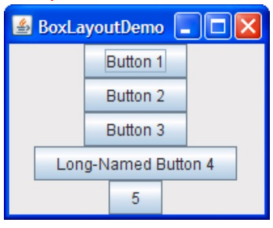
https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

BorderLayout



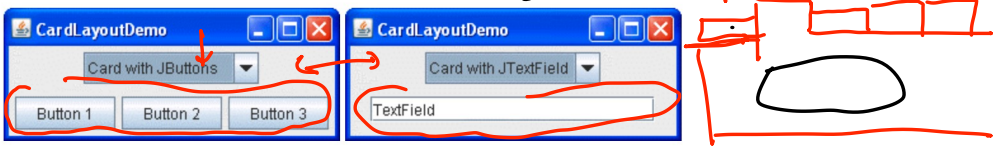
Every content pane is initialized to use a `BorderLayout`. (As Using Top-Level Containers explains, the content pane is the main container in all frames, applets, and dialogs.) A `BorderLayout` places components in up to five areas: top, bottom, left, right, and center. All extra space is placed in the center area. Tool bars that are created using JToolBar must be created within a `BorderLayout` container, if you want to be able to drag and drop the bars away from their starting positions. For further details, see How to Use BorderLayout.

BoxLayout



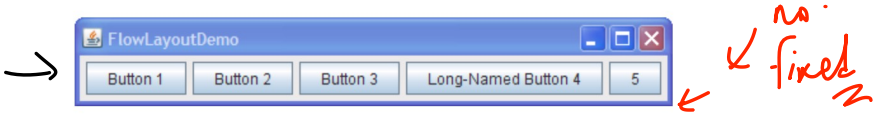
The `BoxLayout` class puts components in a single row or column. It respects the components' requested maximum sizes and also lets you align components. For further details, see How to Use BoxLayout.

CardLayout



The `CardLayout` class lets you implement an area that contains different components at different times. A `CardLayout` is often controlled by a combo box, with the state of the combo box determining which panel (group of components) the `CardLayout` displays. An alternative to using `CardLayout` is using a tabbed pane, which provides similar functionality but with a pre-defined GUI. For further details, see How to Use CardLayout.

FlowLayout

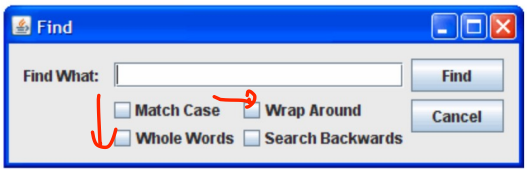


`FlowLayout` is the default layout manager for every `JPanel`. It simply lays out components in a single row, starting a new row if its container is not sufficiently wide. Both panels in CardLayoutDemo, shown previously, use `FlowLayout`. For further details, see How to Use FlowLayout.

GridBagLayout

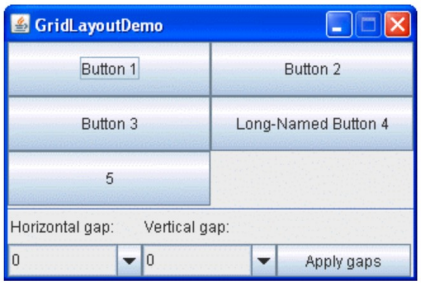
`GridBagLayout` is a sophisticated, flexible layout manager. It aligns components by placing them within a grid of cells, allowing components to span more than one cell. The rows in the grid can have different heights, and grid columns can have different widths. For further details, see How to Use GridBagLayout.

GroupLayout



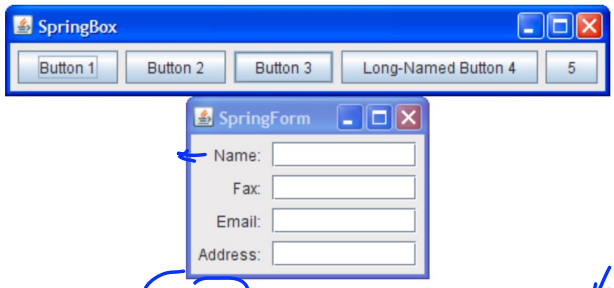
`GroupLayout` is a layout manager that was developed for use by GUI builder tools, but it can also be used manually. `GroupLayout` works with the horizontal and vertical layouts separately. The layout is defined for each dimension independently. Consequently, however, each component needs to be defined twice in the layout. The Find window shown above is an example of a `GroupLayout`. For further details, see How to Use GroupLayout.

