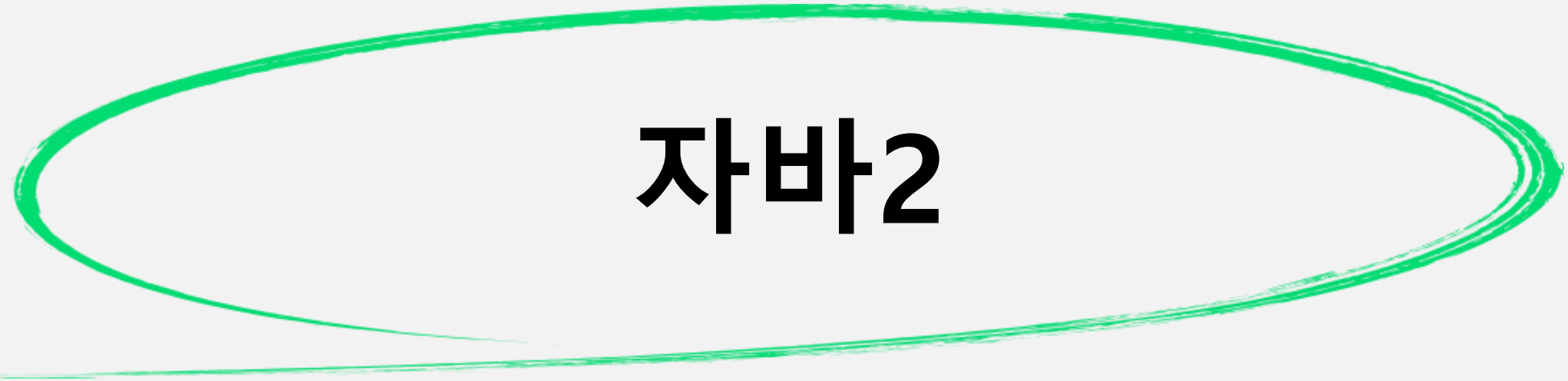


자바2

(GUI-2 : 이벤트 구동 프로그래밍)

Chapter 07
이벤트 구동 프로그래밍

Chapter07의 학습목표

- 자바에서의 이벤트 구동 방법에 대해 알아본다

Event Programming

이벤트 구동 프로그래밍 (Event-driven programming)

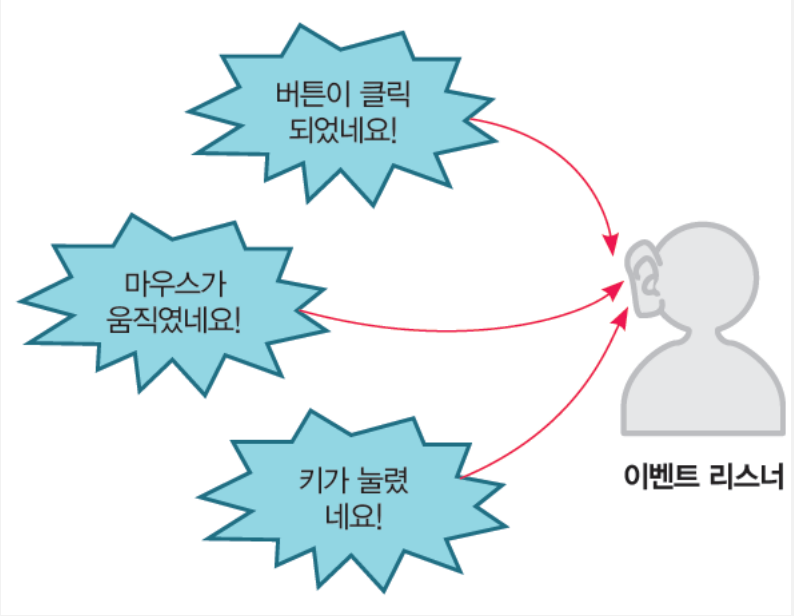
- 버튼 클릭과 같은 특정 이벤트가 발생하면 적절한 처리를 해주는 프로그래밍 방식

이벤트 처리 절차

감시중.. ➡ 버튼을 클릭 ➡ 이벤트 발생! ➡ 코드 실행

이벤트 리스너 (Event listener)

