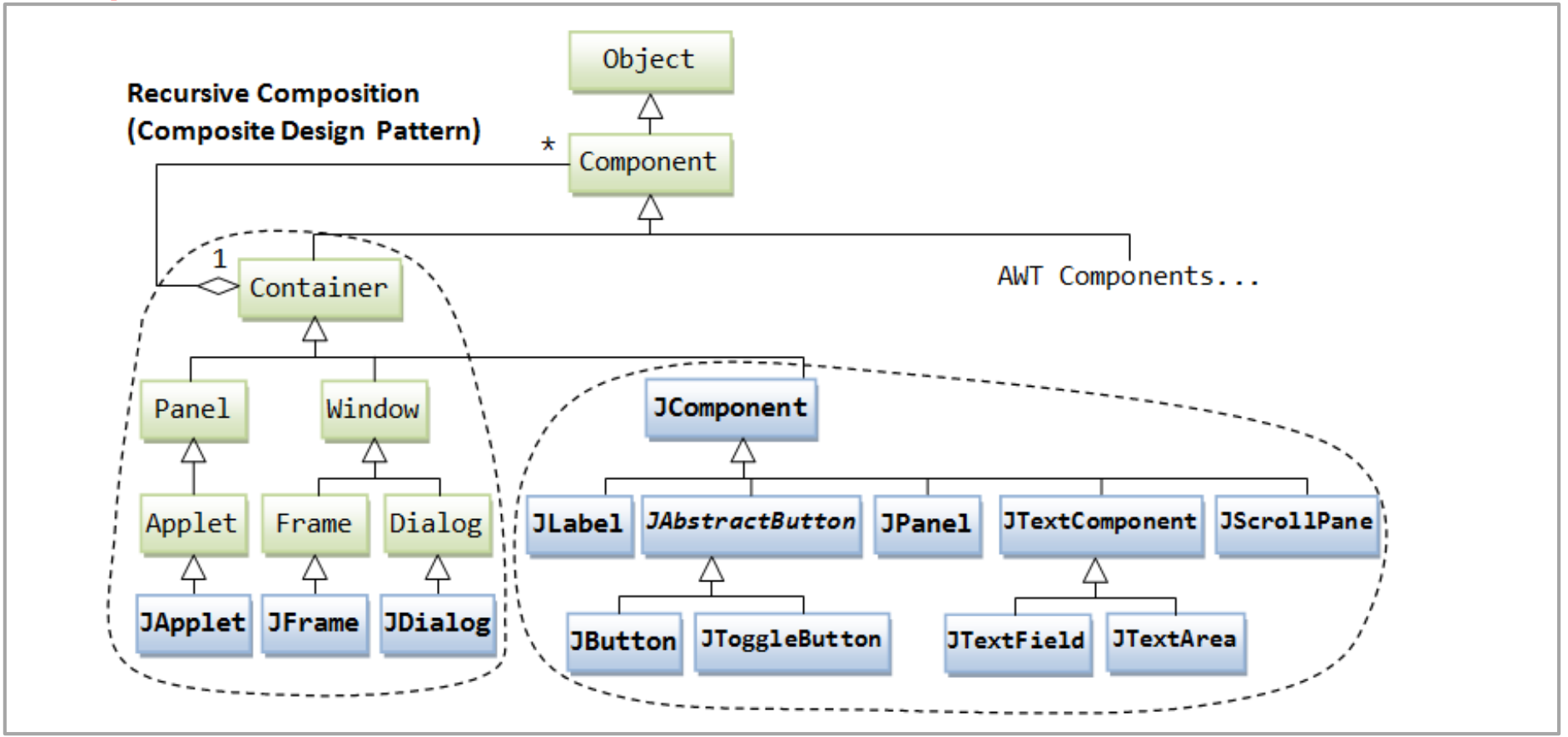
{ JAVA }

선린인터넷
웹 운 영 과

그래픽 사용자 인터페이스 : GUI-4

- Event 처리
- 어뎁터 클래스 맛보기

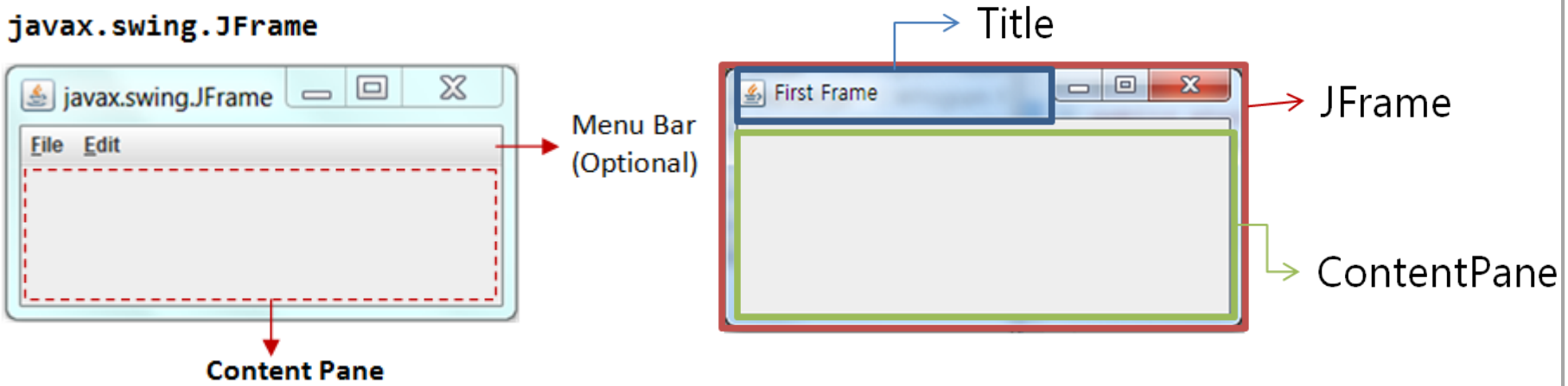
Component & Container 복습



JFrame 복습

❖ 모든 스윙 컴포넌트를 담는 최상위 GUI 컨테이너 : JFrame