GridLayout



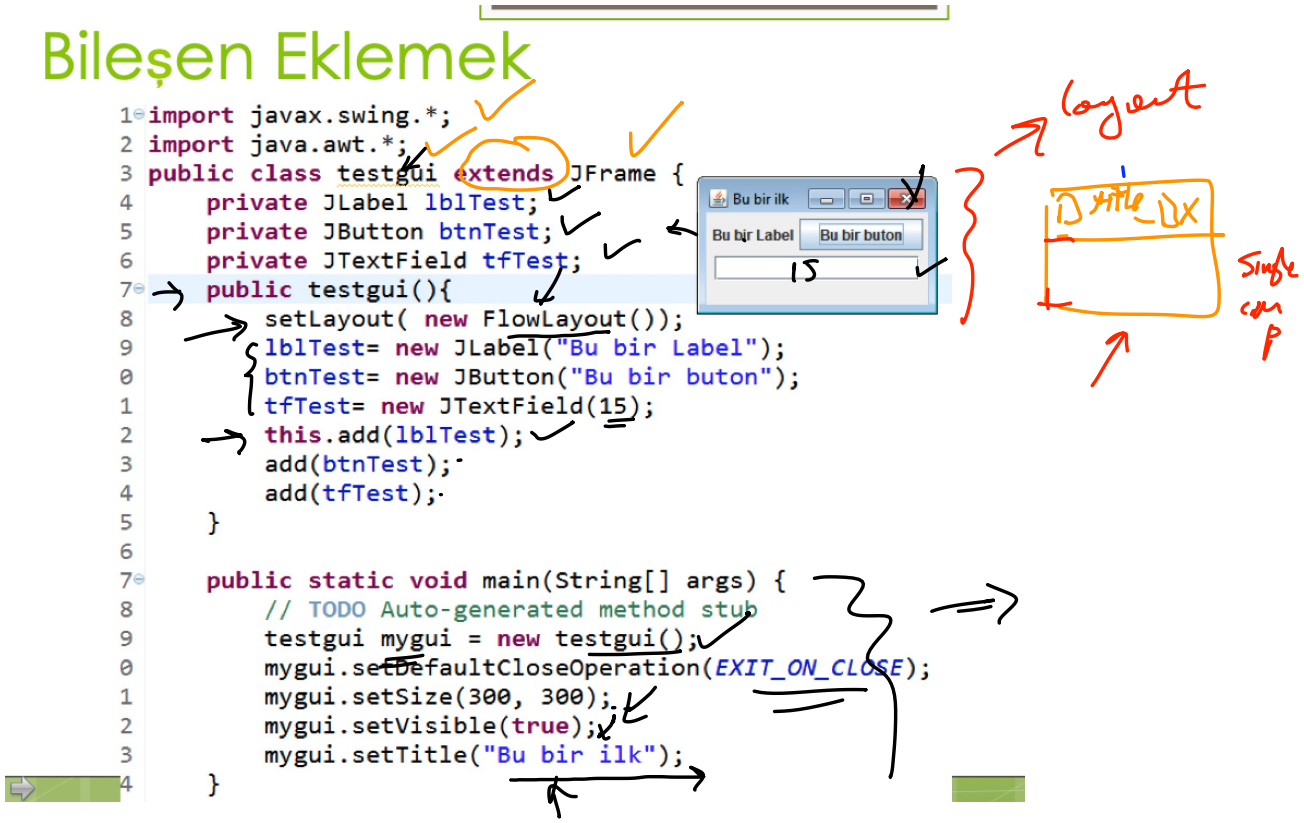
`GridLayout` simply makes a bunch of components equal in size and displays them in the requested number of rows and columns. For further details, see How to Use GridLayout.

SpringLayout

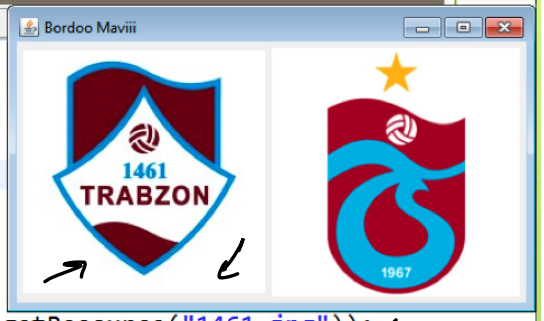


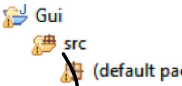
`SpringLayout` is a flexible layout manager designed for use by GUI builders. It lets you specify precise relationships between the edges of components under its control. For example, you might define that the left edge of one component is a certain distance (which can be dynamically calculated) from the right edge of a second component. `SpringLayout` lays out the children of its associated container according to a set of constraints, as shall be seen in How to Use SpringLayout.

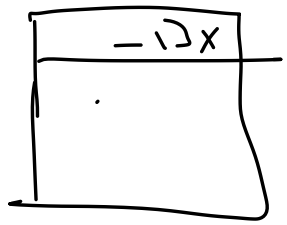
Bileşen Eklemek



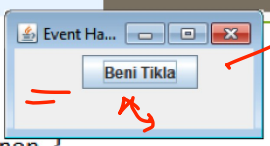
```java
import java.awt.*;
import javax.swing.*;
public class GuiTest02 extends JFrame {
    private ImageIcon img1;
    private ImageIcon img2;
    private JLabel lbl1;
    private JLabel lbl2;
    public GuiTest02(){
        setLayout(new FlowLayout());
        img1= new ImageIcon(getClass().getResource("1461.jpg"));
        img2 = new ImageIcon(getClass().getResource("ts.png"));
        lbl1= new JLabel(img1);
        lbl2= new JLabel(img2);
        add(lbl1);
        add(lbl2);
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        GuiTest02 gui = new GuiTest02();
        gui.setDefaultCloseOperation(EXIT_ON_CLOSE);
        gui.setVisible(true);
        gui.setTitle("Bordoo Maviii");
        gui.pack();
```

