1. 틀 나누기

처음 시작할 때, 나는 자바 프레임 내부에 있는 J패널을 구현하는 방식에 집중했다. 먼저 예시에 나온대로 숫자를 표현하고, 계산한 것을 보여주는 스크린 부분과 다양한 버튼등을 담고있는 메인 패널 부분을 나누어 생각했다.

해당 그림의 왼쪽은 숫자를 보여주는 스크린 오른쪽은 메인 패널.

2. 비율 유지한 상태로 크기 변경하는 창

이러한 방식으로 나누고 나서 가장 먼저 생각하게 된 것은 어떻게 비율을 유지한 상태로 화면크기를 변경할 수 있을까 였다. 하지만 처음에 생각했을 때는 생성자 내부에서 레이아웃을 null로 만든 이후 처음 크기와 위치를 설정해준 이후 계속해서 스크린의 크기를 바꿔주면 상관 없겠지 하는 생각으로 제작하기 시작했다. 그러나 해당 방식은 화면 크기가 바뀌더라도 반영되지 않았고, setSize를 어디에 써야할지도 막막해졌다. 이에 나는 다시 mouseListener를 적용해보았다. 한마디로 창 크기를 바꾸기 위해 마우스를 눌렀다 땠을 경우 사이즈를 바꾸어 주는 방식을 사용해 보려고 했던 것이었다. 그런데 해당 방식의 문제점은 패널 내부에서 클릭 반응이 일어나야 적용되는 함수이기 때문에 사실상 크기 변경 후 내부 패널을 클릭하지 않는 이상 변하지 않는다는 것을 깨닳았다.

이런 것으로 고민하던 중 이미지에 대해서 배우며 PaintComponent함수를 배우던 중 화면 크기에 맞추어 이미지가 변하는 것을 배우게 되었고, 해당 함수를 패널 컴포넌트에 적용시키는 방법은 없을까 생각하게 되었다. 그래서

```
protected void paintComponent(Graphics g){
    super.paintComponent(g);
    Hw4ReSize();
```

패널 내부에 paintComponent를 만든 이후 그 내부에 그 함수가 불릴 때 Hw4ReSize(); 함수를 불러오게 하여 패널의 사이즈를 새로 짜주는 것으로 변경하였다.

2.1 리사이즈 함수

```
    PanelW=getWidth()/2;
    PanelH=getHeight()/2;
    CalculateHeight=getHeight()/2;
    CalculateWidth=getWidth()/2;
    if(CalculateHeight>CalculateWidth) {
        CalculateHeight = CalculateWidth * 3;
        CalculateWidth *= 2;
    }
    else {
        CalculateWidth = CalculateHeight * 2;
        CalculateHeight*=3;
    }
    ButtonW=CalculateWidth/4;
    ButtonH=CalculateHeight/9;
}
```

해당 함수는 크기를 변경해주는 함수인데, Panel의 크기, 버튼의 크기등 대부분의 크기를 변경하도록 해주었다. 패널의 크기 변경의 경우 버튼의 크기를 기준으로 잡아 설정해준 이후 기준이 되는 패널의 크기들을 변환시켜주며 위아래의 크기와 메인 프레임의 크기가 맞도록 설정해주었다.

자세한 버튼의 크기의 경우 숫자를 넣어보며 시행착오로 설정해주었다.

3. 계산 결과 스크린

먼저 해당 계산 결과 또한 마찬가지로 폰트의 크기를 계속해서 변화시켜주어야 하기 때문에 paintComponent를 이용해주었다. 여기에 적힐 내용의 경우 getNum을 전역변수로 만들어 준 이후 다른 클래스에서 해당 값을 변경해주는 것으로 표현해 주는 것을 목표로 삼았다. 그래서 이것을 표현하기 위해 전역변수를 만들어 주었고, 해당 전

역변수를 graphic2D를 이용하여 출력해주어, 크기 전환, 색 변환 등을 자유롭게 해주었다. 이제 가장 중요한 해당 값을 전달하는 부분을 구현해야 했다.