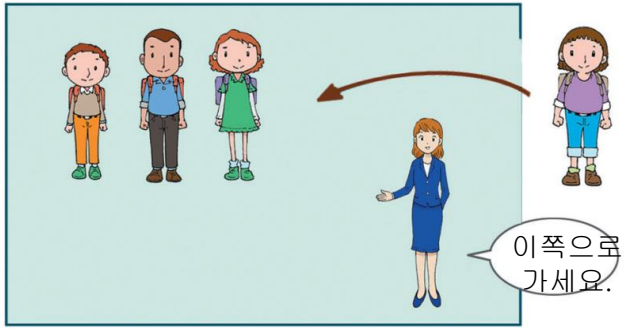
- 컴포넌트가 화면에 보이려면 프레임의 **Content Pane 객체**에 부착되어야 함



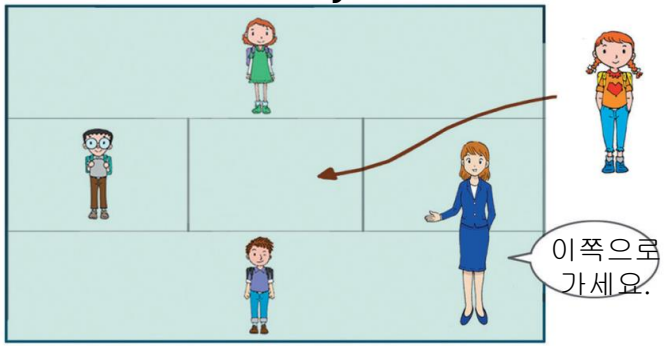

```
Container cp = aJFrame.getContentPane();
aJFrame.setContentPane(aPanel);
```