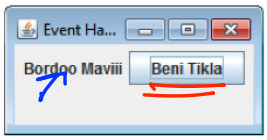


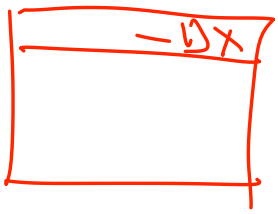




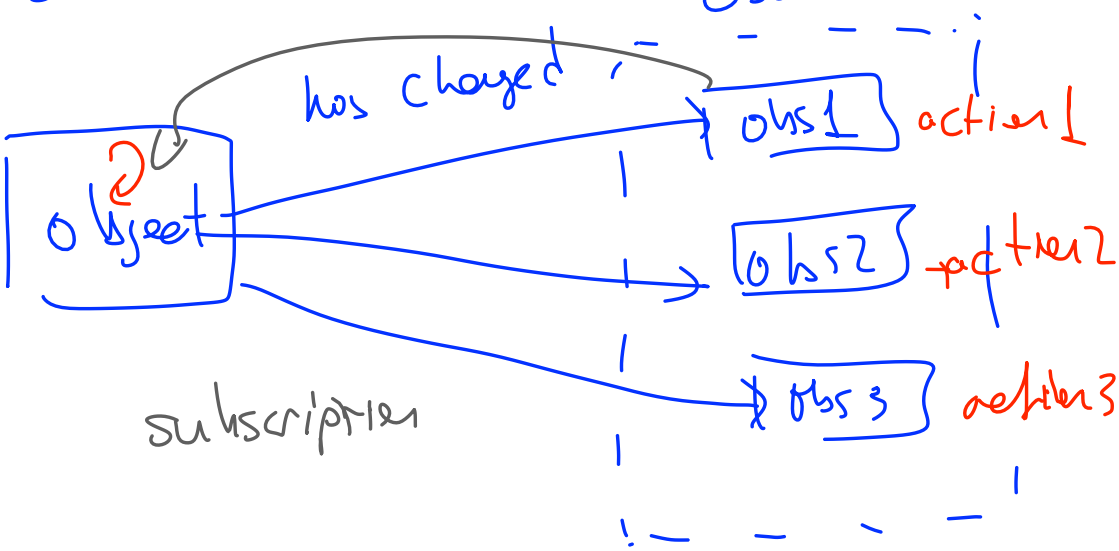

```java
import java.awt.event.*;
import javax.swing.*;
public class GuiEvent extends JFrame {
    private JButton btnTikla;
    private JLabel lblGoster;
    // Butonun tıklandığını yakalayan sınıf
    public class BtnTiklandi implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            lblGoster.setText("Bordoo Maviii");
        }
    }
//Yapılandırıcı
    public GuiEvent() {
        setLayout(new FlowLayout());
        btnTikla = new JButton("Beni Tikla");
        lblGoster = new JLabel("");
        btnTikla.addActionListener(new BtnTiklandi());
        add(lblGoster);
        add(btnTikla);
    }
//main metodu
    public static void main(String[] args) {
        GuiEvent gui = new GuiEvent();
        gui.setDefaultCloseOperation(EXIT_ON_CLOSE);
        gui.setSize(200, 100);
        gui.setVisible(true);
        gui.setTitle("Event Handling");
    }

}
```

