Université de Bouira Département d'Informatique		Module : Interactions Homme Machine (IHM) Niveau : L3 Année : 2023-2024

TP3

- Interface **ActionListener** is used to alert whenever the user **clicks** on **something (** button or menu item**)**. It has only one method: actionPerformed().
- **public** abstract **void actionPerformed(ActionEvent e);**

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
...
    JButton button = new JButton("Click Me");
    button.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

                    ..............Actions.................
    }
    });
```

- Interface **ItemListener** is used to alert whenever the user **clicks** on an **item (**checkbox**)**. It has only one method: itemStateChanged().

```
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;

...
    checkBox.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if(checkBox.isSelected())
            label.setText("Check Box is checked");
            else
            label.setText("Check Box is not checked");
    }
    });
```

- Interface **KeyListener** is used to alert any user interaction on the **keyboard**.

The **KeyListener** interface contains three functions that you must override when creating an object from it:

- **KeyTyped(KeyEvent e)** function: It is called after the character that the user clicked on the keyboard is printed.
- **KeyPressed(KeyEvent e)** function: It is called when the user clicks on any button on the keyboard and before lifting his finger from it.
- **keyReleased(KeyEvent e)** function**:** Called after the user removes his finger from the button he clicked on the keyboard.

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;


    textField.addKeyListener(new KeyListener() {
        @Override
            public void keyTyped(KeyEvent e) {
                    ……….
            }
        @Override
            public void keyPressed(KeyEvent e) {
                    if(e.getKeyCode() == KeyEvent.VK_ENTER)
                    labelResult.setText(textField.getText());
            }
        @Override
            public void keyReleased(KeyEvent e) {
                    ……….
            }
    });
```

- The Java **MouseListener** is notified whenever you change the state of mouse. It is notified against MouseEvent. The MouseListener interface is found in java.awt.event package. It has five methods.
  - public abstract void **mouseClicked**(MouseEvent e);
  - public abstract void **mouseEntered**(MouseEvent e);
  - public abstract void **mouseExited**(MouseEvent e);
  - public abstract void **mousePressed**(MouseEvent e);
  - public abstract void **mouseReleased**(MouseEvent e);

```
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
...
    frame.addMouseListener(new MouseListener() {
        @Override
            public void mouseClicked(MouseEvent e) {
                mousePosition.setText(" Mouse Position: ("+e.getX()+", "+e.getY()
                +")");
                mouseStatus.setText(" Mouse Status: Mouse Clicked");
            }
        @Override
            public void mousePressed(MouseEvent e) {
                mouseStatus.setText(" Mouse Status: Mouse Pressed");
            }
        @Override
            public void mouseReleased(MouseEvent e) {
                mouseStatus.setText(" Mouse Status: Mouse Released");
            }
        @Override
            public void mouseEntered(MouseEvent e) {
                mouseStatus.setText(" Mouse Status: Mouse Entered");
            }
        @Override
            public void mouseExited(MouseEvent e) {
                mouseStatus.setText(" Mouse Status: Mouse Exited");
            }
    });
```

- The Java **MouseMotionListener** is notified whenever you move or drag mouse. It is notified against MouseEvent. The **MouseMotionListener** interface is found in java.awt.event package. It has two methods.

  - public abstract void **mouseDragged(MouseEvent e);**
  - public abstract void **mouseMoved(MouseEvent e);**

```
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionListener;

...

    frame.addMouseMotionListener(new MouseMotionListener() {
        @Override
            public void mouseDragged(MouseEvent e) { }
        @Override
            public void mouseMoved(MouseEvent e) {
            mousePosition.setText(" Mouse Position: ("+e.getX()+", "+e.getY() +")");
        }
    });
```

- The Interface **FocusListener** is used to indicate the current element that the user is interacting with and which he can control using the keyboard.FocusListener contains two functions that you must override when creating an object from it:
- **focusGained(FocusEvent e)** function: Called when the user clicks inside an element.
- **focusLost(FocusEvent e)** function: It is called when the user clicks outside the element he was working with

```
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;

...
    FocusListener fl = new FocusListener() {
    @Override
        public void focusGained(FocusEvent e) {
        e.getComponent().setBackground(Color.yellow);
        }
    @Override
        public void focusLost(FocusEvent e) {
        e.getComponent().setBackground(Color.white);
        }
    };
    nameField.addFocusListener(fl);
    passField.addFocusListener(fl);
```

- Interface **WindowListener** is used to alert you to any change that happens to the window, such as when it is minimized, maximized, opened, exited, active, or inactive.

The WindowListener interface contains five functions that you must override when creating an object from it:

- **windowOpened(WindowEvent e)** function is called after the window is opened.
- **windowClosing(WindowEvent e):** Called while the window is closed
- **windowClosed(WindowEvent):** Called after the window has been closed.
- **windowIconified(WindowEvent):** Called after the window has been hidden by clicking the minimize button above it.
- **windowDeiconified(WindowEvent e)** function is called after it has been rendered again.
- **windowActivated(WindowEvent)** function: It is called if the window is active, that is, if there is no other window or application open above it.
- **WindowDeactivated(WindowEvent e)** function: It is called if the window is inactive, that is, if there is another window or application open above it.

```
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
...

    frame.addWindowListener(new WindowListener() {
        @Override
            public void windowOpened(WindowEvent e) {
            JOptionPane.showMessageDialog(frame, "Event: Window Opened");
            }
        @Override
            public void windowClosing(WindowEvent e) {
            JOptionPane.showMessageDialog(frame, "Event: Window Closing");
            }
        @Override
            public void windowClosed(WindowEvent e) {
            }
        @Override
            public void windowIconified(WindowEvent e) {
            JOptionPane.showMessageDialog(null, "Event: Window Iconified");
```

```
            }
        @Override
            public void windowDeiconified(WindowEvent e) {
            JOptionPane.showMessageDialog(null, "Event: Window Deiconified");
            }
        @Override
            public void windowActivated(WindowEvent e) {
            }
        @Override
            public void windowDeactivated(WindowEvent e) {
            }
});
```