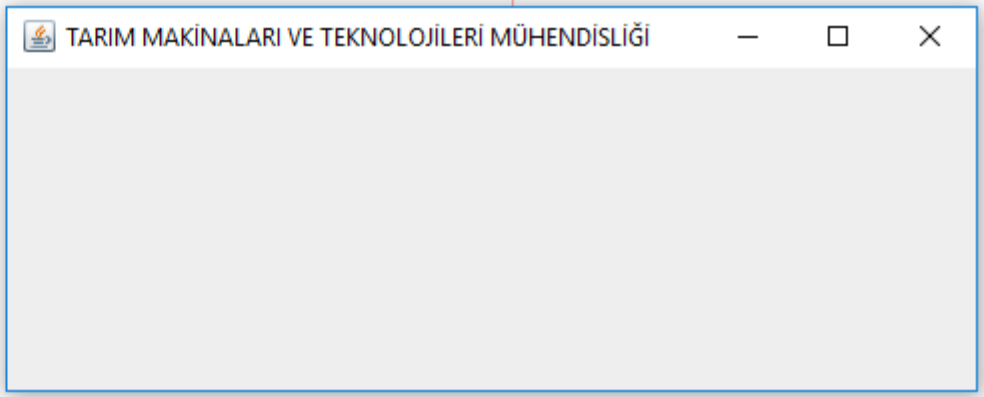
ORNEK 1: Tarım Makinaları ve Teknolojileri Mühendisliği başlıklı çerçeve üretme.

Start Page | NewJFrame.java | NewJFrame.java | NewJFrame.java | Ornek.java | JavaApplication136.java

Source | History

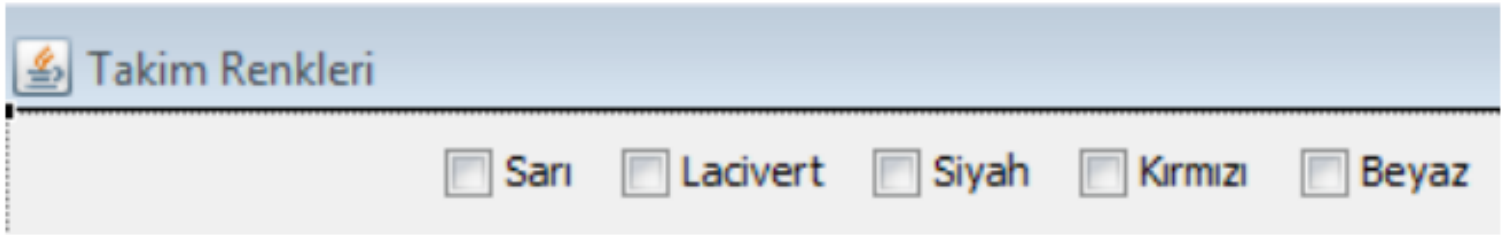
```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package javaapplication135;
7
8  import javax.swing.JFrame;
9  import javax.swing.SwingUtilities;
10 public class Ornek extends JFrame {
11         public Ornek() {
           setTitle("TARIM MAKİNALARI VE TEKNOLOJİLERİ MÜHENDİSLİĞİ");
           setSize(500, 200);
           setLocationRelativeTo(null);
           setDefaultCloseOperation(EXIT_ON_CLOSE);
16         }
17
18         public static void main(String[] args) {
           SwingUtilities.invokeLater(new Runnable() {
                   public void run() {
21
22                     Ornek ex = new Ornek();
23                     ex.setVisible(true);
24                 }
25             });
26         }
27 }
```



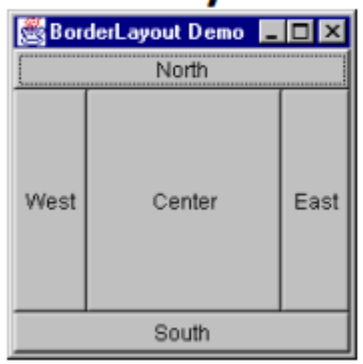
YERLEŞİM PLANI DÜZENLEYİCİLERİ (LAYOUT MANAGERS)

Bir çerçeve veya pano içerisine bileşenler (butonlar, checkboxlar, vs) yerleştirildiğinde o bileşenin çerçevenin neresine ve hangi boyutlarda yerleştirileceğine Yerleşim Planı Düzenleyicileri (Layout karar verir. Grafik uygulamalarında farklı layout yöneticileri kullanılabilir. Bunlardan bazıları;

1. **FlowLayout** *// Bileşenler, çerçeveye satır satır ve soldan sağa doğru yerleştirilir* **.**



2. **BorderLayout** *Bileşenler 5 alana yerleştirilir; Doğu, Batı, Güney, Kuzey ve Merkez olmak üzere.*



