# (INTERMEDIATE)
# JAVA PROGRAMMING

18. Swing – Keyboard Event

# JAVA:
## KEY EVENT HANDLING

# Key 이벤트와 포커스

- 키 입력 시, 다음 세 경우에 Key 이벤트 발생
  - 키를 누르는 순간
  - 누른 키를 떼는 순간
  - 누른 키를 떼는 순간(Unicode 키의 경우에만)
- 키 이벤트를 받을 수 있는 조건
  - 모든 컴포넌트 가능하지만, 현재 포커스(focus)를 가진 컴포넌트
- 포커스(focus)
  - 컴포넌트나 응용프로그램이 키 이벤트를 독점하는 권한
  - 컴포넌트에 포커스 설정 방법 : 다음 2 라인의 코드 필요

```
component.setFocusable(true); // component가 포커스를 받을 수 있도록 설정
component.requestFocus(); // componen에 포커스 강제 지정
```

  - 자바플랫폼마다 실행 환경의 초기화가 서로 다를 수 있기 때문에 다음 코드가 반드시 필요함: component.setFocusable(true);

# 컴포넌트에 포커스 주기

- 스윙 프레임이 만들어질 포커스를 주고자 하는 경우
  - JFrame의 setVisible(true) 이후에  포커스를 주어야 함

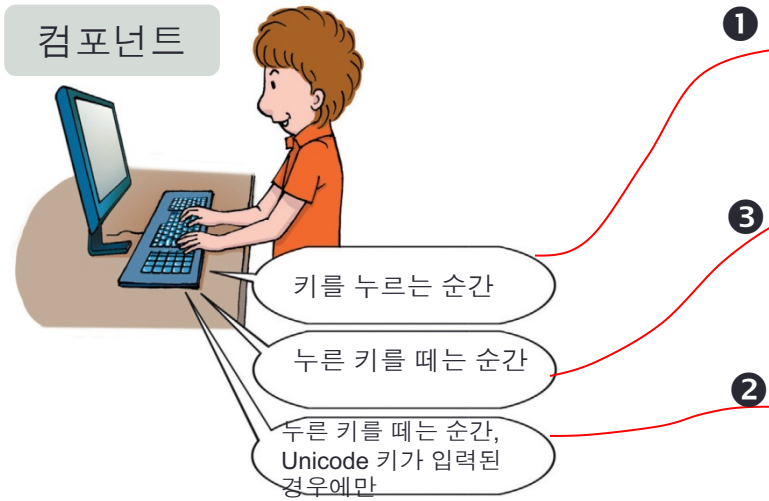
```
setVisible(true); // 스윙 프레임 출력
component.setFocusable(true);
component.requestFocus();
```

- 마우스로 컴포넌트를 클릭할 때 포커스 지정하는 방법
  - 언제든지 필요할 때 포커스 줄 수 있음

```
component.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        Component c = (Component)e.getSource(); // 클릭된 컴포넌트
        c.setFocusable(true);
        c.requestFocus();
    }
}); // 예제 10-8에서 활용하였음
```

# KeyListener의 메소드와 키

- KeyListener의 3 개 메소드




컴포넌트의 키 리스너(KeyListener)

❶
```
void keyPressed(KeyEvent e) {
    // 이벤트 처리 루틴
}
```

❸
```
void keyReleased(KeyEvent e) {
    // 이벤트 처리 루틴
}
```

❷
```
void keyTyped(KeyEvent e) {
    // 이벤트 처리 루틴
}
```

KeyListener의 메소드가 실행되는 순서 ❶ ❷ ❸

- 컴포넌트에 키 이벤트 리스너 등록

```
component.addKeyListener(myKeyListener);
```

# 키는 2가지 종류

1. 유니코드 키
   - 유니코드는 전 세계의 **문자**에 대한 코드 체계
   - 문자들에 대해서만 유니 코드 값 정의
     - A~Z, a~z, 0~9, !, @, & 등
   - 유니코드 키가 눌러진 경우 이벤트 호출 순서
     - keyPressed(), keyTyped(), keyReleased() 순으로 호출

2. 유니코드가 아닌 키
   - 문자 키가 아닌 다음 키들(제어 키)
     - <Function>, <Home>, <Up>, <Delete>, <Control>, <Shift>, <Alt> 등
   - 정의된 유니코드 값 없음
   - 키마다 **키 코드 값(가상 키 코드 값)**이 정의되어 있음
   - 유니코드 키가 아닌 경우 키 이벤트 호출 순서
     - keyPressed(), keyReleased() 만 호출됨