- 이벤트가 발생하면 이벤트 객체가 생성
- 이벤트 리스너 : 발생된 이벤트 객체에 반응하여서 이벤트를 처리하는 객체
- 이벤트 소스에 리스너를 등록하면 이벤트가 발생하였을 때 이벤트 리스너 내부의 처리 메소드가 호출됨



Event Programming

이벤트 리스너 작성 – 1. 이벤트 리스너 클래스 작성

- 리스너를 작성하는 개발자에게 interface로 제공됨
- 이벤트 리스너를 사용하기 위해서는 해당 인터페이스 내부의 추상 메소드를 구현해야 함
- 이벤트의 종류에 따라 다양한 리스너 인터페이스를 제공하기 때문에 이벤트에 맞는 인터페이스를 구현해야 함

```
interface ActionListener{
    public void actionPerformed(ActionEvent e);
}
```

```
class MyListener implements ActionListener{
    //액션이벤트가 발생하면 호출
    public void actionPerformed(ActionEvent e) {
        //액션 이벤트 처리하는 코드란
    }
}
```

Event Programming

이벤트 리스너 작성 – 1. 이벤트 리스너 클래스 작성

이벤트 종류	리스너 인터페이스	추상 메소드	발생 상황
Action	ActionListener	void actionPerformed(ActionEvent)	Action이 발생할 때
Key	KeyListener	void keyPressed(keyEvent)	키가 눌려질 때
		void keyRleased(keyEvent)	눌러진 키가 떼어질 때
		void keyTyped(KeyEvent)	유니코드키가 입력될 때
		void MousePressed(MouseEvent)	마우스버튼이 눌릴 때
Mouse	MouseListener	void MouseReleased(MouseEvent)	눌린 버튼이 떼어질 때
		void MouseClicked(MouseEvent)	클릭될 때
		void MouseEntered(MouseEvent)	커서가 컴포넌트 위에 올라올 때
		void MouseExisted(MouseEvent)	커서가 컴포넌트를 벗어날 때

Event Programming

이벤트 리스너 작성 – 2. 이벤트 리스너를 컴포넌트에 등록

- 각 컴포넌트는 이벤트 리스너를 등록할 수 있는 메소드를 제공

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

class MyListener implements ActionListener{
    //액션이벤트가 발생하면 호출
    public void actionPerformed(ActionEvent e) {
        System.out.println("버튼 클릭!");
    }
}
```

이벤트 리스너 객체를 생성하고
버튼에 이벤트 리스너 객체를 등록

```
public class MyFrame extends JFrame{
    public MyFrame() {
        this.setTitle("Frame test");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel p = new JPanel();
        //배치관리자를 사용하지 않고 절대 위치를 사용
        p.setLayout(null);
        JButton button = new JButton("b1");
        //(x위치, y위치, width, height)
        button.setBounds(20, 5, 95, 30);
        button.addActionListener(new MyListener());
        p.add(button);
        add(p);
        setVisible(true);
    }
```

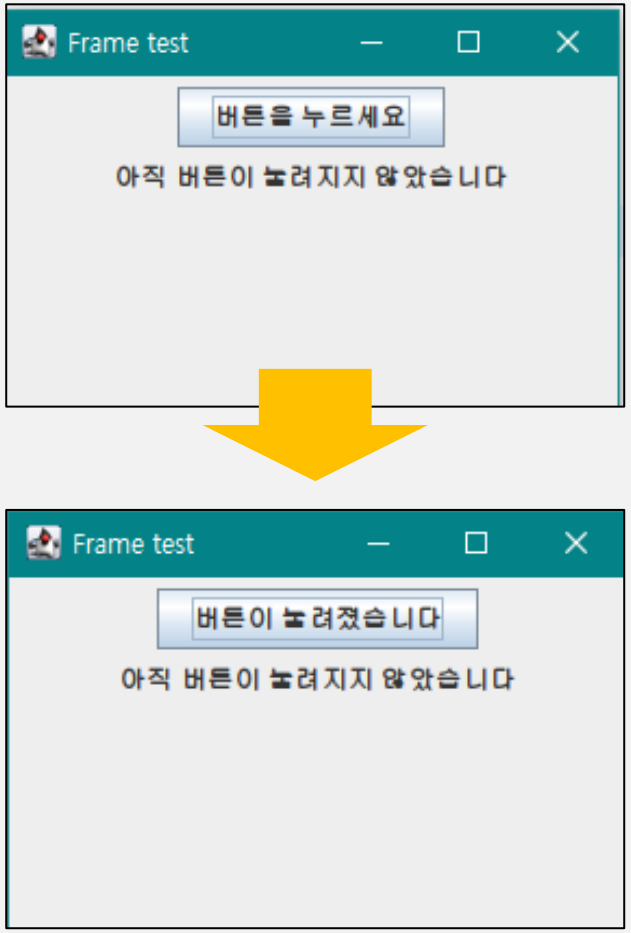
Event Programming

1. 리스너를 독립적인 클래스로 작성

- 리스너를 등록하는 가장 기본적인 방법