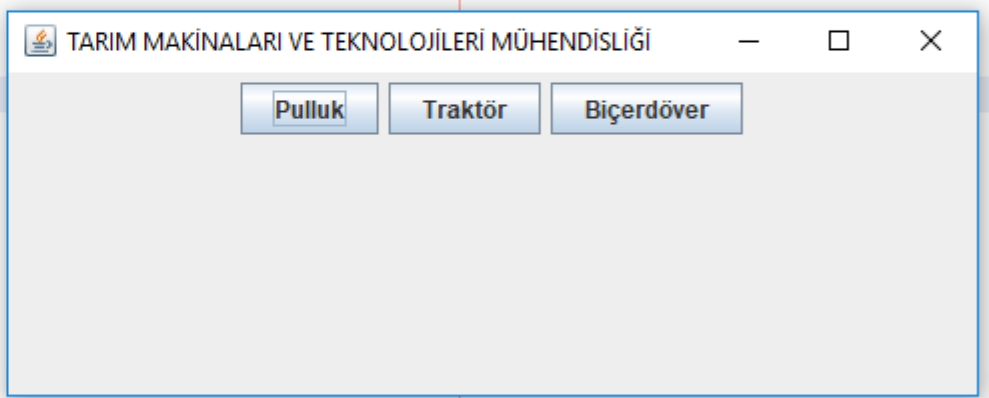
3. **GridLayout** *Bileşenler, Excel tablosuna benzer şekilde ızgaralara yerleştirilir.*

Örnek 2: Form üzerine buton yerleştiren Applet kodu.

Start Page | NewJFrame.java | NewJFrame.java | NewJFrame.java | Ornek1.java | JavaApplication136.java

Source | History

```
1
2   package javaapplication135;
3   import java.awt.FlowLayout;
4   import javax.swing.*;
5   public class Ornek1 extends JFrame {
6           public Ornek1() {
              setTitle("TARIM MAKİNALARI VE TEKNOLOJİLERİ MÜHENDİSLİĞİ");
              setSize(500, 200);
              setLocationRelativeTo(null);
              setDefaultCloseOperation(EXIT_ON_CLOSE);
              setLayout(new FlowLayout() );
12  JButton b1 = new JButton( "Pulluk" );
13  JButton b2 = new JButton( "Traktör" );
14  JButton b3 = new JButton( "Biçerdöver" );
    add( b1 );
    add( b2 );
    add( b3 );
18
19          }
20
21          public static void main(String[] args) {
            SwingUtilities.invokeLater(new Runnable() {
                  public void run() {
24
25                    Ornek1 ex = new Ornek1();
26                    ex.setVisible(true);
27                }
28            });
29          }
30  }
```

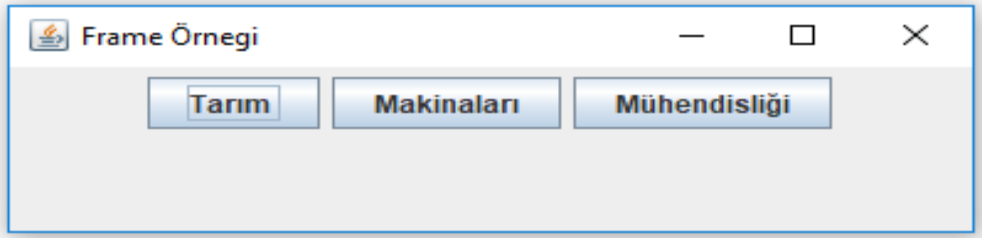


Örnek 3: Form üzerine buton yerleştiren Applet kodu.

Start Page | NewJFrame.java | NewJFrame.java | NewJFrame.java | Ornek1.java | JavaApplication136.j

Source | History

```
2   package javaapplication137;
3
4   import java.awt.FlowLayout;
5   import javax.swing.*;
6   public class Ornek3 {
7
8     public static void main(String[] args) {
9
10       JFrame Cerceve = new JFrame("Frame Örnegi");
11    Cerceve.setLayout(new FlowLayout());
12    JButton b1 = new JButton("Tarım ");
13    JButton b2 = new JButton("Makinaları");
14    JButton b3 = new JButton("Mühendisliği");
15    // Butonlar çerçeveye yerleştirilir
16    Cerceve.add(b1);
17    Cerceve.add(b2);
18    Cerceve.add(b3);
19    Cerceve.setVisible(true);// Çerçeveyi ekranda gösterir
20    Cerceve.pack(); // Nesneleri çerçeve boyutuna göre otomatik
21    // boyutlandırır
22
23        }
24
25    }
26
```



javaapplication137.Ornek3 > main

Output - JavaApplication137 (run)

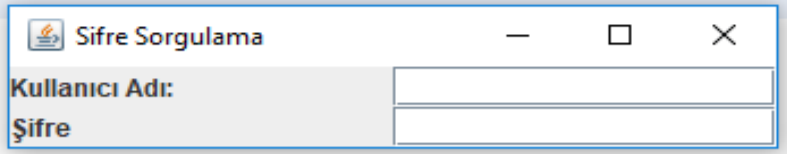
```
run:
```

Örnek 3: GridLayout ile Şifre sorgulama ekranı örneği.

