

## *Lecture 5*

# GUI: 이벤트 처리



---


2018년도 2학기

컴퓨터프로그래밍2

김 영 국

충남대학교 컴퓨터공학과

# 이번 주에 학습할 내용

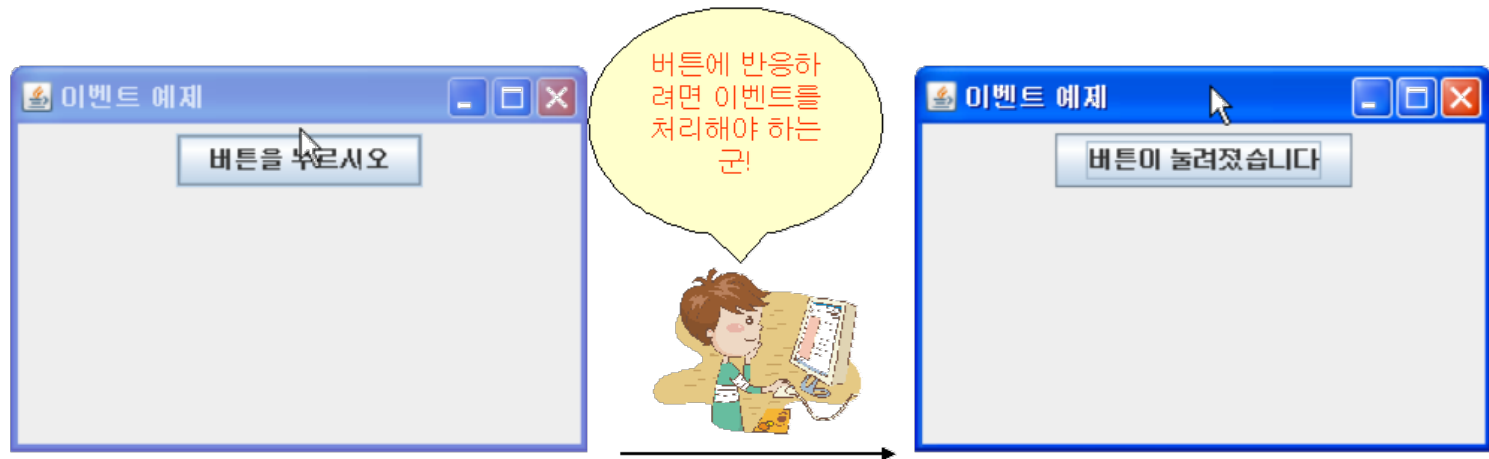
- 
- 이벤트 처리 개요
  - 이벤트
  - 액션 이벤트
  - Key, Mouse, MouseMotion

버튼을  
누르면  
반응하도록  
만들어  
봅시다.



# 학습 목표

- 버튼을 누르면 버튼의 텍스트가 변경되게 한다.



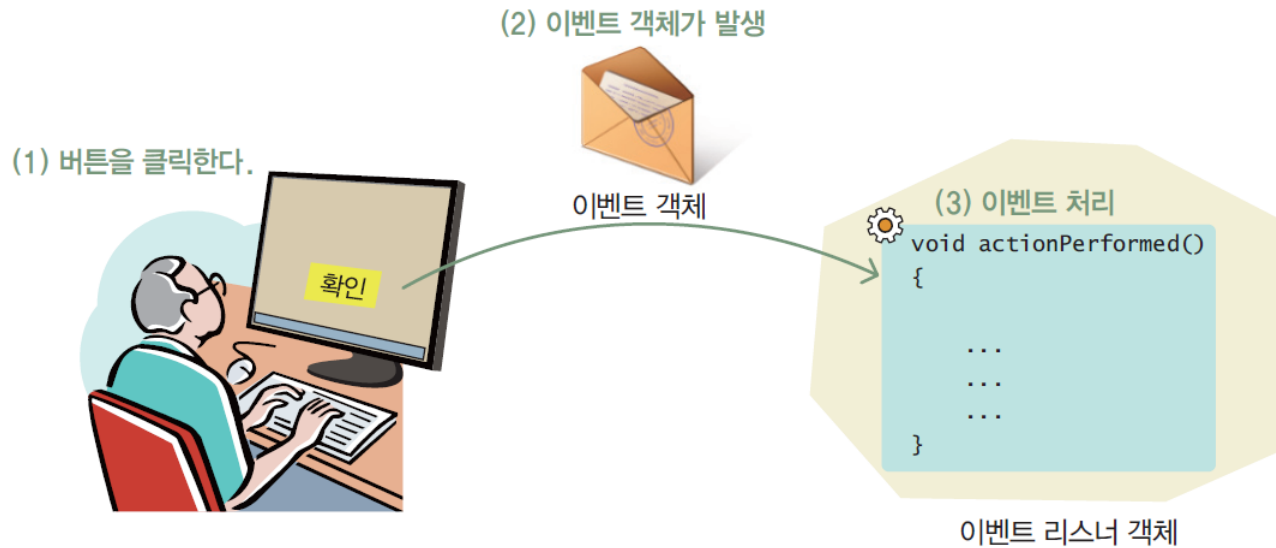
# 이벤트란?

