제 48강

Java GUI Programming 2

교재: p281~291

목차

복습

GUI(Graphical User Interface)

: 컴퓨터를 사용하는 사용자를 위해 만들어진 특정한 기능을 가진 그래픽 요소

JAVA GUI 프로그래밍

: 사용자가 프로그램을 쉽게 다룰 수 있도록 그래픽을 제공하도록 프로그래밍 하는 것

종류

[1] AWT : 자바에서 GUI를 프로그래밍 하기 위해 처음으로 제공한 라이브러리
[2] Swing:: AWT를 대체하기 위해 자바로 작성된 GUI 객체

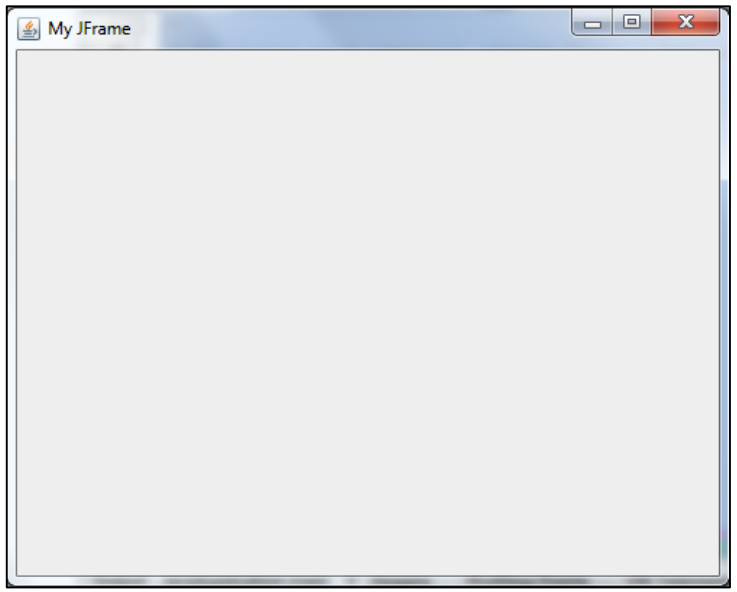
- AWT와 Swing의 차이점: AWT는 운영체제의 자원 활용

복습

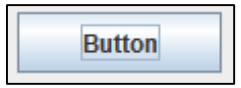
컴포넌트(Component)와 컨테이너(Container)

- 컴포넌트: JAVA에서 GUI를 구성하는 요소
- 컨테이너: 다른 컴포넌트를 포함할 수 있는 컴포넌트

JFrame



JButton



JCheckBox

JSlider

JTextField

JComboBox

JMenu

복습

GUI 프로그래밍 순서

1. extends JFrame

- JFrame 클래스를 상속받아, 나만의 프레임 클래스 작성
- 생성자에서 컴포넌트 및 프레임의 디자인 구현하기

```
public class AddFrame extends JFrame
```

```
public AddFrame() {
```

2. 기본설정 - 생성자에서 하기

: title, size, visible 설정하기

- title 설정 : setTitle("타이틀명");
- size 설정: setSize(가로길이,세로길이);
- visible 설정: setVisible(true);

```
setTitle("JFrame");
setSize(300,300);
setVisible(true);
```

1. 이벤트(Event) 처리

이벤트(Event)란?

: 프로그램을 실행하는 도중 사용자에 의해 발생하는 키보드 입력, 마우스 클릭 등의 동작

이벤트 처리란?