Start Page | NewJFrame.java | NewJFrame.java | NewJFrame.java | Ornek1.java | JavaApplication136.java

Source | History

```
1
2   package javaapplication137;
3
4   import java.awt.GridLayout;
5   import javax.swing.*;
6   public class Password extends JPanel {
7   public static void main(String[] args) {
8   JTextField password, username;
9   password = new JTextField(15);
10  username = new JTextField(15);
11  JFrame Cerceve = new JFrame("Sifre Sorgulama");
12  Cerceve.setLayout(new GridLayout(2, 2));
13  // 2 sutunlu 1. satır
14  Cerceve.add(new JLabel("Kullanıcı Adı:"));
15  Cerceve.add(username);
16  // 2 sutunlu 2. satır
17  Cerceve.add(new JLabel("Şifre"));
18  Cerceve.add(password);
19  Cerceve.setSize(300, 200); //Çerçeve boyutu ayarlanıyor
20  Cerceve.setVisible(true); //Çerçeve ekranda gösteriliyor
21  Cerceve.pack(); //Nesneler çerçeve boyutuna göre yerleştiriliyor
22  }
23  }
24
```



Output - JavaApplication137 (run)

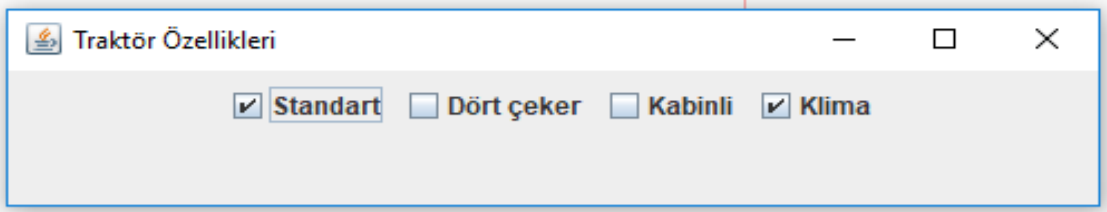
run:

Örnek 4: GridLayout ile traktör özellikleri sorgulama ekranı örneği.

Start Page | NewJFrame.java | NewJFrame.java | NewJFrame.java | Tractor.java | JavaApplication138.java

Source | History

```
1
2   package javaapplication135;
3   import java.awt.FlowLayout;
4   import javax.swing.*;
5   public class Tractor extends JFrame {
6
7
8           public static void main(String[] args) {
9           JFrame frame = new JFrame("Traktör Özellikleri");
10          frame.setLayout(new FlowLayout());
11          JCheckBox tral = new JCheckBox("Standart");
12          tral.setSelected(true); //
13          JCheckBox tra2 = new JCheckBox("Dört çeker");
14          JCheckBox tra3 = new JCheckBox("Kabinli");
15          JCheckBox tra4 = new JCheckBox("Klima");
16          tra4.setSelected(true); //
17          frame.add(tral);
18          frame.add(tra2);
19          frame.add(tra3);
20          frame.add(tra4);
21          frame.pack(); //
22          frame.setVisible(true);
23  }
24  }
25
```