- 이벤트-구동 프로그래밍(Event-driven Programming):

- 프로그램의 실행이 이벤트의 발생에 의하여 결정되는 방식



# 이벤트 처리 과정



< 이벤트 처리의 절차 >

# 이벤트 리스너 작성 과정

## (1) 이벤트 리스너 클래스를 작성한다.

```
class MyListener implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        ... // Action 이벤트를 처리하는 코드가 여기에 들어간다.  
    }  
}
```

액션 이벤트가 발생하면 호출된다.

## (2) 이벤트 리스너를 이벤트 소스에 등록한다.

```
public class MyFrame extends JFrame { // 프레임을 상속하여서 MyFrame 선언  
    ...  
    public MyFrame() // 생성자에서 컴포넌트를 생성하고 추가한다.  
    {  
        button = new JButton("동작"); // 버튼 생성  
        button.addActionListener(new MyListener());  
        ...  
    }  
}
```

이벤트 리스너 객체를 new를 이용하여서 생성하고, 버튼에 이벤트 리스너 객체를 등록한다.

# 이벤트 리스너 작성 과정



## < 리스너 객체의 역할 >



# 이벤트 객체

---

- EventObject 클래스를 상속받는다.
- (예) MouseEvent 클래스

```
java.lang.Object
  java.util.EventObject
    java.awt.AWTEvent
      java.awt.event.ComponentEvent
        java.awt.event.InputEvent
          java.awt.event.MouseEvent
```

```
public void actionPerformed(MouseEvent e) {
    button = (JButton)e.getSource();
    ...
}
```

- 이벤트를 발생시킨 이벤트 소스 등의 여러 가지 정보를 제공한다.



# 리스너를 독립적인 클래스로 작성

```
01 import javax.swing.*;
02 import java.awt.FlowLayout;
03 import java.awt.event.*;
04
05 class MyListener implements ActionListener {
06     public void actionPerformed(ActionEvent e) {
07         JButton button = (JButton) e.getSource();
08         button.setText("마침내 버튼이 눌러졌습니다.");
09     }
10 }
11 class MyFrame extends JFrame {
12     private JButton button;
13     public MyFrame() {
14         this.setSize(300, 200);
15         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16         this.setTitle("이벤트 예제");
17         this.setLayout(new FlowLayout());
18         button = new JButton("버튼을 누르시오");
19         button.addActionListener(new MyListener());
20         this.add(button);
21         this.setVisible(true);
22     }
23 }
```

이벤트 처리를 위한 패키지

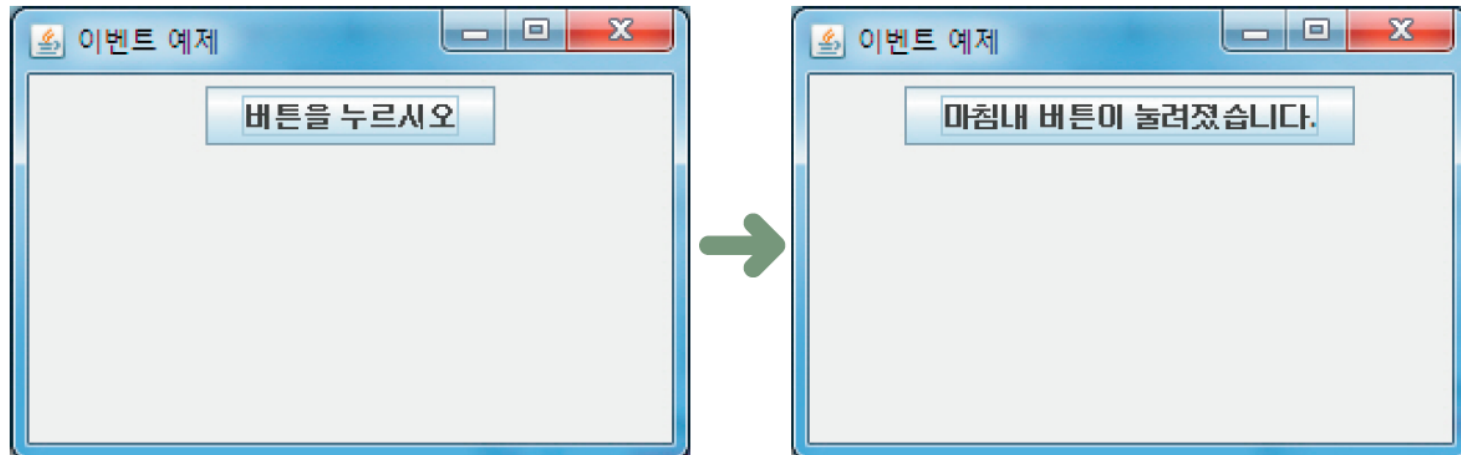
MyListener 클래스를 별도의 클래스로 정의한다.  
ActionListener 인터페이스를 구현한다.

