

10

GUI(GUI) 및 이벤트 구동 프로그래밍

GUI (Graphical User Interface)

- 사용자 인터페이스
 - 프로그램과 사람을 연결짓는 것
- 그래픽 사용자 인터페이스
 - 문자가 아닌 보다 비주얼한 것들을 사용하여 사용자가 편리하게 사용하도록 하는 것
- AWT/Swing
 - Java에서 제공하는 기본 GUI 프레임워크
 - java.awt, javax.swing 패키지

이벤트 구동 프로그래밍 (Event-Driven Programming)

- 이벤트: 불시에 일어나는 사건
 - 예, 마우스 움직였다, 버튼이 눌렸다, 키보드가 눌렸다, 메뉴 아이템이 선택되었다
- 이벤트 구동 프로그램
 - 이벤트 발생과 처리로 제어하는 프로그램
- 이벤트 처리기 (event-handler or event-listener)
 - 이벤트를 처리하는 프로세스
 - 일반적으로, 이벤트를 여러 번 처리하여 정보를 축적하였다가 결과를 보여주는 형태
- Java에서는 액션 이벤트 (action event), 액션 리스너 (action listener)라고 부른다.

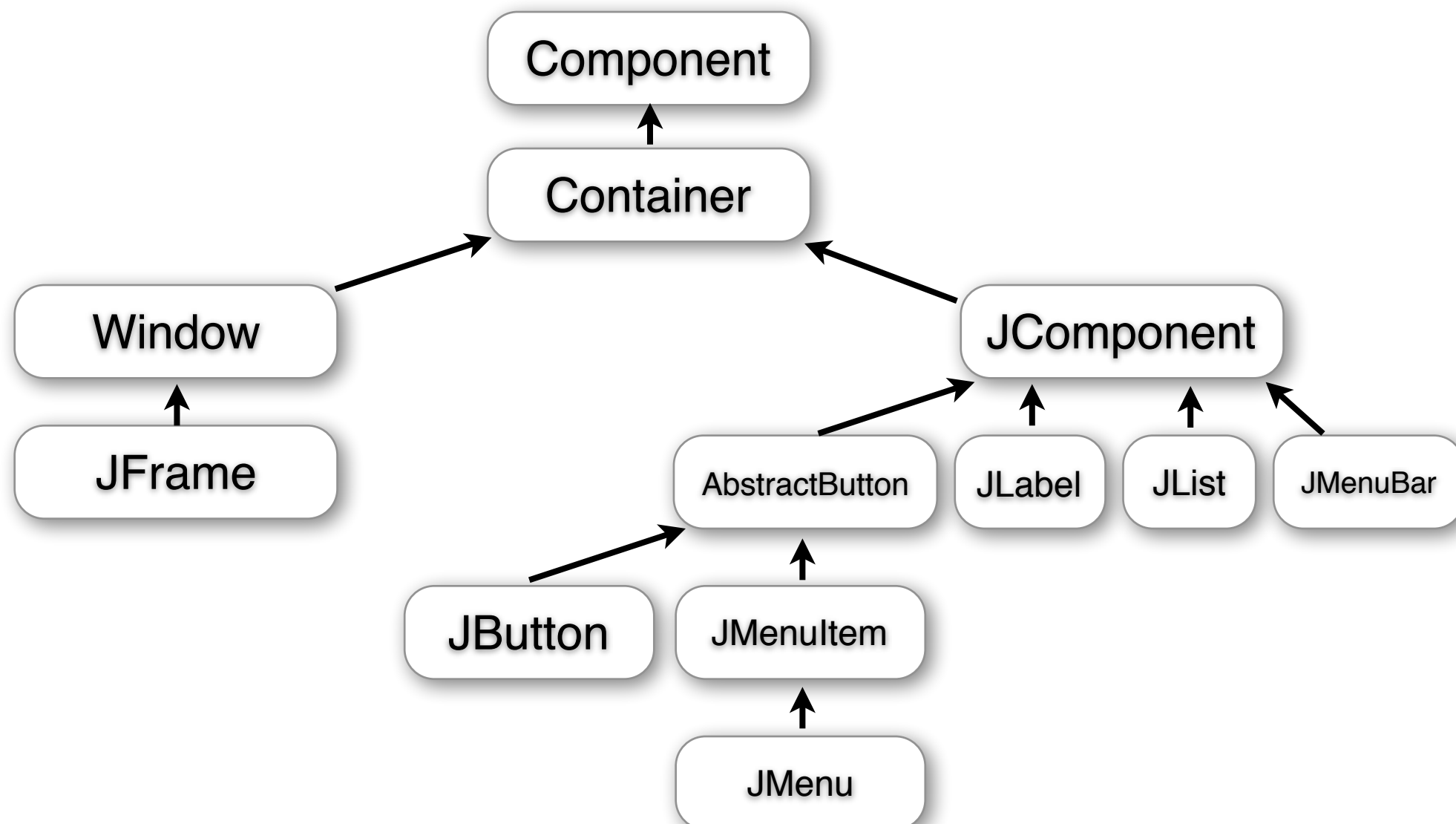
Java AWT/Swing에서의 용어들

- 컴퍼넌트 (component)
 - 화면에서 크기와 위치가 있고 이벤트가 발생하는 객체
 - 예, 라벨(label), 텍스트(text component), 단추(button), 리스트(list)
- 컨테이너 (container): 컴퍼넌트 여러 개를 담을 수 있는 컴퍼넌트
- 패널 (panel): 표준 컨테이너
- 윈도우 (window), 맨 위의 컨테이너
 - 바로 화면에 보여주는 컨테이너
 - 패널을 담은 컨테이너
- 프레임 (frame): 윈도우에 제목과 메뉴가 달린 것
- 대화창 (dialog): 임시 윈도우
- 메뉴바 (menu bar): 메뉴가 있어서 선택할 수 있는 것, 프레임에 달려 있음

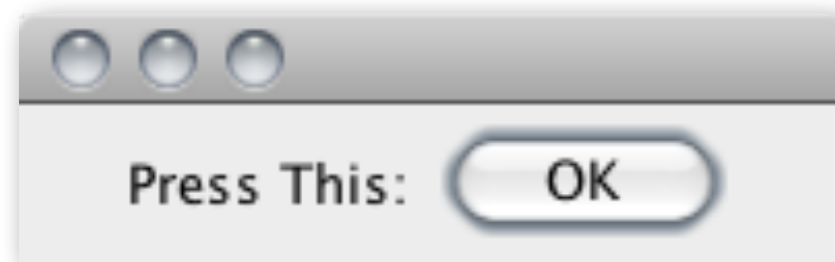
레이아웃 (Layout)

- 컨테이너가 가지고 있는 여러 컴퍼넌트를 어떻게 배열할 것인가에 대한 정책
- 레이아웃 관리자(layout manager)가 관리
- 예
 - 순서대로 (flow): 차례대로 나열
 - 경계 (border): 동서남북, 가운데 등으로 위치 지정
 - 그리드 (grid): 행렬모양으로

AWT/Swing 클래스 구조도



단순 예제, 단추



방법

- JFrame을 상속받아 나만의 프레임을 만든다.
- 라벨 "Press This"를 만든다.
- 단추 "OK"를 만든다.
- 레이아웃을 “순서대로”로 설정한다.
- 라벨을 프레임 컨테이너에 추가한다.
- 단추를 프레임 컨테이너에 추가한다.
- 보여준다.

구현

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

public class ButtonFrame extends JFrame {
    public ButtonFrame() {
        JLabel label = new JLabel("Press This:");
        JButton button = new JButton("OK");
        Container c = getContentPane();
        c.setLayout(new FlowLayout());
        c.add(label);
        c.add(button);
        setSize(200, 60);
        setVisible(true);
    }
    public static void main(String[] args) {
        new ButtonFrame();
    }
}
```

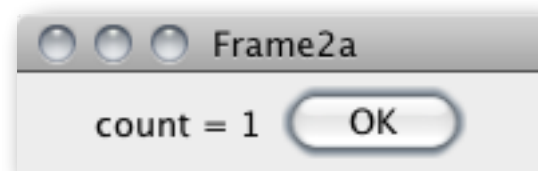
단추를 눌러 보아야 아무 소용 없죠

- 단추 눌림 이벤트가 발생했을 때 무슨 일을 할 지 등록해 두어야 합니다.
- 다음 인터페이스를 만족하는 처리기를 구현

```
public interface ActionListener {  
    public void actionPerformed(ActionEvent e);  
}
```
- 단추에 등록해야 합니다.

예제, 셈하기

- 단추를 누를 때 마다 1씩 증가하는 창을 만드세요.



모델: 카운터