Find: | Previous | Next

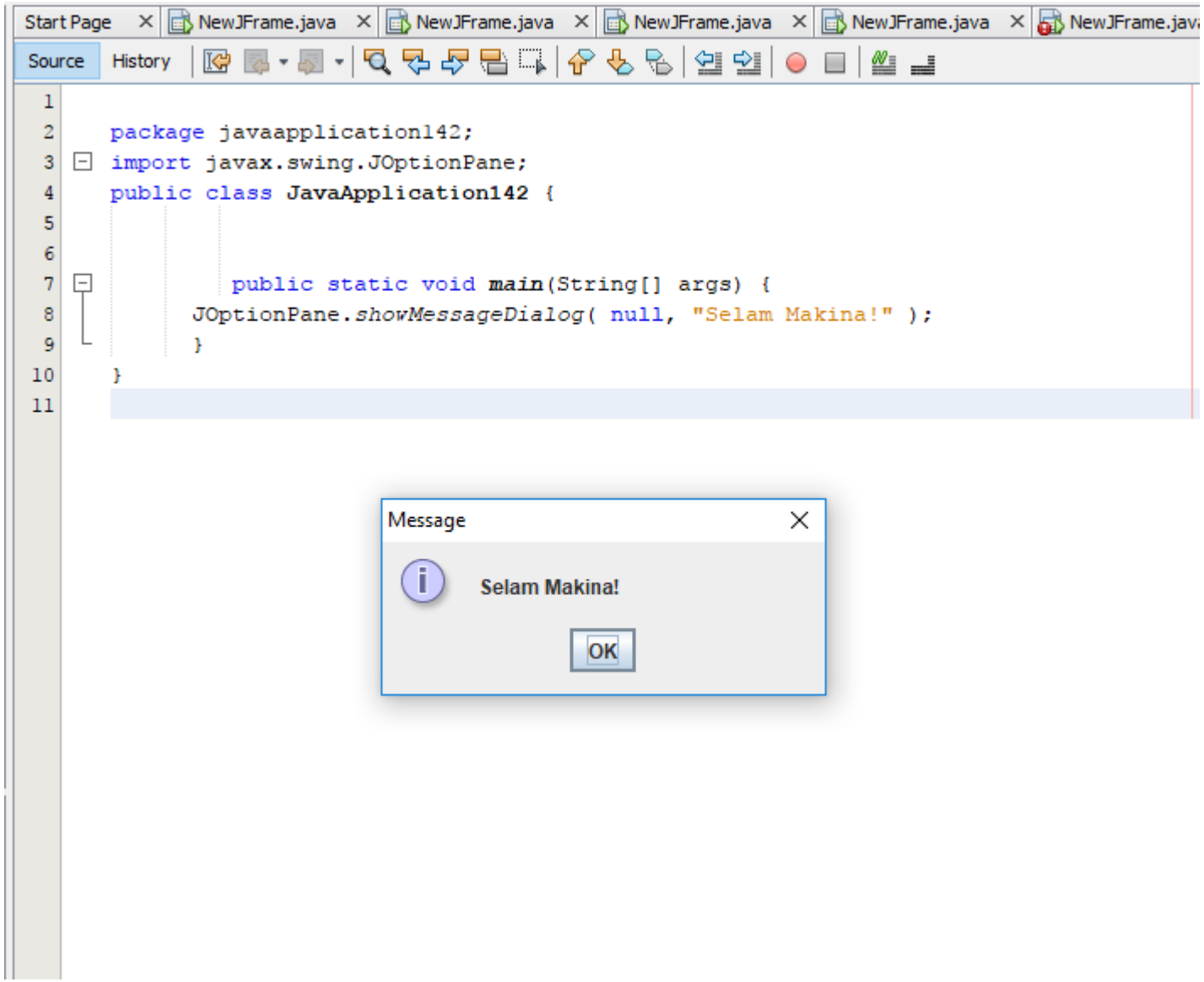
javaapplication135.Tractor > main

Output - JavaApplication135 (run)

```
run:
```

JOptionPane : Mesaj kutusu açmak için kullanılan sınıftır.

Kullanımı: import javax.swing.JOptionPane;



Olay Dinleyicilerini Anlamak

Olaylar (events) GUI programlamanın temelini oluştururlar. Her GUI uygulaması bileşenler ve onlara atanmış olaylarla tümleşik hareket eder. Bundan dolayı olay modelini üç bölüme ayırarak inceleyebiliriz:

- Olay Kaynağı (Event Source)
- Olay Nesnesi (Event Object)
- Olay Dinleyici (Event Listener)

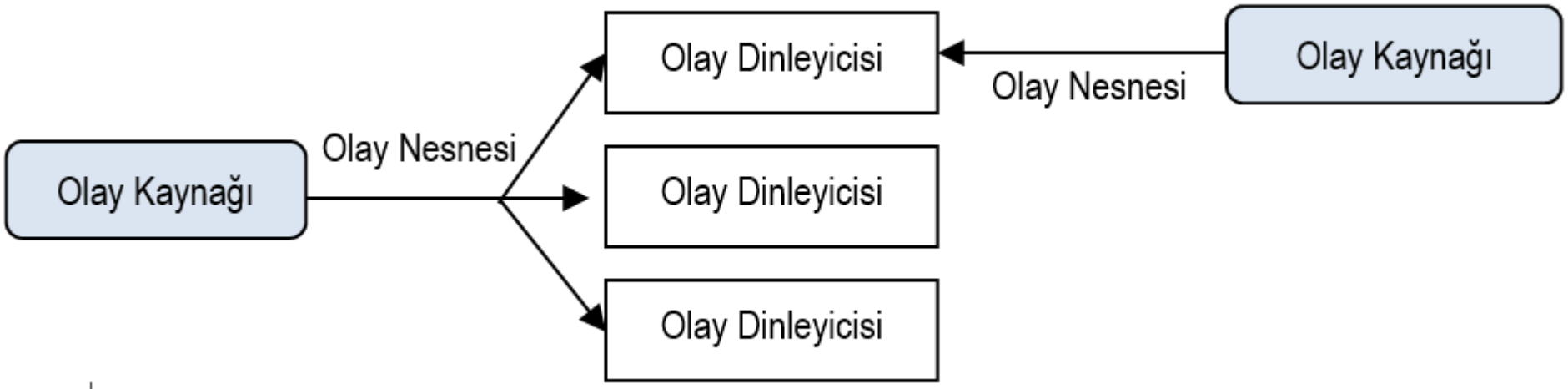
- **Olay Kaynağı**, hangi nesnenin durumunda değişiklik olduğunu gösteren ve olayı oluşturan;

- **Olay Nesnesi** ne tür bir olay oluştuğu ve olayın özelliklerini gösteren;

- **Olay Dinleyici** ise beklenen olayın gerçekleşip gerçekleşmediğini gösteren bölümlerdir.

- Java Platformun her bir bileşen için, o bileşenin işlev ve biçimine göre farklı bir olay dinleyicisi ve olay niteliğine göre de, farklı özelliklere sahip olay nesneleri sunmaktadır.

- Java’da olay modeli oldukça güçlü ve esnektir.
- Birçok olay dinleyici nesne, birçok olay kaynağı nesneden tüm olay türlerini dinleyebilir.
- Örneğin bir uygulama her olay kaynağı için bir dinleyici yaratabilir ya da tüm kaynaklardan üretilen tüm olaylar için tek bir dinleyiciye sahip olabilir.
- Yine bir uygulama bir olay kaynağından üretilen bir olay türü için birden fazla dinleyiciye sahip olabilir.



- Her olay dinleyici yordam, `EventObject` sınıfından türetilen bir nesne, tekil bir argümana sahiptir.
- Bu her zaman `EventObject`'den gelse de bu nesnenin türü genellikle oldukça kesin bir biçimde belirlenir.
- Örneğin fare olaylarını yönetecek yordamlar için argüman, `EvenObject`'in dolaylı alt sınıfı `MouseEvent`'in bir örneğidir.
- Bu `EventObject` sınıf oldukça kullanışlı bir yordam sunar. Bu yordam `Object getSource()` yordamıdır ve olayı tetikleyen nesneyi döndürür.
- `getSource` yordamının bir `Object` döndürdüğüne dikkat edilmelidir. Olay sınıfları bazen `getSource`'a benzeyen; ancak daha sınırlı dönüş türlerine sahip yordamlar sunarlar.
- `ComponentEvent` sınıfı, tıpkı `getSource` gibi olayı tetikleyen nesneyi döndüren bir `getComponent` yordamı sunar. Aradaki fark ise `getComponent`'ın her zaman bir `Component` döndürmesidir.