- 가상 키
  - 유니코드 키든 아니든 모든 키에 자바의 가상 키 코드가 정의되어 있음
    - 가상 키 값으로 어떤 키인지 비교 판단 가능

# 가상 키(Virtual Key)

- 가상 키 코드는 KeyEvent 클래스에 상수로 선언

| 가상 키 | 설명 | 가상 키 | 설명 |
|---|---|---|---|
| VK_0 ~ VK_9 | 0에서 9까지의 키,<br>'0'~'9'까지의 유니코드 값과 동일 | VK_LEFT | 왼쪽 방향 키 |
| VK_A ~ VK_Z | A에서 Z까지의 키,<br>'A'~'Z'까지의 유니코드 값과 동일 | VK_RIGHT | 오른쪽 방향 키 |
| VK_F1 ~ VK_F24 | <F1>~<F24>까지의 키 코드 | VK_UP | <Up> 키 |
| VK_HOME | <Home> 키 | VK_DOWN | <Down> 키 |
| VK_END | <End> 키 | VK_CONTROL | <Control> 키 |
| VK_PGUP | <Page Up> 키 | VK_SHIFT | <Shift> 키 |
| VK_PGDN | <Page Down> 키 | VK_ALT | <Alt> 키 |
| VK_UNDEFINED | 입력된 키의 코드 값을 알 수 없음 | VK_TAB | <Tab> 키 |

# 입력된 키 판별

- 키가 입력되면 키 정보를 가진 이벤트 객체 생성 : KeyEvent 객체
  - KeyEvent 객체가 리스너에 전달됨

1. 키의 **문자 코드**(유니코드) 알아내기, char KeyEvent.getKeyChar()
   - 눌러진 키에 해당하는 문자 코드(유니코드) 리턴
   - 눌러진 키가 문자 키인 경우에만 작동

2. 입력된 키의 **가상 키** 값 알아내기, int KeyEvent.getKeyCode()
   - 모든 키에 대해 작동
   - 입력된 키를 판별하기 위해 가상키(Virtual Key) 값과 비교
     - 가상 키 값은 KeyEvent 클래스의 상수로 정의됨

3. **키 이름** 문자열 리턴 String KeyEvent.getKeyText(int keyCode)
   - Static 메소드
   - 매개변수 keyCode의 코드 값(가상 키)에 해당하는 키의 이름 문자열 리턴
     - F1 키의 경우 "F1", Shift 키의 경우 "SHIFT" 등의 문자열 리턴

# KeyEvent의 getKeyChar()과 getKeyCode()

a 키를 누르는 순간

```
public void keyPressed(KeyEvent e) {
    char keyChar = e.getKeyChar();
    int keyCode = e.getKeyCode();
}
```

keyChar: 키 a의 유니코드 값('a')

keyCode: VK_A

〈F5〉 키를 누르는 순간

```
public void keyPressed(KeyEvent e) {
    char keyChar = e.getKeyChar();
    int keyCode = e.getKeyCode();
}
```

keyChar: CHAR_UNDEFINED

keyCode: VK_F5

w 키를 떼는 순간

```
public void keyTyped(KeyEvent e) {
    char keyChar = e.getKeyChar();
    int keyCode = e.getKeyCode();
}
```

keyChar: 키 w의 유니코드 값('w')

keyCode: VK_UNDEFINED

〈F5〉 키를 떼는 순간

```
public void keyTyped(KeyEvent e) {
    char keyChar = e.getKeyChar();
    int keyCode = e.getKeyCode();
}
```

keyChar:

keyCode:

〈F5〉 키에 대해서는 keyTyped() 메소드가 호출되지 않습니다.

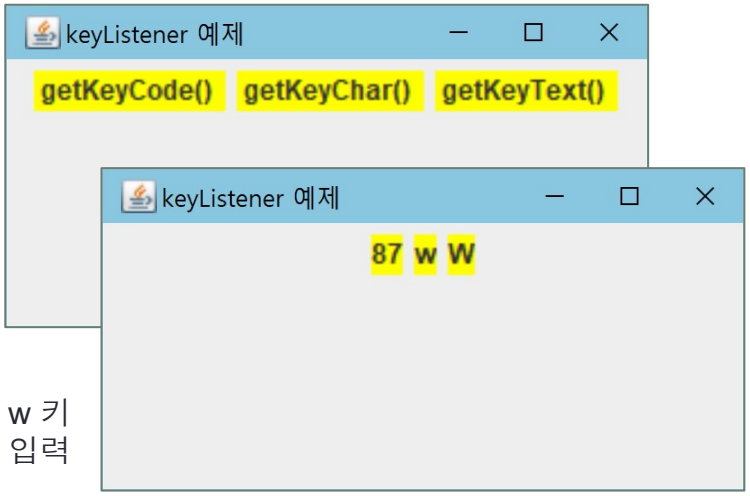
# 예제 10-6 : 다양한 KeyEvent와 KeyListener 활용

