[lab09 보고서]

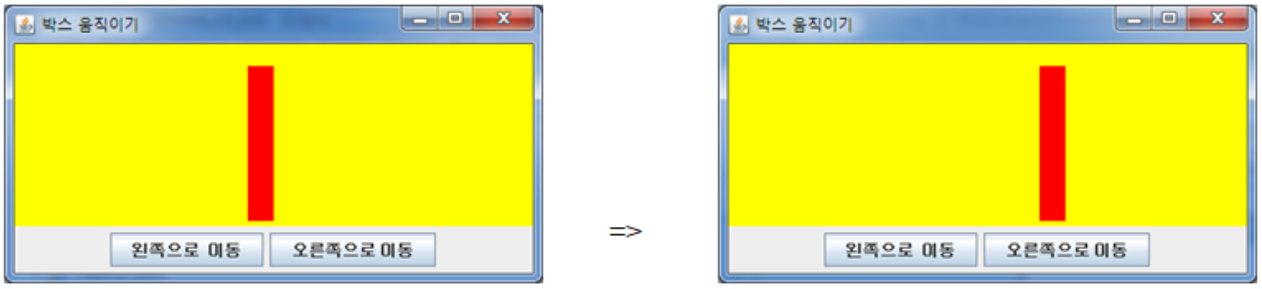
디지털미디어학과

2019111677

김지연

(실습문제)

1. lab08의 다음 프로그램에 아래 지시와 같은 이벤트 처리를 추가하세요.



① 버튼 클릭으로 좌우 이동

② 마우스 드레그로 좌우 이동

③ 키보드의 화살표 키로 좌우 이동

[분석]

버튼을 누르면 막대가 움직이는 프로그램

- 화면 디자인: 패널 2개를 생성하여 BorderLayout으로 panel1은 CENTER에 panel2는 SOUTH에 배치한다. 패널1에는 막대기(도형)를 중앙에 배치한다. 패널1을 MyPanel로 생성하고 MyPanel에 JPanel을 상속받도록 하여 패널1에 도형을 생성한다. JPanel에 있는 기능만으로는 구현이 힘들기 때문에 MyPanel에 JPanel을 상속시켜 새로운 기능을 추가하여 사용하기 위함이다. 도형은 paint메소드를 활용하여 색과 위치를 지정해준다. 위치는 x와 y를 정적 매개변수로 선언하여 지정해준다. 패널2에는 도형을 왼쪽으로 움직이게 하는 버튼 1개와 오른쪽으로 움직이게 하는 버튼 1개를 생성한다. 버튼에는 이벤트 리스너를 등록하여 버튼을 누를 때마다 막대기가 움직이도록 한다.

- 이벤트 처리

① 버튼 클릭으로 좌우 이동: 액션 리스너 클래스를 무명 클래스로 정의한다. Button1은 왼쪽으로 이동하는 버튼 이므로 button1이 눌린다면 x값을 -20만큼 움직이게 하고, repaint()함수를 이용하여 그 위치에 막대기를 다시 그린다. Button2도 마찬가지로 오른쪽으로 이동하는 버튼 이므로 button2가 눌린다면 x값을 +20만큼 움직이게 하고, repaint()함수를 이용하여 막대기를 다시 그린다.

② 마우스 드레그로 좌우 이동: MouseListener와 MouseMotionListener를 무명 클래스로 정의하여 패널에 추가한다. MouseListener의 mouseReleased함수는 마우스에서 손을 떼었을 때 repaint() 함수를 이용하여 그 위치에 막대기를 다시 그리기 위해 사용한다. mouseDragged는 드레그 된 위치의 x값을 막대기의 x값에 대입하여 이동하게 하기 위해 사용한다.

③ 키보드의 화살표 키로 좌우 이동: KeyListener를 무명클래스로 정의하여 패널에 추가한다. keyPressed함수에서 왼쪽 화살표 키보드가 눌리면 -20, 오른쪽 화살표 키보드가 눌리면 +20만큼 이동하도록 한다. Repaint()함수를 이용하여 막대기와 패널을 다시 그린다. 다음으로 키보드 포커스를 요청하여 패널이 포커스를 받을 수 있게 한다.

Move생성자를 호출한다.

