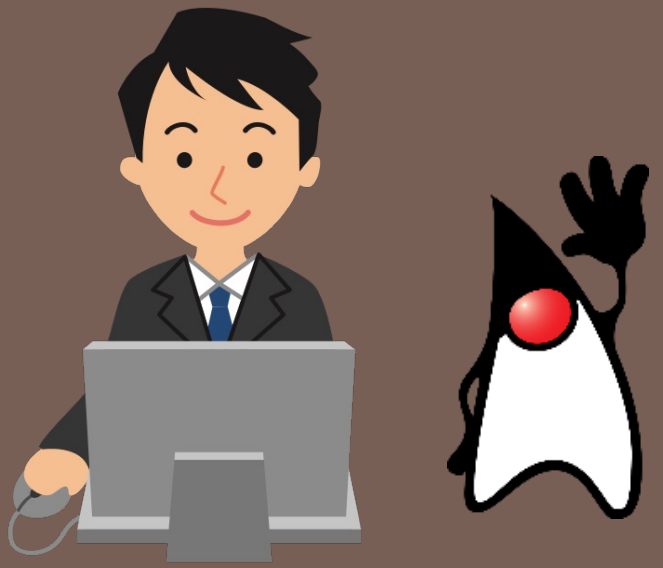
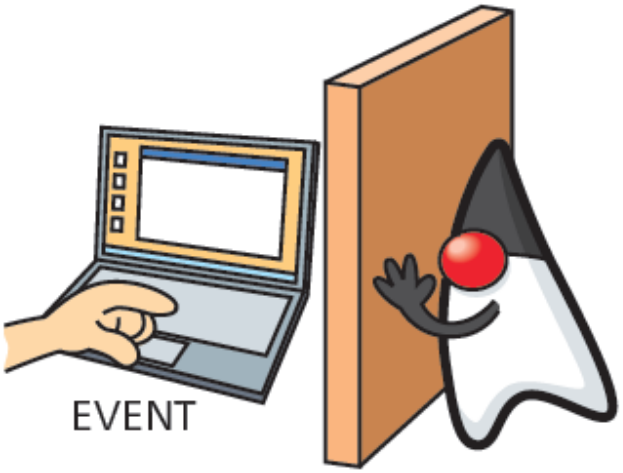
파워자바(개정3판)



10장 이벤트 처리

10장의 목표

1. 버튼에서 발생하는 이벤트를 처리하여서 어떤 작업을 할 수 있나요?
2. 이벤트 처리기를 익명 클래스로 작성할 수 있나요?
3. 이벤트 처리기를 람다식으로 작성할 수 있나요?
4. 키보드 이벤트를 받아서 어떤 작업을 할 수 있나요?
5. 마우스 이벤트를 받아서 어떤 작업을 할 수 있나요?



이벤트 구동 프로그래밍

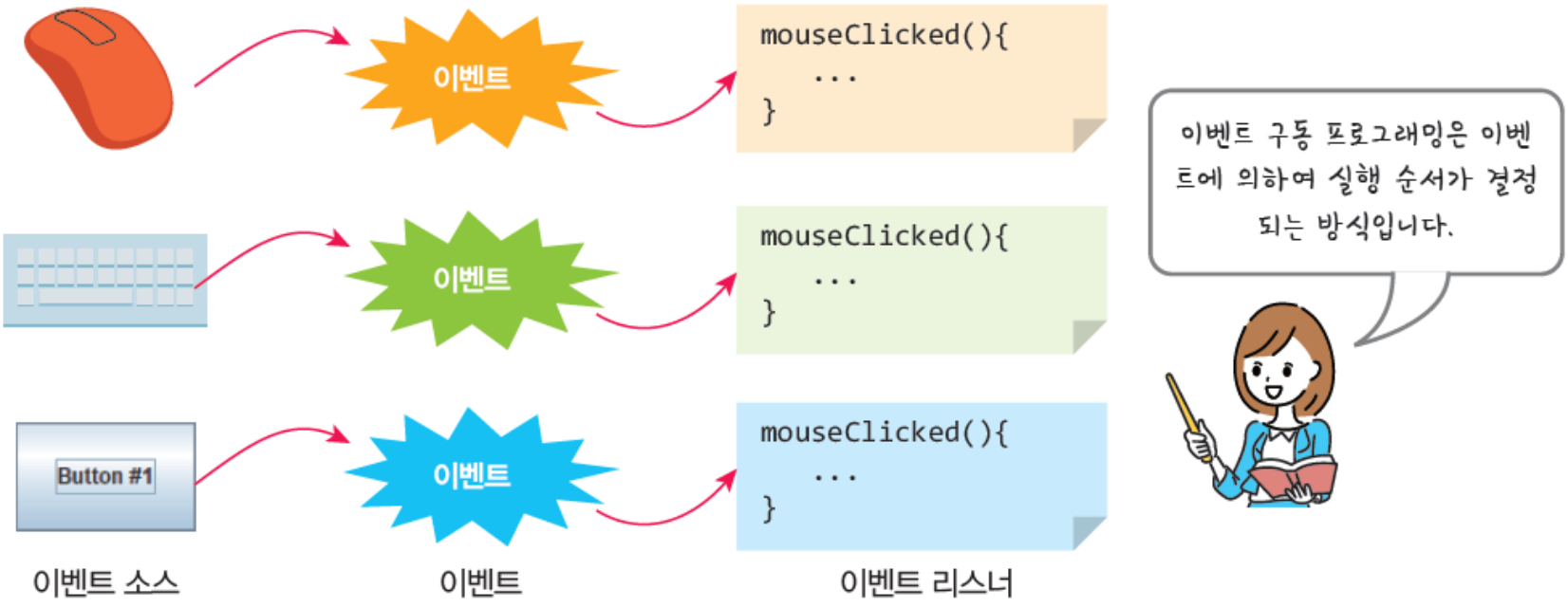
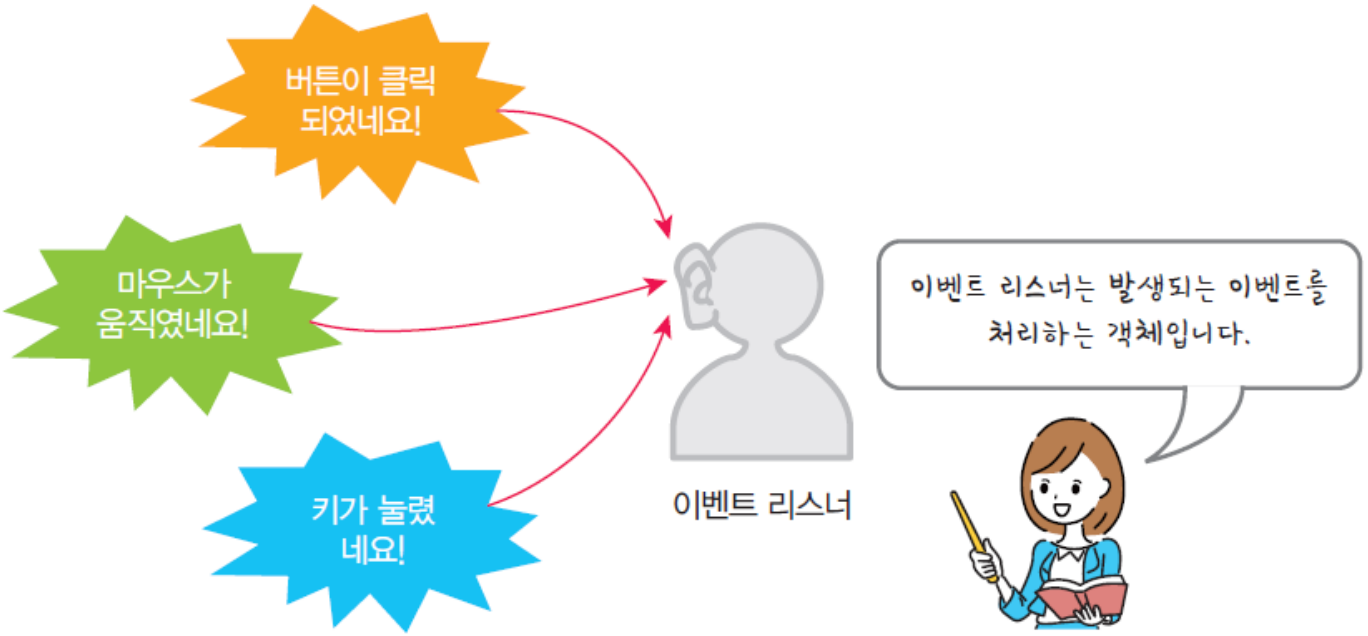


그림 10.1 이벤트 구동 프로그래밍

이벤트와 이벤트 리스너

- 이벤트(event)는 사용자가 버튼을 클릭한다거나 마우스를 움직이거나 키를 누르면 발생한다. 발생된 이벤트 객체에 반응하여서 이벤트를 처리하는 객체를 이벤트 리스너(event listener)라고 한다.



