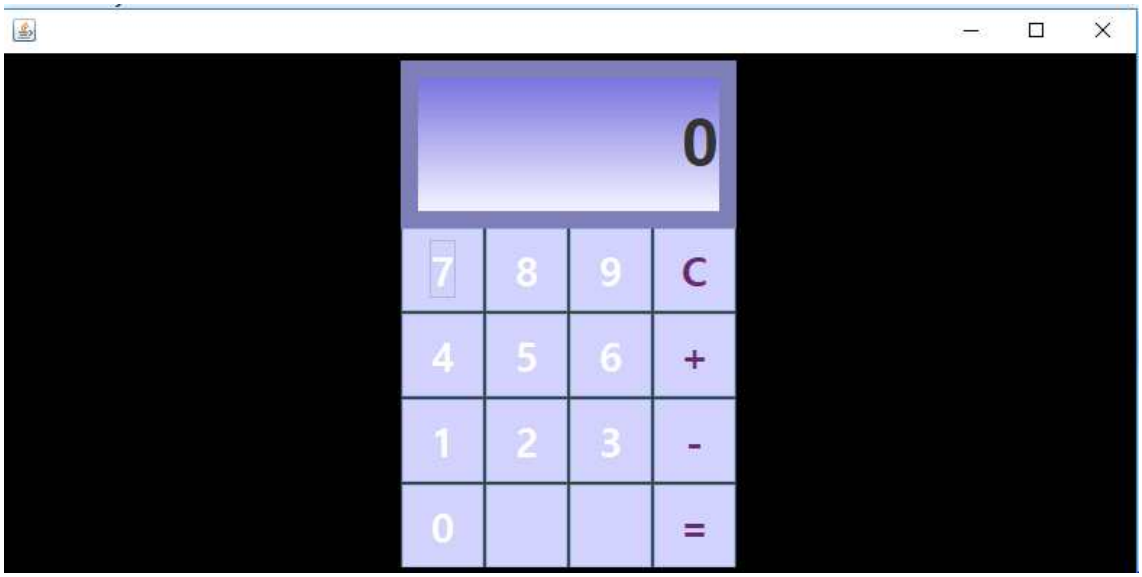
17011653 김지수

<실행 결과>



<보고서>

우선 Swing과 awt의 기능과, awt의 이벤트 기능들을 사용하기 위하여 import를 한다.

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ComponentEvent;
import java.awt.event.ComponentListener;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingConstants;
import javax.swing.border.LineBorder;
import javax.swing.border.TitledBorder;
```

- 전체적인 구조

JFrame을 상속받고 ActionListener인터페이스를 사용하는 클래스 JavaHw4을 생성한다. 이 클래스는 JFrame 위에 4개의 JPanel이 올라가 있는데, 검은색 배경을 가지는 bg 패널을 깔고, 그위에 p1패널을 올린후, p1위에 숫자를 출력하기 위한 p2패널과 버튼을 넣기 위한 p3패널을 만든다.

- 사용된 변수들