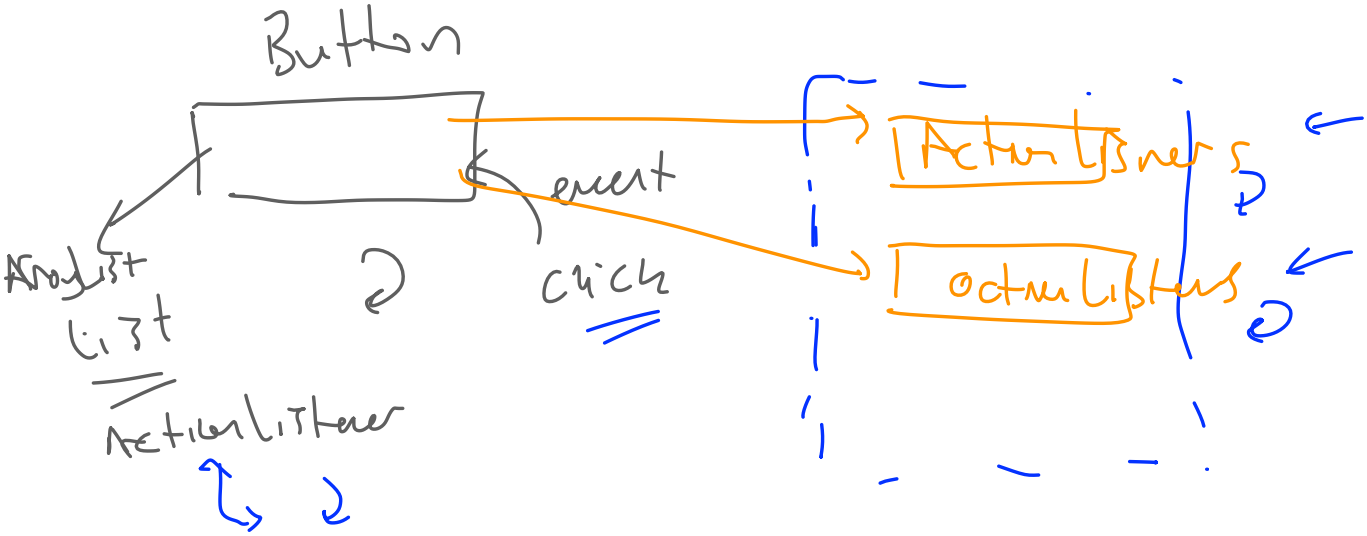




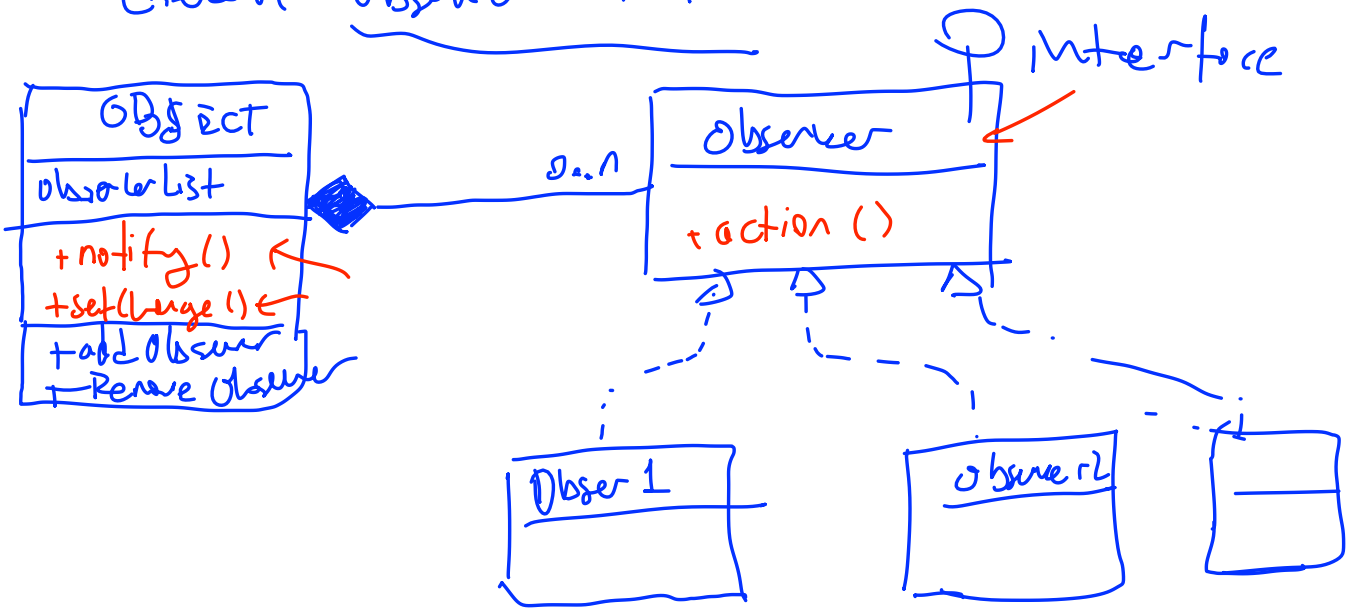


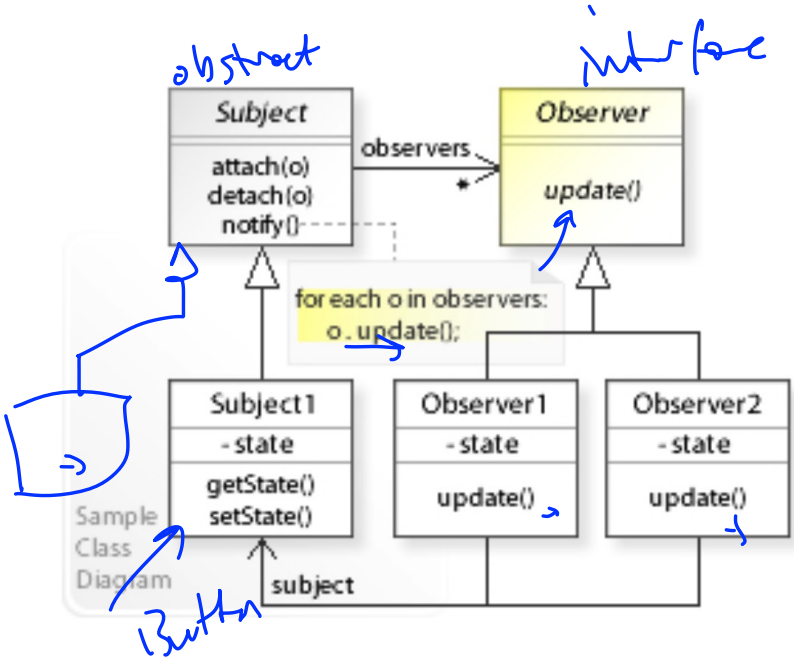
Observer Pattern observer

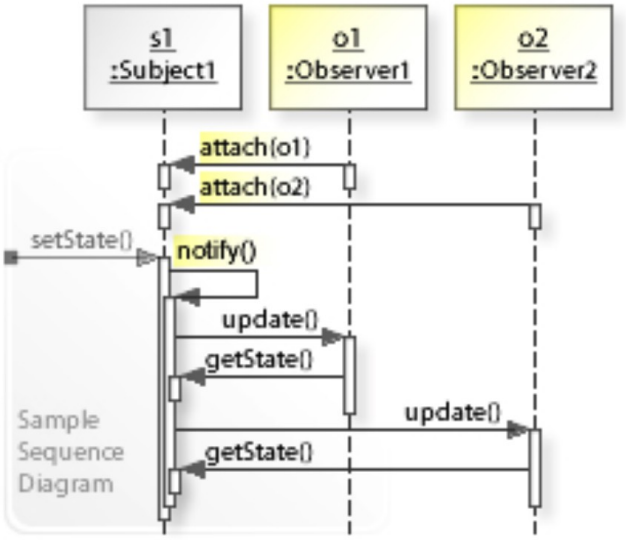




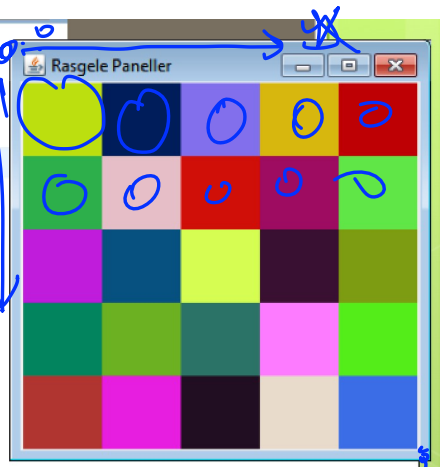
General observer Pattern UML

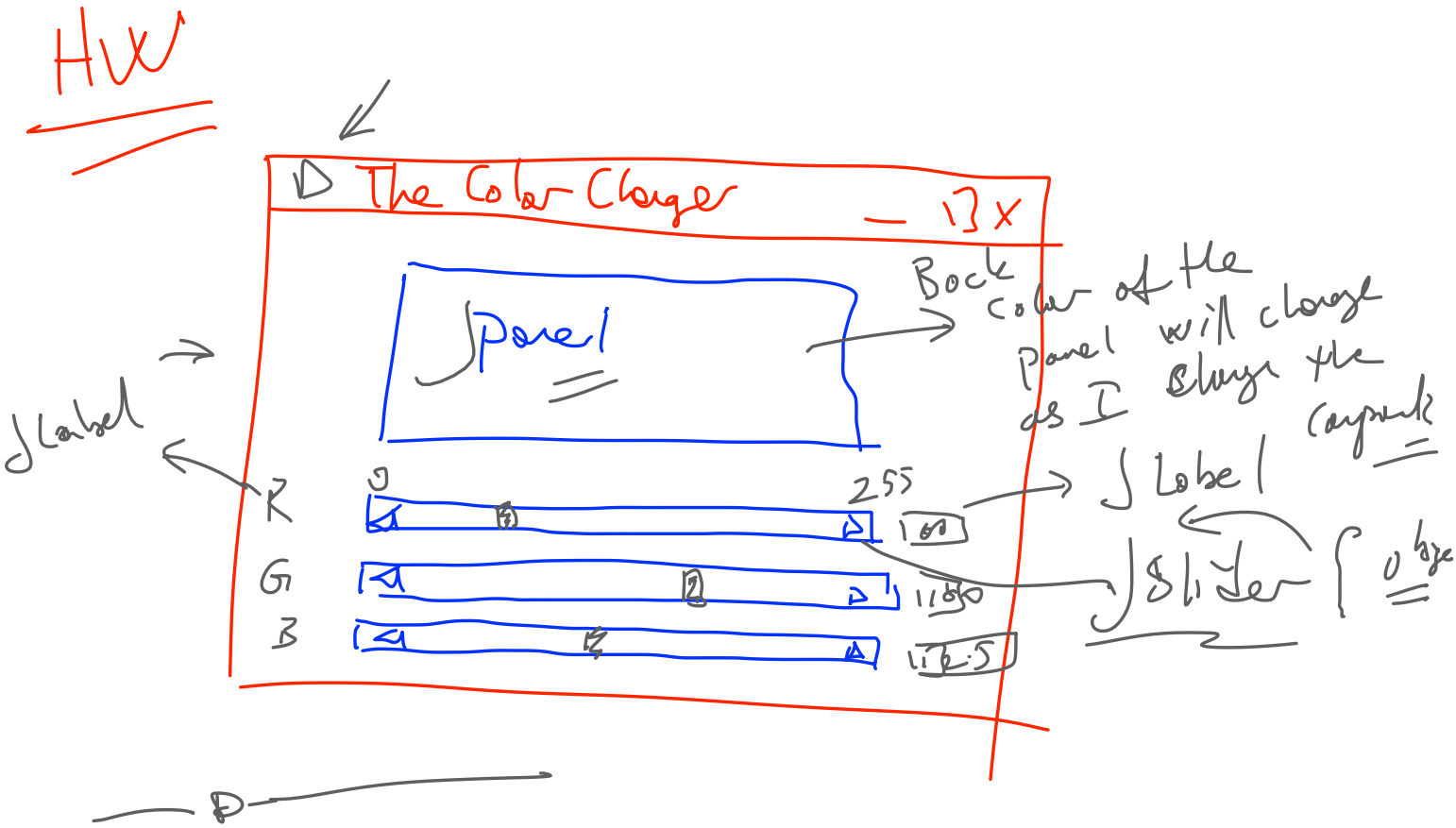







```
4  public class RandomColor extends JPanel {
6       * Tıklandığında Rastgele REnkler Panel sınıfı
8      public Color RandomColor() {
9          int red = (int) (Math.random() * 256);
10         int green = (int) (Math.random() * 256);
11         int blue = (int) (Math.random() * 256);
12         return (new Color(red, green, blue));
13     }
14     public class MouseListener extends MouseAdapter {
15         @Override
16         public void mouseClicked(MouseEvent e) {
17             setBackground(RandomColor());
18         }
19     }
20     public RandomColor() {
21         setBackground(RandomColor());
22         addMouseListener(new MouseListener());
23     }
24     public static void main(String args[]) {
25         JFrame gui = new JFrame();
26         gui.setTitle("Rasgele Paneller");
27         gui.setSize(300, 300);
28         gui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
29         Container pane = gui.getContentPane();
30         pane.setLayout(new GridLayout(5, 5));
31         for (int x = 1; x <= 25; x++) {
32             RandomColor rd = new RandomColor();
33             pane.add(rd);
34         }
35         gui.setVisible(true);
36     }
```






```
private JButton btnTikla2;
private JLabel lblGoster2;
// Butonun tıklandığını yakalayan sınıf
public class BtnTiklandi implements ActionListener {