버튼에 이벤트 리스너 등록

# 리스너를 독립적인 클래스로 작성

```
24 public class ActionEventTest {  
25     public static void main(String[] args) {  
26         MyFrame t = new MyFrame();  
27     }  
28 }
```

실행결과



# 리스너 클래스를 내부 클래스로 작성

```
01 import javax.swing.*;
02 import java.awt.event.*; //이벤트 처리를 위한 패키지
03
04 class MyFrame extends JFrame {
05     private JButton button;
06     private JLabel label;
07
08     public MyFrame() {
09         this.setSize(300, 200);
10         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11         this.setTitle("이벤트 예제");
12         JPanel panel = new JPanel();
13         button = new JButton("버튼을 누르시오");
14         label = new JLabel("아직 버튼이 눌러지지 않았습니다");
15         button.addActionListener(new MyListener());
16         panel.add(button);
17         panel.add(label);
18         this.add(panel);
19         this.setVisible(true);
20     }
21 }
```

button, label 변수가 멤버 변수로 선언되었다. 그 이유는 생성자와 actionPerformed() 메소드에서 사용하기 때문에 멤버 변수로 하여야 한다. 여기서는 전용 멤버로 선언하여도 된다. 내부 클래스는 전용 멤버에도 접근할 수 있다.

버튼에 이벤트 리스너 등록

# 리스너 클래스를 내부 클래스로 작성

MyListener 클래스는 MyFrame 클래스의 내부 클래스로 정의된다. MyListener 클래스는 Action 이벤트를 처리할 수도 있도록 ActionListener 인터페이스를 구현한다.

```
22 private class MyListener implements ActionListener {
23     public void actionPerformed(ActionEvent e) {
24         if (e.getSource() == button) {
25             label.setText("마침내 버튼이 눌러졌습니다.");
26         }
27     }
28 }
29 }
30 public class ActionEventTest {
31     public static void main(String[] args) {
32         MyFrame t = new MyFrame();
33     }
34 }
```

MyListener 클래스 안에서 actionPerformed() 메소드는 반드시 정의되어야 한다. 이 메소드는 사용자가 버튼을 누를 때마다 실행된다. 매개변수인 e는 버튼에 의하여 생성되는 이벤트 객체이다.

멤버인 label에 쉽게 접근할 수 있다.

# 실행 결과



# MyFrame 클래스가 이벤트를 처리

```
01 ...
02
03
04 ...
05
06
07
08
09
10
11 ...
12 }
13
14
15
16
17
18
19 }
20
21 ...
```

```
class MyFrame extends JFrame implements ActionListener {
```

```
    public MyFrame() {
```

```
        ...
```

```
        button = new JButton("버튼을 누르시오");
```

```
        label = new JLabel("아직 버튼이 눌러지지 않았습니다");
```

```
        button.addActionListener(this);
```

```
        ...
```

```
    public void actionPerformed(ActionEvent e) {
```

```
        if (e.getSource() == button) {
```

```
            label.setText("마침내 버튼이 눌러졌습니다.");
```

```
        }
```

```
    }
```

EventTest 클래스는 JFrame 클래스를 상속 받고 동시에 ActionListener를 구현한다. 따라서 프레임이 버튼에서 발생하는 이벤트도 처리할 수 있다.

현재 객체를 이벤트 리스너로 버튼에 등록한다. 즉 자기 자신이 이벤트를 처리한다고 등록한다.

MyFrame 클래스 안에 actionPerformed()가 정의되어 있어야 한다.

# 무명 클래스를 사용하는 방법

```
01 class MyFrame extends JFrame {  
02     ...  
03     public MyFrame() {  
04         ...  
05         button = new JButton("버튼을 누르시오");  
06         button.addActionListener(new ActionListener() {  
07             public void actionPerformed(ActionEvent e) {  
08                 if (e.getSource() == button) {  
09                     label.setText("마침내 버튼이 눌러졌습니다.");  
10                 }  
11             }  
12         });  
13     }  
14     ...  
15 }  
16  
17 }
```

무명 클래스는  
ActionListener 인터페이스를  
구현한다. 무명 클래스의  
객체도 동시에 생성된다.

무명 클래스를 정의한다.  
무명 클래스 안에서  
actionPerformed() 메소드를  
정의한다.



# EventHandler 클래스를 사용하는 방법

---

```
myButton.addActionListener(  
    (ActionListener)EventHandler.create(ActionListener.class, frame, "toFront"));
```

```
myButton.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        frame.toFront();  
    }  
});
```





# 모든 컴포넌트들이 지원하는 이벤트

이벤트 종류	설명
Component	컴포넌트의 크기나 위치가 변경되었을 경우 발생
Focus	키보드 입력을 받을 수 있는 상태가 되었을 때, 혹은 그반대의 경우에 발생
Container	컴포넌트가 컨테이너에 추가되거나 삭제될 때 발생
Key	사용자가 키를 눌렀을 때 키보드 포커스를 가지고 있는 객체에서 발생
Mouse	마우스 버튼이 클릭되었을 때, 또는 마우스가 객체의 영역으로 들어오거나 나갈 때 발생
MouseMotion	마우스가 움직였을 때 발생
MouseWheel	컴포넌트 위에서 마우스 휠을 움직이는 경우 발생
Window	윈도우에 어떤 변화가 있을 때 발생(열림, 닫힘, 아이콘화 등)

