Görsel Programlama

DERS 08

Java da Olay Yönetimi (Event Handling)

Program çalışırken kullanıcı tarafından gerçekleştirilen tüm hareketlere olay(event) denilir. Kullanıcının fareyi hareket ettirmesi, tuşlara basması, bir butona basması, pencerenin kapatılması vb.

Java da bir olay gerçekleştirilirken 4 rol bulunmaktadır:

Java da Olay Yönetimi (Event Handling)

1.Olay Kaynağı (Event Source): Olayı gerçekleştiren bileşendir. Button, Mouse vb. Bu kaynak olay nesnesini oluşturur.

2.Olay Nesnesi (Event Object): Olay zamanında oluşturulur ve olay ile ilgili bilgileri tutan nesnedir.

3.Olay Dinleyicileri (Event Listener) :Belirli bir amaç için yazılmış olan dinleyici , o olay gerçekleştiği zaman bilgilendirilir.

4.Olay İşleyicisileri (Event Handler): Olay dinleyicisi tarafından olay yakalandığında ilgili kodun çalıştırılacağı yerdir.

Java da Olay Yönetimi (Event Handling)

Bir JButton (düğme) için olay dinleyici ve olay kodu örneği oluşturalım.

1. Olay kaynağı oluşturulur: JButton btn= new JButton(“Dialog göster”);

2. Kaynaktan sonra olay dinleyici oluşturulur. Buton'a basılma olayını dinleyebilmek için ActionListener arayüzünü implement eden bir sınıfa ihtiyacımız vardır.

```
class Dinleyicim implement ActionListener{
        //olay gerçekleşince çalışan metot
        public void actionPerformed(ActionEvent e){
                //olayda yapılması istenilen kod.
        }
}
```

3.Yazmış olduğumuz dinleyici olay kuyruğuna eklenir.

```
        btn.addActionListener(new Dinleyicim());
```

ActionListenerOrnek.java

```java
package com.comu.gorsel_programlama.ders08;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class ActionListenerOrnek extends JFrame {
    public ActionListenerOrnek() {
        createGUI();
    }

    private void createGUI() {
        JButton btn = new JButton("Dialog Göster");
        this.getContentPane().add(btn);
        //anonymous class
        ActionListener dinleyici = new ActionListener(){
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(null, "Olay bitti");
            }
        };
        btn.addActionListener(dinleyici);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(150,100);
    }
    public static void main(String[] args) {
        ActionListenerOrnek app = new ActionListenerOrnek();
        app.setVisible(true);
    }
}
```

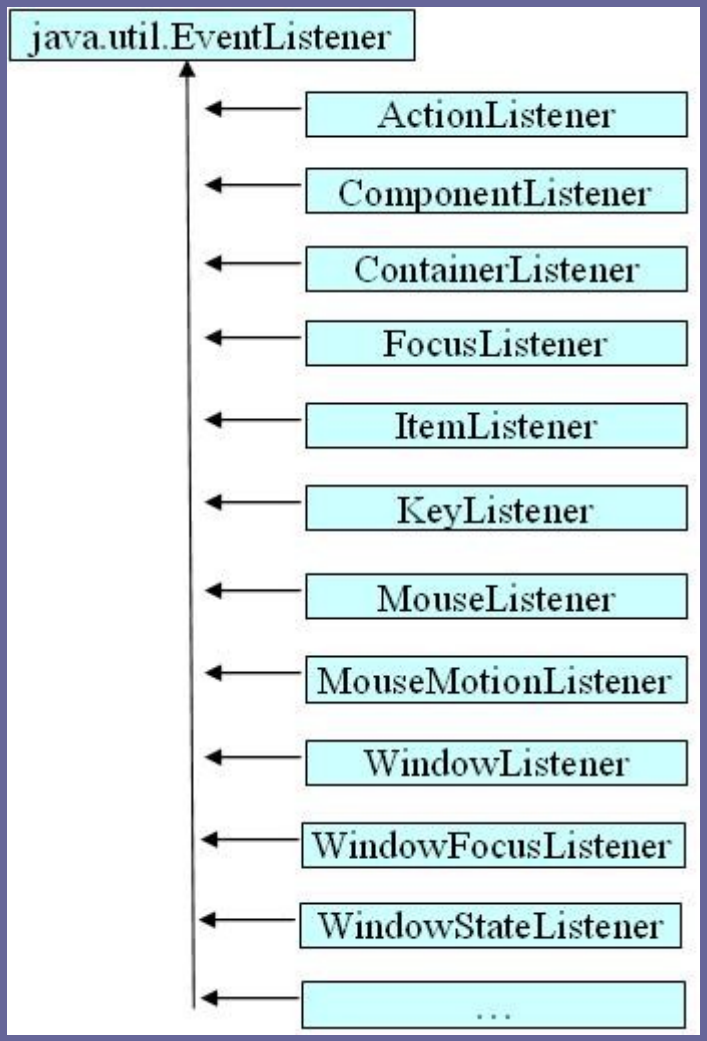
ActionListenerOrnek.java

```
package com.comu.gorsel_programlama.ders08;
import java.awt.event.ActionEvent;
public class ActionListenerOrnek extends JFrame {
    public ActionListenerOrnek() {
        createGUI();
    }
    private void createGUI() {
        JButton btn = new JButton("Dialog Göster");
        this.getContentPane().add(btn);
        //anonymous class
        ActionListener dinleyici = new ActionListener(){
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(null, "Olay bitti");
            }
        };
        btn.addActionListener(dinleyici);
        btn.addActionListener(new Dinleyicim());
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(150,100);
    }
    public static void main(String[] args) {
        ActionListenerOrnek app = new ActionListenerOrnek();
        app.setVisible(true);
    }
}
class Dinleyicim implements ActionListener{

    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(null, "Olay bitti Dinleyicim");
    }
}
```