```
public class MyFrame extends JFrame{
    public MyFrame() {
        this.setTitle("Frame test");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel p = new JPanel();
        JButton button = new JButton("버튼을 누르세요");
        JLabel label = new JLabel("아직 버튼이 눌려지지 않았습니다");
        button.addActionListener(new MyListener());
        p.add(button);
        p.add(label);
        add(p);
        setVisible(true);
    }

    public static void main(String[] args) {
        MyFrame frame = new MyFrame();
    }
}
```



Event Programming

1. 리스너를 독립적인 클래스로 작성

- 리스너를 등록하는 가장 기본적인 방법

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

class MyListener implements ActionListener{
    //액션이벤트가 발생하면 호출
    public void actionPerformed(ActionEvent e) {
        JButton button = (JButton) e.getSource();
        button.setText("버튼이 눌려졌습니다");
        //label.setText("버튼이 눌려졌네요");
    }
}
```

이벤트 객체가 가지고 있는 getSource()메소드를 사용하여 이벤트를 발생한 이벤트 소스를 Object타입으로 반환

label은 MyFrame 클래스 안에 있어서 접근이 어려움

Event Programming

2. 리스너 클래스를 내부 클래스로 작성

- 내부 클래스는 외부 클래스의 멤버 변수들을 자유롭게 사용가능

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

class MyFrame extends JFrame{
    private JButton button;
    private JLabel label;

    public MyFrame() {
        this.setTitle("Frame test");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel p = new JPanel();
        button = new JButton("버튼을 누르세요");
        label = new JLabel("아직 버튼이 눌려지지 않았습니다");
        button.addActionListener(new MyListener());
        p.add(button);
        p.add(label);
        add(p);
        setVisible(true);
    }
```

생성자와 actionPerformed() 메소드에서 사용하기 때문에 멤버 변수로 설정

Event Programming

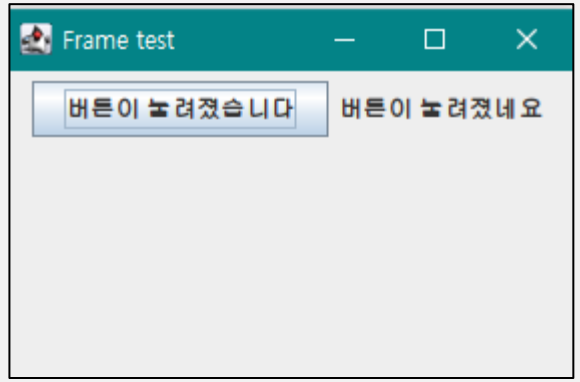
2. 리스너 클래스를 내부 클래스로 작성

- 내부 클래스는 외부 클래스의 멤버 변수들을 자유롭게 사용가능

```
private class MyListener implements ActionListener{
    //액션이벤트가 발생하면 호출
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == button) {
        button.setText("버튼이 눌려졌습니다");
        label.setText("버튼이 눌려졌네요");
        }
    }
}

public static void main(String[] args) {
    MyFrame frame = new MyFrame();
}
}
```

MyListener 클래스는 MyFrame의 내부 클래스로 설정하여 MyFrame의 멤버 변수에 접근할 수 있도록 만든다



Event Programming

3. 클래스가 JFrame을 상속받으면서 동시에 리스너 인터페이스를 구현

- 하나의 클래스 안에서 모든 작업이 가능