# 일부 컴포넌트들이 지원하는 이벤트

이벤트 종류	설명
Action	사용자가 어떤 동작을 하는 경우에 발생
Caret	텍스트 삽입점이 이동하거나 텍스트 선택이 변경되었을 경우 발생
Change	일반적으로 객체의 상태가 변경되었을 경우 발생
Document	문서의 상태가 변경되는 경우 발생
Item	선택 가능한 컴포넌트에서 사용자가 선택을 하였을 때 발생
ListSelection	리스트나 테이블에서 선택 부분이 변경되었을 경우에 발생

컴포넌트	이벤트					
	Action	Caret	Change	Document	Item	ListSelection
버튼(button )	√		√		√	
체크 박스(check box)	√		√		√	
색상 선택(color chooser)			√			

# 일부 컴포넌트들이 지원하는 이벤트

콤보 박스(combo box)	✓				✓	
다이얼로그(dialog )						
파일 선택(file chooser )	✓					
프레임(frame )						
리스트(list )						✓
메뉴 항목(menu item )	✓		✓		✓	
진행바(progress bar )			✓			
라디오 버튼(radio button )	✓		✓		✓	
슬라이더(slider )			✓			
스피너(spinner )			✓			
테이블(table)						✓
텍스트 영역(text area )		✓		✓		
텍스트 필드(text field)	✓	✓		✓		

# 리스너 인터페이스의 요약

리스너 인터페이스	어댑터 클래스	메소드
ActionListener	none	actionPerformed()
AdjustmentListener	none	adjustmentValueChanged()
ComponentListener	ComponentAdapter	componentHidden() componentMoved() componentResized() componentShown()
ContainerListener	ContainerAdapter	componentAdded() componentRemoved()
FocusListener	FocusAdapter	focusGained() focusLost()
ItemListener	none	itemStateChanged()
KeyListener	KeyAdapter	keyPressed() keyReleased() keyTyped()
MouseListener	MouseAdapter	mouseClicked() mouseEntered() mouseExited() mousePressed() mouseReleased()
MouseMotionListener	MouseMotionAdapter	mouseDragged() mouseMoved()
TextListener	none	textValueChanged()
WindowListener	WindowAdapter	windowActivated() windowClosed() windowClosing() windowDeactivated() windowDeiconified() windowIconified() windowOpened()

# 액션 이벤트

- 사용자가 버튼을 클릭하는 경우
- 사용자가 메뉴 항목을 선택하는 경우
- 사용자가 텍스트 필드에서 엔터키를 누르는 경우



# 예제 프로그래밍

ActionEventTest4.java

```
01  ...
02
03  class MyFrame extends JFrame {
04
05      private JButton button1;
06      private JButton button2;
07      private JPanel panel;
08      MyListener listener = new MyListener();
09
10      public MyFrame() {
11          this.setSize(300, 200);
12          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13          this.setTitle("이벤트 예제");
14          panel = new JPanel();
```

← 리스너 객체를 미리 생성시켜 놓는다.

# 예제 프로그래밍

```
15     button1 = new JButton("노란색");
16     button1.addActionListener(listener);
17     panel.add(button1);
18     button2 = new JButton("핑크색");
19     button2.addActionListener(listener);
20     panel.add(button2);
21     this.add(panel);
22     this.setVisible(true);
23 }
24 private class MyListener implements ActionListener {
25     public void actionPerformed(ActionEvent e) {
26         if (e.getSource() == button1) {
27             panel.setBackground(Color.YELLOW);
28         } else if (e.getSource() == button2) {
29             panel.setBackground(Color.PINK);
30         }
31     }
32 }
33 }
34 public class ActionEventTest4 {
35     public static void main(String[] args) {
36         MyFrame t = new MyFrame();
37     }
38 }
```

두 개의 버튼에 동일한 이벤트 리스너 객체를 등록한다.

getSource()를 이용하여서 이벤트 소스를 찾는다.

# 실행 결과







# Key 이벤트

---

## ■ KeyListener 인터페이스 구현

### KeyListener 인터페이스

메소드	설명
keyTyped(KeyEvent e)	사용자가 글자를 입력했을 경우에 호출
keyPressed(KeyEvent e)	사용자가 키를 눌렀을 경우에 호출
keyReleased(KeyEvent e)	사용자가 키에서 손을 떼었을 경우에 호출



# Key 이벤트

## KeyEvent 클래스

메소드	설명
<code>int getKeyChar()</code>	KeyEvent에 들어있는 글자(유니코드)를 반환한다.
<code>int getKeyCode()</code>	<p>KeyEvent에 들어있는 키코드(keycode)를 반환한다. 키코드란 글자가 아니라 키보드 자판의 각각의 키를 가리키는 상수이다. 예를 들어 Escape 키의 키코드는 VK_ESCAPE로 정의되어 있다.</p> <p>예를 들어 눌려진 키가 Enter 키인가를 판단하기 위해서는 다음과 같은 문장을 이용한다.</p> <pre>if (e.getKeyCode() == e.VK_ENTER) {     // 처리할 내용 }</pre>
<code>boolean isActionKey()</code>	이벤트를 발생시킨 키가 액션 키이면 true를 반환한다. 액션 키란 Cut, Copy, Paste, Page Up, Caps Lock, 화살표와 function 키를 의미한다.