public class BtnTikla2Tiklandi implements ActionListener {
    boolean Goster = true;
    @Override
    public void actionPerformed(ActionEvent e) {
        if (Goster) {
            lblGoster2.setText("Bordoo Maviii 2222");
            Goster = false;
        } else {
            lblGoster2.setText("");
            Goster = true;
        }
    }
}
// Yapılandırıcı
public GuiEvent() {
    setLayout(new FlowLayout());
    btnTikla = new JButton("Beni Tikla");
    lblGoster = new JLabel("");
    btnTikla2 = new JButton("Beni Tikla 2");
    lblGoster2 = new JLabel("");
    btnTikla.addActionListener(new BtnTiklandi());
    btnTikla2.addActionListener( new BtnTikla2Tiklandi());
    add(lblGoster);
    add(btnTikla);
    add(lblGoster2);
    add(btnTikla2);
}
```

more than one Buttons

Single Class

```
32          MultipleButtons mb= new MultipleButtons();
33
34
35
36      }
37      @Override
38      public void actionPerformed(ActionEvent e) {
39          // TODO Auto-generated method stub
40
41
42          if(e.getSource()==btn1)
43              JOptionPane.showMessageDialog(this, " message from Button1");
44          else if(e.getSource()==btn2)
45              JOptionPane.showMessageDialog(this, " message from Button2");
46          e.
47
48
49      }
50
51
52  }
53
```

reference

equals(Object obj) : boolean - Object
getActionCommand() : String - ActionEvent
getClass() : Class<?> - Object
getID() : int - AWTEvent
getModifiers() : int - ActionEvent
getSource() : Object - EventObject
getWhen() : long - ActionEvent
hashCode() : int - Object
notify() : void - Object
notifyAll() : void - Object