4. 버튼 생성

버튼 생성은 메인 패널에서 해주었다. 그러나 해당 버튼 내부에 들어갈 숫자의 텍스트 성질(폰트, 크기, 색)등이나 버튼 색, 버튼을 누르면 나오는 숫자, 코드등을 저장해주어야 했기 때문에 버튼을 관리하는 클래스를 하나 더 생성해 주었다. 이곳에는 버튼을 누르면 나오는 코드(0~9 숫자, 10~15 다른 기능)를 저장하는 변수, 해당 버튼 위에 적혀질 폰트의 내용을 저장하는 변수와, 해당 버튼을 꾸밀 paintComponent를 넣어주었다. 그 이후 버튼이 눌러졌을 상황을 가정하고 해당 버튼이 가지고 있는 코드에 따른 폰트를 string안에 넣어주었고,

```
class Hw4Button extends JButton {        //메인 패널 내부에 들어갈 버튼
    int buttonCode;          //버튼에 들어갈 코드
    String otherNum = new String();      //버튼위에 표시될 문자
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (buttonCode < 10) otherNum = Integer.toString(buttonCode);
        else if (buttonCode == 10) otherNum = "C";
        else if (buttonCode == 11) otherNum = "+";
        else if (buttonCode == 12) otherNum = "-";
        else if (buttonCode == 13) otherNum = "=";
        else if(buttonCode == 14) otherNum = "+/-";
        else if(buttonCode == 15) otherNum = "00";
```

이후 해당 버튼안에 들어갈 폰트의 다양한 특성(색, 폰트, 크기)들을 설정해 주었다.

```
int w = getWidth();
int h = getHeight();
Graphics2D g2 = (Graphics2D) g;

if (getModel().isArmed()) { g2.setColor(getBackground().brighter()); }
else if (getModel().isRollover()) { g2.setColor(getBackground().darker()); }
else { g2.setColor(getBackground()); }

g2.fillRect(0, 0, w, h);
Font fp=new Font("Comic Sans MS", 1, (int)(getSize().width/(float)4));
FontMetrics fontMetrics = getFontMetrics(fp);
g2.setFont(fp);
Rectangle stringBounds = fontMetrics.getStringBounds(otherNum, g2).getBounds();


int textX = (w - stringBounds.width) / 2;
int textY = (h - stringBounds.height) / 2 + fontMetrics.getAscent();
g2.setColor(Color.decode("#E2E1E0"));          //폰트 색
g2.drawOval(w/6,h/6,w*10/15,h*10/15);
g2.setStroke(new BasicStroke(2));
if(otherNum.equals("C"))g2.setColor(Color.decode("#ff533d"));
g2.drawString(otherNum, textX, textY);
```

5. 버튼 이벤트 컨트롤

해당 버튼이 작용하도록 이벤트 컨트롤러를 적용해 줄 필요가 있는데, 해당 이벤트 컨트롤러를 만들기 위해서 먼저 생성자에 해당 버튼의 코드를 저장해 주는 작업이 필요했다. 버튼을 생성해준 이후 해당 버튼에 코드를 저장하는데 y축과 x축을 기준으로 버튼의 내용을 저장해주었다. 간단하게 하기 위해서 간단한 식으로 만들어 주었다. 해당 식 `buttonNum= 7 - 3*y + x;`이는 1~9까지의 숫자를 넣기위한 식이고 다른 것의 경우 10~의 코드를 넣어주어 해당 코드에 따라 텍스트와 이벤트 컨트롤을 넣어두려고 했다.

```
for (int y=0;y<4;y++){
    for(int x=0;x<4;x++){
        NumberButton[cnt]=new Hw4Button();
        NumberButton[cnt].addActionListener(this);
        int buttonNum= 7 - 3*y + x;
        if(y<3 && x<3)
            NumberButton[cnt].buttonCode=buttonNum;
        else if(y==0)
            NumberButton[cnt].buttonCode= 10;
        else if(y==1)
            NumberButton[cnt].buttonCode=11;
        else if(y==2)
            NumberButton[cnt].buttonCode=12;

        else if(y==3){
            if(x==0)
                NumberButton[cnt].buttonCode=14;
            if(x == 1)
                NumberButton[cnt].buttonCode = 0;
            if(x == 2)
                NumberButton[cnt].buttonCode = 15;
            if(x==3)
                NumberButton[cnt].buttonCode=13;

        }
```