컨테이너와 배치 개념 복습

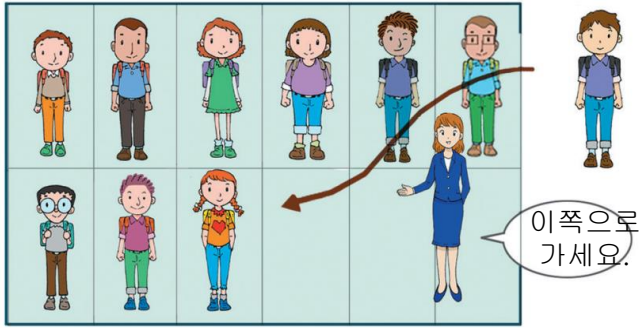
FlowLayout



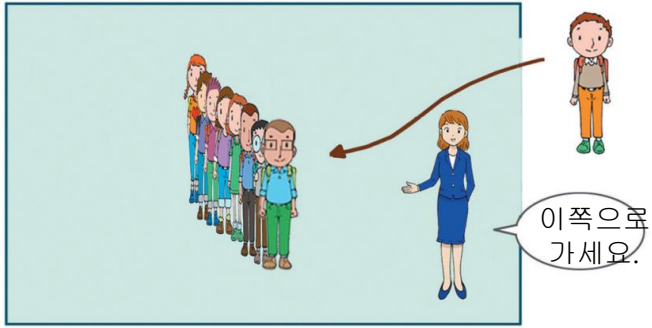
BorderLayout



GridLayout



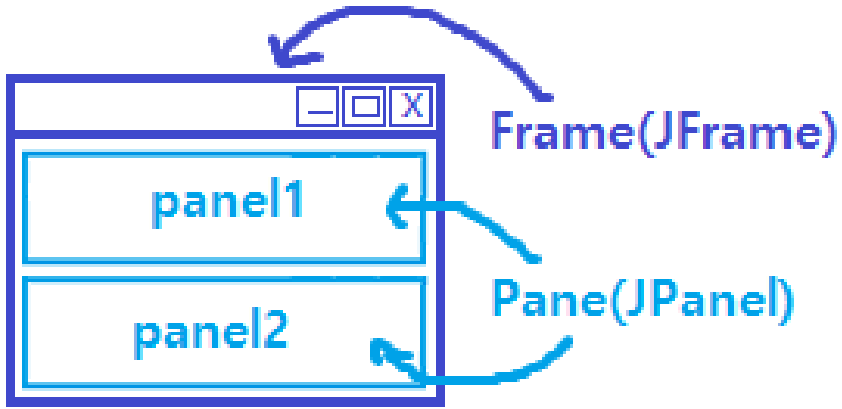
Null : 레이아웃 없음



JPanel 복습

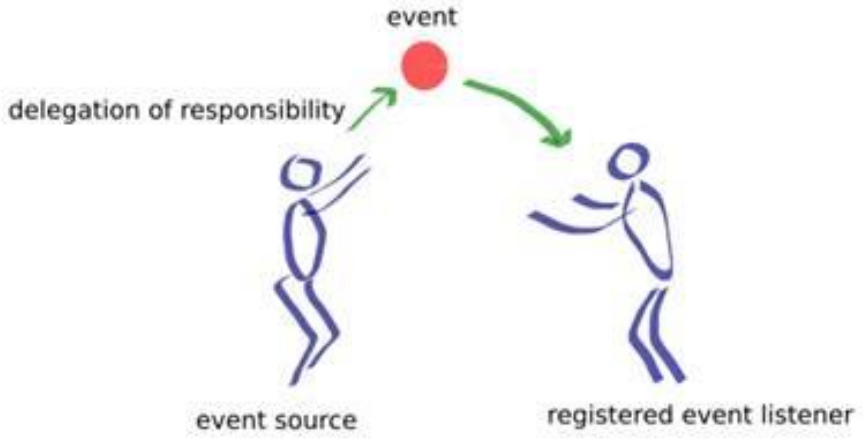
❖ **필요한 컴포넌트들을 그룹별로 묶을 때 사용하는 컨테이너 : JPanel**

- 화면에 보여지기 위해서는 다른 패널 또는 프레임 등 컨테이너에 추가돼야 함



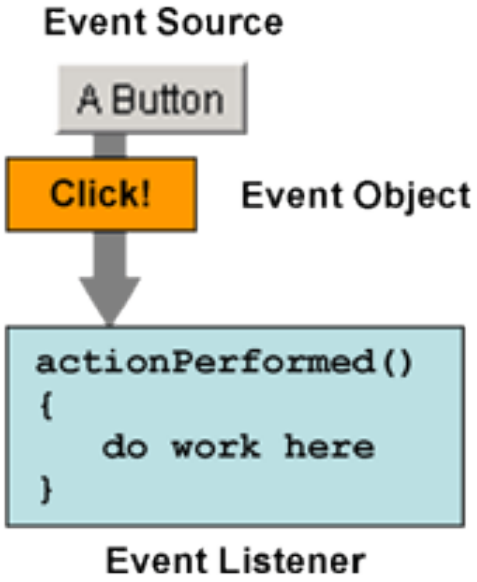
잠시만...