이벤트 처리 과정

❶ 이벤트 리스너를 작성한다. - 어떤 클래스가 이벤트 리스너가 되기 위해서는 리스너 인터페이스를 구현하여야 한다.

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

액션 이벤트가 발생하면 호출된다.

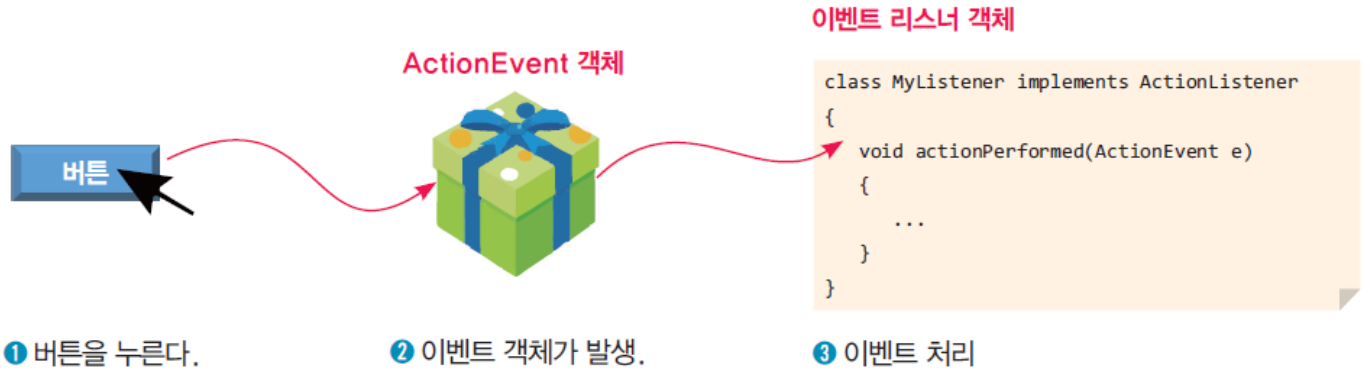


그림 10.2 리스너 객체의 역할

이벤트 처리 과정

❷ 이벤트 리스너를 컴포넌트에 등록한다. - 이벤트 리스너를 컴포넌트에 등록하는 단계이다.

```
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 객체

- 이벤트 객체는 발생된 이벤트에 대한 모든 정보를 리스너로 전달한다. 이벤트 객체는 getSource() 액션 이벤트가 발생하면 호출된다.

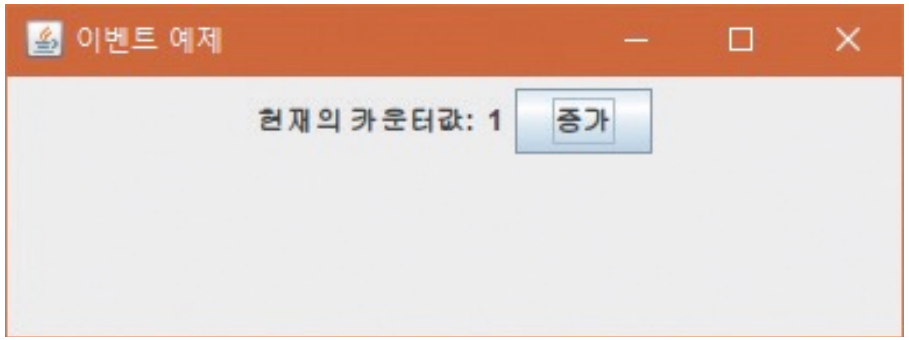
```
public void actionPerformed(ActionEvent e) {
   button = (JButton)e.getSource();
      ...
}
```

중간점검

1. 이벤트가 발생하면 적절한 처리를 해주는 프로그래밍 방식을 무엇이라고 하는가?
2. 이벤트를 처리하는 객체를 무엇이라고 하는가?
3. 이벤트를 처리하는 절차를 요약해보자.

이벤트 처리 방법

- 카운터 프로그램을 예로 들어서 설명해보자. 아래 프로그램에서 “증가” 버튼을 누르면 카운터 값이 하나씩 증가되어서 표시된다. 우리는 버튼에서 발생하는 액션 이벤트를 처리하여야 한다.



이벤트 처리기를 어디에...

이벤트 처리 방법

- (1) 독립적인 클래스로 이벤트 처리기를 작성
- (2) 내부 클래스로 이벤트 처리기를 작성
- (3) 프레임 클래스에 이벤트 처리를 구현
- (4) 무명 클래스를 사용하는 방법
- (5) 람다식을 이용하는 방법

내부 클래스가 이벤트를 처리하는 방법

① java.awt.event 패키지에 이벤트를 처리하는 클래스들이 모여 있음

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

public class EventTest1 extends JFrame {
        private JButton button;
        private JLabel label;
        int counter=0;

        class MyListener implements ActionListener {
                public void actionPerformed(ActionEvent e) {
                        counter++;
                        label.setText("현재의 카운터값:  "+counter);
                }
        }
```

② 내부 클래스로 이벤트를 처리하는 클래스 정의

내부 클래스가 이벤트를 처리하는 방법

```
        public EventTest1() {
                setSize(400, 150);
                setTitle("이벤트 예제");
                setLayout(new FlowLayout());
                button = new JButton("증가");
                label = new JLabel("현재의 카운터값: "+counter);


                button.addActionListener(new MyListener());
                add(label, "Center");
                add(button, "East");
                setVisible(true);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
        public static void main(String[] args) {
                EventTest1 t = new EventTest1();
        }
}
```

③ 버튼에 이벤트 처리 객체를 등록시킨다.

외부 클래스가 이벤트를 처리하는 방법

- 만약 다음과 같이 MyListener를 외부 클래스로 구현했다고 하자. MyListener 외부 클래스에서는 MyFrame 안에 정의된 label 변수에는 접근하기 힘들다

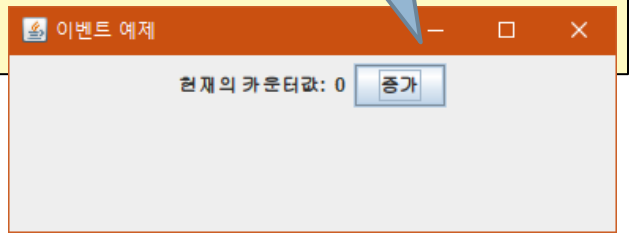
```
class MyListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                JButton button = (JButton) e.getSource();

                // counter++;
                // label.setText("현재의 카운터값: "+counter);
        }


}

public class EventTest2 extends JFrame {
        private JButton button;
        private JLabel label;
        int counter=0;
```

counter와 label은 MyFrame 클래스 안에 있어서 접근하기 어렵다. 물론 완전히 못하는 것은 아니다

외부 클래스가 이벤트를 처리하는 방법

```
        public  EventTest2() {
                setSize(400, 150);
                setTitle("이벤트 예제");
                setLayout(new FlowLayout());
                button = new JButton("증가");
                label = new JLabel("현재의 카운터값:  "+counter);

                button.addActionListener(new MyListener());
                add(label, "Center");
                add(button, "East");
                setVisible(true);

                setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
        public static void main(String[] args) {
                EventTest2 t = new EventTest2();
        }
}
```

버튼을 눌러도 카운터가 증가되지 않는다.



프레임 클래스가 이벤트를 처리하는 방법

- 이 방법은 프레임 클래스가 JFrame을 상속받으면서 동시에 ActionListener 인터페이스도 구현하는 방법이다.

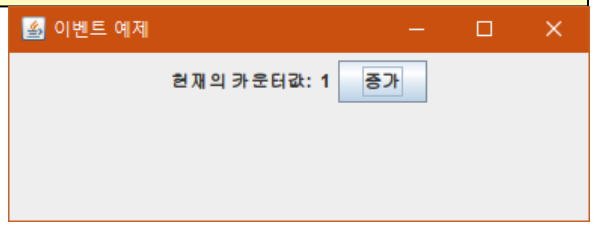
```
public class EventTest3 extends JFrame implements ActionListener {
        private JButton button;
        private JLabel label;
        int counter=0;

        public void actionPerformed(ActionEvent e) {
                counter++;
                label.setText("현재의 카운터값:  "+counter);
        }
        public EventTest3() {
                this.setSize(400, 150);
                this.setTitle("이벤트 예제");
```

MyFrame 클래스는 JFrame 클래스를 상속받고 동시에 ActionListener를 구현한다. 따라서 프레임이 버튼에서 발생하는 이벤트도 처리할 수 있다.

프레임 클래스가 이벤트를 처리하는 방법

```
                JPanel panel = new JPanel();
                button = new JButton("증가");
                label = new JLabel("현재의 카운터값:  "+counter);


                button.addActionListener(this);
                panel.add(label);
                panel.add(button);
                add(panel);
                setVisible(true);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
        public static void main(String[] args) {
                EventTest3 t = new EventTest3();
        }
}
```

현재 객체를 이벤트 리소너로 버튼에 등록한다. 즉 자기 자신이 이벤ㅁ트를 처리한다고 등록한다.



