객체지향프로그래밍응용 과제12 컴퓨터소프트웨어공학과 2-YA 20202296 전채린

[과제01]

01. ThreadTimerEx.java

Thread를 상속받아 1초 단위로 출력하는 타이머 스레드 응용프로그램을 작성하시오.

[소스코드]

// 예제 13-1

// Thread를 상속받아 1초 단위로 출력하는 타이머 스레드 만들기

// 1초 단위로 증가하는 스레드를 만들고 레이블을 이용하여 타이머 값을 출력한다.

// 응용프로그램의 시작과 함께 타이머가 바로 작동한다.

// 타이머 값은 JLabel 컴포넌트를 생성하여 출력하고, Thread 클래스를 상속받는 TimerThread 클래스로 스레드를 만든다.

// 이 스레드는 1초마다 1씩 증가시킨 정수 값을 JLabel의 텍스트에 출력한다.

import java.awt.\*;

import javax.swing.\*;

class TimerThread extends Thread {

private JLabel timerLabel; // 타이머 값이 출력되는 레이블

public TimerThread(JLabel timerLabel) {

this.timerLabel = timerLabel; // 타이머 카운트를 출력할 레이블

}

// 스레드 코드. run()이 종료하면 스레드 종료

public void run() { // run()은 스레드 코드로서 start() 메소드가 호출된 후 스레드가 실행을 시작하는 메소드이다.

int n=0; // 타이머 카운트 값

while(true) { // 무한 루프

timerLabel.setText(Integer.toString(n));

n++; // 카운트 증가

try {

Thread.sleep(1000); // 1초 동안 잠을 잔다.

}

catch(InterruptedException e) {

return; // 예외가 발생하면 스레드 종료

}

}

}

}

public class ThreadTimerEx extends JFrame {

public ThreadTimerEx() {

setTitle("Thread를 상속받은 타이머 스레드 예제");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Container c = getContentPane();

c.setLayout(new FlowLayout());

// 타이머 값을 출력할 레이블 생성

JLabel timerLabel = new JLabel();

timerLabel.setFont(new Font("Gothic", Font.ITALIC, 80));

c.add(timerLabel); // 레이블을 컨텐트팬에 부착

// 타이머 스레드 객체 생성. 타이머 값을 출력할 레이블을 생성자에게 전달

TimerThread th = new TimerThread(timerLabel); // 스레드 객체를 만든다.

setSize(300, 170);

setVisible(true);

th.start(); // 스레드를 동작시킨다. 이 호출의 결과 TimerThread의 run() 메소드가 실행을 시작한다.

}

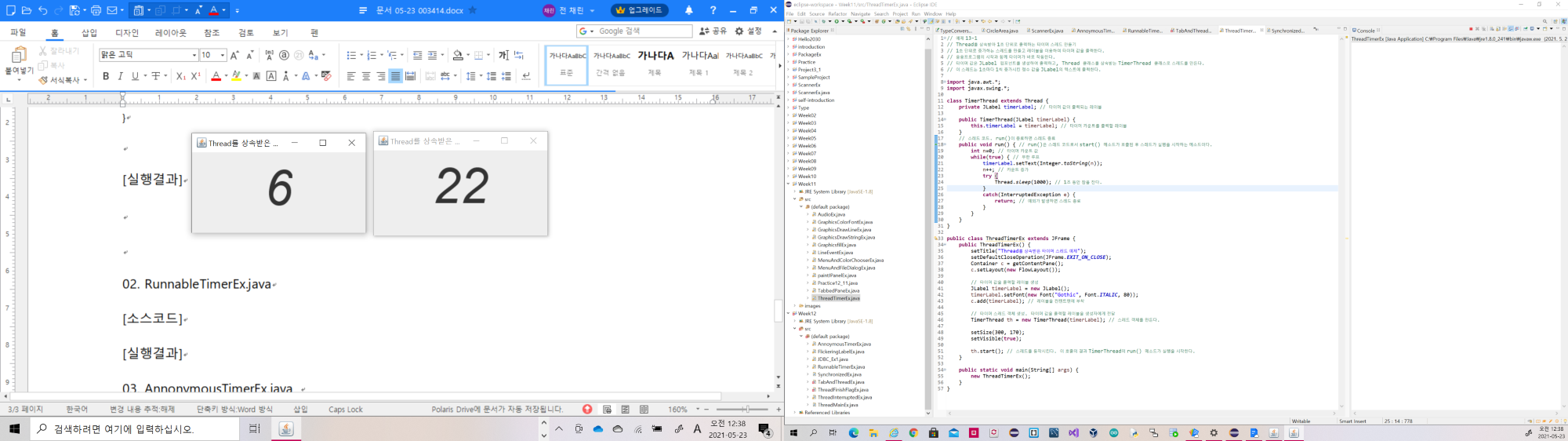
public static void main(String[] args) {

new ThreadTimerEx();