```
class MyFrame extends JFrame implements ActionListener{
    private JButton button;
    private JLabel label;

    public MyFrame() {
        this.setTitle("Frame test");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel p = new JPanel();
        button = new JButton("버튼을 누르세요");
        label = new JLabel("아직 버튼이 눌려지지 않았습니다");
        //현재 객체를 이벤트 리스너로 버튼에 등록
        //자기 자신이 이벤트를 처리한다고 등록하는 것과 같음
        button.addActionListener(this);
        p.add(button);
        p.add(label);
        add(p);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == button) {
        button.setText("버튼이 눌려졌습니다");
        label.setText("버튼이 눌려졌네요");
        }
    }

    public static void main(String[] args) {
        MyFrame frame = new MyFrame();
    }
}
```

Event Programming

4. 무명 클래스를 사용

- 클래스가 정의되면서 바로 사용됨

```
public MyFrame() {
    this.setTitle("Frame test");
    setSize(300, 200);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JPanel p = new JPanel();
    button = new JButton("버튼을 누르세요");
    label = new JLabel("아직 버튼이 눌려지지 않았습니다");
    //무명 클래스는 ActionListener 인터페이스를 구현
    button.addActionListener(new ActionListener() {
        //무명 클래스를 생성하고 actionPerformed메소드를 새로 정의한다
        @Override
        public void actionPerformed(ActionEvent e) {
            if(e.getSource() == button) {
            button.setText("버튼이 눌려졌습니다");
            label.setText("버튼이 눌려졌네요");
            }
        }
    });
```

Event Programming

5. 람다식을 이용

- JDK8부터 사용이 가능한 람다식을 사용
- 람다식은 함수를 객체로 만들어서 메소드에 전달이 가능

```
public MyFrame() {
    this.setTitle("Frame test");
    setSize(300, 200);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JPanel p = new JPanel();
    button = new JButton("버튼을 누르세요");
    label = new JLabel("아직 버튼이 눌려지지 않았습니다");
    //무명 클래스는 ActionListener 인터페이스를 구현
    button.addActionListener(e -> {
        button.setText("버튼이 눌려졌습니다");
        label.setText("버튼이 눌려졌네요");
    });
```

Event Programming

스윙 컴포넌트의 이벤트

- 스윙 컴포넌트가 발생하는 이벤트는 모든 컴포넌트가 공통적으로 지원하는 저수준 이벤트와 일부 컴포넌트만 지원하는 의미적 이벤트로 나뉨

저수준 이벤트 (low-level event)

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

Event Programming

스윙 컴포넌트의 이벤트

- 스윙 컴포넌트가 발생하는 이벤트는 모든 컴포넌트가 공통적으로 지원하는 저수준 이벤트와 일부 컴포넌트만 지원하는 의미적 이벤트로 나뉨

의미적 이벤트 (semantic event)

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

Event Programming

액션 이벤트 (Action Event) 예제 : 배경색 변경

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

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

class MyFrame extends JFrame {
    private JButton button1;
    private JButton button2;
    private JPanel panel;
    MyListener listener = new MyListener();
```

Event Programming

액션 이벤트 (Action Event) 예제 : 배경색 변경

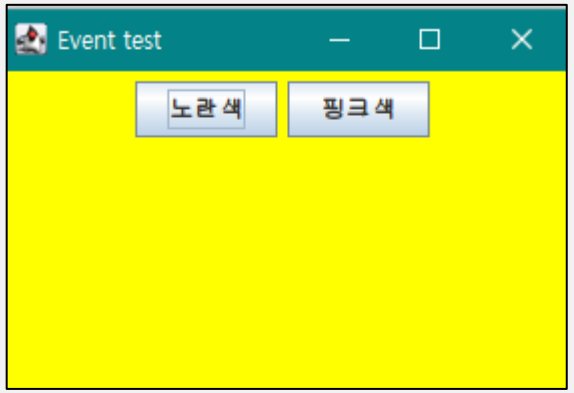
```
public MyFrame() {
    this.setTitle("Event test");
    setSize(300, 200);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    panel = new JPanel();
    button1 = new JButton("노란색");
    button2 = new JButton("핑크색");
    button1.addActionListener(listener);
    button2.addActionListener(listener);
    panel.add(button1);
    panel.add(button2);
    add(panel);
    setVisible(true);
}
```