- Çoğu kez bir olay sınıfı olay hakkında bilgi döndüren yordamlar sunar. Örneğin olayın nerede meydana geldiğini öğrenmek için `MouseEvent` nesnesini incelediğimizde, kullanıcının kaç kez fareye tıkladığı, hangi tamamlayıcı tuşa bastığı vb. gibi bilgileri de alabildiğimizi görürüz.

Olay Bağdaştırıcıları

Bazı dinleyici arayüzleri bir yordamdan daha fazlasını içerir. Ör-neğin MouseListener arayüzü beş yordam içerir:
mousePressed,

mouseRe-leased,

mouseEntered,

mouseExited ve

mouseClicked.

Yalnızca fare tıklarıyla ilgileniyor olunsa bile, beş tane MouseListener yordamı kullanılmalıdır. Yönetmek istemediğiniz olay yordamları boş kalabilirler. Bu-nun için aşağıdaki örneğe bakabilirsiniz:

```
public void mouseClicked(MouseEvent e) {
    }
    public void mousePressed(MouseEvent e) {
    }
```

Swing Bileşenlerinin Desteklediği Diğer Dinleyiciler

Tablo 1. Swing bileşenleri ve bu bileşenlerce desteklenen belirli dinleyicilerin listesi

Bileşen	Dinleyici							
	action	caret	change	document, undoable edit	item	list selection	window	other
button	✓		✓		✓			
check box	✓		✓		✓			
color chooser			✓					
combo box	✓				✓			
dialog							✓	
editor pane		✓		✓				hyperlink
file chooser	✓							
formatted text field	✓	✓		✓				
frame							✓	
internal frame								internal frame
list						✓		list data
menu								menu
menu item	✓		✓		✓			menu key menu drag mouse
option pane								
password field	✓	✓		✓				
popup menu								popup menu
progress bar			✓					
radio button	✓		✓		✓			
slider			✓					
spinner			✓					
tabbed pane			✓					
table						✓		table model table column model cell editor
text area		✓		✓				
text field	✓	✓		✓				
text pane		✓		✓				hyperlink
toggle button	✓		✓		✓			
tree								tree expansion tree will expand tree model tree selection
viewport (scrollpane tarafından kullanılan)			✓					