: 이벤트가 발생했을 때, 어떤 작업을 할 것인지 결정하는 것

1. 이벤트(Event) 처리

[1] 이벤트 객체

: 이벤트 발생 시 이벤트에 대한 정보를 가진 객체
ex) 이벤트의 종류, 이벤트가 발생한 위치(ex. 마우스 좌표), 체크박스의 체크 상태

[2] 이벤트 리스너

: 이벤트 객체에 의해서 호출 되어 해당 이벤트를 처리하는 객체
- 이벤트 리스너 사용 시 해당 인터페이스의 추상 메서드 구현해야함

1. 이벤트(Event) 처리

[2] 이벤트 리스너

이벤트에 따른 이벤트 리스너의 종류

이벤트 종류	리스너 인터페이스	추상메서드	발생 상황
Action	ActionListener	void actionPerformed(ActionEvent)	Action이 발생 시
Key	KeyListener	void KeyPressed(KeyEvent)	키가 눌려질 때
		void KeyReleased(KeyEvent)	눌러진 키를 뗄 때
		void keyTyped(KeyEvent)	유니코드 키 입력 시
Mouse	MouseListener	void MousePressed(MouseEvent)	마우스버튼이 눌릴 때
		void MouseReleased(MouseEvent)	눌린 마우스 뗄 때
		void MouseClicked(MouseEvent)	클릭 시
		void MouseEntered(MouseEvent)	마우스 포인터가 컴포넌트 위에 올라올 때
		vois MouseExisted(MouseEvent)	마우스 포인터가 컴포넌트를 벗어날 때

1. 이벤트(Event) 처리

<실습> Exam-104.java

앞에서 다루었던 예제에 이벤트 요소 넣은 코드

1) 이벤트 리스너에서 이벤트 발생시 할 작업 오버라이딩

```
class Listner1 implements ActionListener{
    @Override
    public void actionPerformed(ActionEvent e) {
            JButton button = (JButton)e.getSource();
            button.setText("btn clicked!");
    }}
```

2) Frame 작성

```
class AddFrame2 extends JFrame{
    public AddFrame2() {
        setTitle("JFrame");
        setSize(300,300);

        JButton button= new JButton("Button");
        button.addActionListener(new Listner1());
        JCheckBox box= new JCheckBox("checkbox");
        JSlider slider= new JSlider();
        JTextField tf= new JTextField("input text",20);

        this.setLayout(new FlowLayout());
        this.add(button);
        this.add(box);
        this.add(slider);
        this.add(tf);
        setVisible(true);

        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

1. 이벤트(Event) 처리

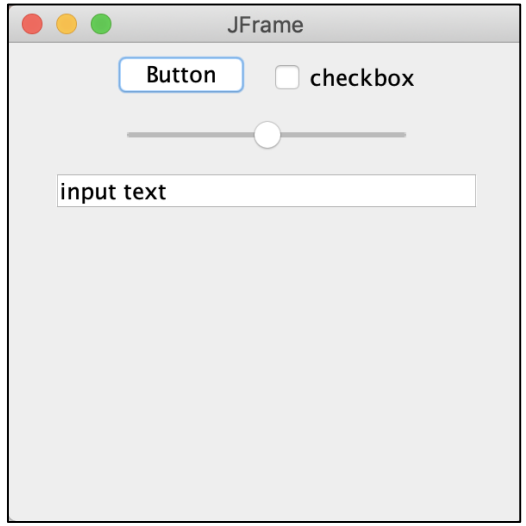
<실습> Exam-104.java

앞에서 다루었던 예제에 이벤트 요소 넣은 코드

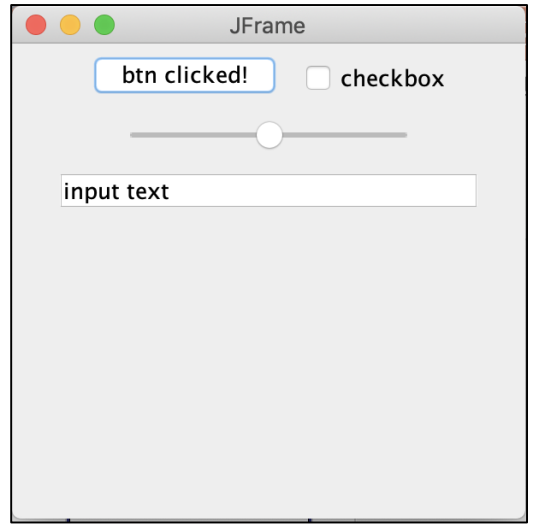
3) 만든 Frame 생성자 호출

```
public class Eve {
    public static void main(String[] args) {
        new AddFrame2();
    }
}
```

<실행 결과>



이벤트 발생 전



이벤트 발생 후

1. 이벤트(Event) 처리

이벤트 리스너 클래스의 구조