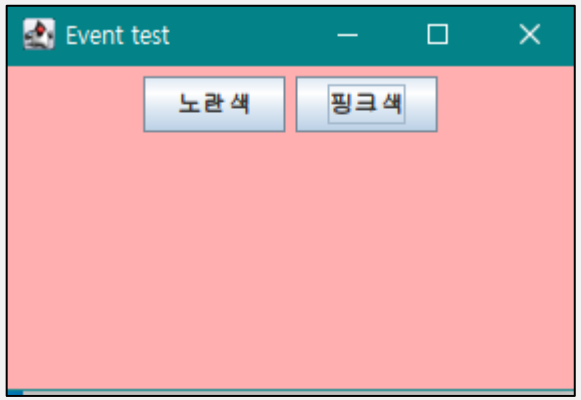
Event Programming

액션 이벤트 (Action Event) 예제 : 배경색 변경

```
    private class MyListener implements ActionListener{
        @Override
        public void actionPerformed(ActionEvent e) {
            if(e.getSource() == button1) {
                panel.setBackground(Color.yellow);
            }
            else if(e.getSource() == button2) {
                panel.setBackground(Color.pink);
            }
        }
    }

    public static void main(String[] args) {
        MyFrame frame = new MyFrame();
    }
}
```





Event Programming

키 이벤트 (Key Event)

- 사용자가 키보드를 이용하여 입력을 하는 경우에 발생
- 키를 누를 때, 버튼에서 손을 뗄 때 발생
- KeyEvent가 발생하기 위해서는 컴포넌트가 반드시 키보드 포커스를 가지고 있어야 함
- requestFocust() 메소드 : 키보드 포커스를 얻는 메소드
- KeyListener 클래스를 구현해야 함

keyTyped 이벤트 : 입력된 유니코드 문자가 전송됨

keyPressed, **keyReleased 이벤트** : 사용자가 키를 누르거나 키에서 손을 뗄 때 이벤트 발생

메소드	호출 상황
keyTyped(KeyEvent e)	사용자가 클자를 입력했을 경우
keyPressed(KeyEvent e)	사용자카 키를 눌렀을 경우
keyReleased(KeyEvent e)	사용자가 키에서 손을 뗐을 경우

Event Programming

키 이벤트 (Key Event) 예제 1 : 키 이벤트 표시하기

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

class MyFrame extends JFrame implements KeyListener {
    private JPanel panel;
    private JTextField field;
    private JTextArea area;
```

Event Programming

키 이벤트 (Key Event) 예제 1 : 키 이벤트 표시하기

```
public MyFrame() {
    setTitle("KeyEvent Test");
    setSize(400, 200);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    panel = new JPanel(new GridLayout(0, 2));
    field = new JTextField(10);
    field.addKeyListener(this);

    panel.add(new JLabel("문자를 입력하시오: "));
    panel.add(field);

    area = new JTextArea(3, 30);

    add(panel, BorderLayout.NORTH);
    add(area, BorderLayout.CENTER);

    setVisible(true);
}
```

```
public void keyTyped(KeyEvent e) {
    display(e, "Key Typed ");
}
public void keyPressed(KeyEvent e) {
    display(e, "Key Pressed ");
}
public void keyReleased(KeyEvent e) {
    display(e, "Key Released ");
}
```

Event Programming

키 이벤트 (Key Event) 예제 1 : 키 이벤트 표시하기

```
    protected void display(KeyEvent e, String s) {
        char c = e.getKeyChar(); //입력한 문자를 얻음
        int keyCode = e.getKeyCode(); //눌려진 유니코드 값을 얻음
        String modifiers =
                "Alt: " + e.isAltDown() + " / "
                + "Ctrl: " + e.isControlDown() + " / "
                + "Shift: " + e.isShiftDown();
        area.append(s + "문자 " + c + "(코드: " + keyCode + ") " + modifiers + "\n");
    }


    public static void main(String[] args) {
        MyFrame frame = new MyFrame();
    }
}
```

Event Programming

키 이벤트 (Key Event) 예제 2 : 키를 입력 받아 그림을 움직이는 프로그램

```
import javax.imageio.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;

class MyPanel extends JPanel{
    BufferedImage img = null;
    Image resizeImage = null;
    int img_x = 100, img_y = 100;

    public MyPanel() {
        try {
            img = ImageIO.read(new File("run.png"));
            resizeImage = img.getScaledInstance(200, 200, Image.SCALE_SMOOTH);
        }
        catch (IOException e) {
            System.out.println("no image");
            System.exit(1);
        }
```

Event Programming

키 이벤트 (Key Event) 예제 2 : 키를 입력 받아 그림을 움직이는 프로그램