그래서 결과적으로 이렇게 넣어주게 되었다. 해당 코드를 보면 각 버튼마다 액션리스너를 넣어줬는데 이것을 위해 인터페이스 엑션리스너를 추가해 주었다. 해당 액션 리스너에는 누른 버튼을 받아와서 해당 버튼의 숫자를 전역변수로 선언해둔 출력 숫자에 더해주는 방식으로 보여주었다. 또한, 출력할 값과, 정답 값을 따로 나누어 정답을 저장해 두는 방식을 이용하였다.

```
if(e.getSource() instanceof Hw4Button){
        if(buttonNum<10){
            PrintNum*=10;
            PrintNum+=buttonNum;
            isAnswer=false;
            CheckNumberPad=1;
        }
        else if(buttonNum==10){
            PrintNum=0;
            AnswerNum=0;
            isAnswer=true;
            operator=1;
            CheckNumberPad=0;
        }
        else if(buttonNum==11){
            AnswerNum+=PrintNum*operator;
            PrintNum=0;
            operator=1;
            isAnswer=true;
            CheckNumberPad=0;
        }
```

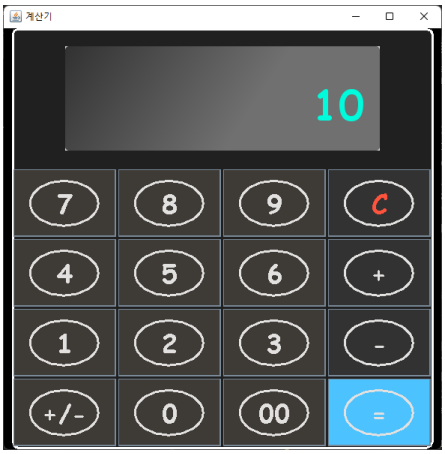
```
        else if(buttonNum==12){
            AnswerNum+=PrintNum*operator;
            PrintNum=0;
            operator=-1;
            isAnswer=true;
            CheckNumberPad=0;
        }
        else if(buttonNum==13){
            AnswerNum+=PrintNum*operator;
            PrintNum=0;
            operator=1;
            isAnswer=true;
            CheckNumberPad=0;
        }
        else if(buttonNum==14) {
            PrintNum *= -1;
            isAnswer = false;
            CheckNumberPad = 1;
        }
```

6. 꾸미기

이제 해당 계산기를 꾸밀 차례였다. 해당 계산기의 레퍼런스의 경우 색감은 윈도우에서 기본으로 제공하는 계산기

를 이용해 주었다. 그러나 교수님께서 주신 레퍼런스에는 다양한 선과 입력창이

있기 때문에 조금 더 다양한 graphics2D의 함수를 이용해 주었다.



결과값이 나타나면 푸른색 빼 줄 경우 붉은색, 더해주거나 디폴트 상태의 경우 하얀색으로 만들어주었다. 또한,

```
if (getModel().isArmed()) { g2.setColor(getBackground().brighter()); }
else if (getModel().isRollover()) { g2.setColor(getBackground().darker()); }
else { g2.setColor(getBackground()); }
```

해당 방식으로 뒷 배경의 색을 받아와 버튼에 커서가 올라오면 뒷색을 좀더 어둡게, 누를경우 더 밝게 만들어 주었다.

7. 추가 버튼

해당 버튼의 경우 요구사항에는 없지만 빈 버튼 두개에 새로운 기능을 만들어 주고 싶었다. 그래서 해당 기능은 계산기에 자주 보이는 0을 두개 붙여주는 00 버튼과 +와 -를 변경해주는 버튼을 추가해 주었다.