```
public class Counter {  
    private int count;  
    public Counter(int start) {  
        count = start;  
    }  
    public void increment() {  
        count++;  
    }  
    public int countOf() {  
        return count;  
    }  
}
```

이벤트 처리기와 프레임 동시 구현

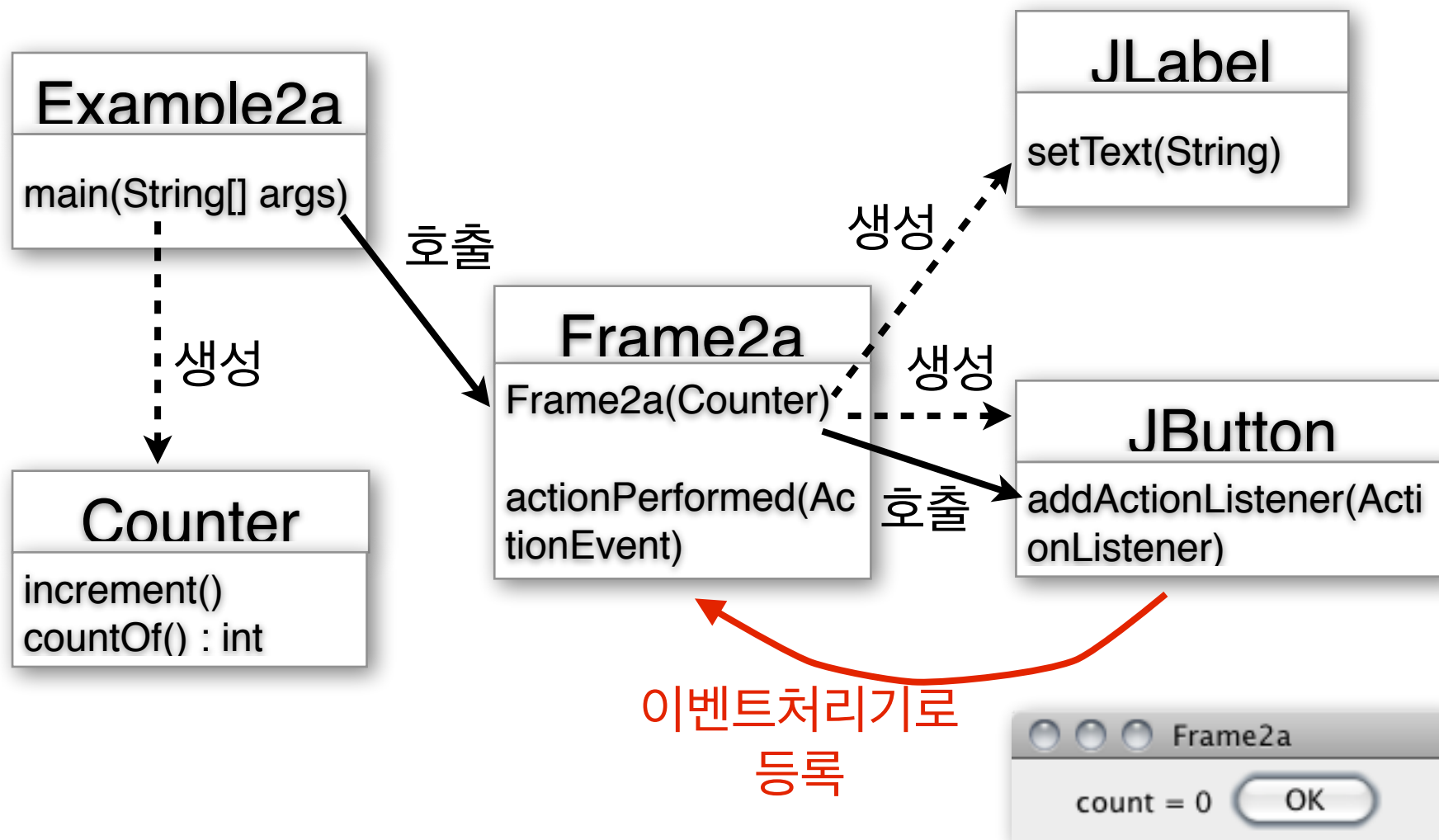
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class Frame2a extends JFrame implements ActionListener {
    private Counter count;
    private JLabel label = new JLabel("count = 0");
    public Frame2a(Counter c) {
        count = c;
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
        JButton button = new JButton("OK");
        cp.add(label); cp.add(button);
        button.addActionListener(this); // 이 객체를 단추의 이벤트 처리기로 등록
        setTitle("Frame2a"); setSize(200, 60);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        count.increment();
        label.setText("count = " + count.countOf());
    }
}
```

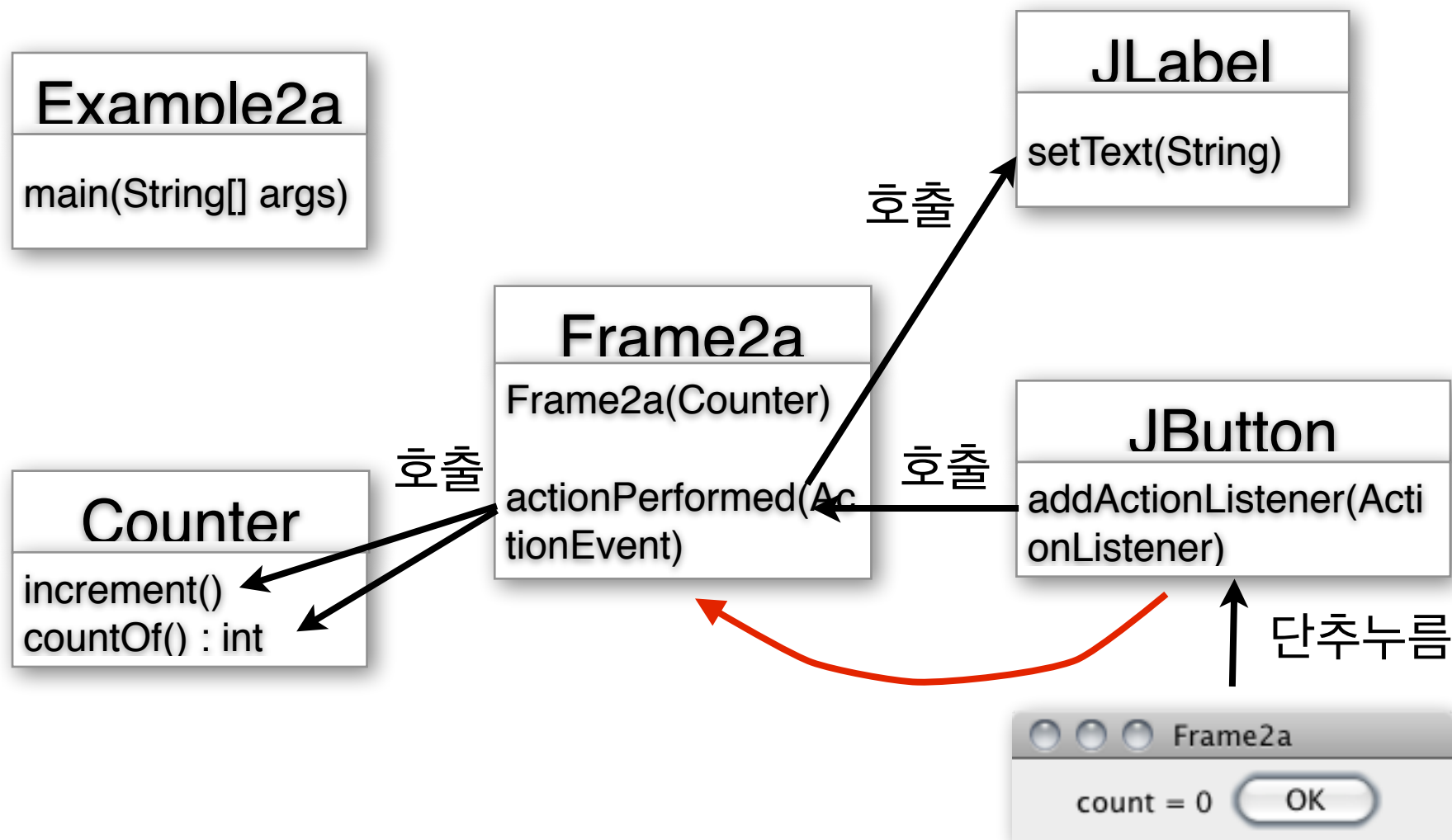
구동 코드

```
public class Example2a {  
    public static void main(String[] args) {  
        Counter model = new Counter(0);  
        Frame2a view = new Frame2a(model);  
    }  
}
```