무명 클래스를 사용하는 방법

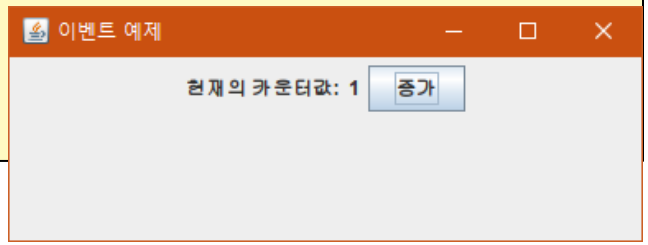
- 무명 클래스는 말 그대로, 이름이 없는 클래스를 작성하여 한 번만 사용하는 것이다. 이것은 처음에는 상당히 이상해 보이지만, 실제로는 익숙해지면 코드를 읽기 쉽게 만든다. 왜냐하면 클래스가 정의되면서 바로 사용되기 때문이다. 안드로이드 프로그래밍에서도 자주 사용된다

```
public class EventTest4 extends JFrame {
        private JButton button;
        private JLabel label;

        int counter=0;

        public EventTest4() {
                this.setSize(400, 150);
                this.setTitle("이벤트 예제");

                JPanel panel = new JPanel();
                button = new JButton("증가");
                label = new JLabel("현재의 카운터값:  "+counter);
```

익명 클래스를 사용하는 방법

```
                button.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                counter++;
                                label.setText("현재의 카운터값:  "+counter);
                        }
                });


                panel.add(label);
                panel.add(button);
                add(panel);
                setVisible(true);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
        public static void main(String[] args) {
                EventTest4 f = new EventTest4();
        }
}
```

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



일반 클래스와 익명 클래스의 비교

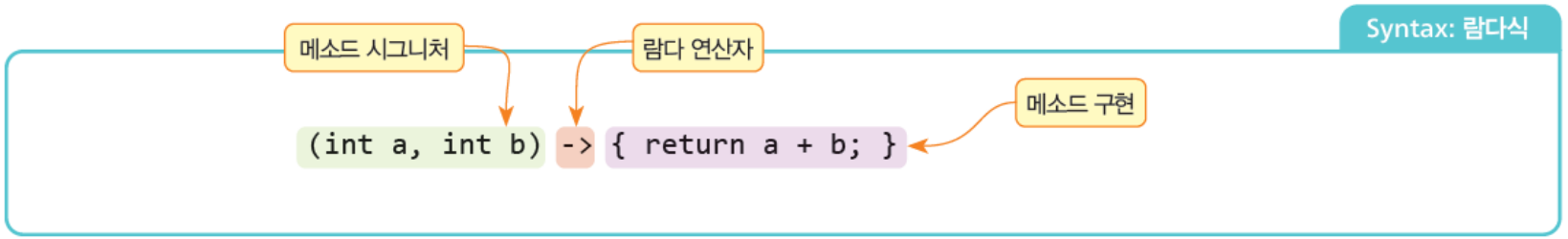
무명 클래스	일반 클래스

ActionListener obj = **new** ActionListener() { ...
 };
	class MyListener **implements** ActionListener {
	 ...
	 }
	 MyListener obj = **new** MyListener();

교재 오타!

람다식을 이용하는 방법

- 람다식(lambda expression)은 이름이 없는 메소드(함수)라고 할 수 있다. 우리가 람다식을 사용하는 이유는 간결함 때문이다



람다식을 이용하는 방법

```
public class EventTest5 extends JFrame {
        private JButton button;
        private JLabel label;
        int counter=0;

        public EventTest5() {
                this.setSize(400, 150);
                this.setTitle("이벤트 예제");
                JPanel panel = new JPanel();
                button = new JButton("증가");
                label = new JLabel("현재의 카운터값: "+counter);

                button.addActionListener(e -> {
                        counter++;
                        label.setText("현재의 카운터값: "+counter);
                });
                panel.add(label);
                panel.add(button);
                add(panel);
                setVisible(true);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
        public static void main(String[] args) {
                EventTest5 t = new EventTest5();
        }
}
```

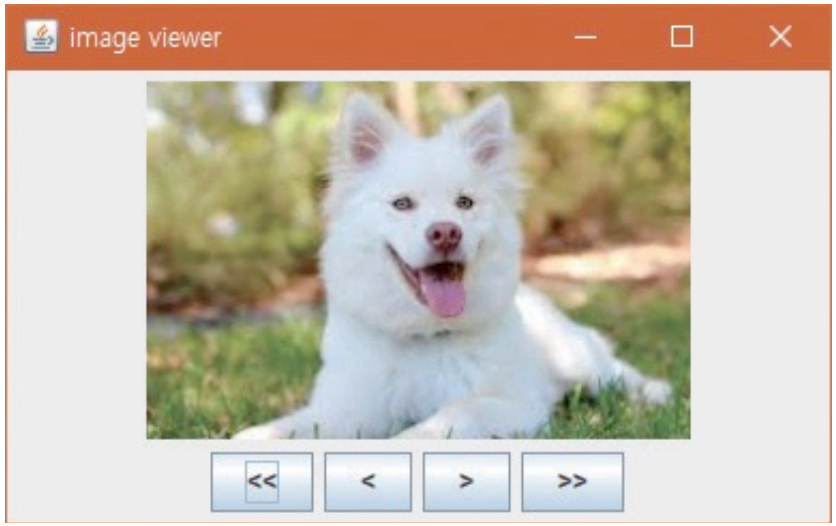
람다식을 이용하여 이벤트를 처리하고 있다. 변수 e는 이벤트를 나타낸다. 람다식은 함수를 객체로 만들어서 메소드에 전달할 수 있다.

중간점검

1. 왜 리스너를 외부 클래스로 작성하지 않는가?
2. 프레임이 액션 이벤트를 처리하려면 어떤 인터페이스를 구현하여야 하는가?
3. 무명 클래스를 설명해보자.
4. 람다식을 설명해보자.

Mini Project: 사진 뷰어 만들기

- 우리는 이번 장에서 각 컴포넌트들을 화면에 배치하는 방법과 레이블에 이미지를 표시하는 방법을 학습하였다. 이것을 이용하여서 다음과 같은 이미지 뷰어를 만들어보자. 물론 아직은 이벤트 처리가 되지 않지만, 다음 장을 미리 예습한 독자라면 이벤트 처리 기능도 추가할 수 있을 것이다



스윙 컴포넌트의 이벤트

- 스윙 컴포넌트가 발생하는 이벤트는 모든 컴포넌트가 공통적으로 지원하는 저수준 이벤트와 일부 컴포넌트만 지원하는 의미적 이벤트로 나눌 수 있다.

저수준 이벤트:
Mouse, MouseMotion, Key, Component, Container, Focus, Window

의미적 이벤트:
Action, Adjustment, Document, Item, Text

저수준 이벤트

- 저수준 이벤트(low-level event)는 모든 컴포넌트에서 발생된다. 예를 들어서 마우스나 키보드로부터 발생되는 이벤트는 저수준 이벤트이다

이벤트 종류	설명
Component	컴포넌트의 크기나 위치가 변경되었을 경우 발생
Focus	키보드 입력을 받을 수 있는 상태가 되었을 때, 혹은 그 반대의 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 변화(열림, 닫힘, 아이콘화 등)가 있을 때 발생

의미적 이벤트

- 의미적 이벤트(semantic event)는 일부 컴포넌트에서만 발생한다. 대표적인 의미적 이벤트는 액션 이벤트이다

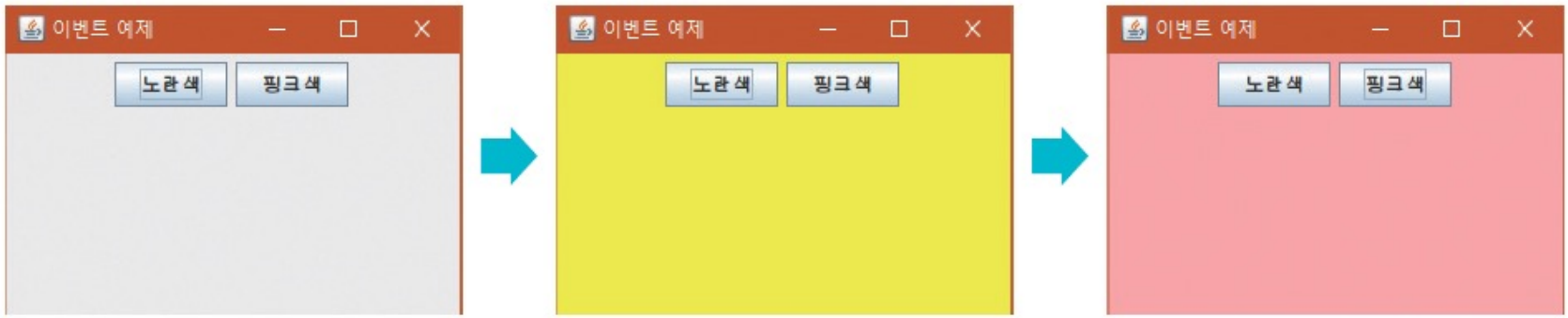
이벤트 종류	설명
Action	사용자가 어떤 동작을 하는 경우에 발생
Caret	텍스트 삽입점이 이동하거나 텍스트 선택이 변경되었을 경우 발생
Change	일반적으로 객체의 상태가 변경되었을 경우 발생
Document	문서의 상태가 변경되는 경우 발생
Item	선택 가능한 컴포넌트에서 사용자가 선택을 하였을 때 발생
ListSelection	리스트나 테이블에서 선택 부분이 변경되었을 경우에 발생

액션 이벤트

- 다음과 같은 경우에 액션 이벤트가 발생한다.
  - 사용자가 버튼을 클릭하는 경우
  - 사용자가 메뉴 항목을 선택하는 경우
  - 사용자가 텍스트 필드에서 엔터키를 누르는 경우

예제: 액션 이벤트

- 두 개의 버튼을 만들어서 패널의 배경 색을 변경하는 프로그램을 작성하여 보자.
- 이벤트 리스너는 하나만 생성한다.