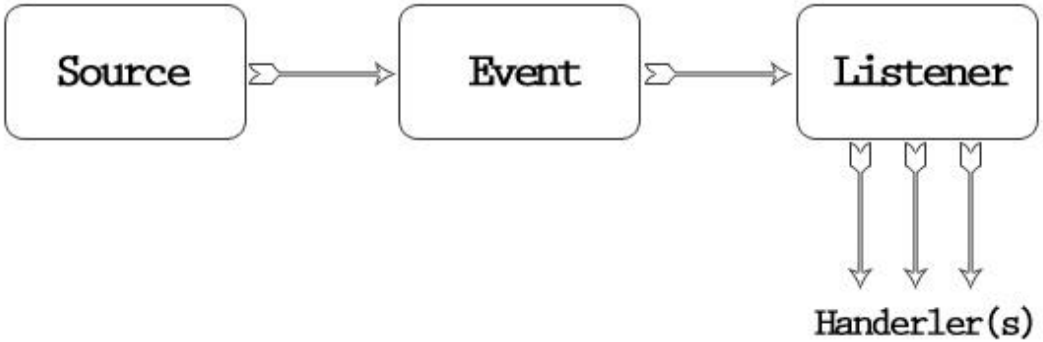
JAVA : Event 처리



Delegation 위임 모델

Event-driven Programming : Delegation model

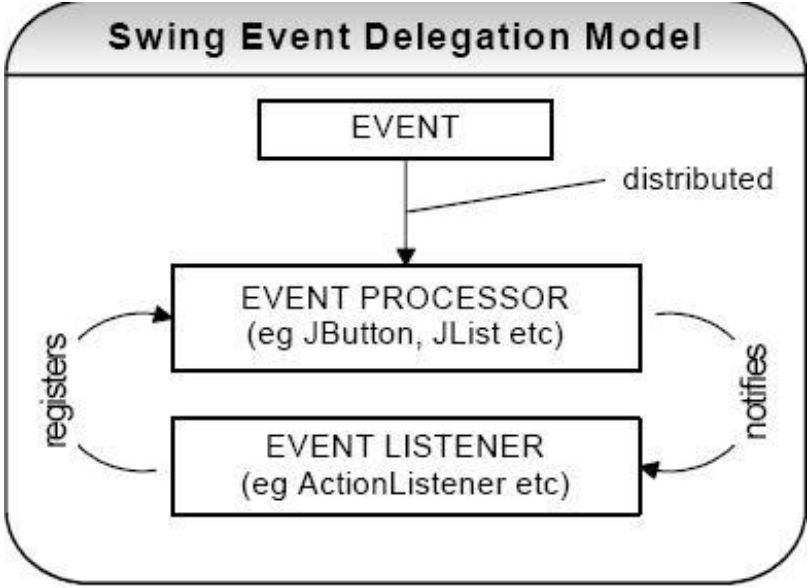




Event-Driving Programming Model

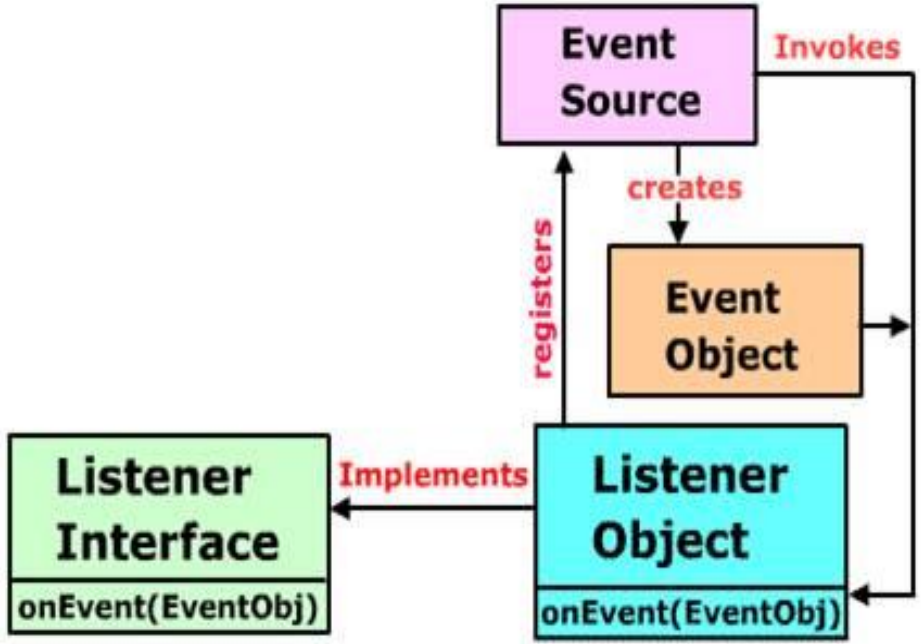
Event-driven Programming : Delegation model