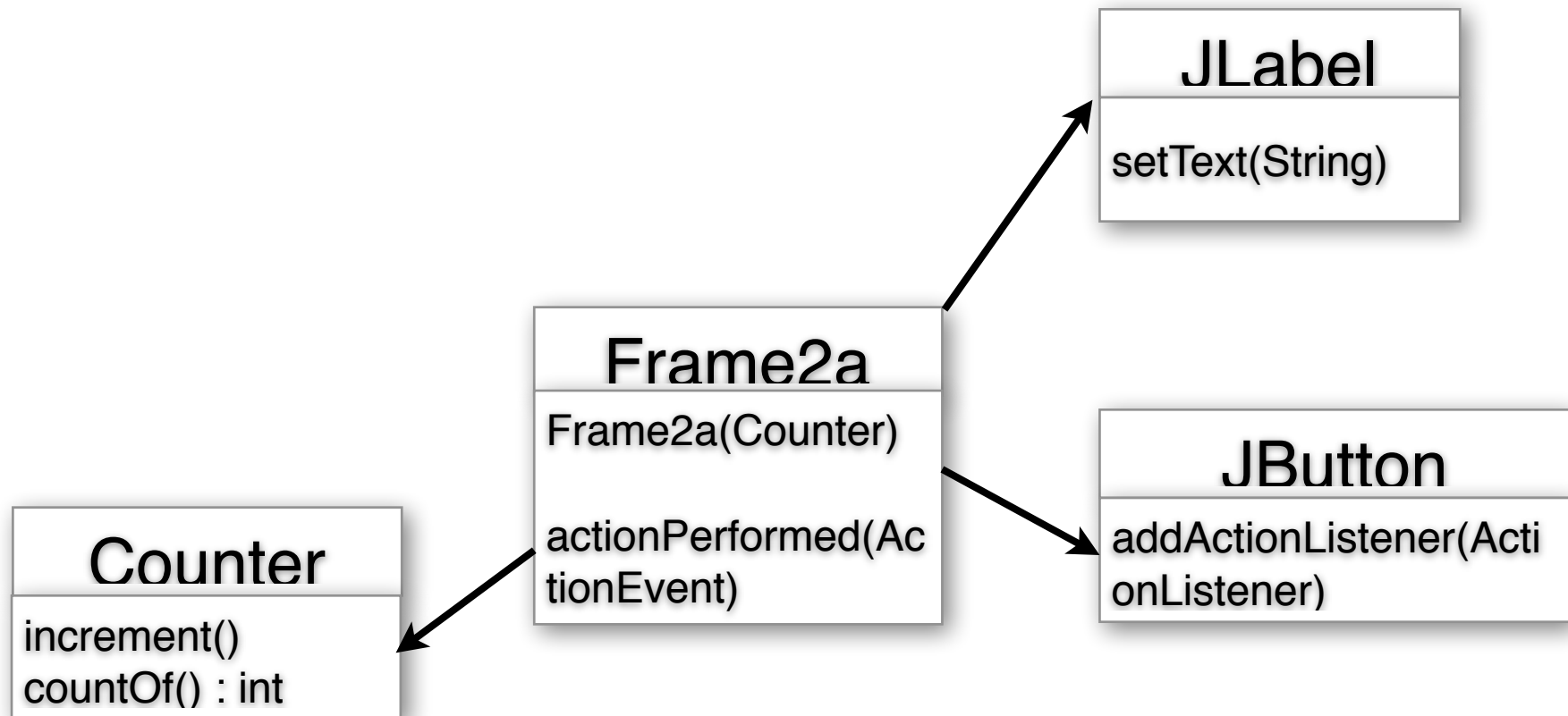
실행 구조: 시작



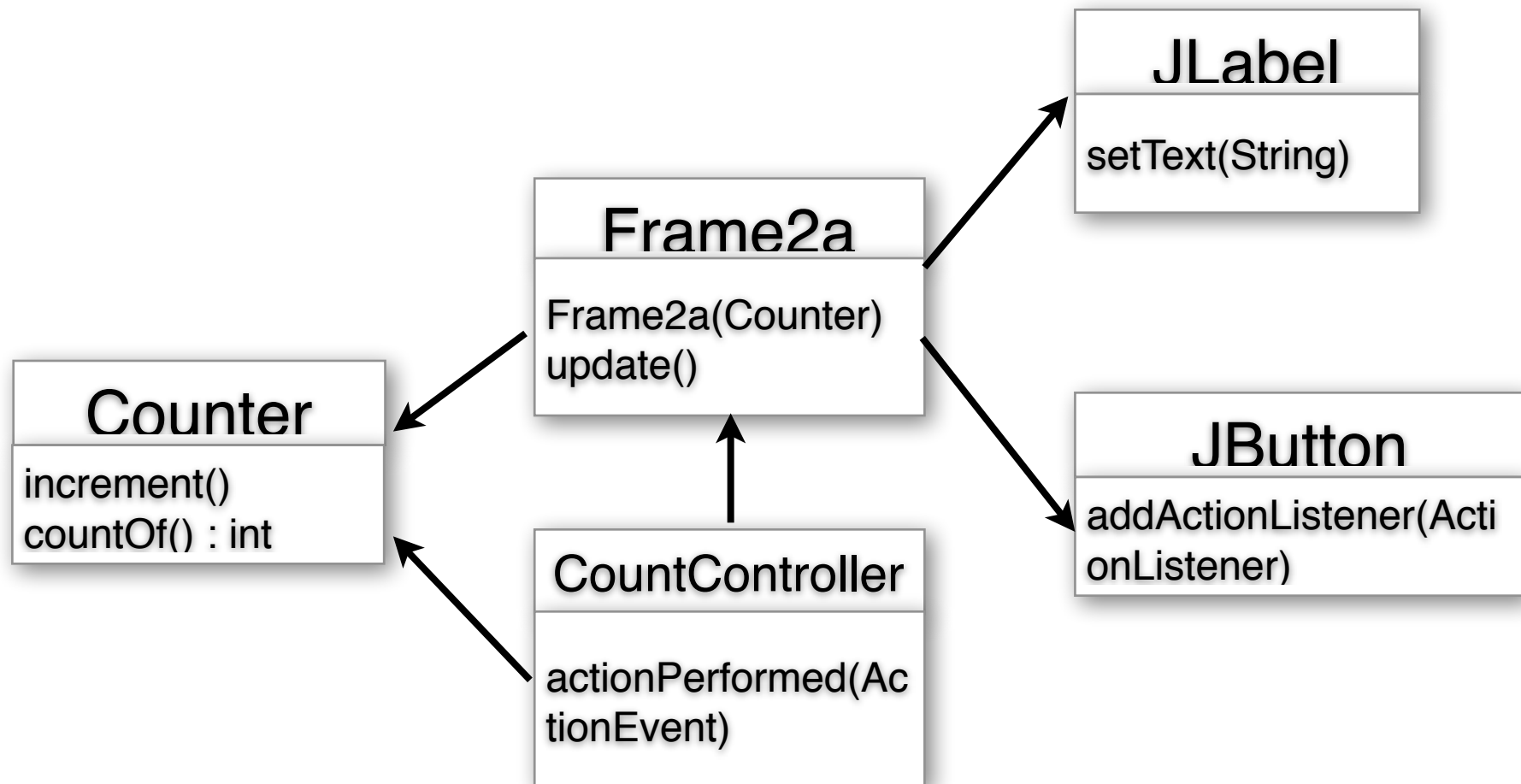
실행 구조: 단추 누름



현재: 출력 뷰와 제어가 합쳐져 있음



대안 1: 출력 뷰와 제어기 분리



출력 뷰

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

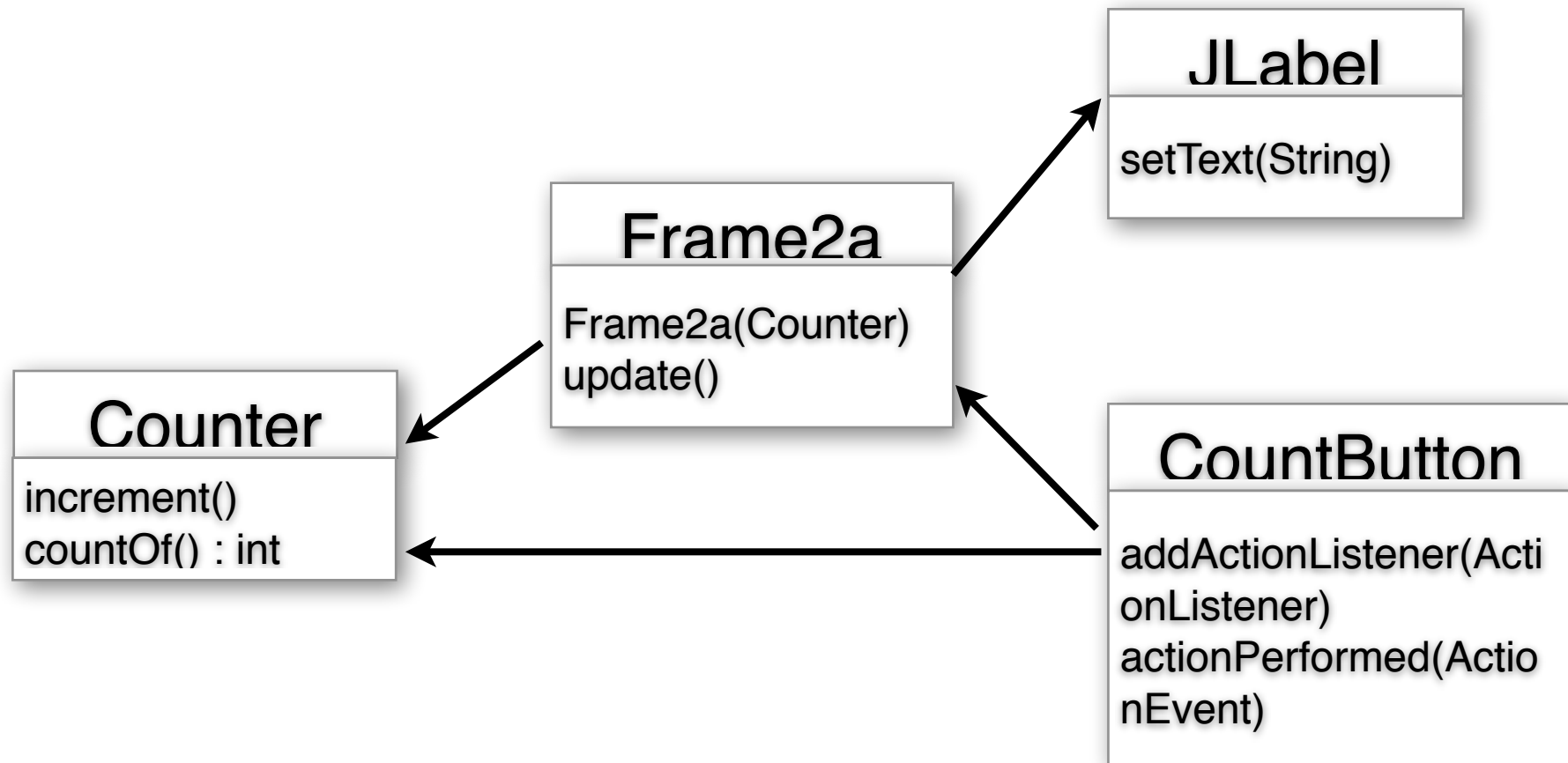
class Frame2b extends JFrame {
    private Counter count;
    private JLabel label = new JLabel("count = 0");
    public Frame2b(Counter c) {
        count = c;
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
        JButton button = new JButton("OK");
        cp.add(label); cp.add(button);
        button.addActionListener(new CountController(count, this));
        setTitle("Frame2b"); setSize(200, 60);
        setVisible(true);
    }
    public void update() {
        // 모델에 계산시키는 부분이 삭제
        label.setText("count = " + count.countOf());
    }
}
```

제어기

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

public class CountController implements ActionListener {
    private Frame2b view;
    private Counter model;
    public CountController(Counter m, Frame2b v) {
        view = v; model = m;
    }
    public void actionPerformed(ActionEvent e) {
        model.increment();
        view.update();
    }
}
```

대안 2: 입력 뷰와 제어를 함께



출력 뷰

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

class Frame2c extends JFrame {
    private Counter count;
    private JLabel label = new JLabel("count = 0");
    public Frame2b(Counter c) {
        count = c;
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
        CountButton button = new CountButton("OK", count, this);
        cp.add(label); cp.add(button);
        // 단추의 이벤트 처리기 등록시키는 부분 삭제
        setTitle("Frame2c"); setSize(200, 60);
        setVisible(true);
    }
    public void update() {
        label.setText("count = " + count.countOf());
    }
}
```

입력 뷰 + 제어기

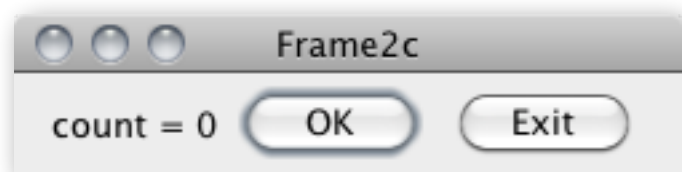
```
import javax.swing.*; import java.awt.event.*;

public class CountButton extends JButton implements
ActionListener {
    private Frame2c view;
    private Counter model;
    public CountButton(String label, Counter m, Frame2c v) {
        super(label);
        view = v; model = m;
        addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        model.increment();
        view.update();
    }
}
```

여러 제어기

- 이벤트 구동 프로그램은 제어기가 여러 개 있을 수 있다.
 - 단추1을 눌렀을 때
 - 단추2를 눌렀을 때
 - 단추3을 눌렀을 때
- 그렇기 때문에 단추에 제어기를 장착하는 것이 유리한 경우가 많다.

예제, 종료 단추를 추가하라



종료 단추

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

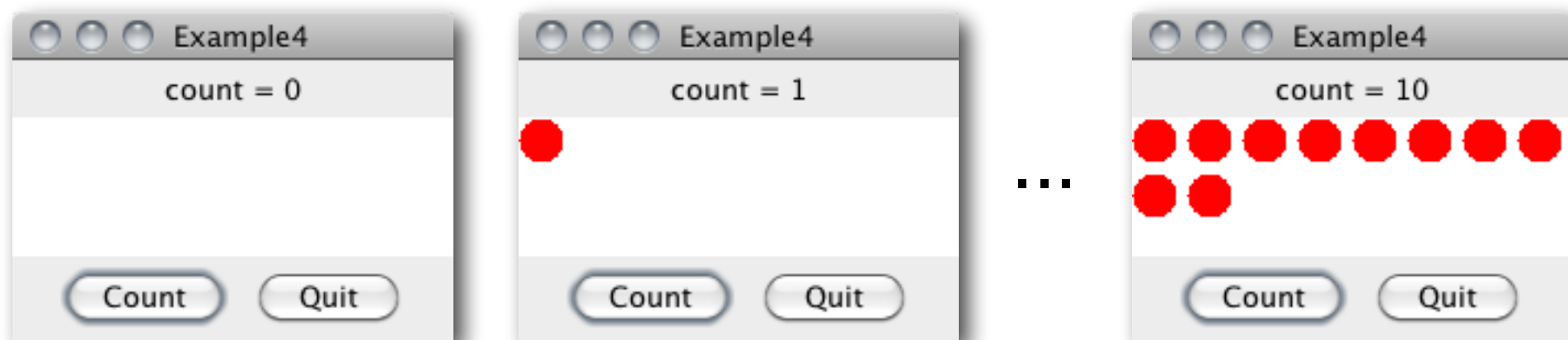
public class ExitButton extends JButton implements
ActionListener {
    public ExitButton(String label) {
        super(label);
        addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
}
```