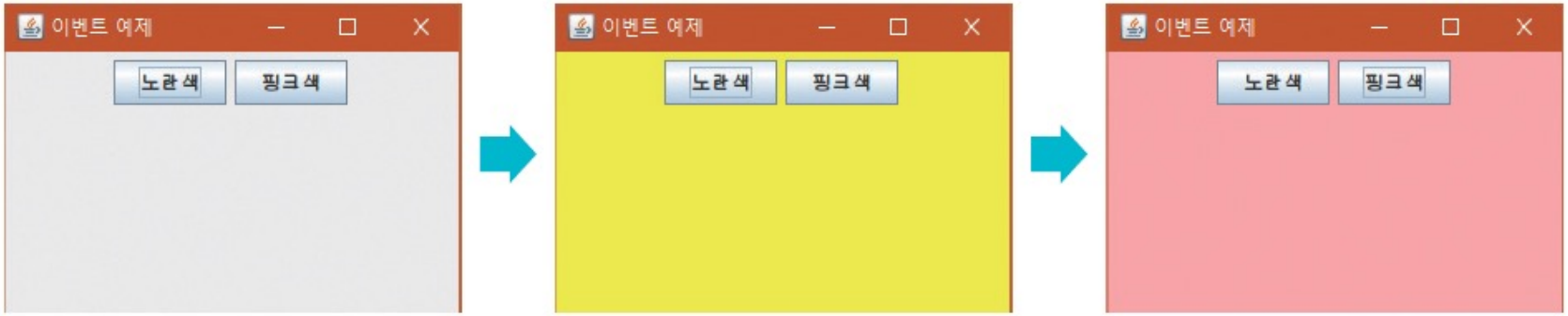
예제: 액션 이벤트

```
public class ChangeBackground extends JFrame {

    private JButton button1;
    private JButton button2;
    private JPanel panel;
    MyListener listener = new MyListener();

    public ChangeBackground() {
        this.setSize(300, 200);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("이벤트 예제");
        panel = new JPanel();

        button1 = new JButton("노란색");
        button1.addActionListener(listener);
        panel.add(button1);
        button2 = new JButton("핑크색");
        button2.addActionListener(listener);

        panel.add(button2);
        this.add(panel);
        this.setVisible(true);
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
```

예제: 액션 이벤트

```
}
    private class MyListener implements ActionListener {
        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) {
        ChangeBackground t = new ChangeBackground();
    }
}
```

실행결과

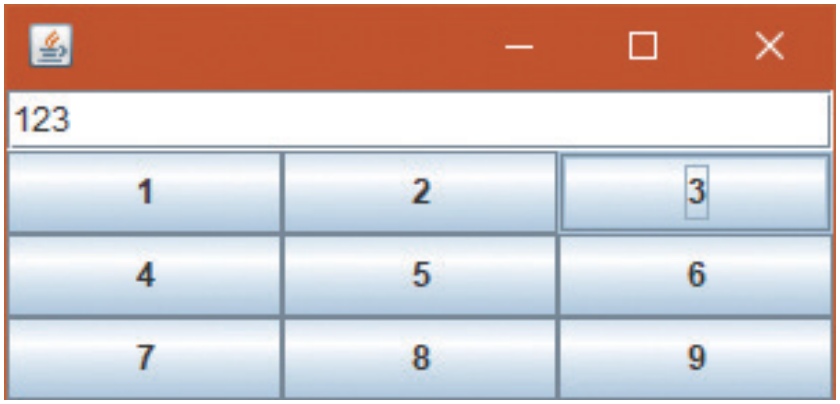


중간점검

1. 버튼을 누르면 발생하는 이벤트는 무엇인가?
2. 버튼에서 발생하는 이벤트를 처리하려면 어떤 인터페이스를 구현해야 하는가?
3. 화면에 버튼이 여러 개 있을 때, 리스너에서는 어떻게 버튼을 구별하면 되는가?

Lab: 키패드 만들기

- 숫자를 입력할 수 있는 키패드 프로그램을 작성하여 보자.



Sol: 키패드 만들기

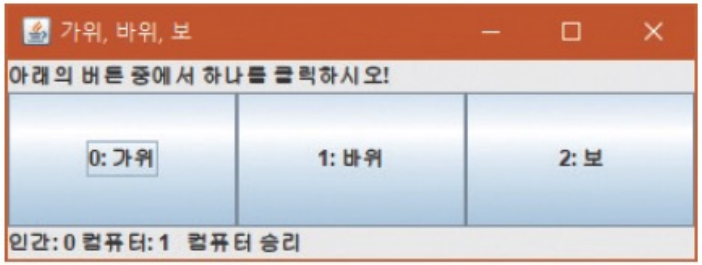
```
public class KeyPad extends JFrame implements ActionListener {
        private JTextField txt;
        private JPanel panel;

        public KeyPad() {
                txt = new JTextField(20);
                add(txt, BorderLayout.NORTH);
                panel = new JPanel();
                panel.setLayout(new GridLayout(3, 3));
                add(panel, BorderLayout.CENTER);
                for (int i = 1; i <= 9; i++) {
                        JButton btn = new JButton("" + i);
                        btn.addActionListener(this);
                        btn.setPreferredSize(new Dimension(100, 30));
                        panel.add(btn);
                }
                pack();
                setVisible(true);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
```

Sol: 키패드 만들기

```
        @Override
        public void actionPerformed(ActionEvent e) {
                String actionCommand = e.getActionCommand();
                txt.setText(txt.getText() + actionCommand);
        }
        public static void main(String[] args) {
                KeyPad f = new KeyPad();
        }
}
```

Lab: 가위 바위 보 게임

- 가위, 바위, 보 게임을 작성하여 보자. 먼저 가위, 바위, 보를 나타내는 버튼을 생성한다. 사용자가 버튼 중에서 하나를 클릭하면 이것을 컴퓨터가 내부에서 생성한 난수값과 비교한다. 그리고 누가 이겼는지를 화면에 출력한다.



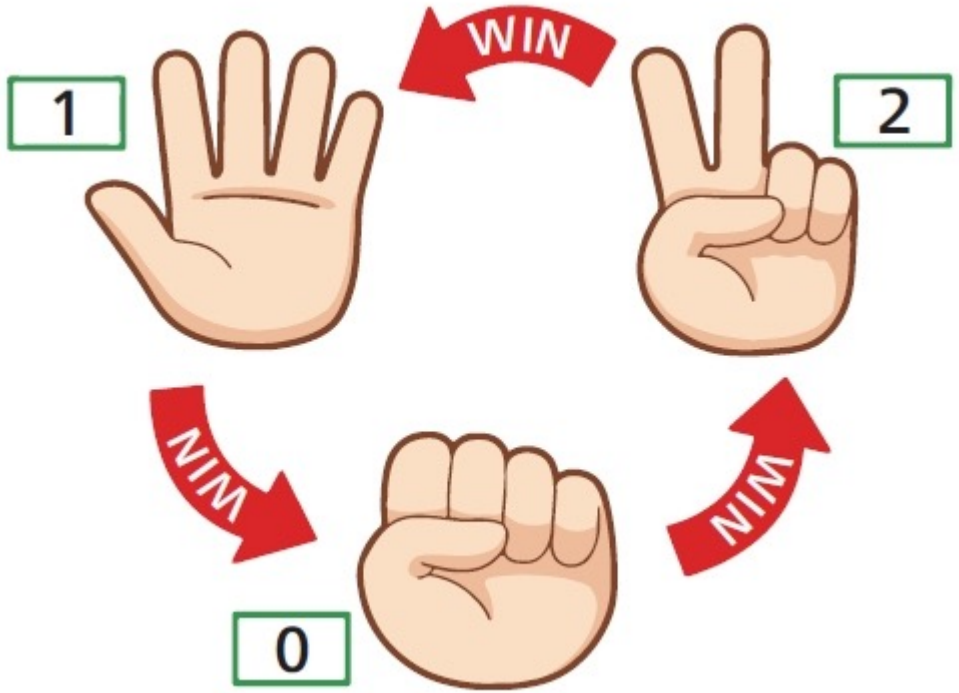
Sol: 가위 바위 보 게임

```
public class RockPaperScissor extends JFrame implements ActionListener {
        final int SCISSOR = 0;
        final int ROCK = 1;
        final int PAPER = 2;

        private JPanel panel;
        private JLabel output, information;
        private JButton rock, paper, scissor;

        public RockPaperScissor() {
                setTitle("가위, 바위, 보");
                setSize(400, 150);

                panel = new JPanel();
                panel.setLayout(new GridLayout(0, 3));      // 그리드 배치 관리자 선택

                information = new JLabel("아래의 버튼 중에서 하나를 클릭하시오!");
                output = new JLabel("Good Luck!");

                rock = new JButton("0: 가위");
                paper = new JButton("1: 바위");
                scissor = new JButton("2: 보");
                rock.addActionListener(this);
                paper.addActionListener(this);
                scissor.addActionListener(this);
```

