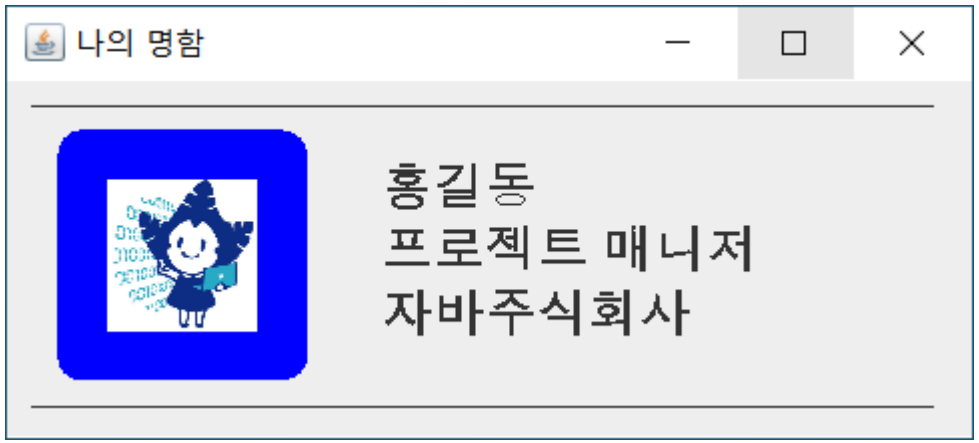

GUI 실습에 있어서 배치 관리자, 프레임의 크기 및 각 컴포넌트의 크기와 간격은 문제에서 별도 명시가 없으면 실행 결과 화면과 유사하게 임의로 설정합니다.

코드는 참고용일 뿐 다르게 작성하셔도 됩니다.

# Lab 10. Chapter 12

# 실습 1> (DrawCard.java)

- 다음과 같은 명함을 화면에 그려보시오.



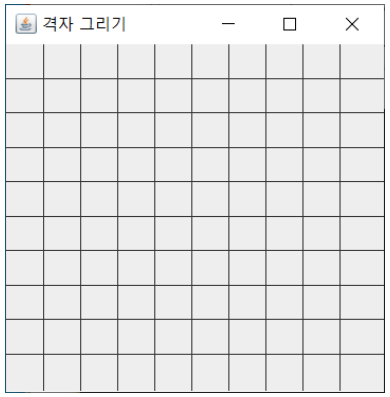

- 명함은 텍스트와 그림, 도형(1개 이상)을 모두 포함하고 있어야 합니다.
  - 도형, 그림, 텍스트의 크기, 위치, 텍스트 내용, 색상 다 임의 설정 가능
  - 버튼이나 라벨 등의 컴포넌트가 아닌 Graphics 를 이용한 drawXXX 또는 fillXXX 메소드를 활용해야 합니다.

# 실습 1>

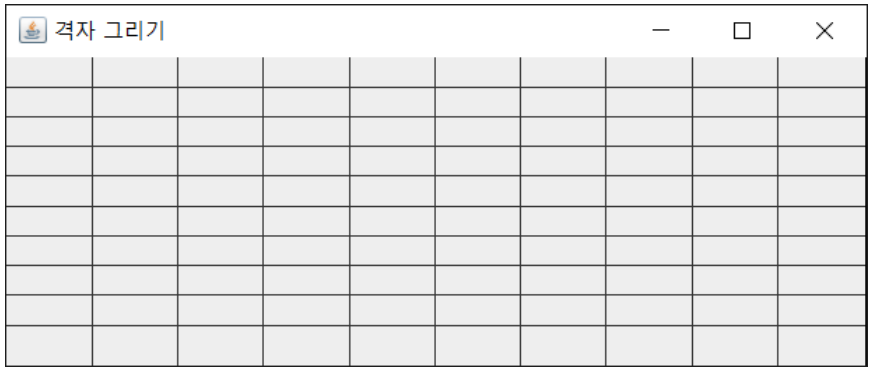
```
public class DrawCard extends JFrame {
    class MyPanel extends JPanel{
        protected void paintComponent(Graphics g)
        {
            super.paintComponent(g);
            // 텍스트 및 도형, 그림 그리는 코드 작성
        }
    }
    public DrawCard()
    {
        setSize(400,180);
        setTitle("나의 명함");
        add(new MyPanel());
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] arge){
        new DrawCard();
    }
}
```

# 실습 2> DrawLine 실습 (DrawLineTest-1.java, DrawLineTest-2.java)

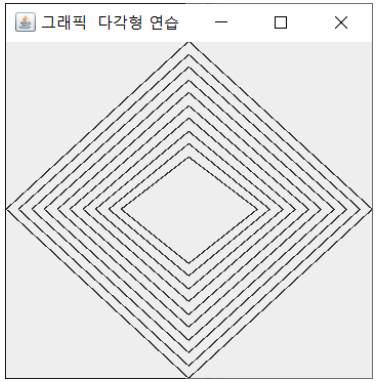
- 아래 그림과 같은 기하학적인 모양을 그리시오. 프레임의 크기를 조절하면 자동으로 크기가 조절됩니다.
- 컨텐트팬을 10x10으로 나누는 격자 그리기