출력 뷰

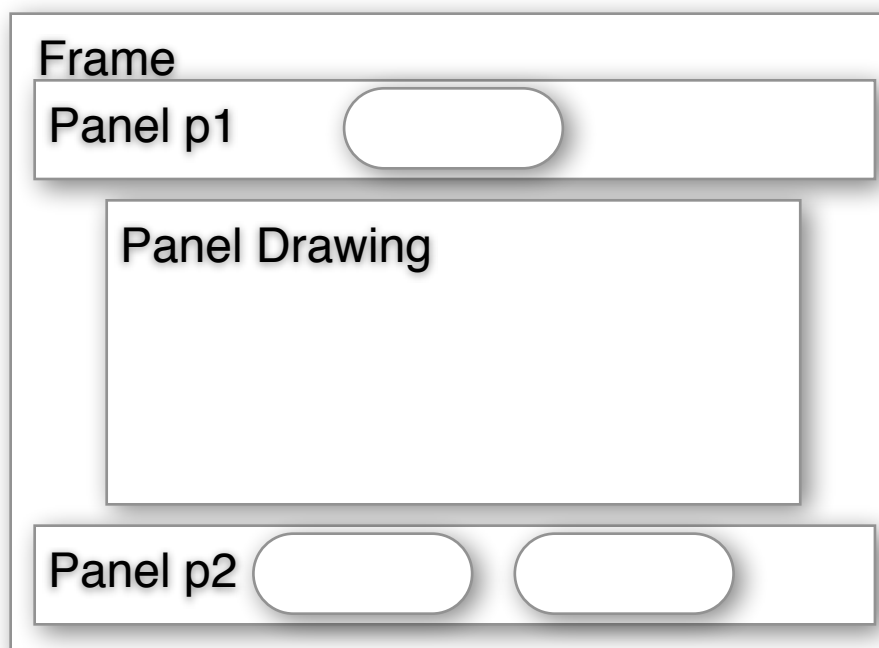
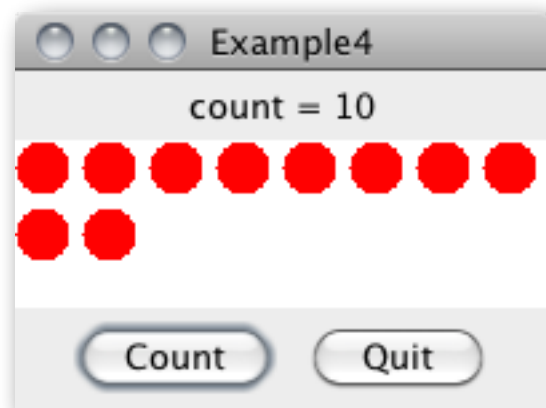
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class Frame2c extends JFrame {
    private Counter count;
    private JLabel label = new JLabel("count = 0");
    public Frame2b(Counter c) {
        count = c;
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
        cp.add(label);
        cp.add(new CountButton("OK", count, this));
        cp.add(new ExitButton("Exit"));
        setTitle("Frame2c"); setSize(200, 60);
        setVisible(true);
    }
    public void update() {
        label.setText("count = " + count.countOf());
    }
}
```

예제, 공 세기



경계 레이아웃 (BorderLayout)



p1 패널은 NORTH로 Frame에 붙이고,
 Drawing 패널은 센터로
 p2 패널은 SOUTH로 붙인다.

Count 버튼, Drawing, Label은 모두 Counter를 공유

Drawing

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

public class Drawing extends JPanel {
    private Counter count;
    public Drawing(Counter model) {
        count = model; setSize(200,80);
    }
    public void paintComponent(Graphics g) {
        g.setColor(Color.white); // 바닥에 흰 칠하고
        g.fillRect(0,0,200,80);
        g.setColor(Color.red);
        int x=0, y=0;
        for(int i=0; i<count.countOf(); i++) { // 빨간 점을 그려준다.
            g.fillRect(x*25, y*25, 20, 20);
            x++; if(x>7) { x=0; y++; }
        }
    }
}
```

CountButton

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

public class CountButton extends JButton implements ActionListener {
    private Frame4 view;
    private Counter model;
    public CountButton(String label, Counter m, Frame4 v) {
        super(label);
        view = v; model = m;
        addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        model.increment();
        view.update();
    }
}
```

Frame

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

public class Frame4 extends JFrame {
    private Counter count;
    private JLabel lab = new JLabel("count = 0");
    private JPanel drawing;

    public void update() {
        lab.setText("count = " + count.countOf());
        drawing.repaint();
    }
}
```


Frame

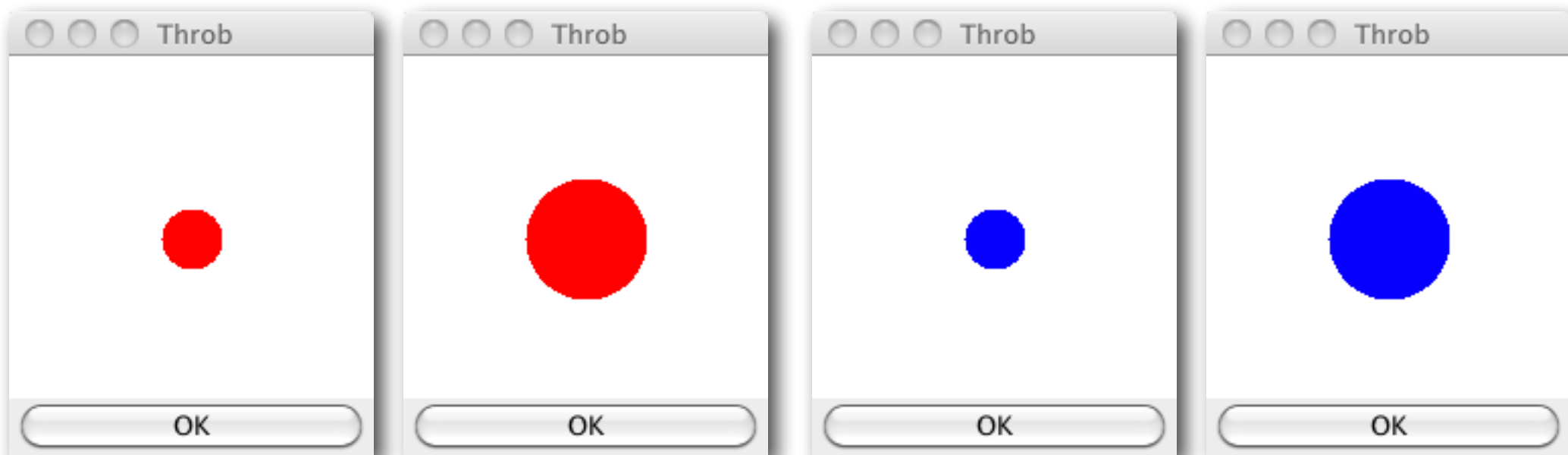
```
public Frame4(Counter c, JPanel panel) {  
    count = c; drawing = panel;  
    Container cp = getContentPane();  
    cp.setLayout(new BorderLayout());  
    JPanel p1 = new JPanel(new FlowLayout());  
    JPanel p2 = new JPanel(new FlowLayout());  
  
    lab = new JLabel("count = " + count.countOf());  
    p1.add(lab);  
  
    p2.add(new CountButton("Count", count, this));  
    p2.add(new ExitButton("Quit"));  
  
    cp.add(p1, BorderLayout.NORTH);  
    cp.add(drawing);  
    cp.add(p2, BorderLayout.SOUTH);  
    setTitle("Example4"); setSize(200,150); setVisible(true);  
}
```

구동 코드

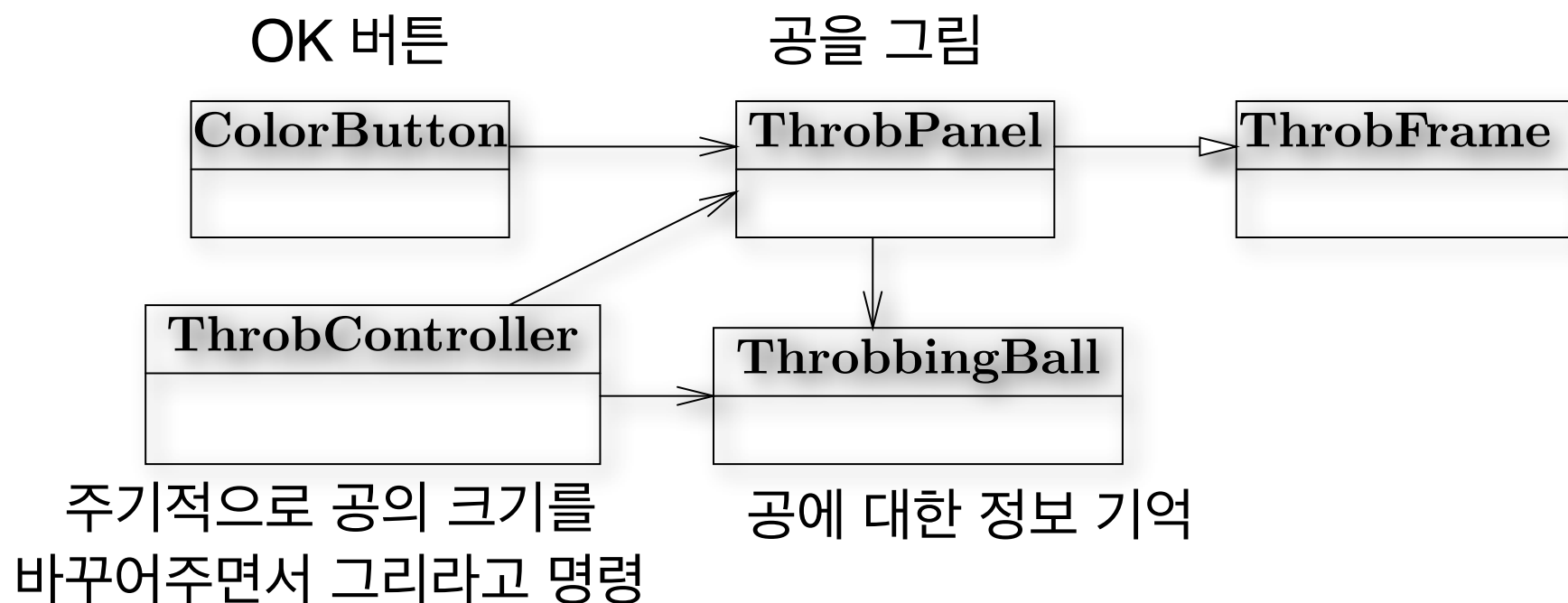
```
public class Example4 {  
    public static void main(String[] args) {  
        Counter model = new Counter(0);  
        Drawing drawing = new Drawing(model);  
        Frame4 view = new Frame4(model, drawing);  
    }  
}
```