- 이벤트가 발생한 **컴포넌트(이벤트 소스 객체)**에 이벤트 처리를 위한 **리스너 객체**를 등록해 이벤트를 처리하는 방법.
- **리스너 객체**는 **이벤트 처리기(핸들러)**를 가지고 있는 객체를 말함.
- 위임 모델은 이벤트 처리를 위한 코드가 어플리케이션 코드와 분리됨으로써 **재사용성이 높아지는 장점**을 가짐.

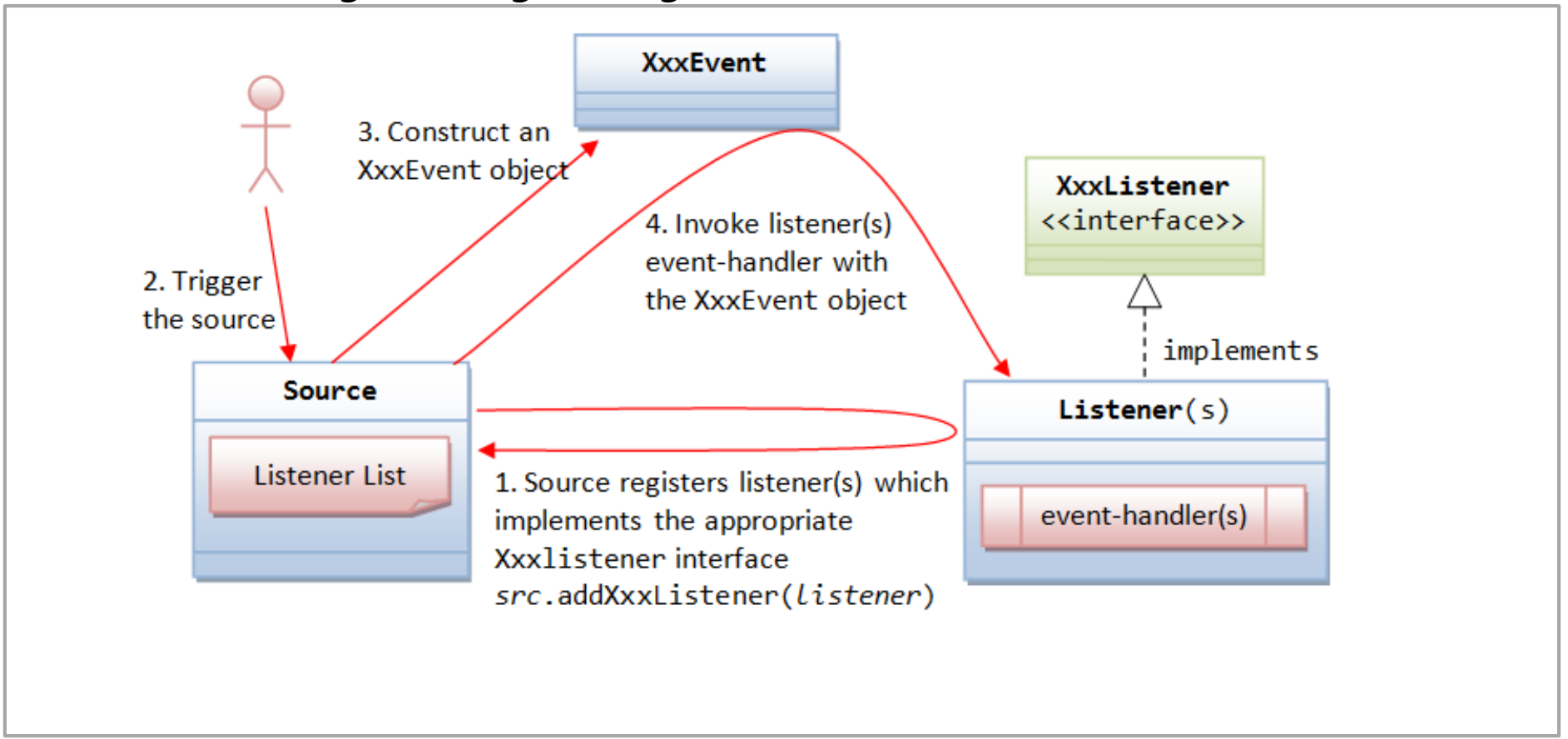


Event-driven Programming : Delegation model