Returns the command string associated with this action. This string allows a "modal" component to specify one of several commands, depending on its state. For example, a single button might toggle between "show details" and "hide details". The source object and the event would be the same in each case, but the command string would identify the intended action.

Note that if a `null` command string was passed to the constructor for this `ActionEvent`, this this method returns `null`.

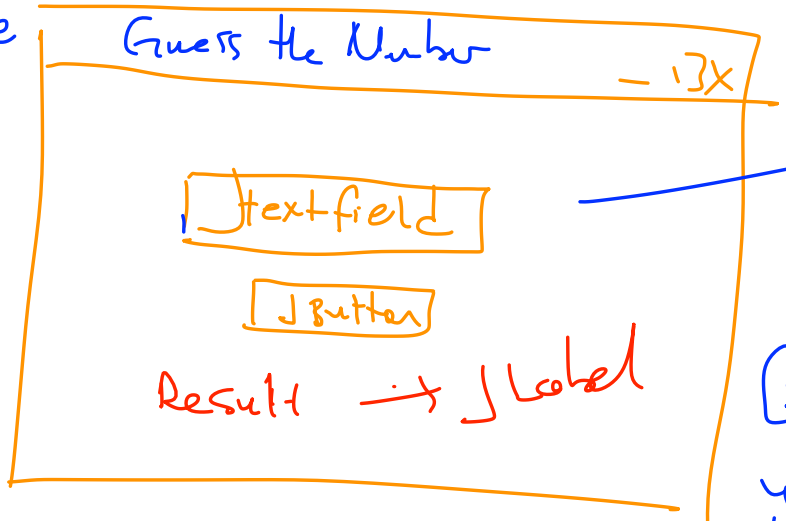
Returns:

Press '^Space' to show Template Proposals

Press 'Tab' from proposal table or click for focus

HW-3

Version One
Classical version



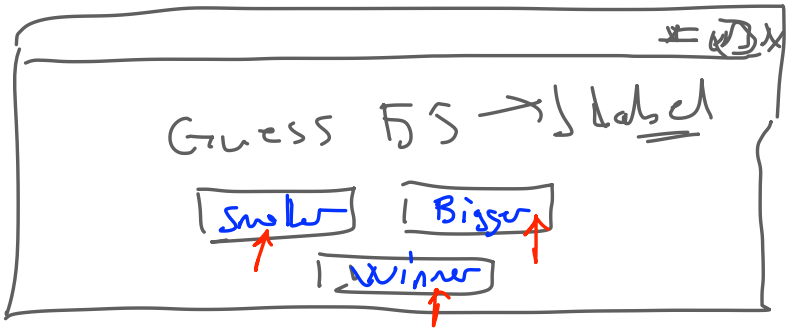
Computer hold a secret integer number between [1 — 100] you try to guess it by entering values in the text field and submitting your answer with button. Result is shown in a label indicating that your guess is ~~[illegible]~~ smaller or bigger then the secret number. → Game continues until you find the

SN.

Hey

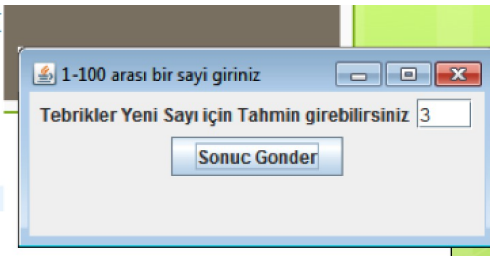
Version two

less Classical



- How as a user will keep a number and let the computer find it)
- Computer should indicate that if you are Cheating.

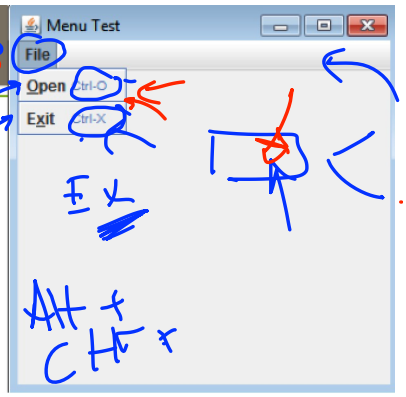
```java
public class RandomNumberGuesser extends JFrame implements ActionListener {
    private JLabel lblSonuc;
    private JTextField txtDeger;
    private JButton btnGonder;
    private int RandomSayi;
    public RandomNumberGuesser() {
        setLayout(new FlowLayout());
        lblSonuc = new JLabel();
        add(lblSonuc);
        btnGonder = new JButton("Sonuc Gonder");
        btnGonder.addActionListener(this);
        txtDeger = new JTextField(3);
        add(txtDeger);
        add(btnGonder);
        RandomSayi = (int) (Math.random() * 100 + 1);
    }
    public static void main(String args[]) {
        RandomNumberGuesser gui = new RandomNumberGuesser();
        gui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        gui.setTitle("1-100 arası bir sayi giriniz");

        gui.pack();
        gui.setVisible(true);

    }
```