예제, 숨쉬는 공, 색 변하는 공

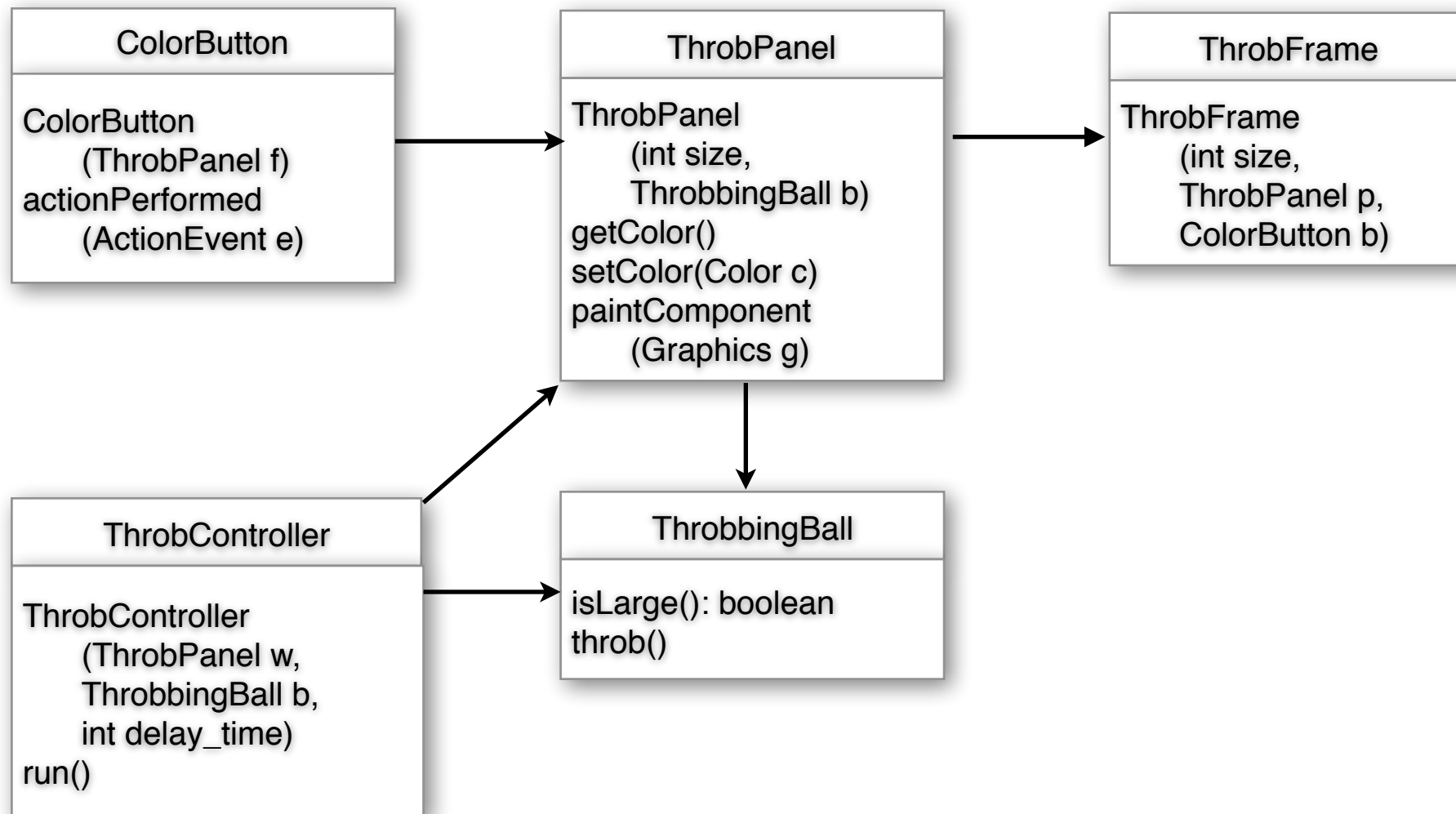
- 크기가 주기적으로 변하는 공을 그려보자.
- OK 단추를 그리면 색도 변하게 하자.



소프트웨어 구조



구체적인 소프트웨어 구조



ThrobbingBall

```
public class ThrobbingBall {  
    private boolean is_large = true;  
    public boolean isLarge() { return is_large; }  
    public void throb() { is_large = !is_large; }  
}
```

ThrobPanel

```
import java.awt.*; import javax.swing.*;
public class ThrobPanel extends JPanel {
    private int panel_size, location, ball_size;
    private Color c = Color.red;
    private ThrobbingBall ball;
    public ThrobPanel(int size, ThrobbingBall b) {
        panel_size = size; location = size/2; ball_size = size/3; ball = b;
        setSize(size, size);
    }
    public Color getColor() { return c; }
    public void setColor(Color new_color) { c = new_color; }
    public void paintComponent(Graphics g) {
        g.setColor(Color.white); g.fillRect(0, 0, panel_size, panel_size);
        g.setColor(c);
        if (ball.isLarge())
            g.fillOval(location-ball_size/2, location-ball_size/2, ball_size, ball_size);
        else
            g.fillOval(location-ball_size/4, location-ball_size/4, ball_size/2, ball_size/2);
    }
}
```

ColorButton

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

public class ColorButton extends JButton implements ActionListener {
    private ThrobPanel view;
    public ColorButton(ThrobPanel f) {
        super("OK");
        view = f;
        addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        Color c = view.getColor();
        if(c==Color.red)
            view.setColor(Color.blue);
        else
            view.setColor(Color.red);
    }
}
```


ThrobFrame

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

public class ThrobFrame extends JFrame {
    public ThrobFrame(int size, ThrobPanel p, ColorButton b) {
        Container cp = getContentPane();
        cp.setLayout(new BorderLayout());
        cp.add(p, BorderLayout.CENTER);
        cp.add(b, BorderLayout.SOUTH);
        setTitle("Throb"); setSize(size, size+40);
        setVisible(true);
    }
}
```

ThrobController

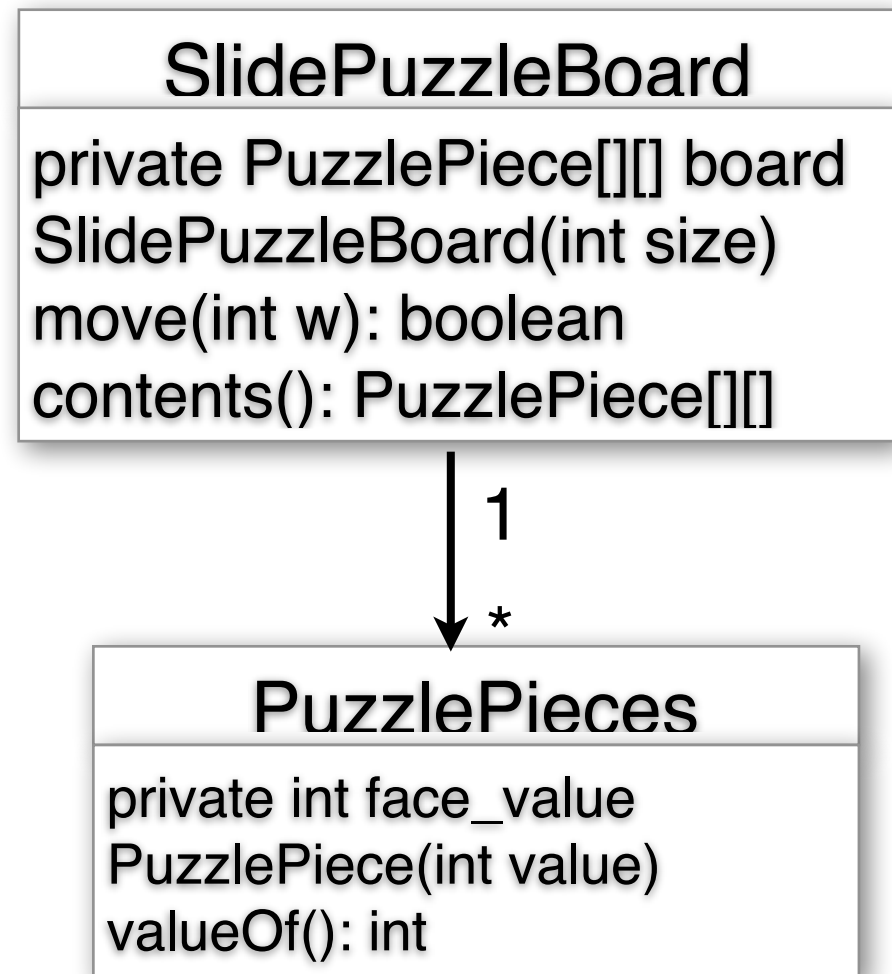
```
public class ThrobController {
    private ThrobPanel writer;
    private ThrobbingBall ball;
    private int time;

    public ThrobController(ThrobPanel w, ThrobbingBall b, int delay_time) {
        writer = w; ball = b; time = delay_time;
    }
    public void run() {
        while(true) {
            ball.throb();
            writer.repaint();
            delay();
        }
    }
    private void delay() {
        try { Thread.sleep(time); } catch (InterruptedException e) {}
    }
}
```