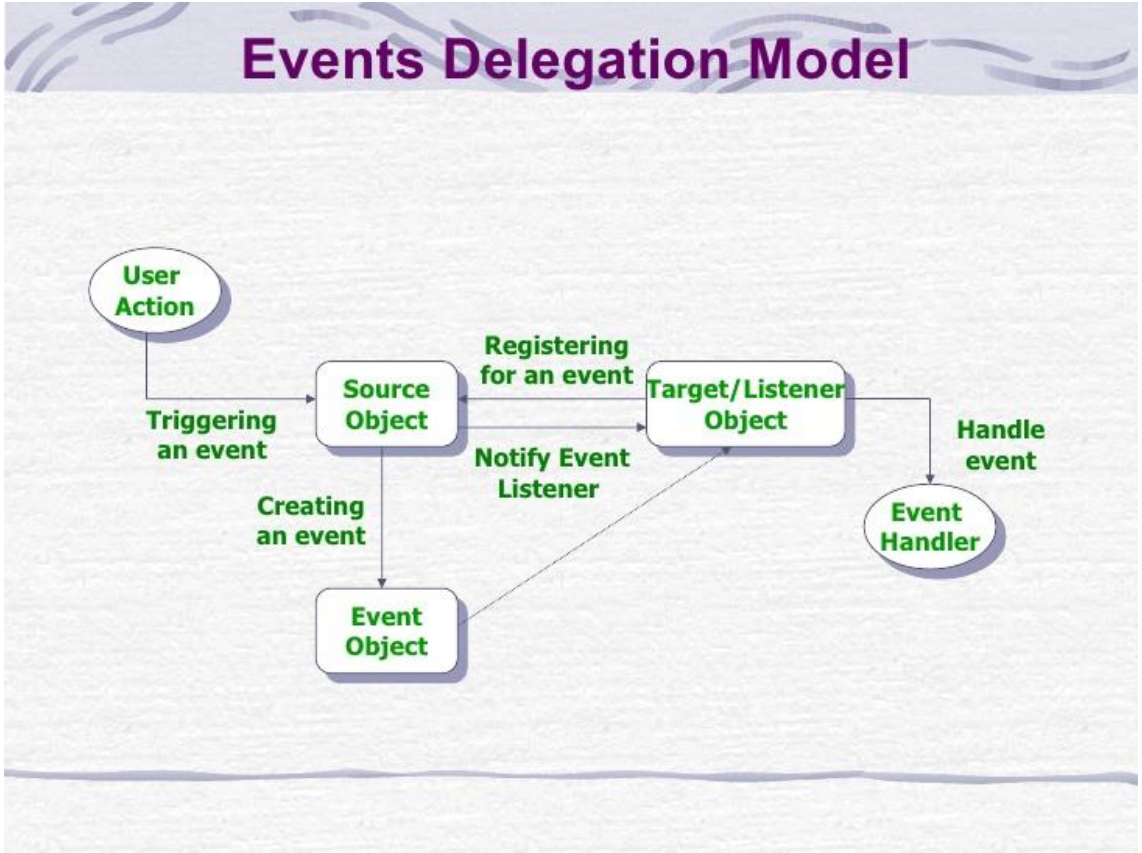
- **사용자의 행동**에 따라 컴포넌트에서 발생할 수 있는 **이벤트 종류**가 결정되어 있음 →교과서 202쪽 표 참조
- 각 **이벤트**마다 **리스너 인터페이스**가 지정되어있음 →교과서 203쪽 표 참조
- **리스너 인터페이스**를 **구현**하여 이벤트 소스 객체에 **등록**하는 방법으로 ***이벤트 프로그래밍***을 함