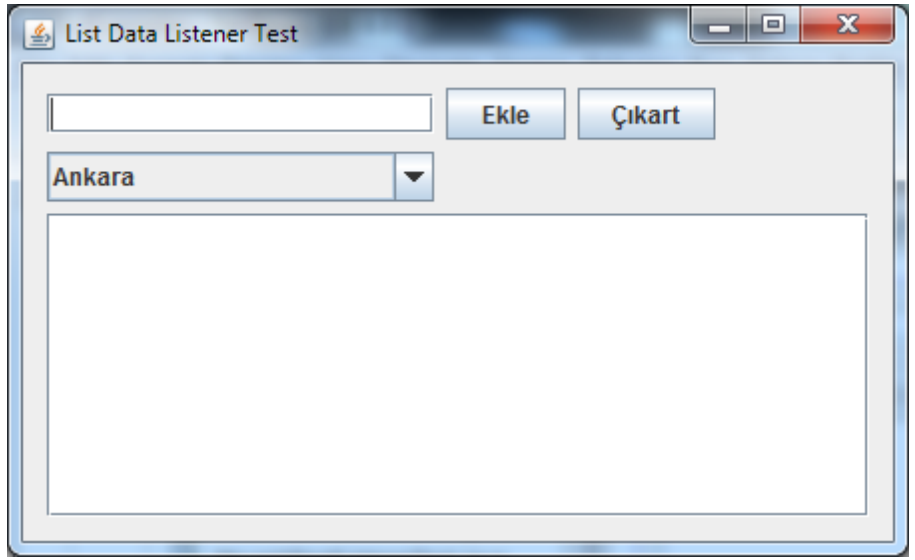
Bir List Data Listener Yazma

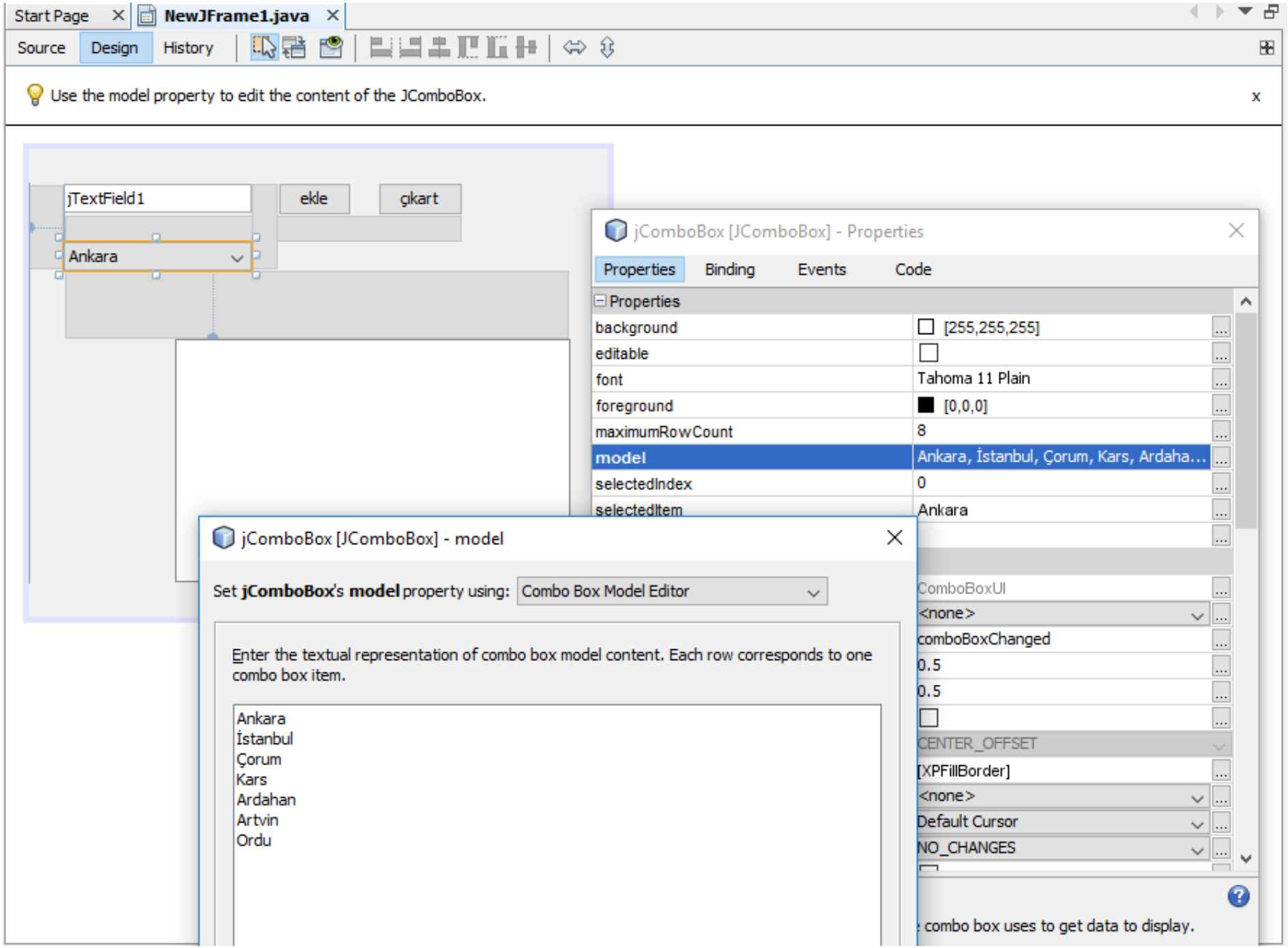
Liste veri olayları değişken bir listenin içeriği değiştiğinde oluşur. Modellerin –bileşenler değil- bu olayları üretmesinden dolayı, liste veri dinleyicileri liste modellerine atanır.

Eğer açık bir biçimde değişken bir liste modeli oluşturulmamışsa, bu liste modeli değişken olmadığı için bu tür olaylar üretemeyecektir.

Tuş dinleyicilerinin nasıl kullanıldığını aşağıdaki ekran görünümünde olduğu gibi `ListDataListenerTest` adlı bir `JForm` oluşturuyor ve üzerine aşağıdaki gibi bir tane `JTextArea`, bir tane `JComboBox`, bir tane `JTextField` ve iki tane de `JButton` ekliyoruz.

Bu örnekte `JTextArea`: `jtaEventTest`, `JComboBox`: `jcbSimpleList`, `JTextField`: `jtfItemName` ve Ekle: `jbAdd`, Çıkart: `jbRemove` olarak adlandırılmıştır.



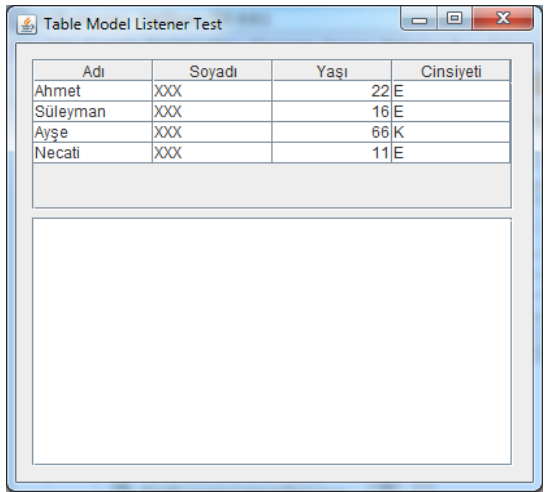

Start Page
NewJFrame1.java
Source
Design
History
Use the model property to edit the content of the JComboBox.
jTextField1
ekle
çıkart
Ankara
jComboBox [JComboBox] - Properties
Properties
Binding
Events
Code
Properties
background
[255,255,255]
editable
font
Tahoma 11 Plain
foreground
[0,0,0]
maximumRowCount
8
model
Ankara, İstanbul, Çorum, Kars, Ardaha...
selectedIndex
0
selectedItem
Ankara
ComboBoxUI
<none>
comboBoxChanged
0.5
0.5
CENTER_OFFSET
[XPFillBorder]
<none>
Default Cursor
NO_CHANGES
combo box uses to get data to display.
jComboBox [JComboBox] - model
Set jComboBox's model property using:
Combo Box Model Editor
Enter the textual representation of combo box model content. Each row corresponds to one combo box item.
Ankara
İstanbul
Çorum
Kars
Ardahan
Artvin
Ordu