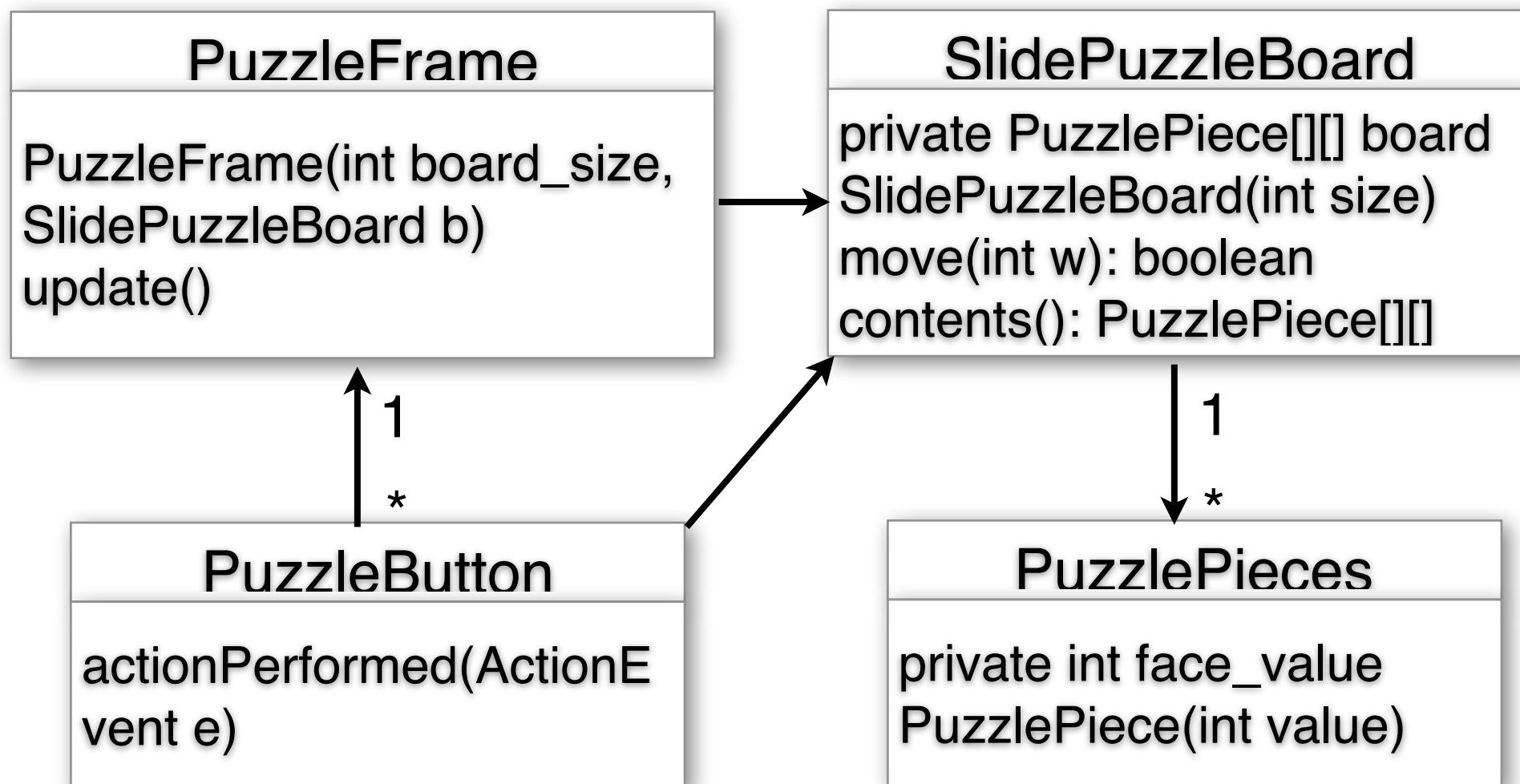
구동 코드

```
public class StartThrob {  
    public static void main(String[] a) {  
        int frame_size = 180;  
        int pause_time = 200;  
        ThrobbingBall b = new ThrobbingBall();  
        ThrobPanel p = new ThrobPanel(frame_size, b);  
        ThrobFrame f = new ThrobFrame(frame_size, p, new ColorButton(p));  
        new ThrobController(p, b, pause_time).run();  
    }  
}
```

예제, 퍼즐판 그리기



출력 뷰, 제어기 추가



PuzzleButton

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

public class PuzzleButton extends JButton implements ActionListener {
    private SlidePuzzleBoard puzzle;
    private PuzzleFrame view;

    public PuzzleButton(SlidePuzzleBoard p, PuzzleFrame v) {
        puzzle = p; view = v;
        addActionListener(this);
    }
    public void actionPerformed(ActionEvent evt) {
        String s = getText();
        if(!s.equals("")) {
            boolean ok = puzzle.move(new Integer(s).intValue());
            if(ok) view.update();
        }
    }
}
```

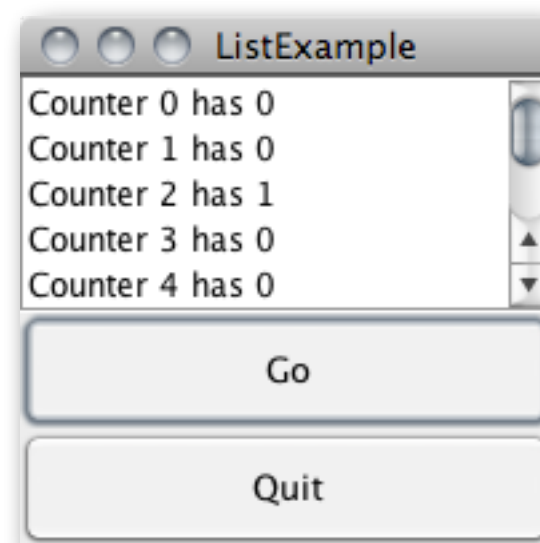
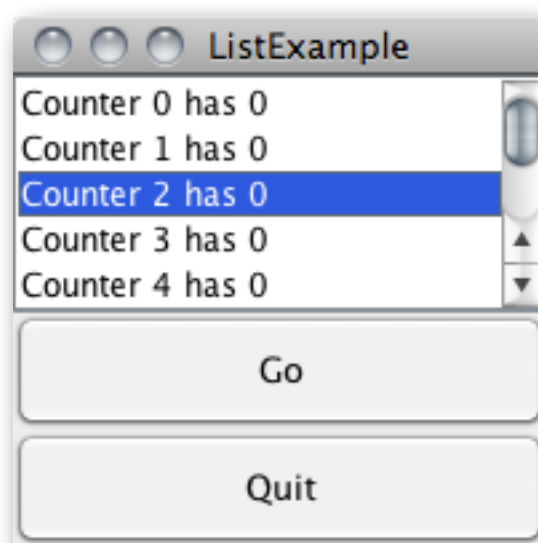
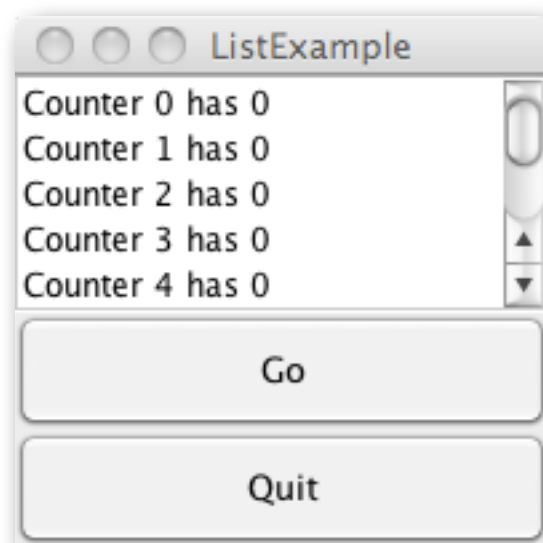
PuzzleFrame

```
public class PuzzleFrame extends JFrame {  
    private SlidePuzzleBoard board;  
    private int size, button_size = 60;  
    private PuzzleButton[][] button;  
  
    public PuzzleFrame(int board_size, SlidePuzzleBoard b) {  
        size = board_size; board = b;  
        button = new PuzzleButton[size][size];  
        Container cp = getContentPane();  
        cp.setLayout(new GridLayout(size, size));  
        for (int i=0; i<size; i++) for(int j=0; j<size; j++) {  
            button[i][j] = new PuzzleButton(board, this);  
            cp.add(button[i][j]);  
        }  
        update();  
        setTitle("PuzzleFrame");  
        setSize(size * button_size + 10, size * button_size + 20);  
        setVisible(true);  
    }  
}
```

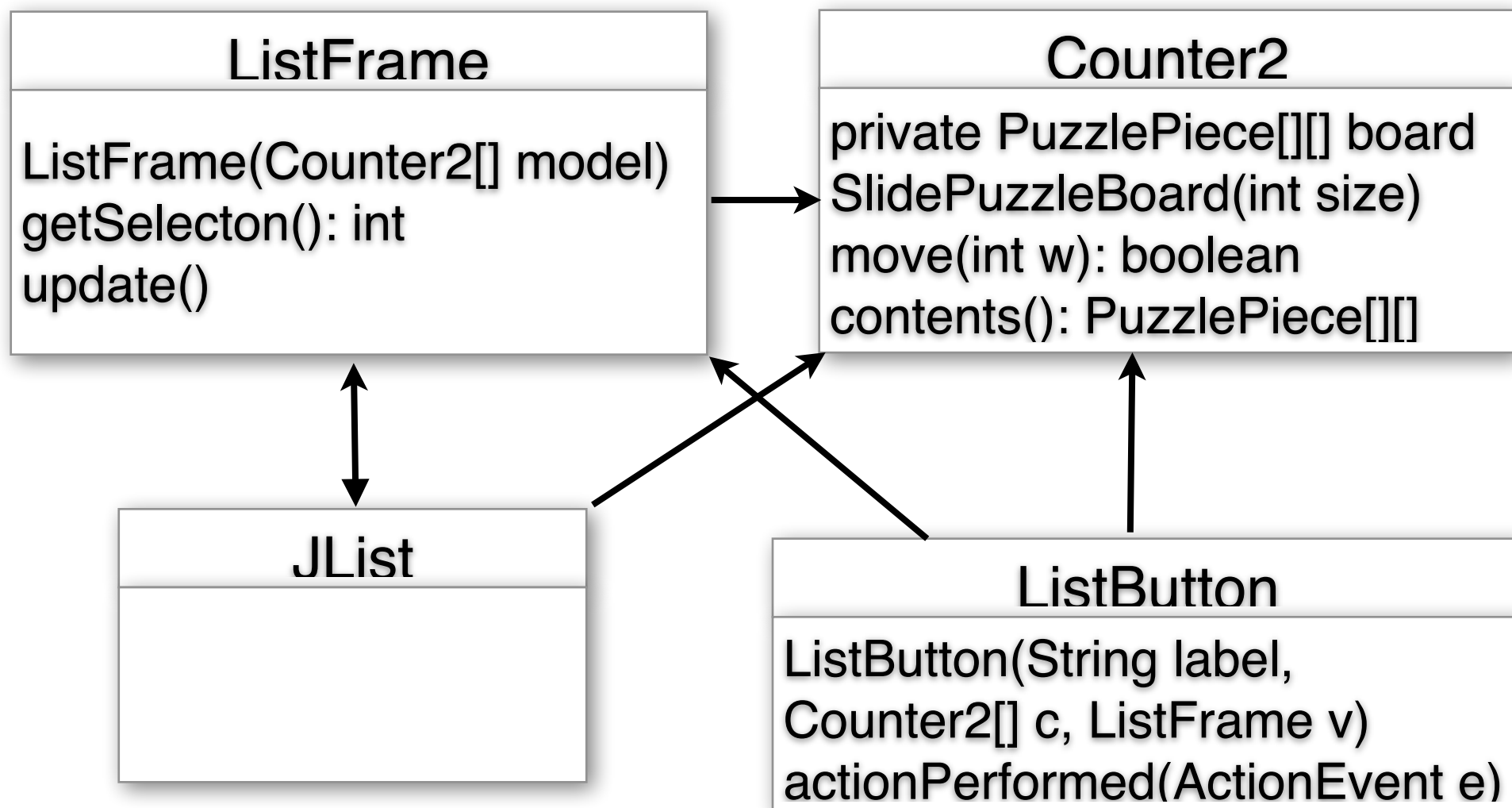
PuzzleFrame