Event-driven Programming : Delegation model

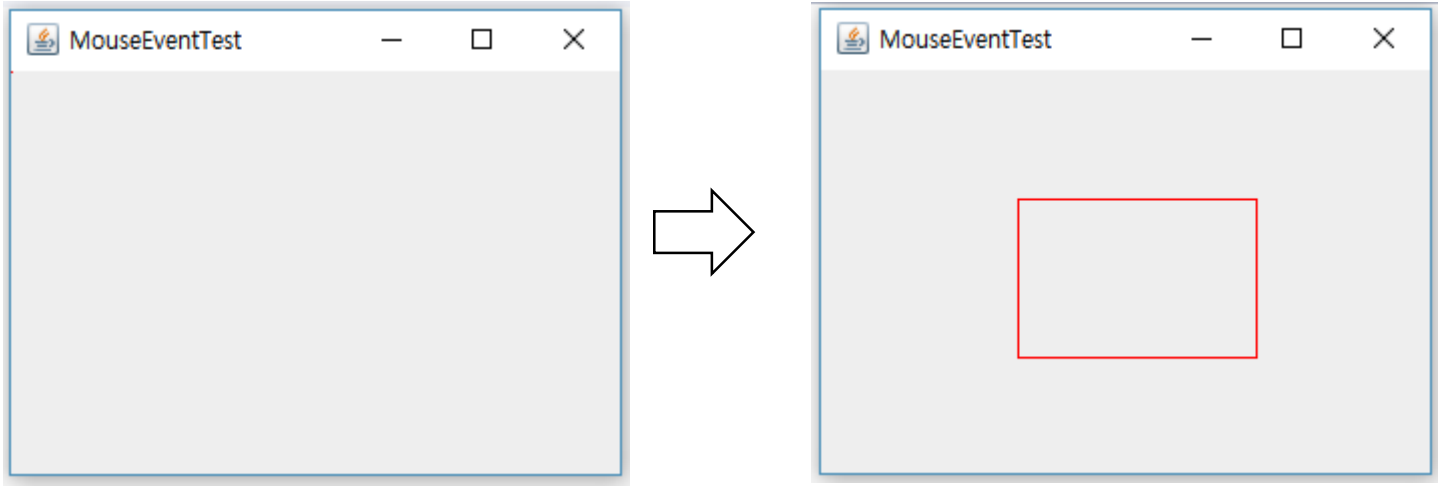


Event-driven Programming : Delegation model



무작정 따라하기 : 어뎁터 클래스가 있어요... → 교과서 203쪽

어뎁터 클래스 맛보기



MouseEventTest 클래스

```
1  import javax.swing.JFrame;
2  public class MouseEventTest {
3      public static void main(String[] args) {
4          JFrame f = new JFrame("MouseEventTest");
5          f.setSize(320, 240);
6          f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
7          f.add(new WhiteBorad());
8          f.setVisible(true);
9      }
10 }
```

WhiteBorad 클래스 (1/2) : 다음 페이지에 코드가 이어짐

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4  public class WhiteBorad extends JPanel {
5      int x, y, w, h;
6      public WhiteBorad() {
7          x=y=w=h=0;
8          addMouseListener(new MouseEventHdl());
9          addMouseMotionListener(new MouseMotionHdl());
10     }
11     public void setStartPoint(int x,int y) {
12         this.x=x;      this.y=y;
13     }
14     public void setEndPoint(int x,int y) {
15         w= Math.abs(this.x-x);   h = Math.abs(this.y-y);
16     }
17     @Override
18     protected void paintComponent(Graphics g) {
19         super.paintComponent(g);
20         g.setColor(Color.RED);
21         g.drawRect(x, y, w, h);
22     }
```

WhiteBorad 클래스 (2/2) : 이전 페이지에서 계속된 코드임

```
23      //아래는 어뎁터 클래스(리스너 클래스의 일종)를 상속받은 내부클래스
24      class MouseEventHdl extends MouseAdapter {
25          public void mousePressed(MouseEvent e) {
26              setStartPoint(e.getX(),e.getY());
27          }
28          public void mouseReleased(MouseEvent e) {
29              setEndPoint(e.getX(),e.getY());
30              repaint();
31          }
32      }
33      class MouseMotionHdl extends MouseMotionAdapter {
34          public void mouseDragged(MouseEvent e) {
35              setEndPoint(e.getX(),e.getY());
36              repaint();
37          }
38      }
39  }
```

다음 시간은...

GUI 계속 …

화 이 팅 !