[소스코드]

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


public class Move extends JFrame {
	static int x = 330, y = 40;
	JPanel panel1 = new MyPanel();
	JPanel panel2 = new JPanel();
	JButton button1, button2;

	class MyPanel extends JPanel{
		public void paint(Graphics g) {
			super.paint(g);
			g.setColor(Color.red);
			g.fillRect(x,y,30,200);
		}
	}

	public Move(){
		setSize(700,330);
		setTitle("박스 움직이기");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setLayout(new BorderLayout());

		panel1.setSize(700,300);
		panel1.setBackground(Color.yellow);
		panel2.setSize(700,100);

		button1 = new JButton("왼쪽으로 이동");
		button2 = new JButton("오른쪽으로 이동");
		button1.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				if(e.getSource() == button1) {
					x -= 20;
					repaint();
				}
			}
		});
		button2.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				if(e.getSource() == button2) {
					x += 20;
					repaint();
				}
```

```
                }
        });



        addKeyListener(new KeyListener() {
                public void keyPressed(KeyEvent e) {
                        int keycode = e.getKeyCode();
                        switch (keycode) {
                        case KeyEvent.VK_LEFT:  x -= 20; break;
                        case KeyEvent.VK_RIGHT: x += 20; break;
                        }
                        repaint();
                }
                public void keyTyped(KeyEvent e) {}
                public void keyReleased(KeyEvent e) {}
        });
        this.requestFocus();
        setFocusable(true);



        addMouseListener(new MouseListener() {
                public void mousePressed(MouseEvent e) {}
                public void mouseReleased(MouseEvent e) {
                        repaint();
                }
                public void mouseEntered(MouseEvent e) {}
                public void mouseExited(MouseEvent e) {}
                public void mouseClicked(MouseEvent e) {}
        });
        addMouseMotionListener(new MouseMotionListener() {

                public void mouseDragged(MouseEvent e) {
                        x = e.getX();
                }
                public void mouseMoved(MouseEvent e) {}
        });


        panel2.add(button1);
        panel2.add(button2);
        add(panel1, BorderLayout.CENTER);
        add(panel2,BorderLayout.SOUTH);
        setVisible(true);
    }

    public static void main(String[] args) {
        new Move();
    }
}
```

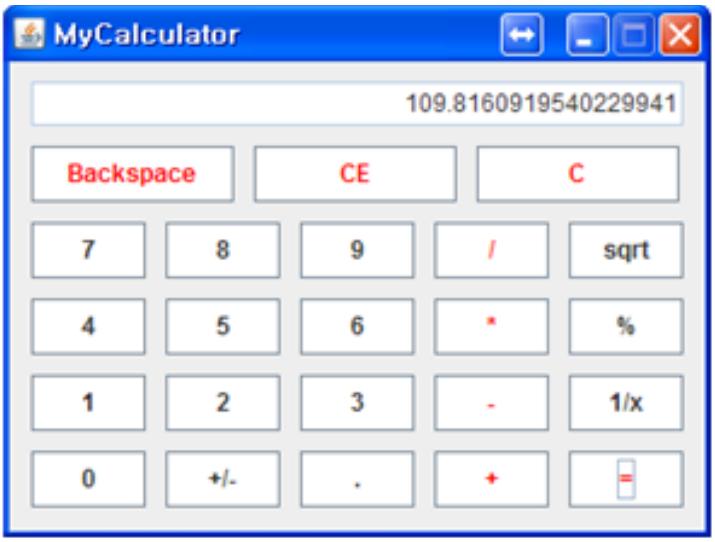
[결과화면]

2. 아래 그림을 참고해서 계산기를 구현해 보세요.

- 우선 화면 디자인을 완성한 후에 계산기 기능의 일부분만 성공하더라도 시도해 보세요.

- 보고서 제출 시, [1. 분석] 부분에 완성도를 표시하세요.

(전체 기능의 절반을 구현한 경우, "완성도 : 50%"로 표시)



[분석] 완성도 40%

계산기를 구현하는 프로그래밍

- 화면 디자인: 전체적인 디자인은 텍스트필드를 상단에 배치하고, 가로로 길이가 긴 버튼 3개를 panelA에 그리드 레이아웃으로 추가하여 중앙에 배치하고, 나머지 키패드들은 panelB에 그리드 레이아웃으로 추가하여 하단에 배치한다. 패널2개, 텍스트 필드 1개, 버튼 배열 2개, 레이블 배열 2개를 생성한다. 텍스트 필드는 25정도로 생성한다. panelA에는 button1과 labels1을 그리드 레이아웃으로 추가한다. 버튼 3개에 label1을 for문을 이용하여 대입한다. panelB에서도 마찬가지로 버튼 20개에 labels2를 for문을 이용하여 대입하고 패널에 추가한다. 패널과 텍스트필드를 BorderLayout을 사용하여 각각 NORTH, SOUTH, CENTER에 배치한다.

- 이벤트 처리: calculator 클래스는 JFrame을 상속받고 ActionListener를 implements하여 actionPerformed를 멤버함수로 직접 오버라이딩 한다. Button2에 현재 객체를 이벤트 리스너로 등록한다. actionPerformed메소드 안에 버튼을 눌렀을 때 일어나는 이벤트를 구현한다. 버튼을 누르면 그 버튼의 텍스트를 tField에 추가한다. 텍스트 필드에 사용자가 누른 버튼의 레이블이 보이게 된다.

구현 못한 점 – 연산자들의 계산을 구현하지 못하였다.

+버튼을 누르면 연산자를 기준으로 앞에 입력 받은 값을 tempString 변수에 저장하고, 문자열로 저장되었기 때문에 tempInt에 정수형으로 형변환 하여 저장한다. 저장한 값을 total 변수에 저장한다. = 버튼을 누르면 연산자 기준 뒤의 숫자를 같은 방식으로 tempInt에 저장하고 total과 tempInt를 더하여 total에 저장한다. Total을 문자열로 변환하여 printScrin에 저장하고 화면에 출력한다.

생각처럼 되지 않았다.

[소스코드]