```
public void update() {  
    PuzzlePiece[][] r = board.contents();  
    for(int i=0; i<size; i++) for(int j=0; j<size; j++) {  
        if (r[i][j]!=null) {  
            button[i][j].setBackground(Color.white);  
            button[i][j].setText("" + r[i][j].valueOf());  
        }  
        else {  
            button[i][j].setBackground(Color.black);  
            button[i][j].setText("");  
        }  
    }  
}
```


예제, 넘김 기능 있는 리스트 (Scrolling List)



소프트웨어 구조



Counter2

```
public class Counter2 {  
    private int count, my_index;  
    public Counter2(int start, int index) {  
        count = start; my_index = index;  
    }  
    public void increment() { count++; }  
    public int countOf() { return count; }  
    public String toString() {  
        return "Counter " + my_index + " has " + countOf();  
    }  
}
```

ListButton

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

public class ListButton extends JButton implements ActionListener {
    private Counter2[] counters;
    private ListFrame view;

    public ListButton(String label, Counter2[] c, ListFrame v) {
        super(label);
        counters = c; view = v;
        addActionListener(this);
    }

    public void actionPerformed(ActionEvent evt) {
        int choice = view.getSelection();
        if(choice != -1) {
            counters[choice].increment();
            view.update();
        }
    }
}
```

ListFrame

```
import java.awt.*; import javax.swing.*;
public class ListFrame extends JFrame {
    private Counter2[] counters;
    private JList items;
    public ListFrame(Counter2[] model) {
        counters = model;
        items = new JList(counters);
        JScrollPane sp = new JScrollPane(items);
        JPanel p = new JPanel(new GridLayout(2,1));
        p.add(new ListButton("Go", counters, this));
        p.add(new ExitButton("Quit"));
        Container cp = getContentPane();
        cp.setLayout(new GridLayout(2,1));
        cp.add(sp); cp.add(p);
        update();
        setTitle("ListExample"); setSize(200,200); setVisible(true);
    }
    public int getSelection() { return items.getSelectedIndex(); }
    public void update() { items.clearSelection(); }
}
```

JList의 더 많은 기능

- 선택될 때 무슨 일을 하고 싶다.
 - `items.addListSelectionListener(ListSelectionListener);`
 - `valueChanged` 메소드가 있어야 한다.
- 여러 개 선택 가능하게 하고 싶다.
 - `items.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION)`
- 선택된 여러 개를 알고 싶다.
 - `items.getSelectedIndices(): int[]`

JTextField

- 사용자가 텍스트를 입력할 수 있는 칸
- `input_text = new JTextField("초기값", 칸 수)`
- `input_text.getText()`: 최종적으로 입력된 문자열
- `input_text.setText("...")`: 칸에 문자열을 바꿈

JTextArea

- JTextField와 같은 입력 칸이지만 크기가 큰 것

```
JTextArea text = new JTextArea("", 20, 40);  
text.setLineWrap(true);  
text.setFont(new Font("Courier", Font.PLAIN, 14));  
JScrollPane sp = new JScrollPane(text);
```


JTextArea의 메소드들

- JComponent
 - getText(): String, setText(String)
 - getCaretPosition(): int, setCaretPosition(int), moveCaretPosition(int)
 - getSelectedText(): String, getSelectionStart(): int, getSelectionEnd(): int
 - cut(), copy(), paste()
 - isEditable(): boolean, setEditable(boolean)
- JTextArea
 - setFont(Font)
 - setLineWrap(boolean)
 - insert(String, int)
 - replaceRange(String, int, int)

JMenu & JMenuBar

```
JMenuBar mbar = new JMenuBar();
JMenu file = new JMenu("File");
mbar.add(file);
JMenu edit = new JMenu("Edit");
    edit.add(new JMenuItem("Cut"));
    edit.add(new JMenuItem("Copy"));
    edit.add(new JMenuItem("Paste"));
    edit.addSeparator();
    JMenu search = new JMenu("Search");
    ...
    edit.add(search);
mbar.add(edit);
setJMenuBar(mbar);
```

메뉴가 선택되었을 때 할 일은 단추의 경우와 같이 등록할 수 있다.

예제, 텍스트 편집기



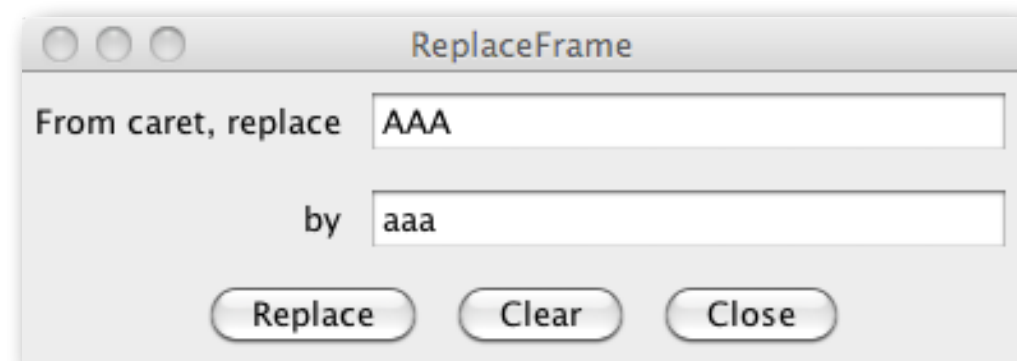
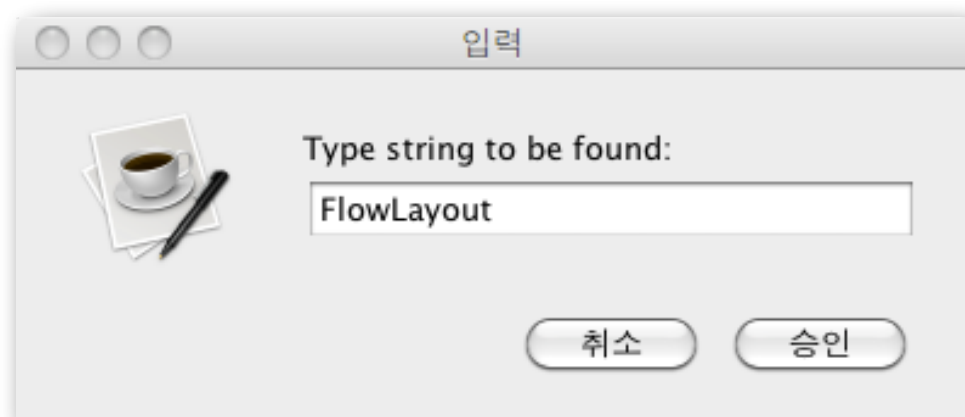
```
import java.awt.*; import java.awt.event.*; import javax.swing.*;

public class ReplaceFrame extends JFrame implements ActionListener {
    private EditModel model;
    private JButton replace = new JButton("Replace");
    private JButton clear = new JButton("Clear");
    private JButton close = new JButton("Close");
    private JTextField find_text = new JTextField("", 20);
    private JTextField replace_text = new JTextField("", 20);
    public ReplaceFrame(EditModel m) {
        model = m;
        Container cp = getContentPane();
        cp.setLayout(new BorderLayout());
        JPanel p1 = new JPanel(new GridLayout(2, 1));
        JPanel p11 = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        p11.add(new JLabel("From caret, replace "));
        p11.add(find_text);
        p1.add(p11);
        JPanel p12 = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        p12.add(new JLabel("by "));
        p12.add(replace_text);
        cp.add(p1, BorderLayout.NORTH);
        cp.add(p12, BorderLayout.SOUTH);
        cp.add(replace, BorderLayout.EAST);
        cp.add(clear, BorderLayout.EAST);
        cp.add(close, BorderLayout.EAST);
    }
}
```