Sol: 가위 바위 보 게임

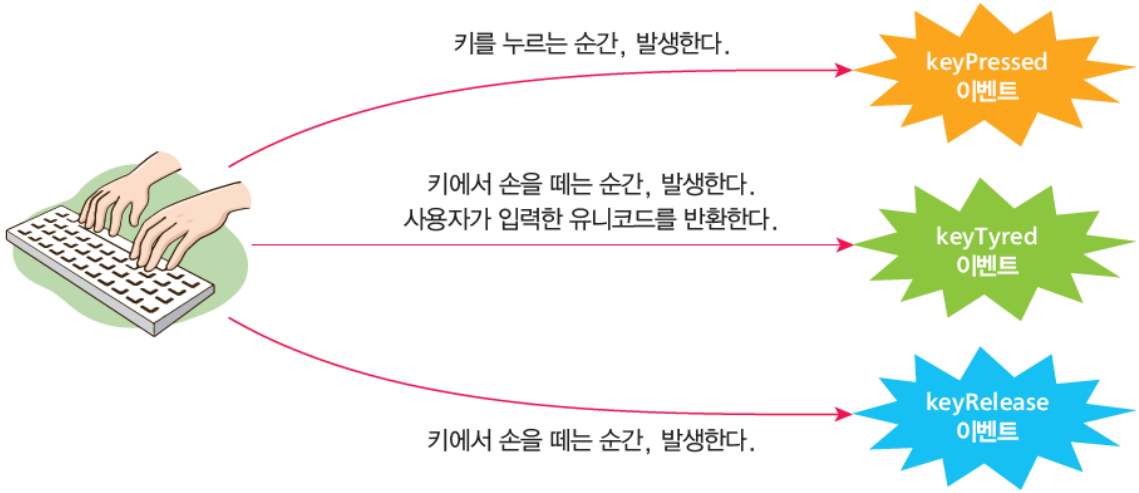
```
                panel.add(rock);
                panel.add(paper);
                panel.add(scissor);

                add(information, BorderLayout.NORTH);
                add(panel, BorderLayout.CENTER);
                add(output, BorderLayout.SOUTH);
                setVisible(true);
        }
        public static void main(String[] args) {
                RockPaperScissor f = new RockPaperScissor();
        }
        @Override
        public void actionPerformed(ActionEvent e) {
                JButton b = (JButton)e.getSource();             // 이벤트 발생 컴포넌트 추출
                int user = Integer.parseInt(""+b.getText().charAt(0)); // 첫 번째 글자 추출
                Random random = new Random();
                int computer = random.nextInt(3);               // 0부터 2까지의 난수 발생
                if( user == computer)
                  output.setText("인간과 컴퓨터가 비겼음");
                else if( user == (computer+1)%3)                // 0은 1한테 진다.
                  output.setText("인간: "+user+" 컴퓨터: "+computer+"  인간 승리");
                else
                  output.setText("인간: "+user+" 컴퓨터: "+computer+"  컴퓨터 승리");
        }
}
```

Sol: 가위 바위 보 게임



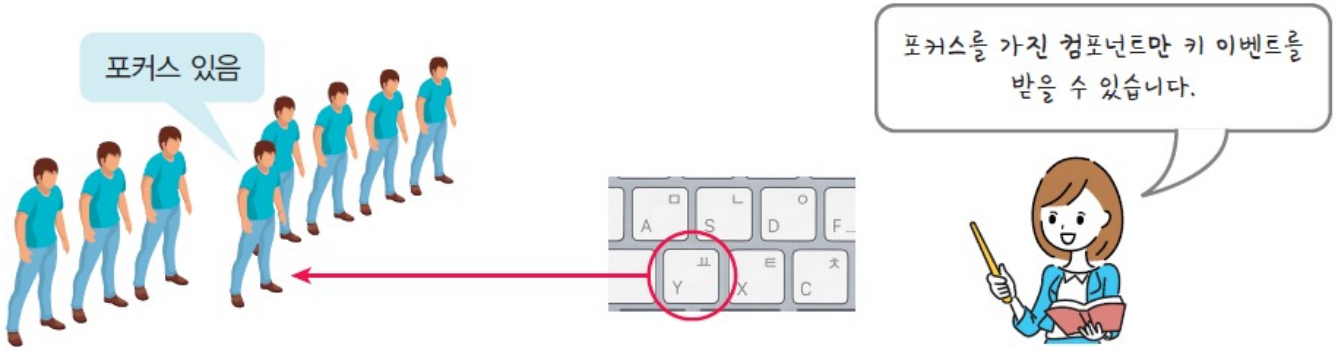
키이벤트

- 키 이벤트(key event)는 사용자가 키보드를 이용하여 입력을 하는 경우에 발생한다.
  - keyPressed 이벤트: 사용자가 키를 누르면 이벤트가 발생한다.
  - keyReleased 이벤트: 사용자가 키에서 손을 떼면 이벤트가 발생한다.
  - keyTyped 이벤트: 입력된 유니코드 문자가 전송된다.



포커스

- 컴포넌트가 키 이벤트를 받으려면 반드시 포커스(focus)를 가지고 있어야 한다. 포커스란 키 입력을 받을 권리이다.
- 일반적으로 오직 한 개의 컴포넌트만 포커스를 가지고 있어서 키 입력을 독점ㅍ하게 된다.




```
panel.setFocusable(true);
panel.requestFocus();
```

KeyListener 인터페이스

- 어떤 클래스가 키보드 이벤트를 처리하려면 KeyListener 인터페이스를 구현하여야 한다.

```
public class MyListener  implements KeyListener {

   public void keyPressed(KeyEvent e) {          }
   public void keyReleased(KeyEvent e) {         }
   public void keyTyped(KeyEvent e) {            }
}
```

사용자가 키를 눌렀을 경우에 호출

사용자가 키에서 손을 떼었을 경우에 호출

사용자가 글자를 입력했을 경우에 호출

키를 누르는 순간, 발생한다.

키에서 손을 떼는 순간, 발생한다.
사용자가 입력한 유니코드를 반환한다.

키에서 손을 떼는 순간, 발생한다.

```
public void keyPressed(KeyEvent e) {
   ...
}
public void keyTyped(KeyEvent e) {
   ...
}
public void keyReleased(KeyEvent e) {
...
}
```

KeyEvent 클래스

메소드	설명
int getKeyChar()	KeyEvent에 들어있는 글자(유니코드)를 반환한다.
int getKeyCode()	KeyEvent에 들어있는 키코드(keycode)를 반환한다. 키코드란 글자가 아니라 키보드 자판의 각각의 키를 가리키는 상수이다. 예를 들어 Escape 키의 키코드는 VK_ESCAPE로 정의되어 있다.
boolean isActionKey()	이벤트를 발생시킨 키가 액션 키이면 true를 반환한다. 액션 키란 Cut, Copy, Paste, Page Up, Caps Lock, 화살표와 function 키를 의미한다.

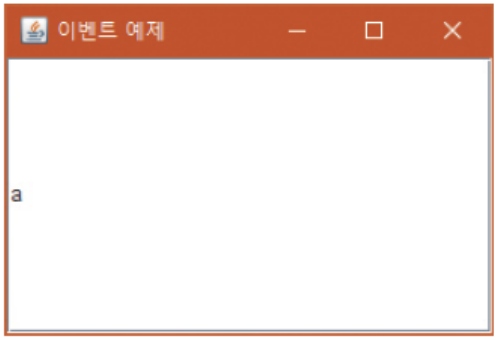
getKeyCode()

- 만약 글자가 아니고 키보드의 어떤 자판을 눌렀는지를 알고 싶으면 e.getKeyCode()를 호출한다.