Java da Olay Yönetimi (Event Handling)

Farklı olaylar için farklı dinleyiciler kullanılır.



Java da Olay Yönetimi (Event Handling)

ActionListener : Button a tıklanınca , metuitem seçilince , TextField da enter a basılınca

ComponentListener: Component görünmez olunca , görünür olunca , taşınırsa yada yeniden boyutlandırılınca

ContainerListener: Bu olay container a bir component eklendiği yada silindiği zaman oluşur.

FocusListener: Bir bileşen klavye focus unu aldığında ve kaybettiğinde oluşur.

ItemListener: Olayı ItemSelectable interface ini gerçekleştiren sınıflarda oluşmaktadır. Mesela CheckBox ,checkboxmetuitem , combobox larda.

KeyListener: Kullanıcı klavye tuşlarına basınca oluşur.

MouseListener: Mouse göstericisi bir componentin alanı içine girdiğinde , çıktığında , mouse butonlarına basılınca .

MouseMotionListener: Mouse göstericisi ekranın üzerinde sürüklenince oluşur.

Java da Olay Yönetimi (Event Handling)

WindowListener: Pencere küçültüldüğünde , büyütüldüğünde , aktif olduğunda yada pasif olduğunda , açıldığında ve kapatıldığında.
WindowFocusListener: Pencereye odaklandığında(focus) , yada odağı kaybettiğinde.
WindowStateListener: Pencere icon şeklinde mi , değil mi , büyütülmüş mü , normal halinde mi ? Gibi olayları dinler

Daha birçok olay dinleyici arayüzleri(interface) vardır. Bunlar hakkında daha ayrıntılı bilgi java.sun.com/tutorial dan elde edilebilir.

Örnek-2

ActionListenerOrnek2.java

```
import java.awt.Container;
public class ActionListenerOrnek2 extends JFrame {
    private JTextField textField1, textField2, textField3;

    private JPasswordField passwordField;

    public ActionListenerOrnek2() {
    private void createGUI() {
        Container container = this.getContentPane();
        container.setLayout(new FlowLayout());
        textField1 = new JTextField(10);
        container.add(textField1);
        textField2 = new JTextField("Buraya yazi girin");
        container.add(textField2);
        textField3 = new JTextField("Değiştirilemeyen TextField");
        textField3.setEditable(false);
        container.add(textField3);
        passwordField = new JPasswordField("Gizli Yazı");
        container.add(passwordField);
        TextFieldHandler handler = new TextFieldHandler();
        textField1.addActionListener(handler);
        textField2.addActionListener(handler);
        textField3.addActionListener(handler);
        passwordField.addActionListener(handler);
    }
    private class TextFieldHandler implements ActionListener {

    public static void main(String[] args) {
        ActionListenerOrnek2 app = new ActionListenerOrnek2();
        app.setSize(325, 100);
        app.setVisible(true);
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

```
private class TextFieldHandler implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String string = "";
        if (e.getSource() == textField1) {
            string = "textField1:" + e.getActionCommand();
        } else if (e.getSource() == textField2) {
            string = "textField2:" + e.getActionCommand();
        } else if (e.getSource() == textField3) {
            string = "textField3:" + e.getActionCommand();
        } else if (e.getSource() == passwordField) {
            string = "passwordField:"
                    + new String(passwordField.getPassword());
        }
        JOptionPane.showMessageDialog(null, string);
    }