Bir TableModelListener Yazma

- Her bir JTable nesnesi, gösterdiği veriyi yöneten bir tablo modeli içerir. Normalde eğer belirlenen özel bir TableModel yoksa bir JTable nesnesi bir DefaultTable nesnesini miras alır; ancak bu model nor-malde yalnızca stringleri yönetir.
- Nesneleri yönetmek, hesaplamalar gerçekleştirmek, veritabanı ya da başka uygulamalardan veri almak için TableModel arayüzünü uygulayan, kendi özel TableModel nesnesinin tasarlanması gerekmektedir.

- Bir tablo model nesnesi aracılığıyla yönetilen verilerin değişimini denetlemek için, JTable sınıfının, olayları yakalamak için addTableMo-delListener()'ı çağıran ve olayları yanıtlamak için tableChanged()'ın üzerine bildirilen TableModelListener arayüzünü uygulaması gerekir.

Şimdi tablo model dinleyicilerinin nasıl kullanıldığını bir uygula-mayla daha yakından inceleyelim. Bunun için aşağıdaki ekran görü-nümünde olduğu gibi, TableModelListenerTest adlı bir JForm oluşturu-yor ve üzerine bir tane JTextArea ve bir tane de JTable ekliyoruz. Bi-zim örneğimizde JTextArea: jtaEventTest ve JTable bileşeni de jtbSimpleList olarak adlandırılmıştır.



Bir WindowListener Yazma

Bu dinleyiciler: WindowListener, WindowFocusListener ve WindowState-Listener'dir. Bu üç dinleyicinin tümü WindowEvent nesnesini yönetir. Bu üç olay dinleyicinin tümünde içerilen yordamlar soyut WindowAdap-ter sınıfı aracılığıyla uygulanırlar.

Bir pencereye (çerçeve ya da iletişim penceresi gibi) uygun bir dinleyici atandığında, pencere olayları, pencere etkinliği ya da duru-mu hemen belirdikten sonra üretilirler.

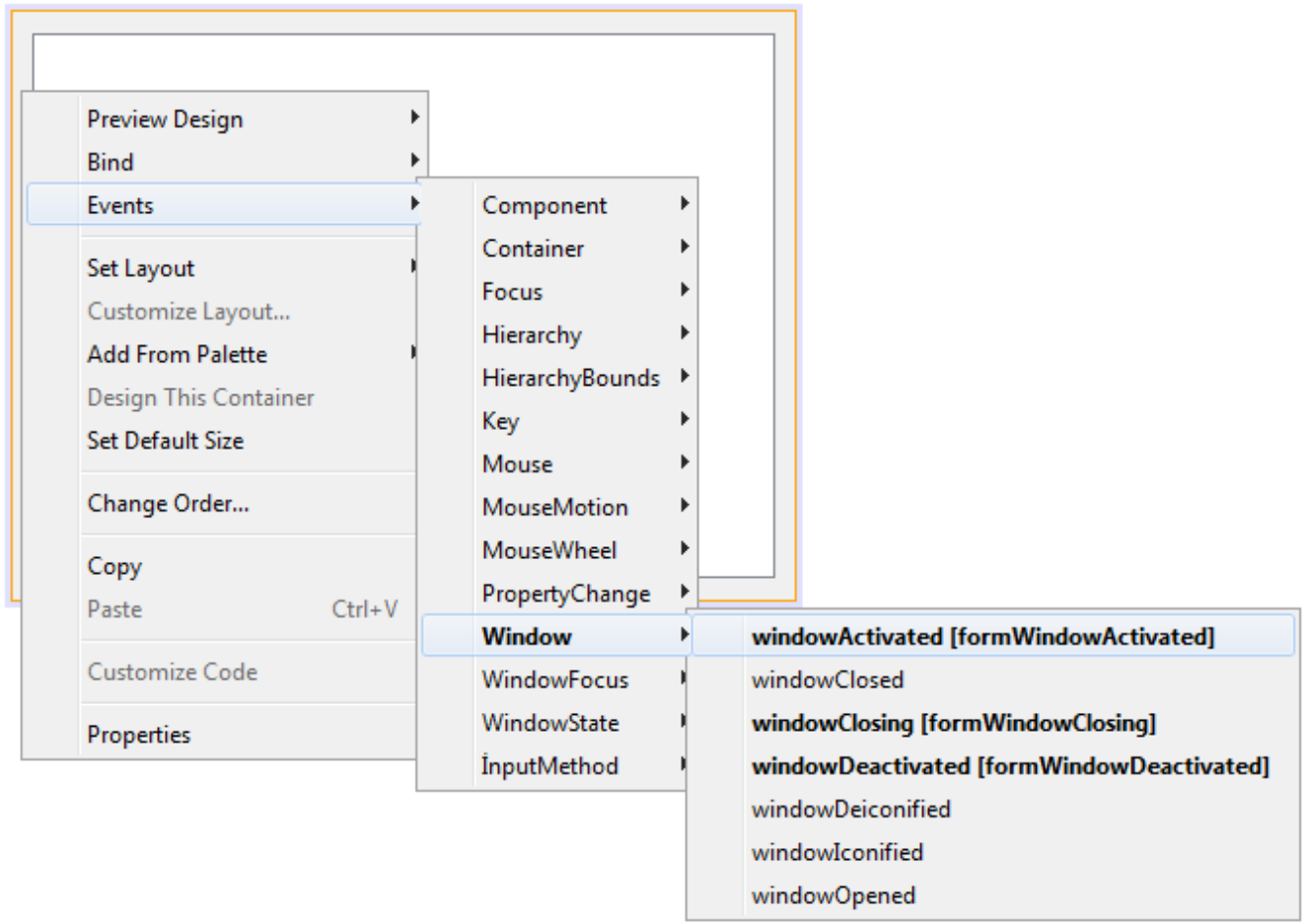
Aşağıda pencere etkinlikleri ve durumlarının bir pencere olayını nasıl önceledi̇ği görülebilir:

- Bir pencere açma — Bir pencerenin ilk kez gösterilmesi.
- Bir pencere kapama — Bu pencerenin ekrandan kaldırılması.
- Bir pencereyi simge durumundan küçültme — Bu pencerenin masaüstüne simge durumunda küçültülmesi.
- Pencereyi önceki boyuta getirme — Bu pencerenin yeniden eski konumuna döndürülmesi.
- Odaklanılmış pencere
- Etkin pencere (çerçeve ya da iletişim penceresi) — Bu pencere ya odaklanılmış bir penceredir ya da odaklanılmış bir pencereye sa-hiptir.
- Pasif pencere — Bu pencere odağını yitirmiştir.
- Pencereyi ekranı kapla konumuna getirme — Pencerenin boyu-tunu izin verilen en yüksek değere getirme, ya dikey ya yatay ya da her iki yöne doğru.

Pencere olaylarının izlenebileceği ve yönetilebileceği olay yordamlar oluşturulabilir.

İlgili sınıfa yalnızca gereksinim duyulan olay yordamları eklenmiş olacaktır.

Bunun için alttaki ekran görünümünde de görebileceği üzere; çizgiyle belirlenmiş olan çerçeve üzerinde sağ tık yapıp, açılan pence-reden Events -> Window -> windowActivated seçeneğini seçilmelidir.



WindowListener APIsi

Pencere dinleyici APIsi üç pencere dinleyici ara yüzü ve `WindowEvent` sınıfından oluşur. İçerdiği yordamlar aşağıda verilmiştir:

- WindowListener Arayüzü
- WindowFocusListener Arayüzü
- WindowStateListener Arayüzü
- WindowEvent Sınıfı

Bu üç ara yüzden gelen yordamlar `WindowAdapter` sınıf aracılığıyla kullanılabilir.

WindowListener Ara Yüzü

Yordam	Kullanım Amacı
`windowOpened(WindowEvent)`	Dinlenen pencere ilk kez gösterildikten hemen sonra çağrılır.
`windowClosing(WindowEvent)`	Kullanıcı dinlenen pencereyi kapatmak istediğinde çağrılır. Pencereyi gerçekten kapatmak için, dinleyicinin pencere `dispose` ya da `setVisible(false)` yordamlarını çağırması gerekir.
`windowClosed(WindowEvent)`	Dinlenen pencere kapatıldıktan hemen sonra çağrılır.
`windowIconified(WindowEvent)` `windowDeiconified(WindowEvent)`	Dinlenen pencerenin sırasıyla, simge durumuna küçültüldüğü ya da yeniden önceki boyutuna getirilmesinden hemen sonra çağrılır.
`windowActivated(WindowEvent)` `windowDeactivated(WindowEvent)`	Dinlenen pencerenin, sırasıyla etkin olması ya da etkinliğini yitirmesinden hemen sonra çağrılır.

WindowFocusListener Arayüzü

Yordam	Kullanım Amacı
`windowGainedFocus(WindowEvent)` `windowLostFocus(WindowEvent)`	Dinlenen pencere odaklanıldığı ya da pencere odağını yitirdikten hemen sonra çağrılır.

WindowStateListener Arayüzü

Yordam	Kullanım Amacı
`windowStateChanged(WindowEvent)`	Dinlenen pencerenin durumu, simge durumunda küçültülerek, önceki boyuta getirilerek ya da ekranı kapla durumuna getirilerek vb. değiştirildikten hemen sonra çağrılır. Döndürülecek olası değerler aşağıdaki gibidir: `NORMAL`: Herhangi bir durum değişikliği yok. `ICONIFIED`: Simge durumunda küçültülmüş. `MAXIMIZED_HORIZ`: Yatay olarak ekranı kaplamış. `MAXIMIZED_VERT`: Dikey olarak ekranı kaplamış. `MAXIMIZED_BOTH`: Her iki yöne doğru ekranı kaplamış. Bazı pencere yöneticileri, her iki yöne doğru ekranı kaplama özelliğini desteklerken, yatay ve dikey kaplama özelliklerini desteklemez. `Java.awt.Toolkit, isFrameStateSupported(int)` yordamıyla hangi pencere durumlarının desteklendiği öğrenilebilir.