상수	설명	상수	설명
VK_ENTER	Enter 키	VK_F1	F1 키
VK_END	End 키	VK_F2	F2 키
VK_CAPSLOCK	Capslock 키	...	
VK_ALT	Alt 키	VK_F12	F12 키
VK_LEFT	왼쪽 화살표	VK_PAGE_DOWN	Page Down 키
VK_RIGHT	오른쪽 화살표	VK_PAGE_UP	Page Up 키

예제: 키 이벤트 정보 출력하기

- 키 이벤트가 어떤 정보들을 반환하는지를 알기 위하여 텍스트 필드에 글자를 입력할 때 발생하는 키 이벤트 정보를 출력해본다. 프로그램을 실행한 후에 여러 가지 키를 눌러보자. 콘솔 창에 출력되는 문자열을 관찰한다.



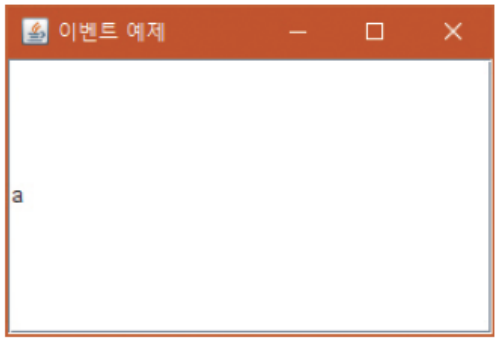

```
Key Pressed  a 65 false false false
KeyTyped  a 0 false false false
Key Released  a 65 false false false
```

예제: 키 이벤트 정보 출력하기

```
public class KeyEventTest extends JFrame implements KeyListener { // (1)
        public KeyEventTest() {
                setTitle("이벤트 예제");
                setSize(300, 200);
                JTextField tf = new JTextField(20);
                tf.addKeyListener(this); // (2)
                add(tf);
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }

        public void keyTyped(KeyEvent e) {
                display(e, "KeyTyped ");
        }

        public void keyPressed(KeyEvent e) {
                display(e, "KeyPressed ");
        }

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

예제: 키 이벤트 정보 출력하기

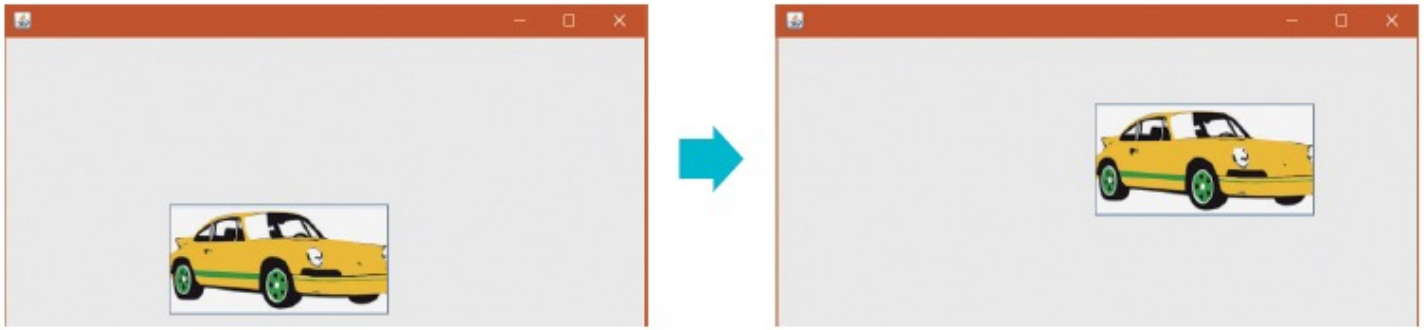
```
        protected void display(KeyEvent e, String s) {
                char c = e.getKeyChar();
                int keyCode = e.getKeyCode();
                String modifiers = e.isAltDown() + " " + e.isControlDown() + " " +
e.isShiftDown();
                System.out.println(s + " " + c + " " + keyCode + " " + modifiers);
        }

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




```
Key Pressed  a 65 false false false
KeyTyped  a 0 false false false
Key Released  a 65 false false false
```

예제: 자동차 움직이기

- 키보드의 화살표 키로 움직이는 자동차를 작성하여 보자. 프레임에 패널을 붙이고 패널에서 자동차 이미지를 그린다. 패널 안에서 키 이벤트를 처리하여 화살표 키가 입력되면 화면의 자동차를 움직인다. 여기서 주의할 점은 패널이 키 입력을 받으려면 반드시 setFocusable(true)를 호출하여야 한다.



예제: 자동차 움직이기

```
public class MoveCar extends JFrame {
        int img_x=150, img_y=150;
        JButton button;

        public MoveCar() {
                setSize(600, 300);
                button = new JButton("");
                ImageIcon icon = new ImageIcon("d:\\car.png");
                button.setIcon(icon);

                JPanel panel = new JPanel();
                panel.setLayout(null);
                button.setLocation(img_x, img_y);
                button.setSize(200, 100);

                panel.add(button);
                panel.requestFocus();

                panel.setFocusable(true);
```

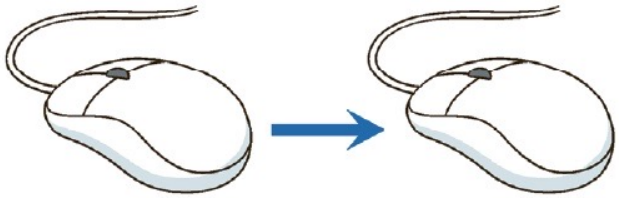
예제: 자동차 움직이기

```
            panel.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;
                    }
                    button.setLocation(img_x, img_y);
                }
                public void keyReleased(KeyEvent arg0) {       }
                public void keyTyped(KeyEvent arg0) {                  }
            });
            add(panel);
            setVisible(true);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
        public static void main(String[] args) {
            MoveCar f = new MoveCar();
        }
}
```

중간점검

1. 컴포넌트가 키 이벤트를 받으려면 어떤 사전 작업을 해야 하는가?
2. 컴포넌트가 포커스를 받으려면 어떤 메소드를 호출하여야 하는가?
3. 위쪽 화살표 키를 인식하려면 어떤 메소드를 사용하여야 하는가?
4. 'q' 키가 입력되면 전체 프로그램을 종료하고 싶다. 어떻게 하면 되는가?
5. 'a' 키를 입력하면 어떤 키 이벤트들이 발생하는가? 순서대로 말해보자.

Mouse와 MouseMotion 이벤트



이동

클릭

MouseListener 인터페이스

```
public class MyListener implements MouseListener {
    public void mousePressed(MouseEvent e) {          }
    public void mouseReleased(MouseEvent e) {         }
    public void mouseEntered(MouseEvent e) {          }
    public void mouseExited(MouseEvent e) {           }
    public void mouseClicked(MouseEvent e) {          }
}
```

사용자가 컴포넌트를 클릭한 경우에 호출된다.

마우스 컴포넌트 위에서 떼어지면 호출된다.

마우스 커서가 컴포넌트로 들어가면 호출된다.

마우스 커서가 컴포넌트에서 나가면 호출된다.

마우스 컴포넌트 위에서 눌려지면 호출된다.

MouseMotionListener 인터페이스

```
public class MyClass implements MouseMotionListener {
      public void mouseDragged(MouseEvent e) {        }
      public void mouseMoved(MouseEvent e) {          }
}
```

마우스를 드래그하면 호출된다.

마우스가 클릭되지 않고 이동하는 경우에 호출된다.

마우스 이벤트 관찰

```
Mouse pressed (# of clicks: 1) X=93 Y=47
Mouse dragged X=93 Y=48
Mouse dragged X=94 Y=48
...
Mouse dragged X=117 Y=66
Mouse dragged X=118 Y=66
Mouse released (# of clicks: 1) X=118 Y=66
```

버튼을 클릭하였을 때 발생

버튼을 클릭한 채로 움직이면 발생

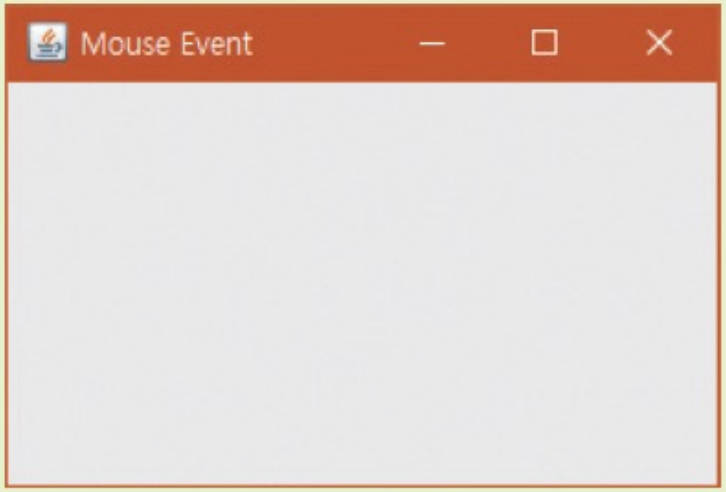
버튼에서 손을 떼면 발생

실행 결과

MouseEvent 객체

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

예제: 마우스 이벤트 정보 출력하기




```
Mouse moved X=240 Y=0
Mouse moved X=241 Y=0
Mouse exited X=242 Y=-1
Mouse entered X=119 Y=2
Mouse moved X=119 Y=2
Mouse moved X=156 Y=25
Mouse moved X=205 Y=57
Mouse moved X=256 Y=95
Mouse exited X=653 Y=419
```