# Key 이벤트

## InputEvent 클래스

메소드	설명
<code>int getID()</code>	이벤트의 타입을 반환한다. 가능한 타입은 다음과 같다. <code>MouseEvent.MOUSE_PRESSED</code> , <code>MouseEvent.MOUSE_RELEASED</code> , <code>MouseEvent.MOUSE_CLICKED</code> .
<code>getComponent()</code>	이벤트를 일으킨 컴포넌트를 반환한다. <code>getSource()</code> 를 사용하여도 된다.
<code>int getWhen()</code>	이벤트가 발생한 시각을 반환한다.
<code>boolean isAltDown()</code> <code>boolean isControlDown()</code> <code>boolean isMetaDown()</code> <code>boolean isShiftDown()</code>	이벤트가 발생한 시각에 키보드의 수식키들의 상태를 반환한다.

# Key 이벤트 예제

KeyEventTest.java

```
01  ...
02
03  class MyFrame extends JFrame implements KeyListener { // ①
04
05      public MyFrame() {
06          setTitle("이벤트 예제");
07          setSize(300, 200);
08          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
09
10          JTextField tf = new JTextField(20);
11          tf.addKeyListener(this); // ②
12
13          add(tf);
14          setVisible(true);
15      }
16
17
18      public void keyTyped(KeyEvent e) { // ③
19          display(e, "KeyTyped ");
20      }
21
```

키 이벤트 리스너로 만들기 위하여  
KeyListener 인터페이스를 구현한다.

addKeyListener() 메소드로  
현재의 객체를 이벤트 리스너로  
추가한다.

키가 입력되면 호출된다.

# Key 이벤트 예제

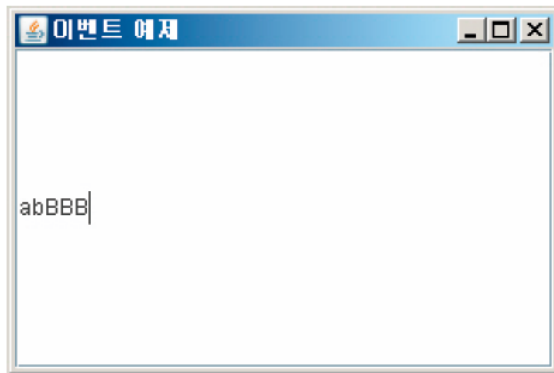
```
21
22 public void keyPressed(KeyEvent e) {
23     display(e, "KeyPressed ");
24 }
25
26 public void keyReleased(KeyEvent e) {
27     display(e, "Key Released ");
28 }
29
30 protected void display(KeyEvent e, String s) {
31     char c = e.getKeyChar();
32     int keyCode = e.getKeyCode();
33     String modifiers = e.isAltDown() + " " + e.isControlDown() + " "
34         + e.isShiftDown();
35     System.out.println(s + " " + c + " " + keyCode + " " + modifiers);
36 }
37 }
38
39 public class KeyEventTest {
40     public static void main(String[] args) {
41         MyFrame f = new MyFrame();
42     }
43 }
```

← 키를 누르는 순간 호출된다.

← 키에서 손을 떼는 순간 호출된다.

← 눌러진 키의 유니코드값을 얻는다.

# 실행 화면



## 실행결과

```
KeyPressed a 65 false false false  
KeyTyped a 0 false false false  
Key Pressed a 65 false false false  
KeyPressed b 66 false false false  
KeyTyped b 0 false false false  
Key Pressed b 66 false false fals
```

# 자동차 게임 예제

CarGameTest.java

```
01 ...
02 class MyPanel extends JPanel {
03     BufferedImage img = null;
04     int img_x = 100, img_y = 100;
05
06     public MyPanel() {
07         try {
08             img = ImageIO.read(new File("car.gif"));
09         } catch (IOException e) {
10             System.out.println("no image");
11             System.exit(1);
12         }
13         addKeyListener(new KeyListener() {
14             public void keyPressed(KeyEvent e) {
15                 int keycode = e.getKeyCode();
16                 switch (keycode) {
17                     case KeyEvent.VK_UP:      img_y -= 10; break;
18                     case KeyEvent.VK_DOWN:    img_y += 10; break;
19                     case KeyEvent.VK_LEFT:    img_x -= 10; break;
20                     case KeyEvent.VK_RIGHT:   img_x += 10; break;
21                 }
12
```

이미지를 읽는다. 오류가 발생하면 실행을 종료한다.

키 리스너를 무명 클래스로 작성해서 패널에 붙인다. 화살표 키가 입력되면 이미지의 좌표를 변경한다.