}

}

[실행결과]



02. RunnableTimerEx.java

Runnable 인터페이스를 구현하여 1초 단위로 출력하는 타이머 스레드 응용프로그램을 작성하시오.

[소스코드]

// 예제 13-2

// Runnable 인터페이스를 구현하여 1초 단위로 출력하는 타이머 스레드 만들기

// Runnable 인터페이스를 구현하여 다음 그림과 같이 1초 단위로 카운팅하여 출력하는 스레드를 만들고, 레이블을 이용하여 카운트 값을 출력한다.

import java.awt.\*;

import javax.swing.\*;

class TimerRunnable implements Runnable {

private JLabel timerLabel; // 타이머 값이 출력되는 레이블

public TimerRunnable(JLabel timerLabel) {

this.timerLabel = timerLabel; // 초 카운트를 출력할 레이블

}

// 스레드 코드. run()이 종료하면 스레드 종료

public void run() { // run()은 스레드 코드로서 start() 메소드가 호출된 후 스레드가 실행을 시작하는 메소드이다.

int n=0; // 타이머 카운트 값

while(true) { // 무한 루프

timerLabel.setText(Integer.toString(n)); // 레이블에 카운트 값 출력

n++; // 카운트 증가

try { // try-catch 블록이 없으면 컴파일 오류가 발생한다. sleep()에 의해 잠을 자는 경우 예외 발생에 대비하기 위해서이다.

Thread.sleep(1000); // 1초 동안 잠을 잔다.

}

catch(InterruptedException e) {

return; // 예외가 발생하면 스레드 종료

}

}

}

}

public class RunnableTimerEx extends JFrame {

public RunnableTimerEx() {

setTitle("Runnable을 구현한 타이머 스레드 예제");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Container c = getContentPane();

c.setLayout(new FlowLayout());

// 타이머 값을 출력할 레이블 생성

JLabel timerLabel = new JLabel();

timerLabel.setFont(new Font("Gothic", Font.ITALIC, 80));

timerLabel.setForeground(Color.GREEN);

c.add(timerLabel); // 레이블을 컨텐트팬에 부착

// 타이머 스레드로 사용할 Runnable 객체 생성. 타이머 값을 출력할 레이블을 생성자에게 전달

TimerRunnable runnable = new TimerRunnable(timerLabel); // 스레드 코드로 작동할 run()이 구현된 Runnable 객체를 만든다.

Thread th = new Thread(runnable); // 스레드 객체 생성

setSize(250, 150);

setVisible(true);

th.start(); // 스레드를 동작시킨다. 이 호출의 결과 TimerRunnable의 run() 메소드가 실행을 시작한다.