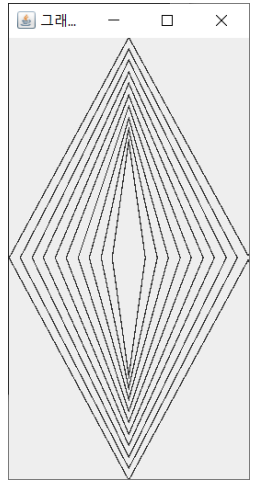

프레임
크기 조절 ➔




- 컨텐트팬에 꽉차는 마름모 10개 그리기




프레임
크기 조절 ➔



# 실습 2>

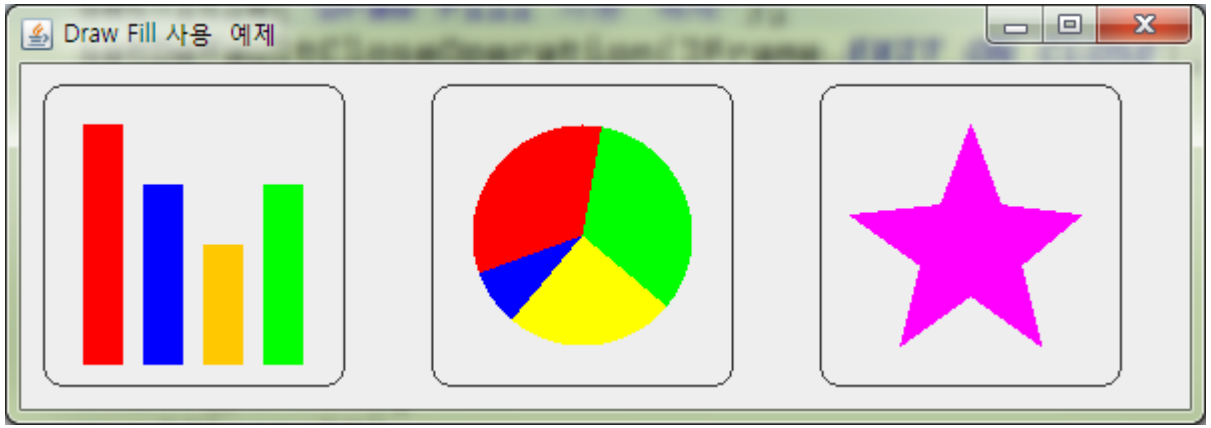
```
public class DrawLineTest extends JFrame {
    class MyPanel extends JPanel{
        protected void paintComponent(Graphics g)
        {
            super.paintComponent(g);

            // 선 그리기
            // 반복문 활용

        }
    }
    public DrawLineTest ()
    {
        setSize(400,300);
        setTitle("격자 그리기");
        add(new MyPanel());
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] arge){
        new DrawLineTest ();
    }
}
```