HW-3

```java
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        int guess = (int) Integer.parseInt(txtDeger.getText());
        if (e.getSource() == btnGonder) {
            if (guess > RandomSayi)
                lblSonuc.setText("Daha küçük tahmin");
            else if (guess < RandomSayi)
                lblSonuc.setText("Daha Büyük Tahmin");
            else {
                Toolkit.getDefaultToolkit().beep(); // bir uyarı ses çıkarır
                lblSonuc.setText("Tebrikler\n Yeni Sayı için Tahmin girebilirsiniz");
                RandomSayi = (int) (Math.random() * 100 + 1);
            }

        }
```

```java
public class GuiMenu extends JFrame implements ActionListener {
    private JMenuBar menubar;
    private JMenu File;
    private JMenuItem Open;
    private JMenuItem Exit;

    public GuiMenu() {
        menubar = new JMenuBar();
        File = new JMenu("File");

        Open = new JMenuItem("Open");
        Open.setMnemonic('O');
        KeyStroke ctrlX = KeyStroke.getKeyStroke(KeyEvent.VK_X,
                InputEvent.CTRL_MASK);
        KeyStroke ctrlO = KeyStroke.getKeyStroke(KeyEvent.VK_O,
                InputEvent.CTRL_MASK);
        Open.setAccelerator(ctrlO);
        Open.addActionListener(this);
        Exit = new JMenuItem("Exit");
        Exit.setMnemonic('x');
        Exit.setAccelerator(ctrlX);

        menubar.add(File);
        File.add(Open);
        File.addSeparator();
        File.add(Exit);
        this.add(menubar);
        this.setJMenuBar(menubar);
        Exit.addActionListener(this);
    }
```