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class KeyListenerEx extends JFrame {
    private JLabel [] keyMessage;

    public KeyListenerEx() {
        setTitle("keyListener 예제");

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(new FlowLayout());

        c.addKeyListener(new MyKeyListener());

        keyMessage = new JLabel [3];
        keyMessage[0] = new JLabel(" getKeyCode() ");
        keyMessage[1] = new JLabel(" getKeyChar() ");
        keyMessage[2] = new JLabel(" getKeyText() ");

        for(int i=0; i<keyMessage.length; i++) {
            c.add(keyMessage[i]);
            keyMessage[i].setOpaque(true);
            keyMessage[i].setBackground(Color.YELLOW);
        }
        setSize(300,150);
        setVisible(true);

        c.setFocusable(true);
        c.requestFocus();
    }

    class MyKeyListener extends KeyAdapter {
        public void keyPressed(KeyEvent e) {
            int keyCode = e.getKeyCode();
            char keyChar = e.getKeyChar();

            keyMessage[0].setText(Integer.toString(keyCode));
            keyMessage[1].setText(Character.toString(keyChar));
            keyMessage[2].setText(e.getKeyText(keyCode));
        }
    }

    public static void main(String [] args) {
        new KeyListenerEx();
    }
}
```

컴포넌트의 바탕색이 보이도록 하기 위해서는 컴포넌트가 불투명하기 지정될 필요 있음

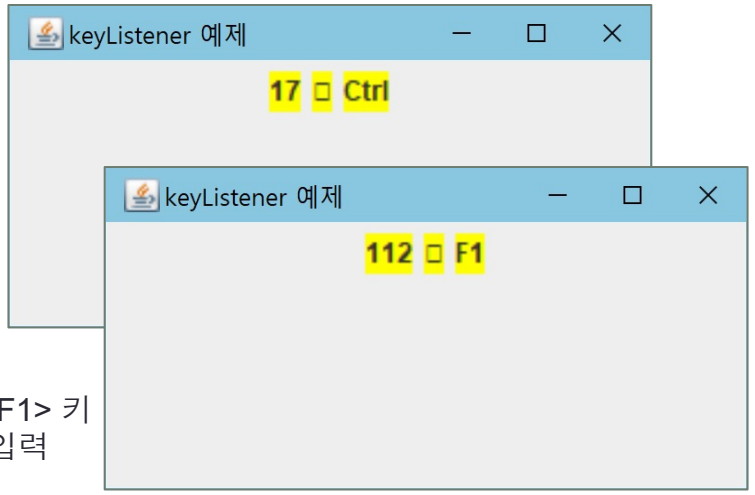
# 실행 결과

초기화면

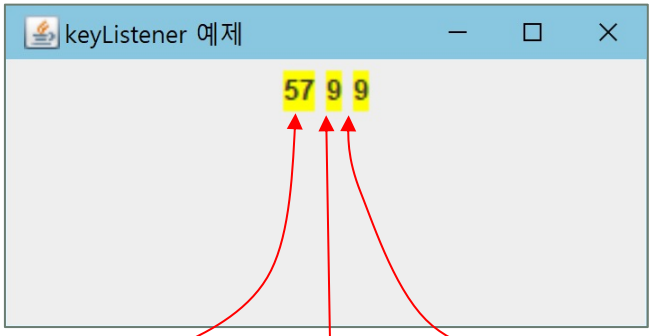



w 키 입력

<Control> 키 입력



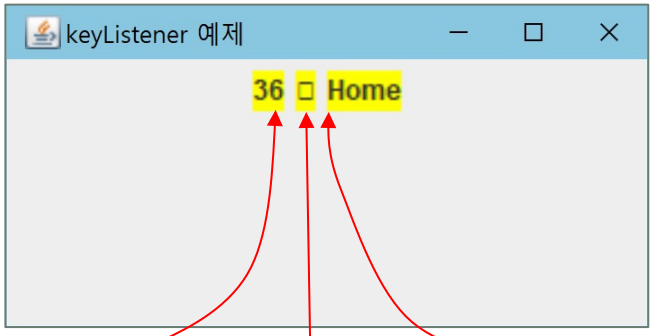

<F1> 키 입력




키 9의 키코드

키 9의 유니코드 문자

키 9의 이름 문자열




<Home> 키 코드

<Home>키에 대응하는 문자 없음

<Home> 키의 이름 문자열

# 예제 10-7 : <F1> 키를 입력받으면 컨텐트팬의 배경을 초록색으로, % 키를 입력받으면 노란색으로 변경

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class KeyCodeEx extends JFrame {
    private JLabel la = new JLabel();

    public KeyCodeEx() {
        setTitle("Key Code 예제 : F1키:초록색, % 키 노란색");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c = getContentPane();

        c.addKeyListener(new MyKeyListener());
        c.add(la);

        setSize(300,150);
        setVisible(true);

        c.setFocusable(true);
        c.requestFocus();
    }
```