```
class Listner1 implements ActionListener{
    @Override
    public void actionPerformed(ActionEvent e) {
            JButton button = (JButton)e.getSource();
            button.setText("btn clicked!");
    }}
```

이벤트 발생 시 e 객체에 이벤트 정보가 전달

getSource( )의 리턴타입이 Object 여서 강제형변환

```
JButton button = (JButton)e.getSource();
```

e.getSource( ) : 현재 이벤트가 발생된 컴포넌트

```
button.setText("btn clicked!");
```

button.setText("문자열"); : 현재 버튼의 문자열 재지정

1. 이벤트(Event) 처리

[3] 마우스 관련 이벤트

이벤트 종류	리스너 인터페이스	추상 메서드	발생 상황
Mouse	MouseMotionListener	void mouseDragged(MouseEvent e)	마우스가 드래그 될 때
		void mouseMoved(MouseEvent e)	마우스가 움직일 때

마우스 이벤트 발생시 얻을 수 있는 정보

메서드	메서드의 활용
int getX( )	현재 마우스 포인터의 X좌표
int getY( )	현재 마우스 포인터의 Y좌표
short getButton( )	현재 클릭한 버튼(왼쪽,오른쪽)
int getClickCount( )	마우스를 클릭한 횟수

1. 이벤트(Event) 처리

<실습> Exam-105.java

간단한 마우스이벤트 처리

```
public class MouseEx extends JFrame{
    JPanel jp = new JPanel();
    JLabel la;
    MouseEx(){
        setTitle("mouse event");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(jp);
        jp.addMouseListener(new MouseListener1());
        jp.addMouseMotionListener(new MouseListener1());
        la=new JLabel("마우스를 위에 올려보세요");
        jp.add(la);
        setSize(300,300);
        setVisible(true);
    }
    public static void main(String[] args) {
        new MouseEx();
    }
```

MouseEx 클래스의 내부 클래스로 작성

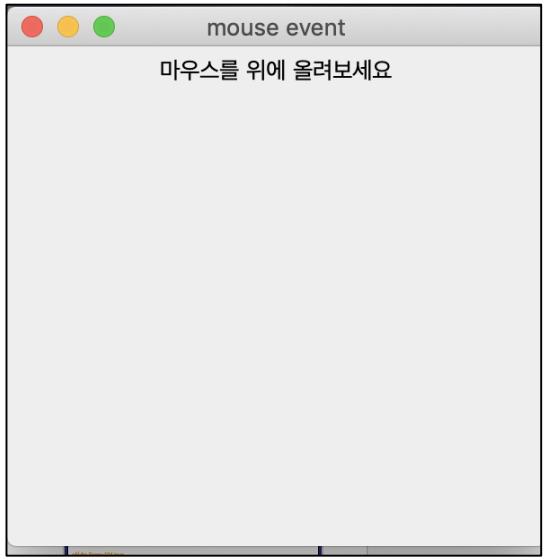
→ 멤버 변수 접근을 위해!

```
class MouseListener1 implements MouseListener,MouseMotionListener{
    @Override
    public void mouseDragged(MouseEvent e) {
        la.setText("mouse Dragged("+e.getX()+","+e.getY()+")");
    }
    @Override
    public void mouseMoved(MouseEvent e) {
        la.setText("mouse Moved("+e.getX()+","+e.getY()+")");
    }
    @Override
    public void mouseClicked(MouseEvent e) {
        la.setText("마우스 클릭횟수: "+e.getClickCount());
    }
    public void mousePressed(MouseEvent e) {
    public void mouseReleased(MouseEvent e) {
    @Override
    public void mouseEntered(MouseEvent e) {
        JPanel p =(JPanel)e.getSource();
        p.setBackground(Color.RED);
    }
    @Override
    public void mouseExited(MouseEvent e) {
        JPanel p =(JPanel)e.getSource();
        p.setBackground(Color.YELLOW);
    }
}
```

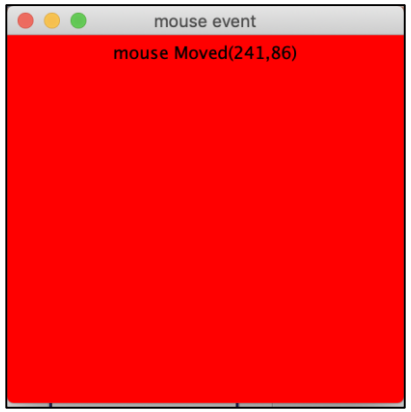
1. 이벤트(Event) 처리

<실습> Exam-105.java

간단한 마우스이벤트 처리

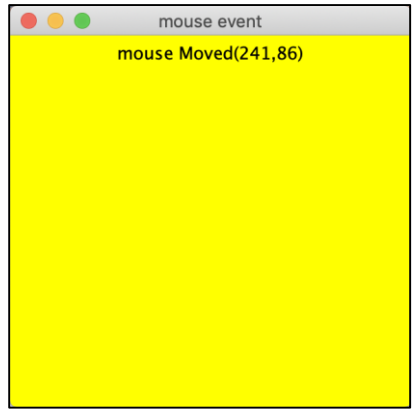