# 자동차 게임 예제

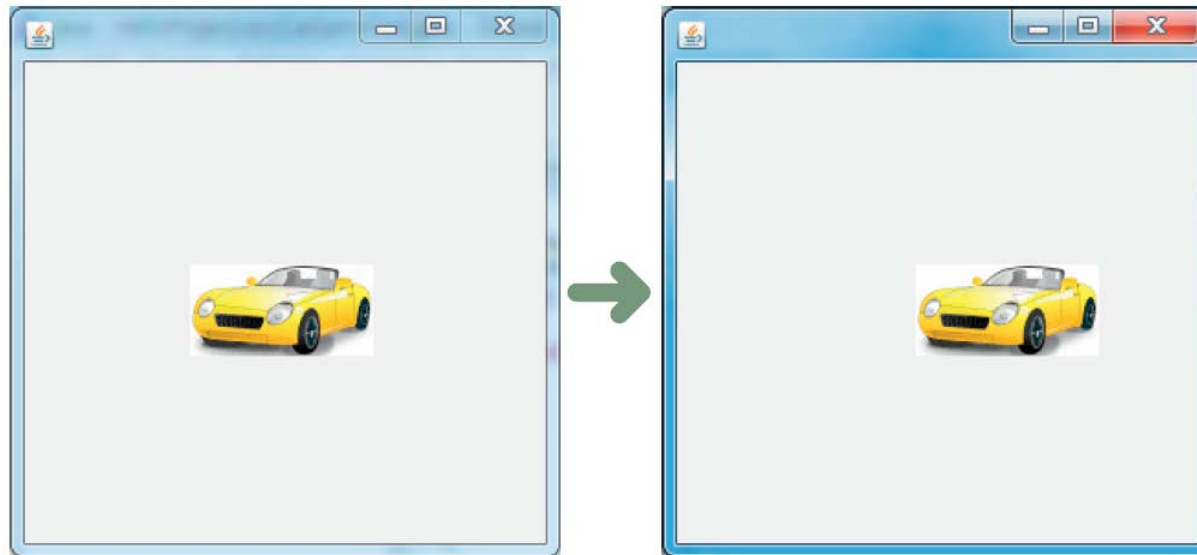
```
22         repaint();
23     }
24     public void keyReleased(KeyEvent arg0) {
25     }
26     public void keyTyped(KeyEvent arg0) {
27     }
28     });
29     this.requestFocus();
30     setFocusable(true);
31 }
32
33 public void paintComponent(Graphics g) {
34     super.paintComponent(g);
35     g.drawImage(img, img_x, img_y, null);
36 }
37 }
38
39 public class CarGameTest extends JFrame {
40     public CarGameTest() {
41         setSize(300, 300);
42         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
43         add(new MyPanel());
44         setVisible(true);
45     }
46
47     public static void main(String[] args) {
48         CarGameTest s = new CarGameTest();
49     }
50 }
```

← 키보드 포커스를 요청하고 패널이 포커스를 받을 수 있게 한다. 디폴트로는 패널이 키보드 포커스를 받을 수 없다.

← 자동차 이미지를 화면의 (img\_x, img\_y) 위치에 그린다.



# 실행 결과



키보드의 화살표 키를 이용해서 화면 안의 자동차를 움직여 보자.



# 신호등 예제

MyFrame.java

```
01  ...
02
03  class MyPanel extends JPanel implements ActionListener {
04      boolean flag = false
05      private int light_number = 0;
06
07      public MyPanel() {
08          setLayout(new BorderLayout());
09          JButton b = new JButton("traffic light turn on");
10          b.addActionListener(this);
11          add(b, BorderLayout.SOUTH);
12      }
13
14      @Override
15      protected void paintComponent(Graphics g) {
16          // TODO Auto-generated method stub
```

# 신호등 예제

```
17      super.paintComponent(g);
18      g.setColor(Color.BLACK);
19      g.drawOval(100, 100, 100, 100);
20      g.drawOval(100, 200, 100, 100);
21      g.drawOval(100, 300, 100, 100);
22      if (light_number == 0) {
23          g.setColor(Color.RED);
24          g.fillOval(100, 100, 100, 100);
25      } else if (light_number == 1) {
26          g.setColor(Color.GREEN);
27          g.fillOval(100, 200, 100, 100);
28      } else {
29          g.setColor(Color.YELLOW);
30          g.fillOval(100, 300, 100, 100);
31      }
32  }
33
```

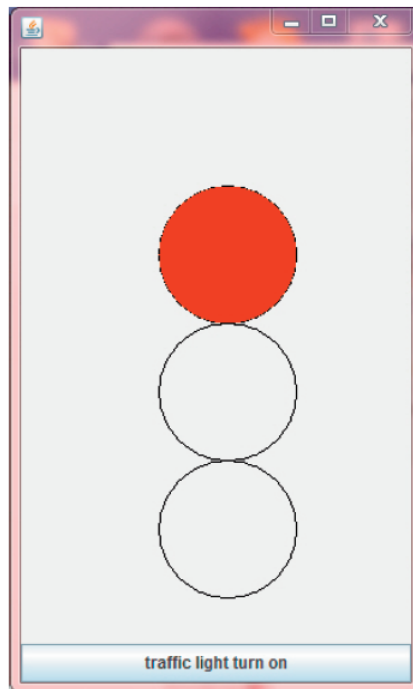
← 신호등을 그린다.