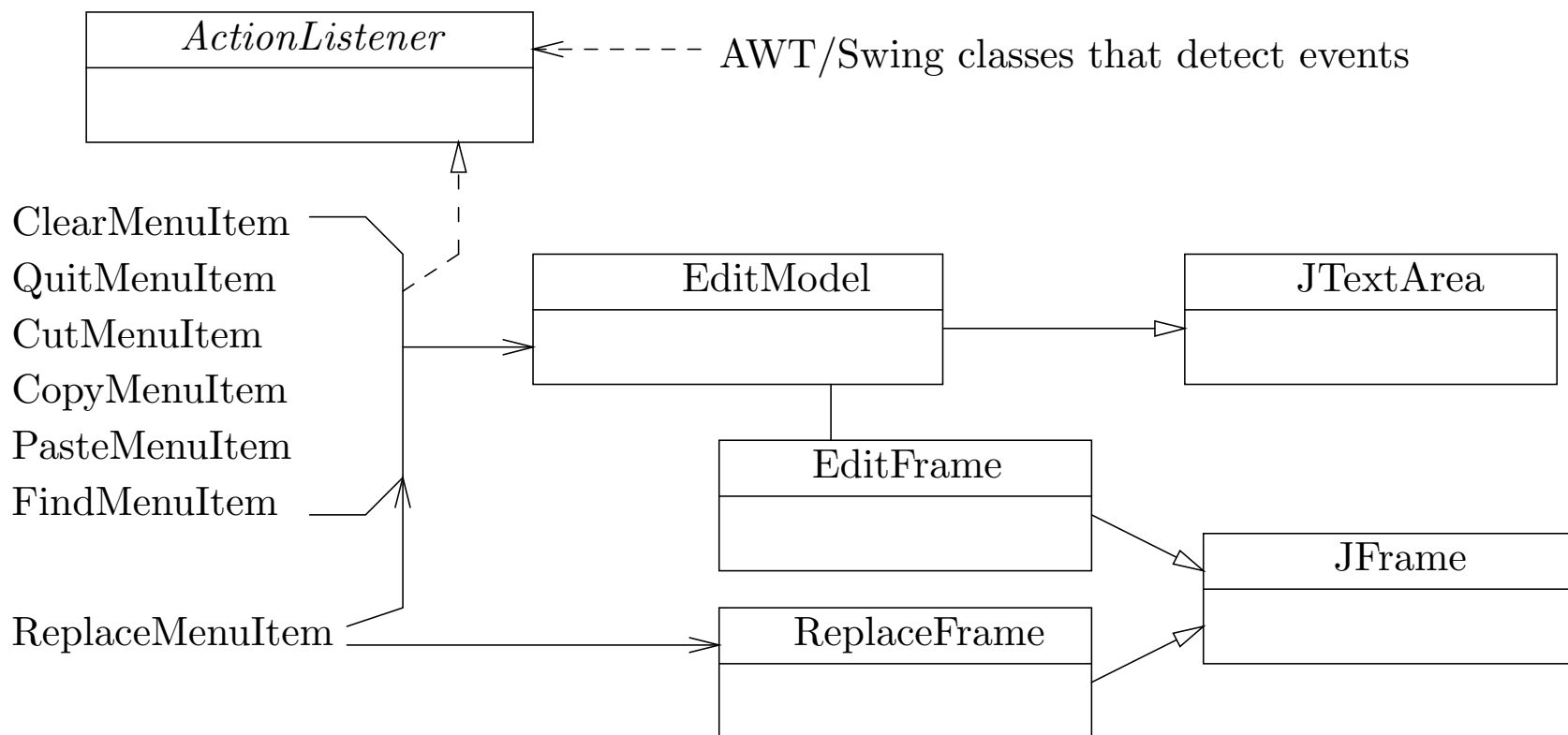
예제, 텍스트 편집기



예제, 텍스트 편집기



간략한 클래스 구조도



EditModel

```
import java.awt.*; import javax.swing.*;
public class EditModel extends JTextArea {
    public EditModel(String initial_text, int rows, int cols) {
        super(initial_text, rows, cols);
        setLineWrap(true); setFont(new Font("Courier", Font.PLAIN, 14));
    }
    public void clear() { setText(""); }
    private int find(String s, int position) {
        int index = getText().indexOf(s, position);
        if(index != -1) {
            setCaretPosition(index + s.length());
            moveCaretPosition(index);
        }
        return index;
    }
    public int findFromStart(String s) { return find(s, 0); }
    public int findFromCaret(String s) { return find(s, getCaretPosition()); }
}
```

QuitMenuItem

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

public class QuitMenuItem extends JMenuItem implements
ActionListener {
    public QuitMenuItem(String label) {
        super(label);
        addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
}
```


추상클래스: EditorMenuItem

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

public abstract class EditorMenuItem extends JMenuItem
implements ActionListener {
    private EditModel buffer;
    public EditorMenuItem(String label, EditModel model) {
        super(label);
        buffer = model;
        addActionListener(this);
    }
    public EditModel myModel() { return buffer; }
    public abstract void actionPerformed(ActionEvent e);
}
```

ClearMenuItem

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

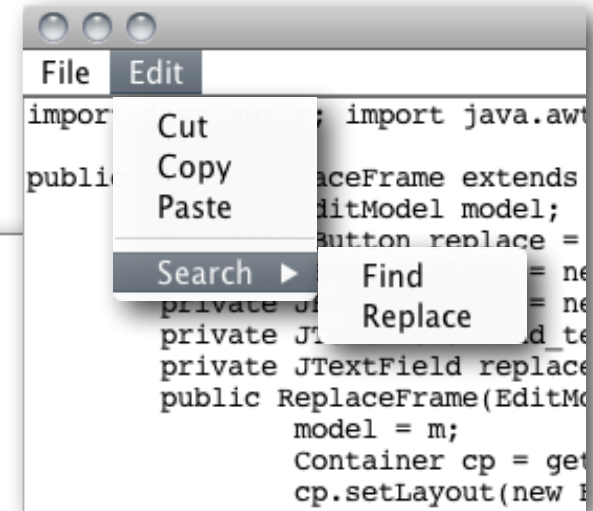
public class ClearMenuItem extends EditorMenuItem {
    public ClearMenuItem(String label, EditModel model) {
        super(label, model);
    }
    public void actionPerformed(ActionEvent e) {
        myModel().clear();
    }
}
```

cut/copy/pasteMenuItem도 유사하게 구현 가능

EditFrame

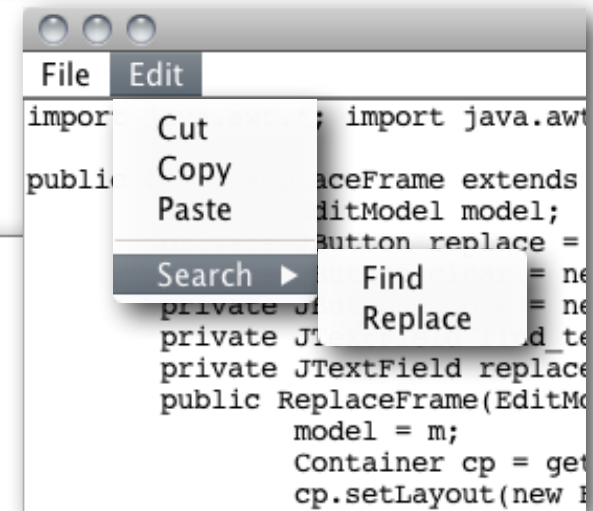
```
import java.awt.*; import javax.swing.*;

public class EditFrame extends JFrame {
    private EditModel buffer = new EditModel("", 15, 50);
    public EditFrame() {
        // ReplaceFrame second_frame = new ReplaceFrame(buffer);
        Container cp = getContentPane();
        cp.setLayout(new BorderLayout());
        JMenuBar mbar = new JMenuBar();
        JMenu file = new JMenu("File");
        file.add(new ClearMenuItem("New", buffer));
        file.add(new QuitMenuItem("Exit"));
        mbar.add(file);
        JMenu edit = new JMenu("Edit");
        edit.add(new CutMenuItem("Cut", buffer));
        edit.add(new CopyMenuItem("Copy", buffer));
        edit.add(new PasteMenuItem("Paste", buffer));
        edit.addSeparator();
    }
}
```



ClearMenuItem

```
// JMenu search = new JMenu("Search");
// search.add(new FindMenuItem("Find", buffer));
// search.add(new ReplaceMenuItem("Replace",
//                                second_frame));
// edit.add(search);
mbar.add(edit);
setJMenuBar(mbar);
JScrollPane sp = new JScrollPane(buffer);
cp.add(sp, BorderLayout.CENTER);
setTitle("EditFrame"); pack(); setVisible(true);
}
public static void main(String[] args) {
    new EditFrame();
}
}
```



FindMenuItem

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

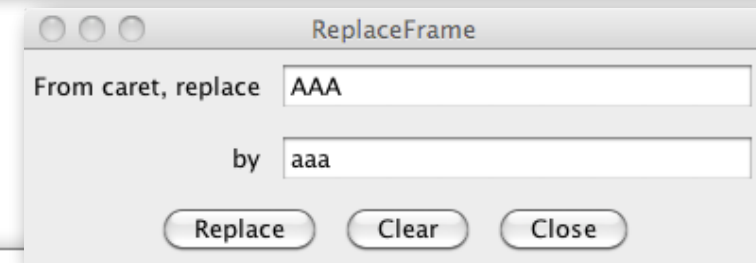
public class FindMenuItem extends EditorMenuItem {
    public FindMenuItem(String label, EditModel model) {
        super(label, model);
    }
    public void actionPerformed(ActionEvent e) {
        String s = JOptionPane.showInputDialog(this, "Type string to be found:");
        if(s != null) {
            if(myModel().findFromCaret(s) == -1) {
                int response = JOptionPane.showConfirmDialog(this,
                    "String " + s + " not found. Restart search from beginning of
                    buffer?");
                if(response == JOptionPane.YES_OPTION) {
                    if(myModel().findFromStart(s) == -1)
                        JOptionPane.showMessageDialog(this,
                            "String " + s + " not found.");
                }
            }
        }
    }
}
```

ReplaceMenuItem

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

public class ReplaceMenuItem extends JMenuItem implements
ActionListener {
    private ReplaceFrame view;
    public ReplaceMenuItem(String label, ReplaceFrame v) {
        super(label);
        view = v;
        addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        view.setVisible(true);
    }
}
```

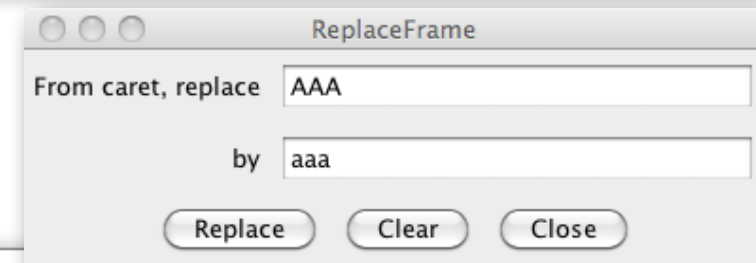
ReplaceFrame



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

public class ReplaceFrame extends JFrame implements ActionListener {
    private EditModel model;
    private JButton replace = new JButton("Replace");
    private JButton clear = new JButton("Clear");
    private JButton close = new JButton("Close");
    private JTextField find_text = new JTextField("", 20);
    private JTextField replace_text = new JTextField("", 20);
    public ReplaceFrame(EditModel m) {
        model = m;
        Container cp = getContentPane();
        cp.setLayout(new BorderLayout());
        JPanel p1 = new JPanel(new GridLayout(2, 1));
        JPanel p11 = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        p11.add(new JLabel("From caret, replace "));
        p11.add(find_text);
        p1.add(p11);
    }
}
```

ReplaceFrame



```
JPanel p12 = new JPanel(new FlowLayout(FlowLayout.RIGHT));
p12.add(new JLabel("by "));
p12.add(replace_text);
p1.add(p12);
cp.add(p1, BorderLayout.CENTER);
JPanel p2 = new JPanel(new FlowLayout());
p2.add(replace); p2.add(clear); p2.add(close);
cp.add(p2, BorderLayout.SOUTH);
replace.addActionListener(this);
clear.addActionListener(this);
close.addActionListener(this);
setTitle("ReplaceFrame"); pack();
setVisible(false);
}
```


ReplaceFrame

```
public void actionPerformed(ActionEvent e) {  
    if(e.getSource() == close) {  
        setVisible(false);  
    }  
    else if(e.getSource() == clear) {  
        find_text.setText("");  
        replace_text.setText("");  
    }  
    else if(e.getSource() == replace) {  
        String find = find_text.getText();  
        int location = model.findFromCaret(find);  
        if(location == -1)  
            JOptionPane.showMessageDialog(this,  
                "String " + find + " not found.");  
        else  
            model.replaceRange(replace_text.getText(), location,  
                location+find.length());  
    }  
}
```