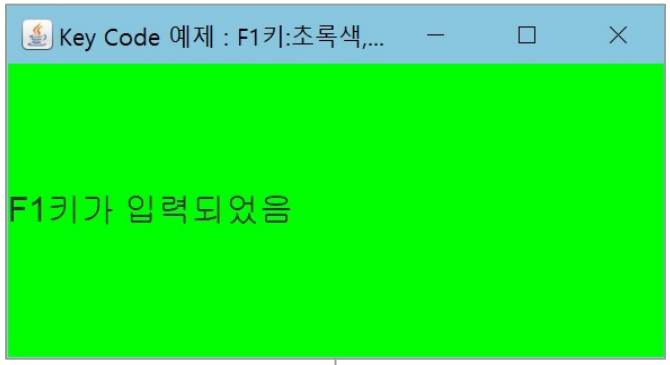
```
    class MyKeyListener extends KeyAdapter {
        public void keyPressed(KeyEvent e) {
            la.setText(e.getKeyText(e.getKeyCode()));

            if(e.getKeyChar() == '%')
                contentPane.setBackground(Color.YELLOW);
            else if(e.getKeyCode() == KeyEvent.VK_F1)
                contentPane.setBackground(Color.GREEN);
        }
    }
    public static void main(String [] args) {
        new KeyCodeEx();
    }
}
```

% 키를 판별하기 위해 e.getKeyChar() 호출

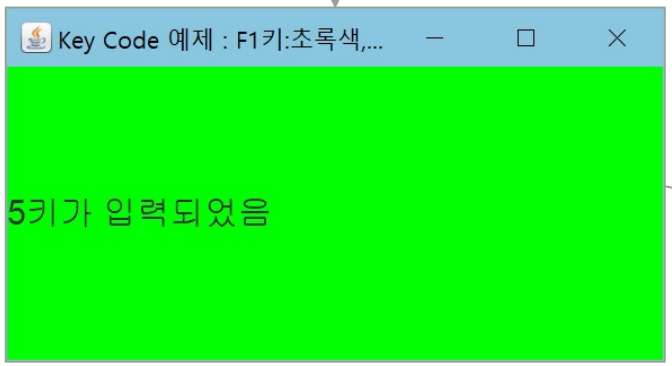
키 입력을 받을 수 있도록 포커스를 준다.

F1 키를 판별하기 위해 e.getKeyCode() 호출
KeyEvent.VK_F1 값과 비교

# 예제 10-7 실행

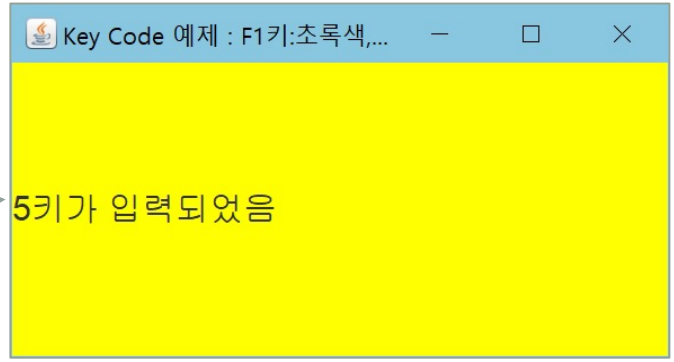



<F1> 키를 누른 경우




키 5 를 누르면 노란색 배경으로 변하지 않음.