# 신호등 예제

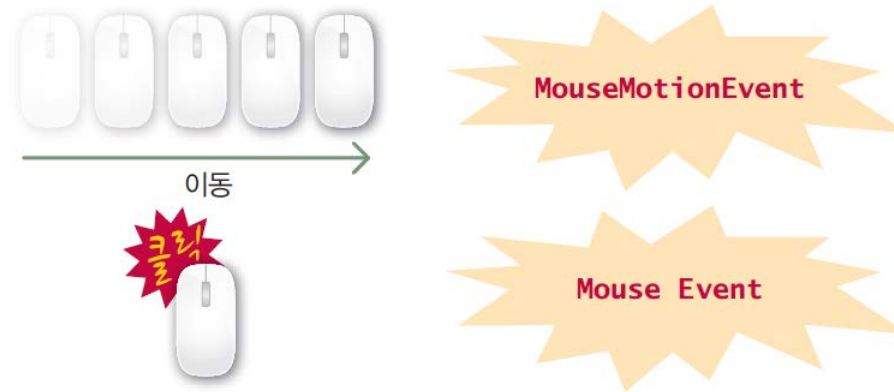
```
34     @Override
35     public void actionPerformed(ActionEvent arg0) {
36         // TODO Auto-generated method stub
37         if (++light_number >= 3)
38             light_number = 0;
39         repaint();
40     }
41 }
42
43 public class MyFrame extends JFrame {
44     public MyFrame() {
45         add(new MyPanel());
46         setSize(300, 500);
47         setVisible(true);
48     }
49
50     public static void main(String[] arg) {
51         new MyFrame();
52     }
53 }
```

버튼이 눌리면 신호를 바꾼다.

# 실행 결과



# Mouse와 MouseMotion 이벤트



## MouseListener 인터페이스

메소드	설명
<code>mouseClicked(MouseEvent e)</code>	사용자가 컴포넌트를 클릭한 경우에 호출된다.
<code>mouseEntered(MouseEvent e)</code>	마우스 커서가 컴포넌트로 들어가면 호출된다.
<code>mouseExited(MouseEvent e)</code>	마우스 커서가 컴포넌트에서 나가면 호출된다.
<code>mousePressed(MouseEvent e)</code>	마우스가 컴포넌트위에서 눌러지면 호출된다.
<code>mouseReleased(MouseEvent e)</code>	마우스가 컴포넌트위에서 떼어지면 호출된다.

# Mouse와 MouseMotion 이벤트

메소드	설명
<code>mouseDragged(MouseEvent e)</code>	마우스 드래그하면 호출된다.
<code>mouseMoved(MouseEvent e)</code>	마우스가 클릭되지 않고 이동하는 경우에 호출된다.

## MouseMotionListener 인터페이스

### 실행결과

Mouse pressed (# of clicks: 1) X=118 Y=81 ← 버튼을 눌렀을 때 발생  
Mouse released (# of clicks: 1) X=118 Y=81 ← 버튼에서 손을 떼면 발생  
Mouse clicked (# of clicks: 1) X=118 Y=81 ← 버튼이 한번 클릭되면 발생

### 실행결과

Mouse pressed (# of clicks: 1) X=93 Y=47 ← 버튼을 클릭하였을 때 발생  
Mouse dragged X=93 Y=48  
Mouse dragged X=94 Y=48  
...  
Mouse dragged X=117 Y=66  
Mouse dragged X=118 Y=66 } ← 버튼을 클릭한채로 움직이면 발생  
Mouse released (# of clicks: 1) X=118 Y=66 ← 버튼에서 손을 떼면 발생



# 마우스 이벤트 객체

## MouseEvent 클래스

메소드	설명
<code>int getClickCount()</code>	빠른 연속적인 클릭의 횟수를 반환한다. 예를 들어 2이면 더블 클릭을 의미한다.
<code>int getX()</code> <code>int getY()</code> <code>Point getPoint()</code>	이벤트가 발생했을 당시의 (x, y) 위치를 반환한다. 위치는 컴포넌트에 상대적이다.
<code>int getXOnScreen()</code> <code>int getYOnScreen()</code> <code>int getLocationOnScreen()</code>	절대 좌표 값 (x, y)을 반환한다. 이들 좌표값은 가상 화면에 상대적이다.
<code>int getButton()</code>	어떤 마우스 버튼의 상태가 변경되었는지를 반환한다. NOBUTTON, BUTTON1, BUTTON2, BUTTON3 중의 하나이다.



# Mouse와 MouseMotion 이벤트 예제

MouseEventTest.java

```
01  ...
02
03  class MyFrame extends JFrame implements MouseListener, MouseMotionListener {
04
05      public MyFrame() {
06          setTitle("Mouse Event");
07          setSize(300, 200);
08          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
09
10          JPanel panel = new JPanel();
11          panel.addMouseListener(this);
12          panel.addMouseMotionListener(this);
13          add(panel);
14          setVisible(true);
15
16      }
17
```

전체 프레임을 마우스 리스너로 등록하고  
마우스 이벤트가 발생하면 콘솔에 이벤트  
정보를 기록한다.

패널에 Mouse 리스너와  
MouseMotion 리스너를 붙  
인다.