```
public void mouseMoved(MouseEvent e) {
    la.setText("mouse Moved("+e.getX()+","+e.getY()+")");
}
```

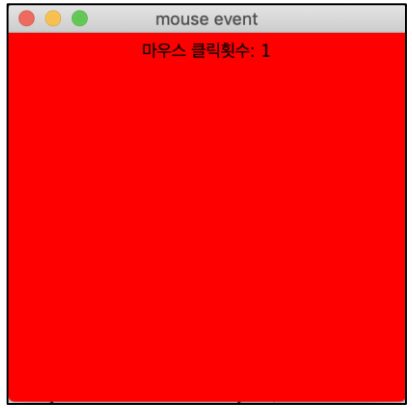



```
public void mouseEntered(MouseEvent e) {
    JPanel p =(JPanel)e.getSource();
    p.setBackground(Color.RED);
}
```

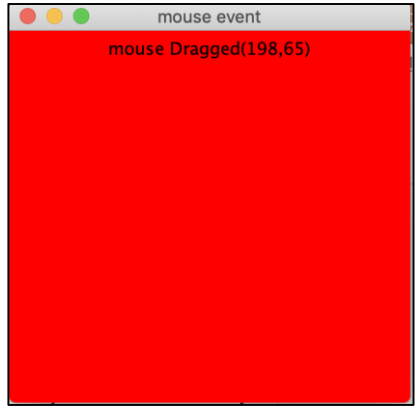
```
public void mouseExited(MouseEvent e) {
    JPanel p =(JPanel)e.getSource();
    p.setBackground(Color.YELLOW);
}
```



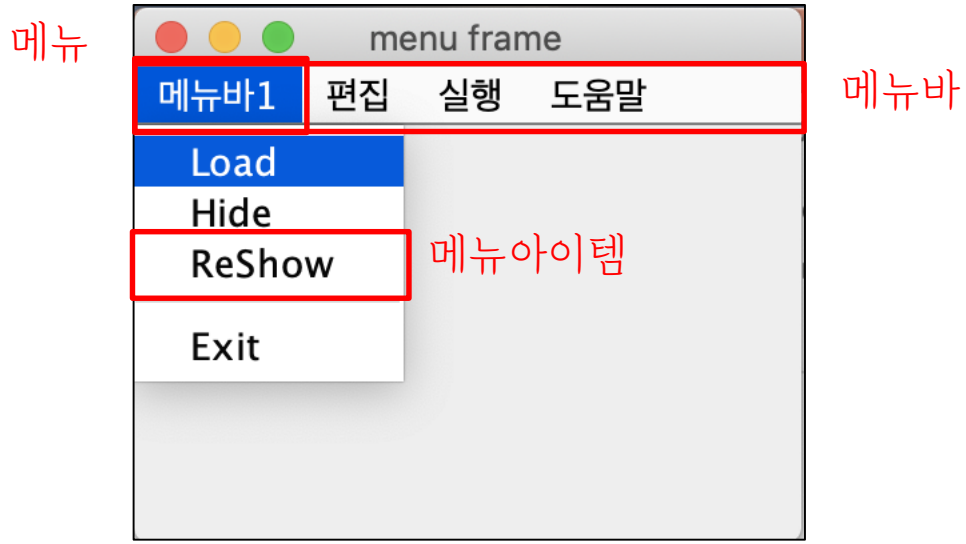