```
private JButton[] buttons;
private String[] labels = {"7", "8", "9", "C", "4", "5", "6", "+", "1", "2", "3", "-",
                           "0", " ", " ", " ", "="};
private JLabel tField;
private JPanel bp;
private JPanel p1;
private JPanel p2;
private JPanel p3;
private String tempName;
private String tempString;
private int tempNum1 = 0;
private String tempW1;
private String tempNum2 = "";
private boolean is_ready = false;
private String onText = "";
private String who = ""; // action performed에서 사용할 변수를 내부에 선언하면
                            이벤트 발생마다 초기화가 돼서 문제가 생겼었다.
```

1) JavaHW4() // 생성자

검은색 배경을 가지는 bg 패널과, 계산기를 보여줄 p1패널을 생성한다.
p1패널 위쪽엔 텍스트를, 아래엔 버튼을 넣기 위해 p1에 Border레이아웃을 넣었다.

생성자 Hw4에서는 계산과정과 결과를 출력할 p2패널 위에 JLabel인 tField를 생성하고, 텍스트는 0을 출력하게 한다. 텍스트를 오른쪽에 위치시키기 위해 p2의 레이아웃을 그리드로 설정하고, setHorizontalAlignment()함수를 이용하여 위치시킨후, paintComponent를 오버라이딩 해서 GradientPaint함수를 이용하여 그라데이션 효과를 넣었다.

그리고 계산 버튼을 만들 배치관리자가 GridLayout인 p3을 추가한다.
p3 위에 16개의 버튼(JButton)을 생성하고 버튼마다 버튼을 입력받을 수 있게 액션리스너를 추가한다.

setForeground, setBackground, setBorder 함수 등을 이용하여 계산기글씨의 색과 배경색을 꾸며주고 버튼의 레이블은 String lables에서 받아온다.

add함수를 통해 p2는 NORTH 방향에, p3은 CENTER에 위치시켜 추가해준다.

창의 크기가 바뀔때마다 화면크기에 따라 버튼크기 및 배치가 적절히 잘 변화시키기 위해서 addComponentListener를 추가하여 오버라이딩 된 componentResized함수 안에서 현재 창의 크기를 받아와 각각의 패널들의 크기를 조정시켜주었다.

(화면크기에 따른 크기 조정하기 위한 함수를 찾아서 오버라이딩된 함수들이 언제 호출되는지 계속 출력해보면서 함수를 익혀서 사용하는 시행착오를 겪었다.)

2) public static boolean isNumeric(String str)

String이 숫자모양 인지 판단하는 메소드 이다.
try catch 문을 이용하여 String을 double 형으로 선언을 시도해본 후, 오류 발생시 숫자가 아님을 알아낸다.

3) public void actionPerformed(ActionEvent e) // 계산을 위한 액션 메쏘드

- 우선 tempName에 getActionCommand함수를 이용하여 사용자가 누른 버튼 이름을

불러온다.

- tempString에 getText함수를 이용하여 여태까지 tField에 받아진 내용을 저장한다.

- 만약 사용자가 누른 버튼의 이름이 C라면, if문과 equals함수를 이용하여 tField의 Text를 다 날려버린후 메소드를 그대로 종료시킨다.

- 버튼을 눌렀을 때 텍스트 필드에 0뿐만이라면 그 0을 지워버린다. (처음에 이 함수가 없어서 3을 출력하려할 때 03이 되었었다.

- if ((tempName.equals("+") || tempName.equals("-")))문을 이용하여 사칙연상에 사용될 버튼들을 처리하는 부분을 작성했다.

우선 만약 is_ready가 false일 경우엔 (직전에 연산이 완료되지않은 연산기호가 없다) tempNum1에서 여태까지 tField에 올라온 모든 숫자를 그대로 저장하고 is_ready값을 true로 바꿔준다. who에 어떤 기호인지 저장한다.

만약 is_ready값이 ture일때는 tempNum2값을 temp에 정수형으로 바꿔 저장한후 이전에 받아두었던 who를 이용하여 연산을 한후 결과값을 출력하고, 새로받은 연산기호를 who에 업데이트 해준다. 그리고 다음 연산을 위해 tempNum2을 비워준다.

또한, 교수님이 올려주신 예시 계산기처럼, +나-를 누른후의 숫자만 띄워주기 위해서 텍스트 화면에 표시될 onText를 비워주었다.

- else if (tempName.equals("=") && is_ready && !tempNum2.equals(""))
이문은 =을 입력받았을 때 저장된 연산 기호가 있고, tempNum2가 공백이 아닐경우에 Calculate함수를 이용하여 계산을 한 후에 계산 결과를 출력하고, 처음 상태로 변수들을 초기화 해준다.

- 만약 =버튼을 눌렀으나 is_ready가 ture가 아닐경우나 tempNum2인 오른쪽 피 연산자가 공백일 경우엔 아무일도 하지 않는다.
- 아무경우도 아니라면 누른 버튼을 출력한다.

- 만약 is_ready가 ture이고 방금 누른 버튼이 숫자일 경우엔 String형 tempNum2에 방금 누른 버튼을 더한다.

4) protected int Calculate(int a, int b, String s) // 계산 메소드

a와 b를 s에 따라서 사칙연산한다.

5) public static void main(String[] args) // main 메소드

main 메소드 에서는 생성자 JavaHw4을 실행한다.