# Mouse와 MouseMotion 이벤트 예제

Mouse 리스너의 메소드 구현

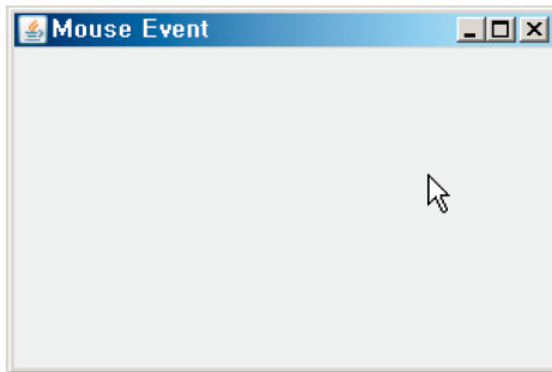
```
17
18 public void mousePressed(MouseEvent e) {
19     display("Mouse pressed (# of clicks: " + e.getClickCount() + ")", e);
20 }
21
22 public void mouseReleased(MouseEvent e) {
23     display("Mouse released (# of clicks: " + e.getClickCount() + ")", e);
24 }
25
26 public void mouseEntered(MouseEvent e) {
27     display("Mouse entered", e);
28 }
29
30 public void mouseExited(MouseEvent e) {
31     display("Mouse exited", e);
32 }
33
34 public void mouseClicked(MouseEvent e) {
35     display("Mouse clicked (# of clicks: " + e.getClickCount() + ")", e);
36 }
37
```

# Mouse와 MouseMotion 이벤트 예제

```
38 public void mouseDragged(MouseEvent e) {
39     display("Mouse dragged", e);
40 }
41
42 public void mouseMoved(MouseEvent e) {
43     display("Mouse moved", e);
44 }
45 }
46 protected void display(String s, MouseEvent e) {
47     System.out.println(s + " X=" + e.getX() + " Y=" + e.getY());
48 }
49 }
50
51 public class MouseEventTest {
52     public static void main(String[] args) {
53         MyFrame f = new MyFrame();
54     }
55 }
```

← MouseMotion 리스너의 메소드 구현

# 실행 화면



## 실행결과

```
Mouse moved X=139 Y=78  
Mouse pressed (# of clicks: 1) X=139 Y=78  
Mouse released (# of clicks: 1) X=139 Y=78  
Mouse clicked (# of clicks: 1) X=139 Y=78  
Mouse moved X=139 Y=77  
Mouse moved X=141 Y=77
```

# 자동차 게임 예제

CarGame2.java

```
01  ...
02
03  class MyPanel extends JPanel {
04      BufferedImage img = null;
05      int img_x = 0, img_y = 0;
06
07      public MyPanel() {
08          try {
09              img = ImageIO.read(new File("car.gif"));
10          } catch (IOException e) {
11              System.out.println("no image");
12              System.exit(1);
13          }
14          addMouseListener(new MouseListener() {
15              public void mousePressed(MouseEvent e) {
16                  img_x = e.getX();
17                  img_y = e.getY();
18                  repaint();
19              }
20              public void mouseReleased(MouseEvent e) {}
21              public void mouseEntered(MouseEvent e) {}
22              public void mouseExited(MouseEvent e) {}
23              public void mouseClicked(MouseEvent e) {}
```

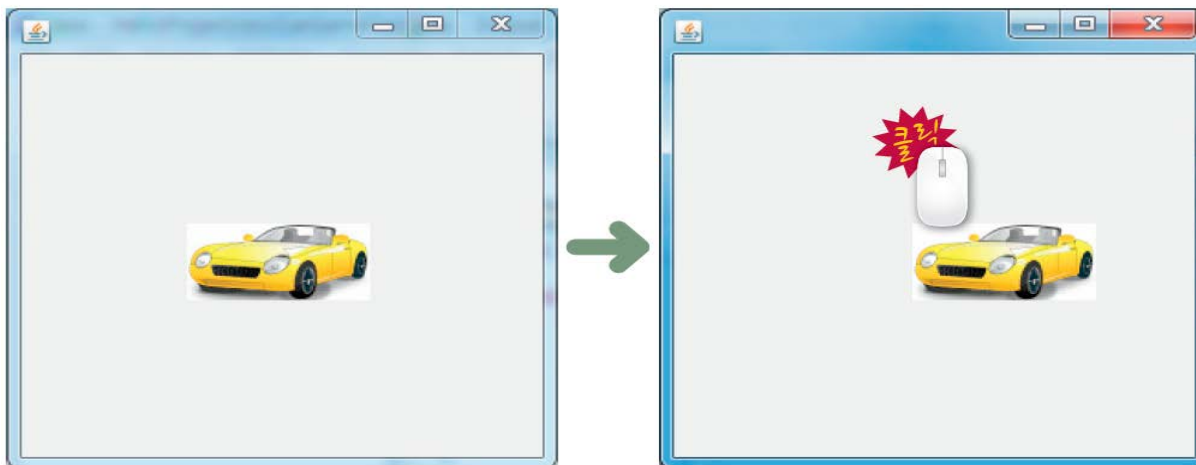
패널에 마우스 리스너를 추가한다.  
무명 클래스로 작성한다.

마우스 버튼이 눌리면 위치를  
얻어서 이미지의 좌표로 저장한다.

# 자동차 게임 예제

```
24     });  
25 }  
26  
27 public void paintComponent(Graphics g) {  
28     super.paintComponent(g);  
29     g.drawImage(img, img_x, img_y, null);  
30 }  
31 }  
32  
33 ...
```

실행결과



# Q & A