%키나 5키는 키보드에서 동일한 키이지만
if(e.getKeyChar() == '%')로 비교하였기 때문



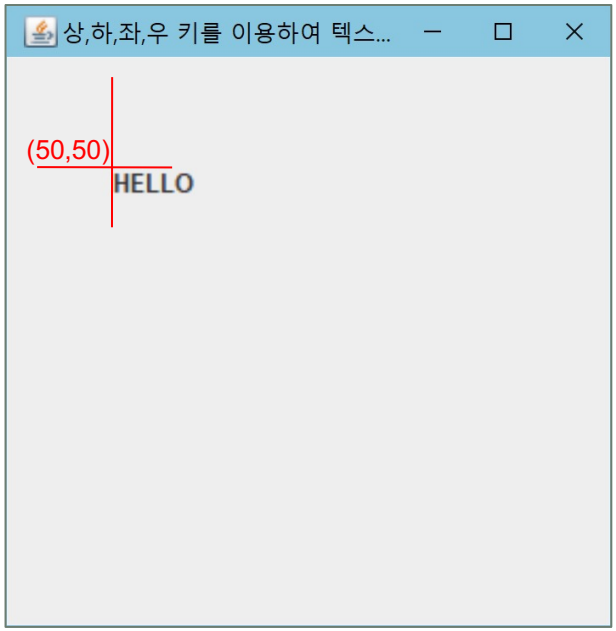

% 키를 누른 경우 노란색으로 변경

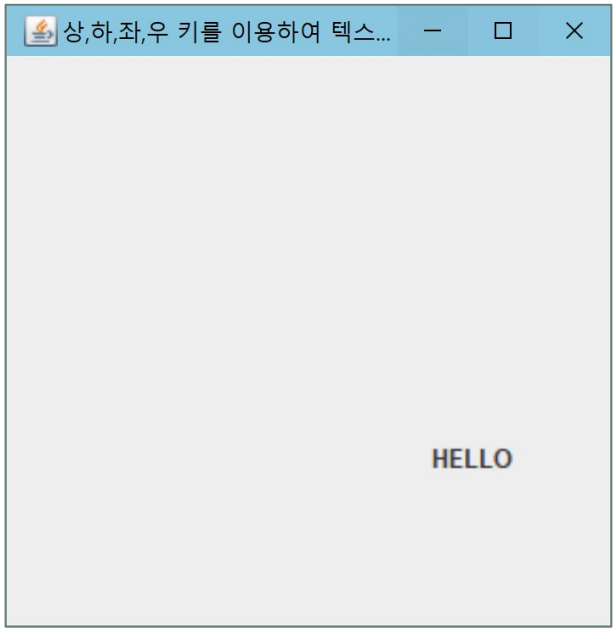
% 키는 Shift키+5키이므로,
키 5에 대한 문자열 출력

# 예제 10-8 : 상(UP), 하(DOWN), 좌(LEFT), 우(RIGHT) 키로 "HELLO" 문자열을 마음대로 움직이기

상, 하, 좌, 우 키를 이용하여 "HELLO" 문자열을 움직이는 응용프로그램을 작성하라.

"HELLO" 문자열은 JLabel 컴포넌트로 만들어 컨텐트팬에 부착하고 상, 하, 좌, 우 키를 움직이면 키 방향으로 한 번에 10픽셀씩 움직인다. 이를 위해 컨텐트팬의 배치관리자를 삭제하여야 한다. "HELLO" 문자열을 초기에 (50, 50) 위치에 출력하라.




상,하,좌,우 키를 움직이면 한 번에 10픽셀씩 "HELLO" 텍스트는 상,하,좌,우로 이동한다.
이 텍스트는 프레임의 영역을 벗어나서 움직일 수 있다.

# 예제 소스: 상,하,좌,우 키로 텍스트 움직이기

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class FlyingTextEx extends JFrame {
    private final int FLYING_UNIT = 10;
    private JLabel la = new JLabel("HELLO");
    public FlyingTextEx() {
        setTitle("상,하,좌,우 키를 이용하여 텍스트 움직이기");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(null);
        c.addKeyListener(new MyKeyListener());

        la.setLocation(50,50);
        la.setSize(100,20);
        c.add(la);
        setSize(300,300);
        setVisible(true);
        c.setFocusable(true);
        c.requestFocus();

        c.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e) {
                Component com = (Component)e.getSource();
                com.setFocusable(true);
                com.requestFocus();
            }
        });
    }
```

```
    class MyKeyListener extends KeyAdapter {
        public void keyPressed(KeyEvent e) {
            int keyCode = e.getKeyCode();

            switch(keyCode) {
                case KeyEvent.VK_UP:
                    la.setLocation(la.getX(), la.getY()-FLYING_UNIT);
                    break;
                case KeyEvent.VK_DOWN:
                    la.setLocation(la.getX(), la.getY()+FLYING_UNIT);
                    break;
                case KeyEvent.VK_LEFT:
                    la.setLocation(la.getX()-FLYING_UNIT, la.getY());
                    break;
                case KeyEvent.VK_RIGHT:
                    la.setLocation(la.getX()+FLYING_UNIT, la.getY());
                    break;
            }
        }
    }

    public static void main(String [] args) {
        new FlyingTextEx();
    }
}
```