예제: 마우스 이벤트 정보 출력하기

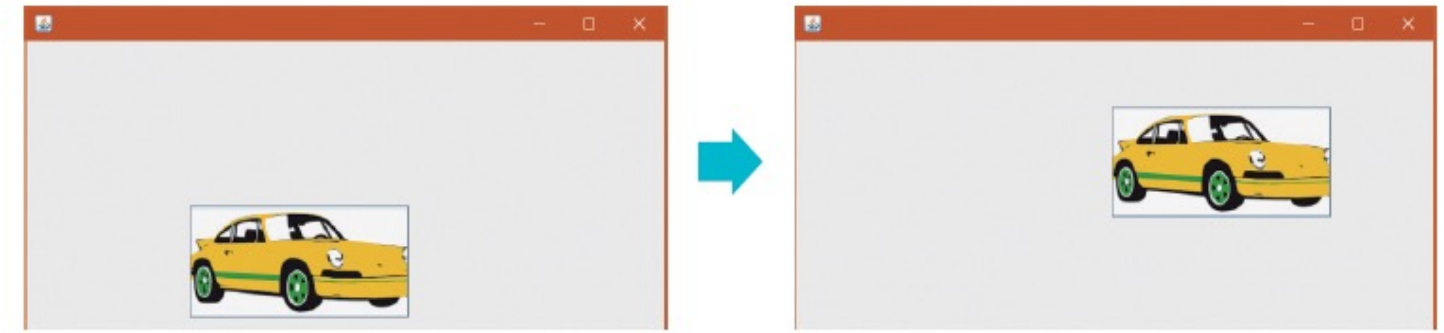
```
class MouseEventTest extends JFrame implements MouseListener, MouseMotionListener {
        public MouseEventTest() {
                setTitle("Mouse Event");
                setSize(300, 200);

                JPanel panel = new JPanel();
                panel.addMouseListener(this);
                panel.addMouseMotionListener(this);
                add(panel);
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }

        public void mousePressed(MouseEvent e) {
                display("Mouse pressed (# of clicks: " + e.getClickCount() + ")", e);
        }
        public void mouseReleased(MouseEvent e) {
                display("Mouse released (# of clicks: " + e.getClickCount() + ")", e);
        }
```

예제: 마우스 이벤트 정보 출력하기

```
        public void mouseEntered(MouseEvent e) {
                display("Mouse entered", e);
        }
        public void mouseExited(MouseEvent e) {
                display("Mouse exited", e);
        }
        public void mouseClicked(MouseEvent e) {
                display("Mouse clicked (# of clicks: " + e.getClickCount() + ")", e);
        }
        public void mouseDragged(MouseEvent e) {
                display("Mouse dragged", e);
        }
        public void mouseMoved(MouseEvent e) {
                display("Mouse moved", e);
        }
        protected void display(String s, MouseEvent e) {
                System.out.println(s + " X=" + e.getX() + " Y=" + e.getY());
        }
        public static void main(String[] args) {
                MouseEvent f = new MouseEvent();
        }
}
```

예제: 마우스로 자동차 이동하기

- 앞에서 키보드의 화살표 키를 이용하여 자동차를 움직이는 프로그램을 작성한 바 있다. 이번에는 마우스 클릭으로 자동차를 이동하는 프로그램을 작성하여 보자.



예제: 마우스로 자동차 이동하기

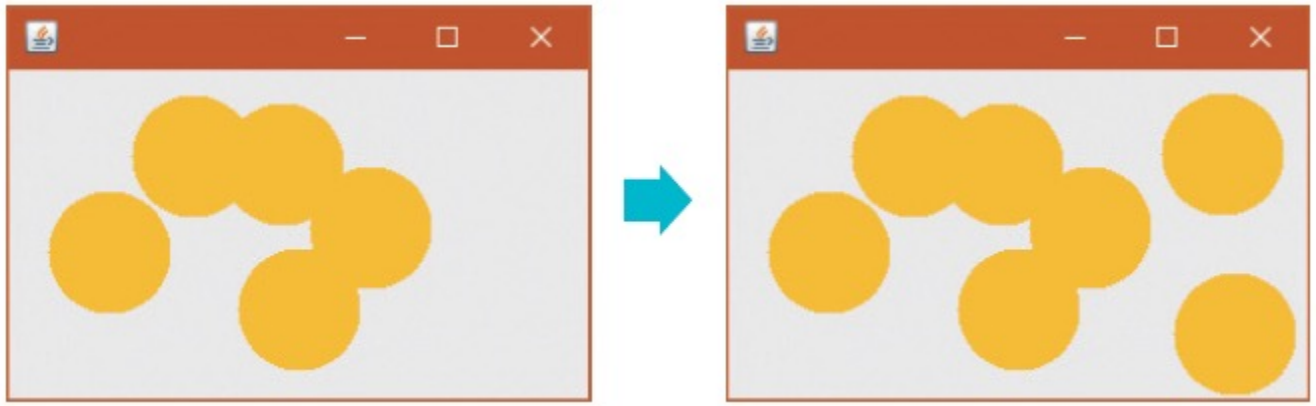
```
public class MoveCar extends JFrame {

    int img_x=150, img_y=150;
    JButton button;

    public MoveCar() {
        setSize(600, 300);
        button = new JButton("");
        ImageIcon icon = new ImageIcon("car.png");

        button.setIcon(icon);
        JPanel panel = new JPanel();
        panel.setLayout(null);
        button.setLocation(img_x, img_y);
        button.setSize(200, 100);
        panel.add(button);
        panel.requestFocus();
        panel.setFocusable(true);
```

예제: 마우스로 자동차 이동하기

```
                panel.addMouseListener(new MouseListener() {
                        public void mousePressed(MouseEvent e) {
                                img_x = e.getX();
                                img_y = e.getY();
                                button.setLocation(img_x, img_y);
                        }
                        public void mouseReleased(MouseEvent e) {       }
                        public void mouseEntered(MouseEvent e) {        }
                        public void mouseExited(MouseEvent e) {         }
                        public void mouseClicked(MouseEvent e) {        }
                });
                add(panel);
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }

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

예제: 마우스로 원 그리기

- 우리는 아직 원을 그리는 방법을 학습하지 않았지만 프레임에서 getGraphics()을 호출하여 Graphics 객체를 얻은 후에 fillOval()을 호출하면 간단히 원을 그릴 수 있다. 마우스를 클릭하는 곳에 원을 그려보자.



예제: 마우스로 원 그리기

```
public class DrawCircle extends JFrame implements MouseListener {
        public DrawCircle() {
                addMouseListener(this);
                setSize(300, 300);
                setLayout(null);
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
        public void mouseClicked(MouseEvent e) {
                Graphics g = getGraphics();
                g.setColor(Color.ORANGE);
                g.fillOval(e.getX()-30, e.getY()-30, 60, 60);
        }

        public void mouseEntered(MouseEvent e) {               }
        public void mouseExited(MouseEvent e) {  }
        public void mousePressed(MouseEvent e) {}
        public void mouseReleased(MouseEvent e) {              }

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

중간점검

1. 사용자가 마우스를 더블클릭하였는지를 인식하려면 어떻게 해야 하는가?
2. 마우스 이벤트 리스터나 마우스의 현재 좌표를 알려면 어떻게 해야 하는가?
3. 위의 원 그리기 프로그램에서 색상과 원의 크기를 난수로 해보자.

어댑터 클래스

- 이벤트를 처리하기 위해서는 리스너 인터페이스에서 정의되어 있는 모든 메소드를 구현해야 한다.
- 어댑터 클래스(Adaptor Class)이다. 인터페이스를 구현해놓은 클래스이다.
- 어댑터 클래스를 상속받아서 원하는 메소드만을 재정의하는 것이 가능해진다.

```
public abstract class MouseAdapter implements MouseListener, MouseWheelListener,
MouseMotionListener {

    public void mouseClicked(MouseEvent e) {}
    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
    public void mouseWheelMoved(MouseWheelEvent e){}
    public void mouseDragged(MouseEvent e){}
    public void mouseMoved(MouseEvent e){}
}
```

리스너 인터페이스를 구현하는 방법

```
class MyListener  implements MouseListener {
   public void mouseClicked(MouseEvent e) {          // 작성하기 원하는 메소드
                        …
               }
   public void mousePressed(MouseEvent e) {}         // 불필요한 메소드
   public void mouseReleased(MouseEvent e) {}
   public void mouseEntered(MouseEvent e) {}
   public void mouseExited(MouseEvent e) {}
   public void mouseWheelMoved(MouseWheelEvent e){}
   public void mouseDragged(MouseEvent e){}
   public void mouseMoved(MouseEvent e){}
}
```

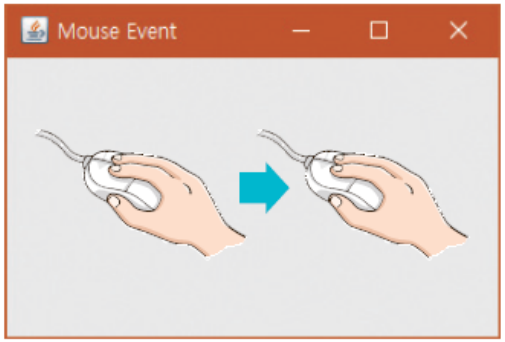
어댑터 클래스를 상속받는 방법

```
class MyListener  extends MouseAdaptor {

        public void mouseClicked(MouseEvent e){
                if( e.getX > 300 ){
                        ...
                }
        }
}
```

필요한 메소드만
재정의한다.