```
public void mouseClicked(MouseEvent e) {
    la.setText("마우스 클릭횟수: "+e.getClickCount());
}
```




```
public void mouseDragged(MouseEvent e) {
    la.setText("mouse Dragged("+e.getX()+","+e.getY()+")");
}
```



1. 이벤트(Event) 처리



[Menu 만들기]

1) JMenuBar 객체 생성

```
JMenuBar mb = new JMenuBar();
```

2) JMenu 생성

```
JMenu screenMenu = new JMenu("메뉴바1");
```

3) JMenuItem 생성

4) JMenu에 JMenuItem 추가

```
screenMenu.add(new JMenuItem("Load"));
```

5) JMenuBar에 JMenu 추가

```
mb.add(screenMenu);
```

6) 현재 프레임의 메뉴바를 생성한 JMenuBar 객체로 지정

```
setJMenuBar(mb);
```

1. 이벤트(Event) 처리

<실습> Exam-106.java

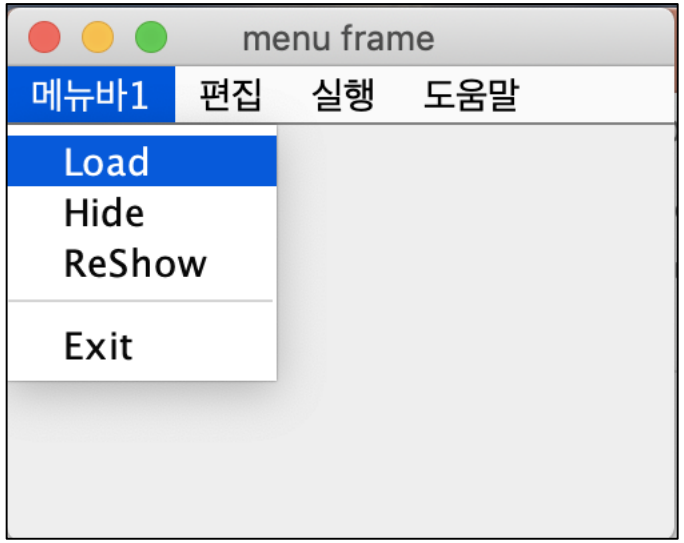
메뉴 만들기

```
public class MenuTest extends JFrame{
    MenuTest(){
        setTitle("menu frame");
        createMenu();
        setSize(250,200);
        setVisible(true);
    }
    void createMenu() {
        JMenuBar mb = new JMenuBar();
        JMenu screenMenu = new JMenu("메뉴바1");

        screenMenu.add(new JMenuItem("Load"));
        screenMenu.add(new JMenuItem("Hide"));
        screenMenu.add(new JMenuItem("ReShow"));
        screenMenu.addSeparator();
        screenMenu.add(new JMenuItem("Exit"));

        mb.add(screenMenu);
        mb.add(new JMenu("편집"));
        mb.add(new JMenu("실행"));
        mb.add(new JMenu("도움말"));
        setJMenuBar(mb);
    }
    public static void main(String[] args) {
                new MenuTest();
    }
}
```

<실행 결과>



1. 이벤트(Event) 처리

<실습> Exam-107.java

해당 메뉴에 기능 추가

```
public class MenuTest2 extends JFrame{
    MenuTest2(){
    void createMenu() {
        JMenuBar mb = new JMenuBar();
        JMenu screenMenu = new JMenu("메뉴바1");

        screenMenu.add(new JMenuItem("팝업"));
        screenMenu.add(new JMenuItem("Hide"));
        screenMenu.add(new JMenuItem("ReShow"));
        screenMenu.addSeparator();
        screenMenu.add(new JMenuItem("Exit"));

        mb.add(screenMenu);
        mb.add(new JMenu("편집"));
        mb.add(new JMenu("실행"));
        mb.add(new JMenu("도움말"));

        setJMenuBar(mb);

        JMenuItem item=  new JMenuItem("팝업");
        item.addActionListener(new MenuActionListener());
        screenMenu.add(item);
        mb.add(item);
    }
    public static void main(String[] args) {
}
```

```
class MenuActionListener implements ActionListener{
    @Override
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(null, "팝업");
    }}
```

<실행 결과>