```
        addKeyListener(new KeyListener() {
            public void keyPressed(KeyEvent e) {
                int keycode = e.getKeyCode();
                switch (keycode) {
                case KeyEvent.VK_UP:         img_y -= 10;    break;
                case KeyEvent.VK_DOWN:   img_y += 10;    break;
                case KeyEvent.VK_LEFT:   img_x -= 10;    break;
                case KeyEvent.VK_RIGHT: img_x += 10;    break;
                }
                repaint();
            }
            public void keyReleased(KeyEvent arg0) {}
            public void keyTyped(KeyEvent arg0) {}
        });
        this.requestFocus();
        setFocusable(true);
    }
```

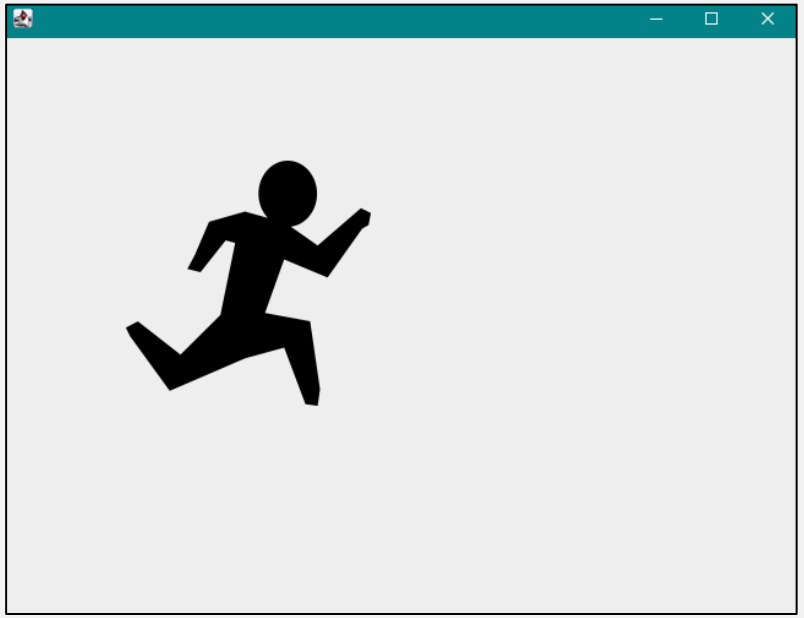
Event Programming

키 이벤트 (Key Event) 예제 2 : 키를 입력 받아 그림을 움직이는 프로그램

```
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawImage(resizeImage, img_x, img_y, null);
    }
}

class MyFrame extends JFrame{
    public MyFrame() {
        setSize(300, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        add(new MyPanel());
        setVisible(true);
    }

    public static void main(String[] args) {
        MyFrame frame = new MyFrame();
    }
}
```



Event Programming

마우스와 마우스 모션 이벤트 (Mouse, MouseMotion Event)

- 사용자가 마우스 버튼을 누르거나 마우스를 움직일 경우에 발생
- 마우스 클릭 : MouseListener 클래스를 구현
- 마우스 모션 : MouseMotionListener 클래스를 구현

mouse Pressed() -> mouseReleased() -> mouseClicked() 순서로 호출

인터페이스	메소드	호출 상황
MouseListener	mouseClicked(MouseEvent e)	사용자가 컴포넌트를 클릭한 경우
	mouseEntered(MouseEvent e)	마우스 커서가 컴포넌트로 들어갔을 경우
	mouseExited(MouseEvent e)	마우스 커서가 컴포넌트에서 나갔을 경우
	mousePressed(MouseEvent e)	마우스가 컴포넌트 위에서 눌려졌을 경우
	mouseReleased(MouseEvent e)	마우스가 컴포넌트 위에서 떼어졌을 경우
MouseMotionListener	mouseDragged(MouseEvent e)	마우스를 드래그 했을 경우
	mouseMoved(MouseEvent e)	마우스가 클릭되지 않고 이동하는 경우

Event Programming

마우스 이벤트 예제 : 마우스 입력을 받아 그림을 움직이는 프로그램

키 이벤트 예제에서 리스너 부분만 수정