예제: 마우스 드래그 이벤트 출력하기

- 마우스 어댑터 클래스를 사용하여서 마우스 드래그 이벤트를 처리해보자. 마우스 드래그 이벤트가 발생하면 이벤트 정보를 콘솔에 출력한다.




```
java.awt.event.MouseEvent
[MOUSE_DRAGGED,(94,54), ...
java.awt.event.MouseEvent
[MOUSE_DRAGGED,(95,55), ...
```

예제: 마우스 드래그 이벤트 출력하기

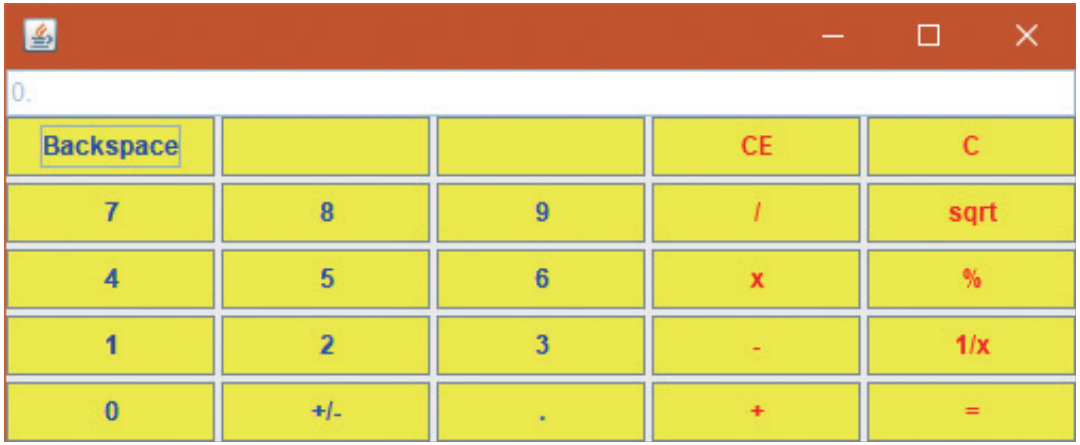
```
public class MouseDrag extends JFrame {
        JPanel panel;

        public MouseDrag() {
                setTitle("Mouse Event");
                setSize(300, 200);

                panel = new JPanel();
                panel.addMouseMotionListener(new MouseAdapter() {
                        public void mouseDragged(MouseEvent e) {
                                System.out.println(e);
                        }
                });
                add(panel);
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }

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

중간점검

1. 어댑터 클래스의 장점은 무엇인가?
2. 무명 클래스로 KeyAdaptor 객체를 생성하고 getKeyCode()만 재정의하는 코드를 작성해 보자.

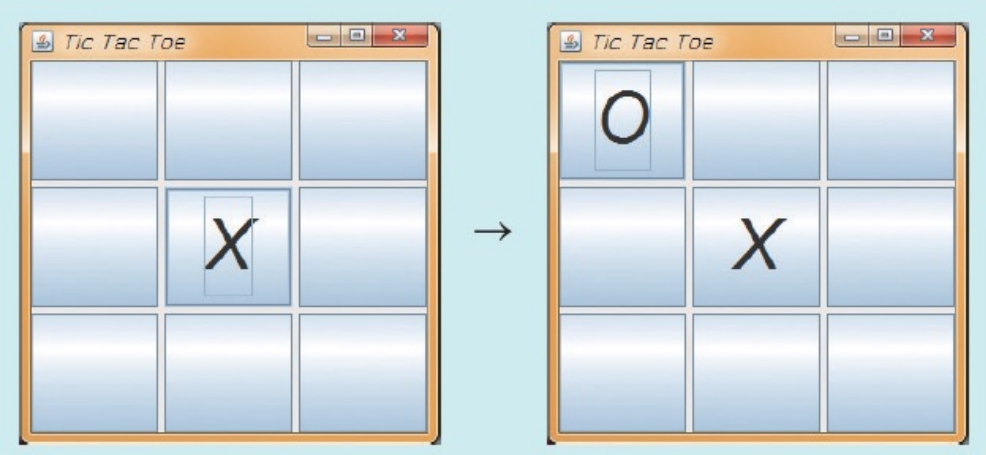
Mini Project: 계산기 프로그램

- 이번 장에서 버튼의 액션 이벤트를 처리하여 기초적인 계산 기능을 부여해 보자



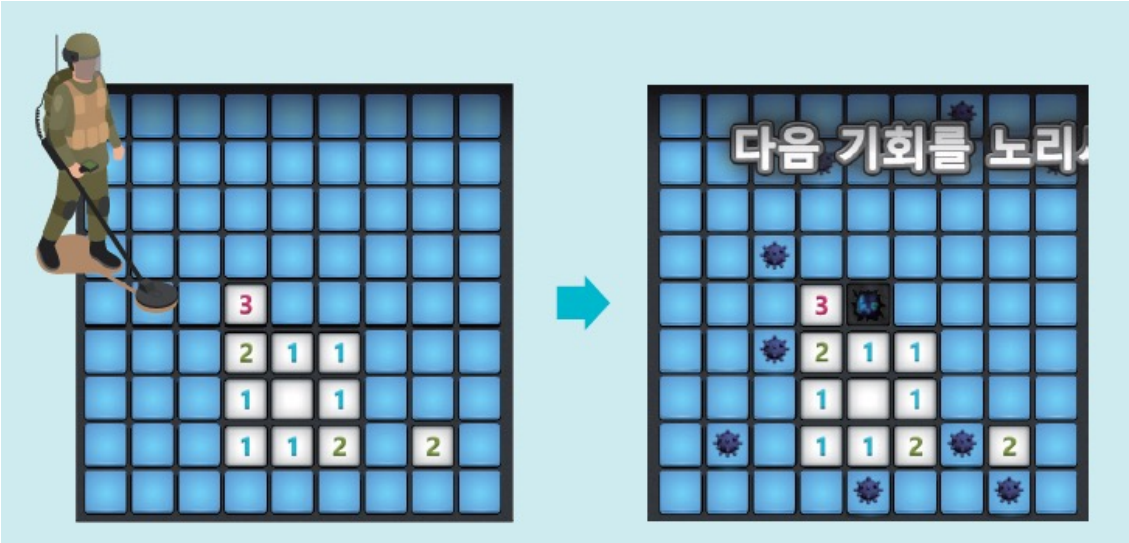
Mini Project: Tic-Tac-Toe 게임

- Tic-Tac-Toe 게임을 작성하여 보자. Tic-Tac-Toe 게임은 3×3칸을 가진 게임판을 만들고, 경기자가 동그라미 기호(O)와 가위표 기호(X)을 번갈아 가며 게임판에 놓는 게임이다



Mini Project: 지뢰 찾기 게임

- 게임의 목표는 “지뢰”가 숨겨진 직사각형 보드에서 인접 지뢰의 개수 힌트의 도움을 받아 어떤 지뢰도 폭파시키지 않고 안전하게 제거하는 것이다



Summary

- 이벤트 구동 프로그래밍이란 이벤트의 발생에 의하여 프로그램의 실행 흐름이 결정되는 프로그래밍 방식이다. 자바 GUI 프로그래밍도 여기에 속한다.
- 이벤트는 버튼을 클릭하거나, 마우스를 움직이거나 하면 발생한다.
- 이벤트 리스너란 이벤트를 받아서 처리하는 객체이다.
- 이벤트 리스너 클래스는 여러 가지 방법으로 구현할 수 있다. 많이 사용되는 방법은 내부 클래스로 작성하거나 무명 클래스로 작성하는 방법이다.
- 내부 클래스로 이벤트 리스너를 작성하면 외부 클래스의 멤버에 자유롭게 접근할 수 있어서 편리하다.
- 키 이벤트는 KeyListener 인터페이스를 구현하여 처리한다. getKeyChar(), getKeyCode() 등의 메소드가 제공된다.
- 마우스 이벤트는 MouseListener나 MouseMotionListener를 구현하여 처리한다. mouseClicked()와 같은 메소드들이 제공된다.
- 어댑터 클래스는 리스너 인터페이스의 메소드들을 전부 구현해놓은 클래스로서, 이것을 상속받아서 필요한 메소드만 오버라이드할 수 있다.

Q & A