}

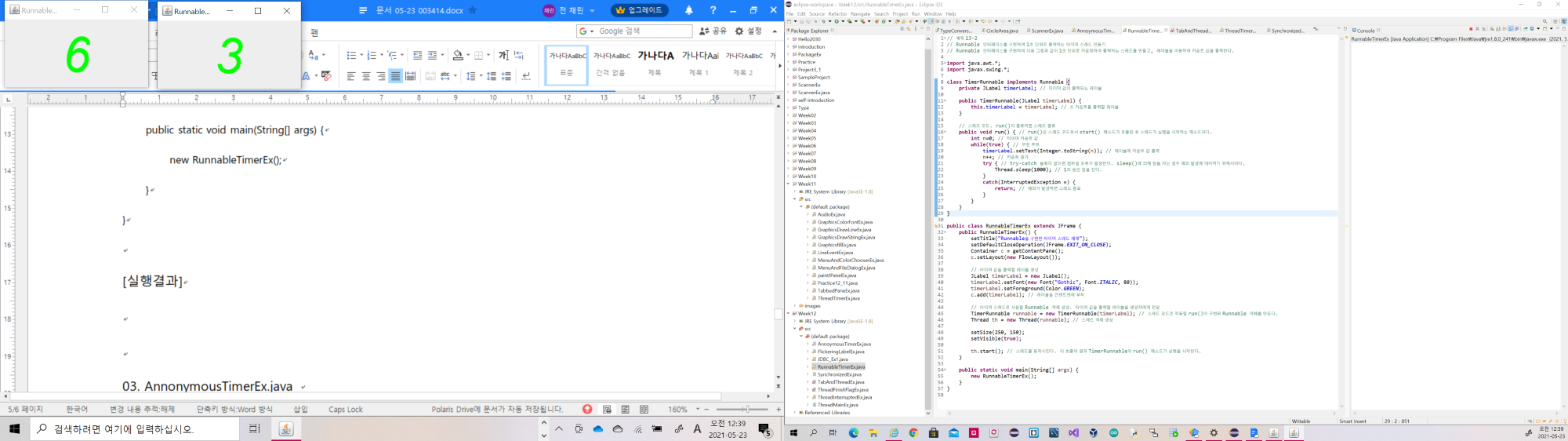
public static void main(String[] args) {

new RunnableTimerEx();

}

}

[실행결과]



03. AnnonymousTimerEx.java

익명 클래스로 1초 단위로 출력하는 타이머 스레드 응용프로그램을 작성하시오.

[소스코드]

// 익명 클래스로 1초 단위로 출력하는 타이머 스레드 만들기

// 익명 클래스로 1초 단위로 카운팅하여 출력하는 스레드를 만들고, 레이블을 이용하여 카운트 값을 출력한다.

import java.awt.\*;

import javax.swing.\*;

public class AnnoymousTimerEx extends JFrame {

static JLabel timerLabel;

public AnnoymousTimerEx() {

setTitle("익명 클래스로 구현한 타이머 스레드 예제");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Container c = getContentPane();

c.setLayout(new FlowLayout());

// 타이머 값을 출력할 레이블 생성

timerLabel = new JLabel();

timerLabel.setFont(new Font("Gothic", Font.ITALIC, 80));

timerLabel.setForeground(Color.RED);

c.add(timerLabel); // 레이블을 컨텐트팬에 부착

setSize(300, 170);

setVisible(true);

}

public static void main(String[] args) {

new AnnoymousTimerEx();

Thread th = new Thread(new Runnable() {

@Override

public void run() { // run()은 스레드 코드로서 start() 메소드가 호출된 후 스레드가 실행을 시작하는 메소드이다.

int n=0; // 타이머 카운트 값

while(true) { // 무한 루프

timerLabel.setText(Integer.toString(n)); // 레이블에 카운트 값 출력

n++; // 카운트 증가

try { // try-catch 블록이 없으면 컴파일 오류가 발생한다. sleep()에 의해 잠을 자는 경우 예외 발생에 대비하기 위해서이다.

Thread.sleep(1000); // 1초 동안 잠을 잔다.