```
import java.awt.*;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class Calculator extends JFrame implements ActionListener {



	private JPanel panelA;
	private JPanel panelB;
	private JTextField tField;
	private JButton[] button1;
	private String[] labels1 = {
			"Backspace", "CE", "C"
	};
	private JButton[] button2;
	private String[] labels2 = {
			"7", "8", "9", "/", "sqrt",
			"4", "5", "6", "*", "%",
			"1", "2", "3", "-", "1/x",
			"0", "+/-", ".", "+", "="
```

```
	};

	public Calculator() {
		setTitle("계산기");

		tField = new JTextField(25);
		panelA = new JPanel();
		panelB = new JPanel();

		panelA.setLayout(new GridLayout(0,3,8,8 ));
		button1 = new JButton[3];
		for (int rows = 0; rows < 3; rows++) {
			button1[rows] = new JButton(labels1[rows]);
			button1[rows].setForeground(Color.RED);
			panelA.add(button1[rows]);
		}


		panelB.setLayout(new GridLayout(0,5, 8, 8));
		button2 = new JButton[20];
		for(int rows = 0; rows < 20; rows++) {
			button2[rows] = new JButton(labels2[rows]);
			button2[rows].setForeground(Color.BLACK);
			panelB.add(button2[rows]);
			button2[rows].addActionListener(this);
		}

		setLayout(new BorderLayout(7,7));
		add(tField,BorderLayout.NORTH);
		add(panelA,BorderLayout.CENTER);
		add(panelB,BorderLayout.SOUTH);
		setVisible(true);
		pack();

	}

	/*
	  String printScrin = ""; //화면에 값을 출력하는 변수
	  int total = 0; //최종값을 저장할 변수
	  String tempStirng = ""; //입력한 값을 가질 변수
	  int tempInt; //입력한 값을 정수로 변환하여 가질 변수
	  int operate = 0;
	  */
	public void actionPerformed(ActionEvent e) {

		String tButton = e.getActionCommand();
		tField.setText(tField.getText()+tButton);

		/*
		if(e.getSource() == button2[18]) {//+을 누르면 이전에 입력한 값이
정수로 저장됨
				tempInt = Integer.parseInt(tempStirng);
				total = total + tempInt;
				tempStirng = "";
				operate = 1;
```

```
                        }

                if(e.getSource() == button2[19]) { //=을 누르면 + 계산, -*/추후 추가
                        tempInt = Integer.parseInt(tempStirng);
                           switch(operate){
                           case 1 : total = total + tempInt; break;
                           }
                           printScrin = String.valueOf(total);
                           tField.setText(printScrin);
                           total = 0;
                }
                */
        }


        public static void main(String[] args) {
                new Calculator();
        }


}
```

[결과화면]