}
```

Örnek-3

MouseOlayOrnek.java

```
package com.comu.gorsel_programlama.ders08;

import java.awt.BorderLayout;

public class MouseOlayOrnek extends JFrame implements MouseListener {

    private JLabel durum;
    public MouseOlayOrnek() {
        super("Mouse Olay Dinleyici");createGUI();
    }
    private void createGUI() {
        durum = new JLabel();
        this.getContentPane().add(durum,BorderLayout.LINE_START);
        this.addMouseListener(this);
        this.setSize(250,100);
    }
    public void mouseClicked(MouseEvent e) {
    public void mouseEntered(MouseEvent e) {
    public void mouseExited(MouseEvent e) {
    public void mousePressed(MouseEvent e) {
    public void mouseReleased(MouseEvent e) {
    public static void main(String[] args) {
        MouseOlayOrnek app = new MouseOlayOrnek();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        app.setVisible(true);
    }
}
```

Java da Olay Yönetimi (Event Handling)

Interface i gerçekleştiren sınıf arayüz içindeki tüm soyut metotları gerçekleştirmelidir. Yukarıdaki örnekte göründüğü gibi ihtiyaç olmasada metotlar içleri boş olsalarda yazılmalıdırlar.

Bu olayı çözmek için Event Adapter (olay adaptörleri) geliştirilmiştir. Bunlar listener ların tüm metotlarını içlerinde barındıran sınıflardır. Bu sınıfların istediğimiz olayının üzerine yükleme(overriding) ile kullanırız.

Örnek-4

MouseOlay2.java

```
package com.comu.gorsel_programlama.ders08;

import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

import javax.swing.JFrame;

public class MouseOlay2 extends JFrame {
    int xPos,yPos;
    public MouseOlay2() {
        createGUI();
    }
    private void createGUI() {
        this.addMouseListener(new FareOlayiAdaptoru());
        this.setSize(400,100);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    private class FareOlayiAdaptoru extends MouseAdapter{..
    public static void main(String[] args) {
        MouseOlay2 app = new MouseOlay2();
        app.setVisible(true);
    }
}
```

Örnek-4

MouseOlay2.java

```
package com.comu.gorsel_programlama.ders08;
import java.awt.event.MouseAdapter;
public class MouseOlay2 extends JFrame {
    int xPos,yPos;
    public MouseOlay2() {createGUI();}
    private void createGUI() {
        this.addMouseListener(new FareOlayiAdaptoru());
        this.setSize(400,100);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    private class FareOlayiAdaptoru extends MouseAdapter{
        @Override
        public void mouseClicked(MouseEvent e) {
            xPos = e.getX();
            yPos = e.getY();
            String baslik = "Tıklama sayisi:"+e.getClickCount();
            if (e.isMetaDown()){
                baslik +=" sag fare tusu";
            }else if (e.isAltDown()){
                baslik +=" orta fare tusu";
            }else{
                baslik +="sol fare tusu";
            }
            setTitle(baslik);
            repaint();
        }
    }
    public static void main(String[] args) {
        MouseOlay2 app = new MouseOlay2();
        app.setVisible(true);
    }
}
```

Örnek-5

PencereAdaptorOrnek.java

```
package com.comu.gorsel_programlama.ders08;
import java.awt.event.WindowAdapter;
public class PencereAdaptorOrnek extends JFrame {
    public PencereAdaptorOrnek() {
        createGUI();
    }
    private void createGUI() {
        this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        this.setSize(350,100);
        this.addWindowListener(new WindowAdapter(){

            @Override
            public void windowClosing(WindowEvent e) {
                int durum =0;
                Object[] a = {"Evet","Hayır"};
                durum = JOptionPane.showOptionDialog(e.getWindow(),
                        "Programı Kapatmak İstediğinizden Emin misiniz?",
                        "Doğrulama", JOptionPane.YES_NO_OPTION,
                        JOptionPane.INFORMATION_MESSAGE, null,a,"Hayır");
                if (durum ==JOptionPane.YES_OPTION){
                    System.exit(0);
                }
            }
        });
    }
    public static void main(String[] args) {
        PencereAdaptorOrnek app = new PencereAdaptorOrnek();
        app.setVisible(true);
    }
}
```

Görsel Programlama

DERS 08