}

catch(InterruptedException e) {

return; // 예외가 발생하면 스레드 종료

}

}

}

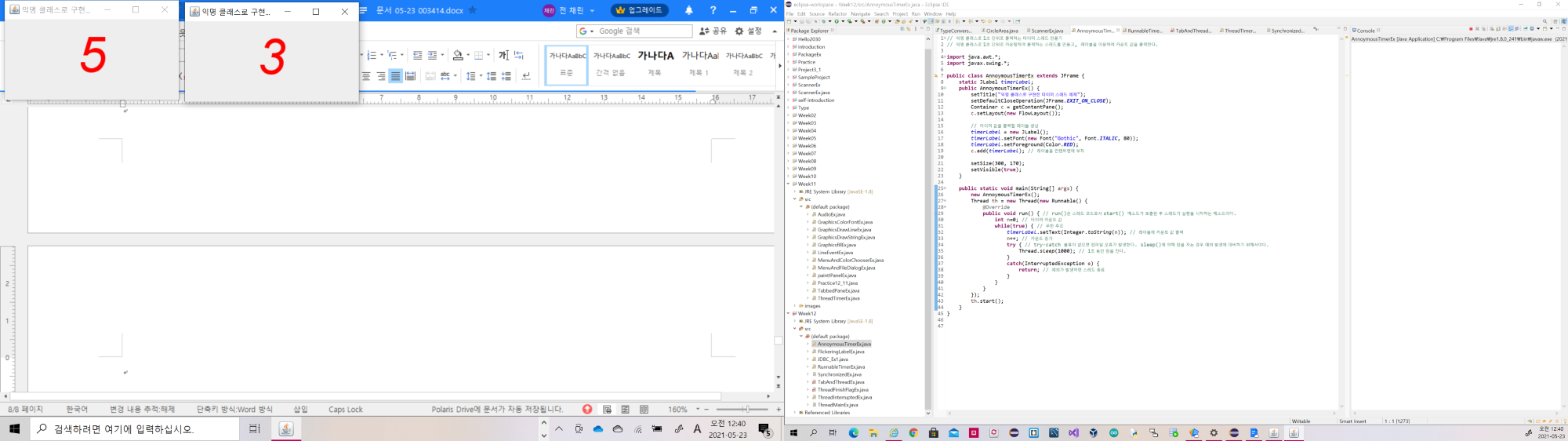
});

th.start();

}

}

[실행결과]



04. SynchronizedEx.java

SynchronizedEx를 사용하여 잘 동작하는 집계판 응용프로그램을 작성하시오.

[소스코드]

// SynchronizedEx를 사용하여 잘 동작하는 집계판 응용

// // 먼저 [그림 13-16]의 코드는 synchronized를 사용한 예이다.

// SharedBoard의 멤버 add()는 집계판의 이전 값을 읽고 10을 더하여 기록하는 기능을 구현하고 synchronized 메소드로 지정하였다.

// 이것은 add()가 집계 합을 나타내는 공유 변수 sum에 접근하고 있으며,

// add()가 집계 합을 나타내는 공유 변수 sum에 접근하고 있으며,

// add()는 여러 StudentThread 스레드(학생들)가 동시에 호출할 수 있는 임계 영역이기 때문이다.

// main()은 이름이 "kitae", "hyosoo"인 2개의 StudentThread 스레드를 생성하며,

// 스레드는 루프를 돌면서 SharedBoard의 add() 메소드를 10번 호출한다.

public class SynchronizedEx {

public static void main(String[] args) {

SharedBoard board = new SharedBoard(); // 집계판 공유 데이터 생성

// 스레드 생성 시 집계판의 주소를 알려준다. 두 스레드는 집계판에 동시에 접근한다.

Thread th1 = new StudentThread("kitae", board); // "kitae" 이름의 스레드 생성

Thread th2 = new StudentThread("hyosoo", board); // "hyosoo" 이름의 스레드 생성

// 두 스레드를 실행시킨다.

th1.start();

th2.start();

}

}

// 공유데이타인 집계판을 시뮬레이션하는 클래스

// 두 StudentThread 스레드에 의해 동시 접근됨

class SharedBoard {

private int sum = 0; // 집계판의 합

synchronized public void add() {

int n = sum;

Thread.yield(); // 현재 실행 중인 스레드 양보

n += 10; // 10 증가

sum = n; // 증가한 값을 집계합에 기록

System.out.println(Thread.currentThread().getName() + " : " + sum);

}

public int getSum() { return sum; }

}

// 학생을 시뮬레이션하는 스레드 클래스

class StudentThread extends Thread {

private SharedBoard board; // 집계판의 주소

public StudentThread(String name, SharedBoard board) {

super(name);

this.board = board;

}

// 집계판을 10번 접근하여 카운팅한다.