```
    addMouseListener(new MouseListener() {
        public void mousePressed(MouseEvent e) {
            img_x = e.getX();
            img_y = e.getY();
            repaint();
        }
        public void mouseReleased(MouseEvent e) {}
        public void mouseEntered(MouseEvent e) {}
        public void mouseExited(MouseEvent e) {}
        public void mouseClicked(MouseEvent e) {}
    });

}
```

Event Programming

어댑터 클래스 (Adaptor Class)

- 미리 리스너의 모든 메소드를 구현해놓은 클래스
- 이벤트를 처리하기 위해서 리스너 인터페이스에서 정의도어 있는 모든 메소드를 구현해야 하는 불편함이 있음
- 각 리스너에 대응되는 어댑터 클래스를 상속받아 원하는 메소드만 정의 가능

인터페이스	어댑터 클래스
ComponentListener	ComponentAdapter
ContainerListener	ContainerAdapter
FocusListener	FocusAdapter
KeyListener	KeyAdapter
MouseListener	MouseAdapter
MouseMotionListener	MouseMotionAdapter
WindowListener	WindowAdapter

Event Programming

어댑터 클래스의 문제점

- 어댑터는 클래스이기 때문에 상속을 받아 사용해야 함 (구현이 아님)
- 다중 상속이 불가능 : 마우스를 처리하는 프레임 = MouseAdaptor + JFrame 사용 불가
- MouseAdaptor를 내부 클래스로 정의하여 사용가능

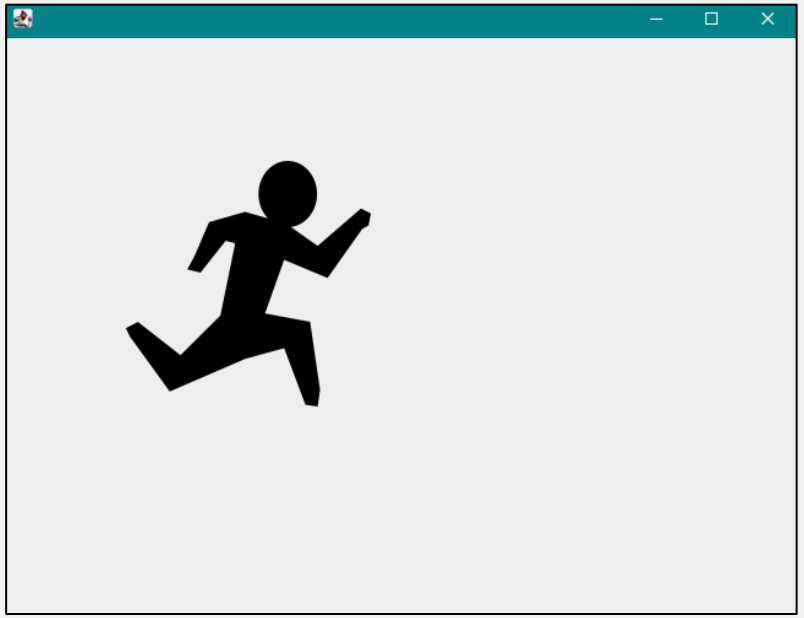
```
public class MyClass extends MouseAdapter {
    public MyClass() {
            // ...
            someObject.addMouseListener(this);
    }
    public void mouseClicked(MouseEvent e) {
            // ...
    }
}
```