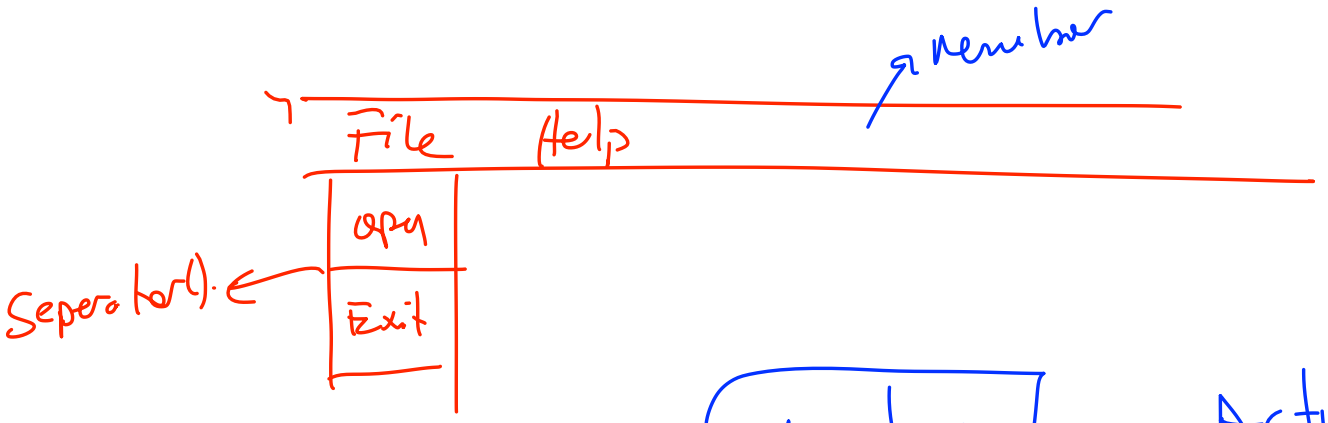
Open

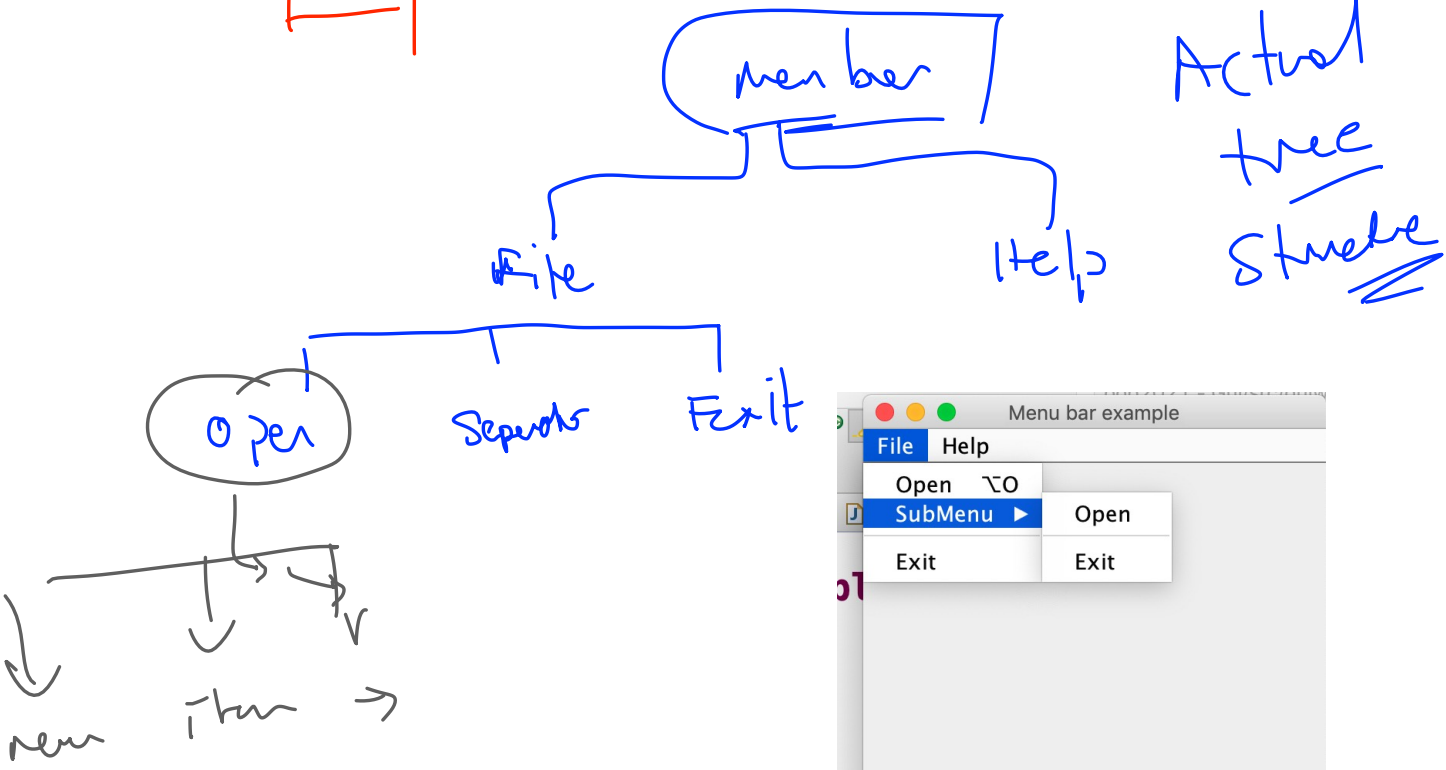
Menu BAR

Alt + Ctrl + X

Mnemonic

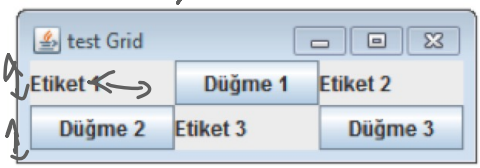
```java
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        GuiMenu gui = new GuiMenu();
        gui.setLayout(new FlowLayout(FlowLayout.LEFT));
        gui.setVisible(true);
        gui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        gui.setTitle("Menu Test");
        gui.setSize(300, 300);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        if (e.getSource() == Exit)
            System.exit(0);
        } else if (e.getSource() == Open) {
            JOptionPane.showMessageDialog(this, "OPEN tıklandı");

        }
    }
```



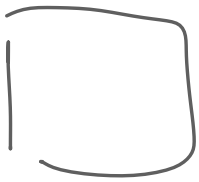


GridLayout

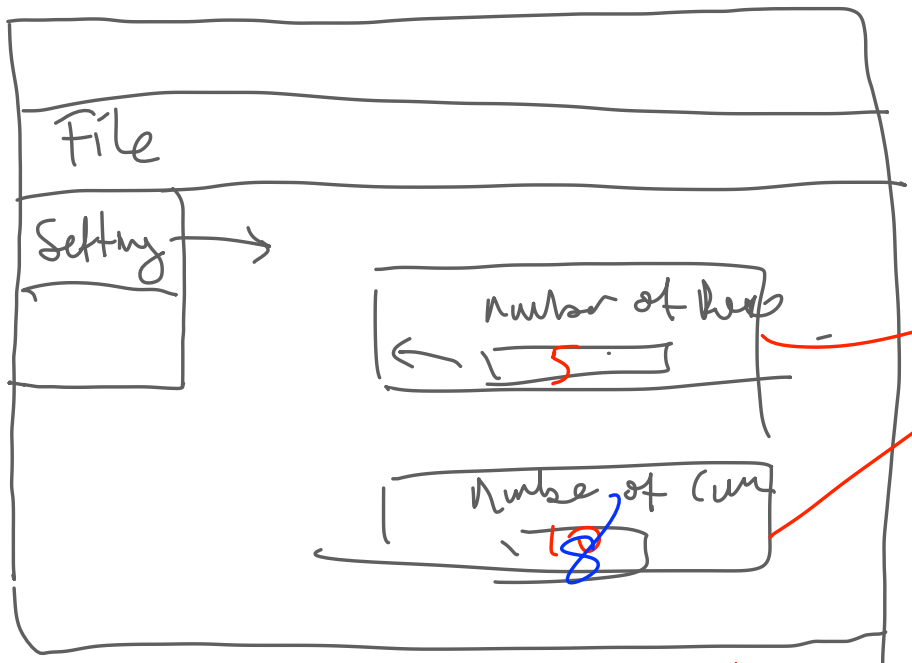
```
5  public class GuiGrid extends JFrame {
6      private JLabel lbl1 = new JLabel("Etiket 1");
7      private JLabel lbl2 = new JLabel("Etiket 2");
8      private JLabel lbl3 = new JLabel("Etiket 3");
9      private JButton btn1 = new JButton("Düğme 1");
0      private JButton btn2 = new JButton("Düğme 2");
1      private JButton btn3 = new JButton("Düğme 3");
2
3      public GuiGrid() {
4          // 2 satır, 3 sütün, 4px yatay bileşenler arası boşluk
5          // 5px düşey pikseller arası boşluk
6          // setLayout(new GridLayout(2,3,4,5));
7          setLayout(new GridLayout(2, 3));
8          add(lbl1);
9          add(btn1);
0          add(lbl2);
1          add(btn2);
2          add(lbl3);
3          add(btn3);
4          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
5          setTitle("test Grid");
6
7          setVisible(true);
8          pack();
```



with equal size



HW-5



JOptionPane
Show Input Dialog

B + #Number of the Button

JButton title B+#

8

5

B1 B2 B3

B40

Dynamically

Click

JOptionPane ("Hello from B40")