public void run() {

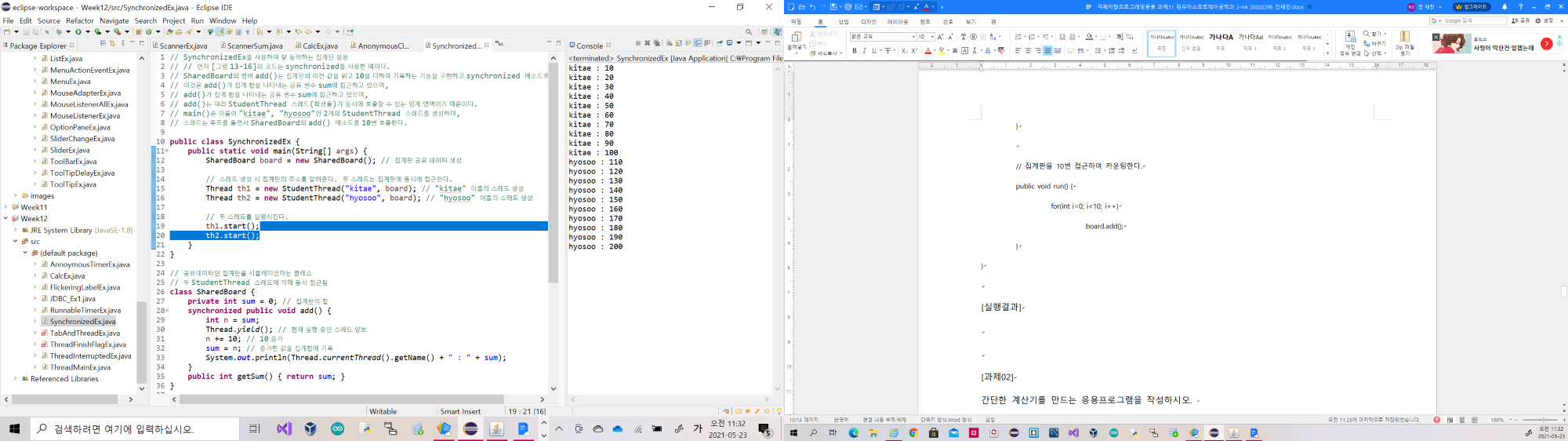
for(int i=0; i<10; i++)

board.add();

}

}

[실행결과]



[과제02]

간단한 계산기를 만드는 응용프로그램을 작성하시오.

[소스코드]

// 간단한 계산기 만들기

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class CalcEx extends JFrame {

private String[] btnName = {"더하기+", "빼기-", "곱하기\*", "나누기/"};

private JButton[] btn = new JButton[4];

public CalcEx() {

setTitle("간단 계산기 예제");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Container c = getContentPane();

c.setLayout(new FlowLayout());

// 학번 이름 레이블

c.add(new JLabel("20202296 전채린"));

// 숫자 입력 패널

JPanel pnIn1 = new JPanel(new FlowLayout(FlowLayout.LEFT));

JTextField tf1 = new JTextField(15);

JPanel pnIn2 = new JPanel(new FlowLayout(FlowLayout.LEFT));

JTextField tf2 = new JTextField(15);

pnIn1.add(new JLabel("숫자1"));

pnIn1.add(tf1);

pnIn2.add(new JLabel("숫자2"));

pnIn2.add(tf2);

c.add(pnIn1);

c.add(pnIn2);

// 버튼 패널

JPanel pnBtn = new JPanel(new GridLayout(0, 1, 5, 5));

// 결과 레이블

JLabel lbResult = new JLabel("결과값 : ");

lbResult.setFont(new Font("Gothic", Font.BOLD, 20));

// 버튼 생성 및 이벤트 처리

for(int i=0; i<btnName.length; i++) {

btn[i] = new JButton(btnName[i]);

pnBtn.add(btn[i]);

// 익명 클래스로 버튼 이벤트 처리

btn[i].addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int num1 = Integer.parseInt(tf1.getText());

int num2 = Integer.parseInt(tf2.getText());

JButton btn = (JButton)e.getSource(); // 이벤트 소스 버튼 알아내기

if(btn.getText().equals("더하기+")) // 버튼의 문자열이 "더하기+"인지 비교

lbResult.setText("결과값 : " + (num1 + num2));

else if(btn.getText().equals("빼기-")) // 버튼의 문자열이 "빼기-"인지 비교

lbResult.setText("결과값 : " + (num1 - num2));

else if(btn.getText().equals("곱하기\*")) // 버튼의 문자열이 "곱하기\*"인지 비교

lbResult.setText("결과값 : " + (num1 \* num2));

else // 버튼의 문자열이 "나누기/"

lbResult.setText("결과값 : " + (num1 / num2));

lbResult.setForeground(Color.RED);

}

});

}

pnBtn.add(lbResult);

c.add(pnBtn);

setSize(250, 400);

setVisible(true);

}

public static void main(String[] args) {

new CalcEx();

}

}

[실행결과]