```
public class MyClass extends JFrame {
    addMouseListener(new MyMouseAdapter());
            class MyMouseAdapter() extends MouseAdapter {
                    public void mouseClicked(MouseEvent e) {
                    // ...
                    }
            }
}
```

Event Programming

Adapter class 예제 : 키를 입력 받아 그림을 움직이는 프로그램 수정

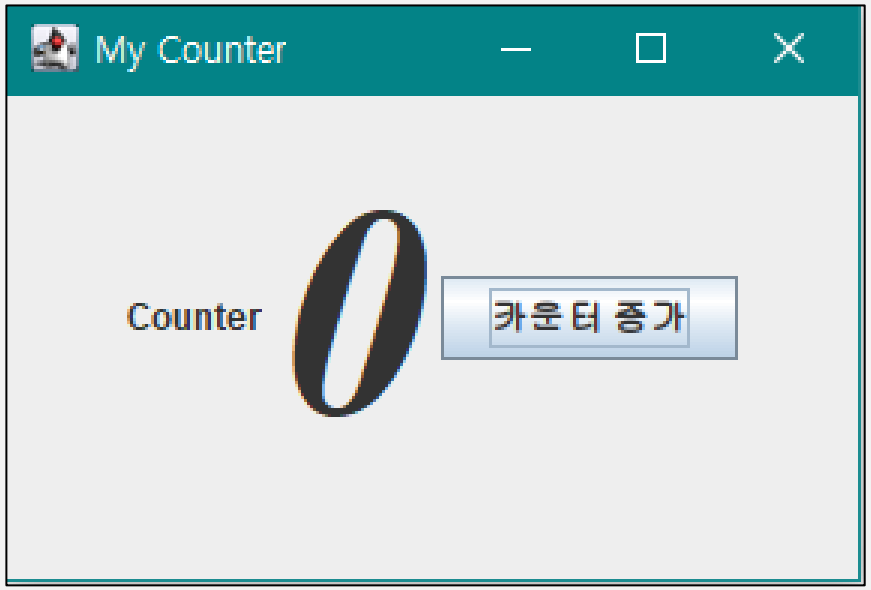
```
addMouseListener(new MouseAdapter() {
    public void mousePressed(MouseEvent e) {
        img_x = e.getX();
        img_y = e.getY();
        repaint();
    }
});
```

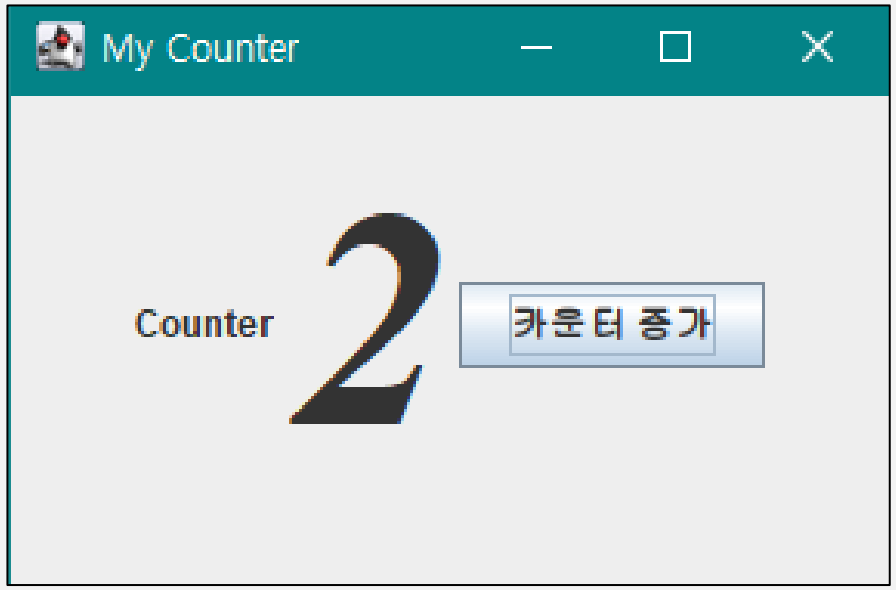


Event Programming

카운터 작성하기

카운터의 초기값은 0이고 “카운터 증가” 버튼을 누르면 카운터 값이 하나씩 증가되어서 화면에 표시되는 프로그램을 작성





Event Programming

카운터 작성하기

```
class MyFrame extends JFrame implements ActionListener {
    private JLabel label, label1;
    private JButton button;
    private int count = 0;

    public MyFrame() {
        setSize(300, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel panel = new JPanel();
        label = new JLabel("Counter");
        label1 = new JLabel(String.valueOf(count));
        //레이블에 폰트를 설정
        label1.setFont(new Font("Serif", Font.BOLD | Font.ITALIC, 100));
```

Event Programming

카운터 작성하기

```
        button = new JButton("카운터 증가");
        button.addActionListener(this);

        panel.add(label);
        panel.add(label1);
        panel.add(button);

        add(panel);
        setSize(300, 200);
        setTitle("My Counter");
        setVisible(true);
    }
```

```
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        count++;
        label1.setText(String.valueOf(count));
    }

    public static void main(String[] args) {
        MyFrame frame = new MyFrame();
    }
}
```

수고하셨습니다.