# 실습 3> 도형 그리기 및 칠하기

- 다음의 출력결과를 나타내는 프로그램을 작성하시오. (DrawFillTest.java)
- 출력 결과

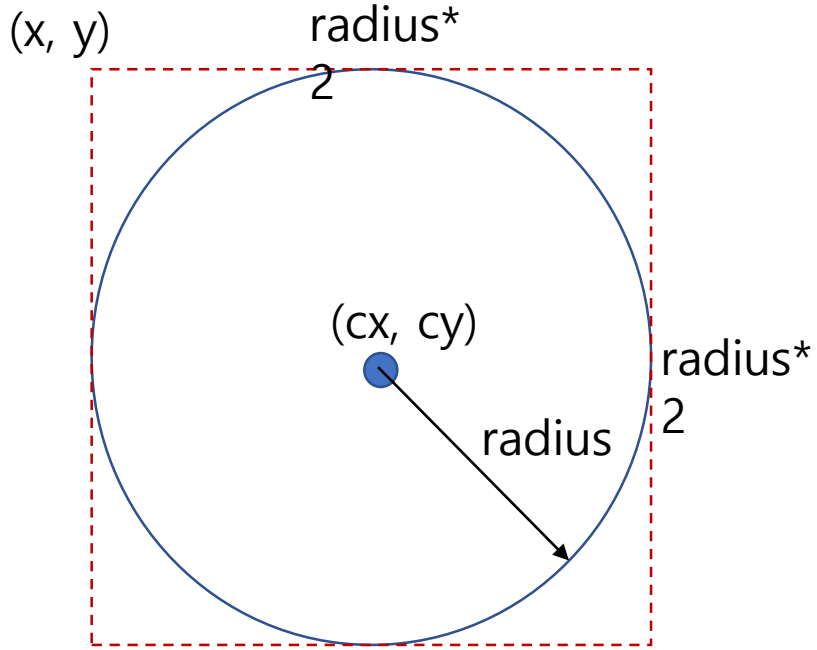


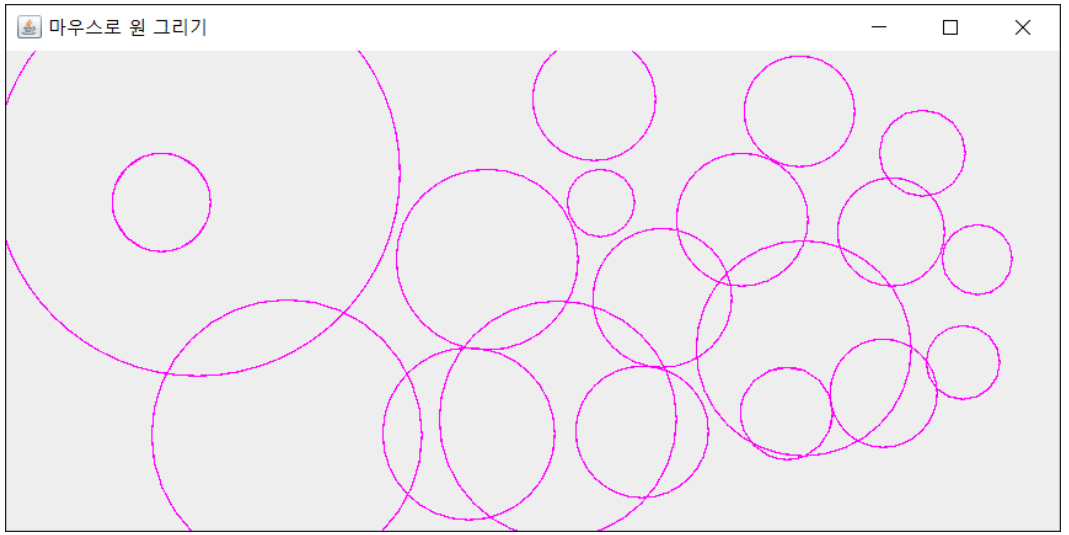

- 3개의 사용자 정의 JPanel을 각각 따로 생성해서 프레임에 그리드 레이아웃으로 붙이기.(막대 길이, 비율, 색상, 별 위치는 임의로 설정)
- fillRect, fillArc, fillPolygon 활용

```java
public class DrawFillTest extends JFrame {
    class MyPanel1 extends JPanel{
        protected void paintComponent(Graphics g){
            super.paintComponent(g);
            // 막대그래프 (fillRect 활용)
        }
    }
    class MyPanel2 extends JPanel{
        protected void paintComponent(Graphics g){
            super.paintComponent(g);
            // 원형 차트 (fillArc 활용)
        }
    }
    class MyPanel3 extends JPanel{
        protected void paintComponent(Graphics g){
            super.paintComponent(g);
            // 별 그리기 (fillPolygon 활용)
        }
    }
    public DrawFillTest (){
        setSize(1000,400);
        setTitle("Draw Fill 사용예제");
        setLayout(new GridLayout(0, 3));
        add(new MyPanel1()); add(new MyPanel2()); add(new MyPanel3());
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] arge){
        new DrawFillTest ();
    }
}
```

# 실습 4> 마우스로 원 그리기(MouseCircle.java)

- 마우스를 찍어 중심을 잡고 드래깅하여 놓으면 원을 그리는 코드를 작성하시오.




```
Vector <Circle> v = new Vector<>();
.....
for(Circle c :  v){
   // x 계산
   // y 계산
   g.drawOval(x, y, c.radius*2,
c.radius*2);
}
```

- (참고 예제) → 교재 슬라이드 41, MyPaintApp 참고
- 원을 표현하는 Circle 클래스(cx, cy, radius)를 만든다.
- Vector를 생성한다. Vector <Circle> v = new Vector<>();
- 마우스가 눌러질 때 cx, cy 좌표를 저장한다.
- 마우스가 떼질 때 dx, dy 좌표를 저장한다.
  - 두 좌표를 이용하여 radius 계산(거리 계산) : $\sqrt{(dx-cx)^2+(dy-cy)^2}$
  - Circle 객체를 생성하여 new Circle(cx, cy, radius)
  - Vector<Circle> 타입의 벡터 v에 저장하면 그려진 모든 원을 기억할 수 있다.
  - repaint()를 호출한다.
- paintComponent()에서는
  벡터 v에 있는 모든 원을 그리면 된다.

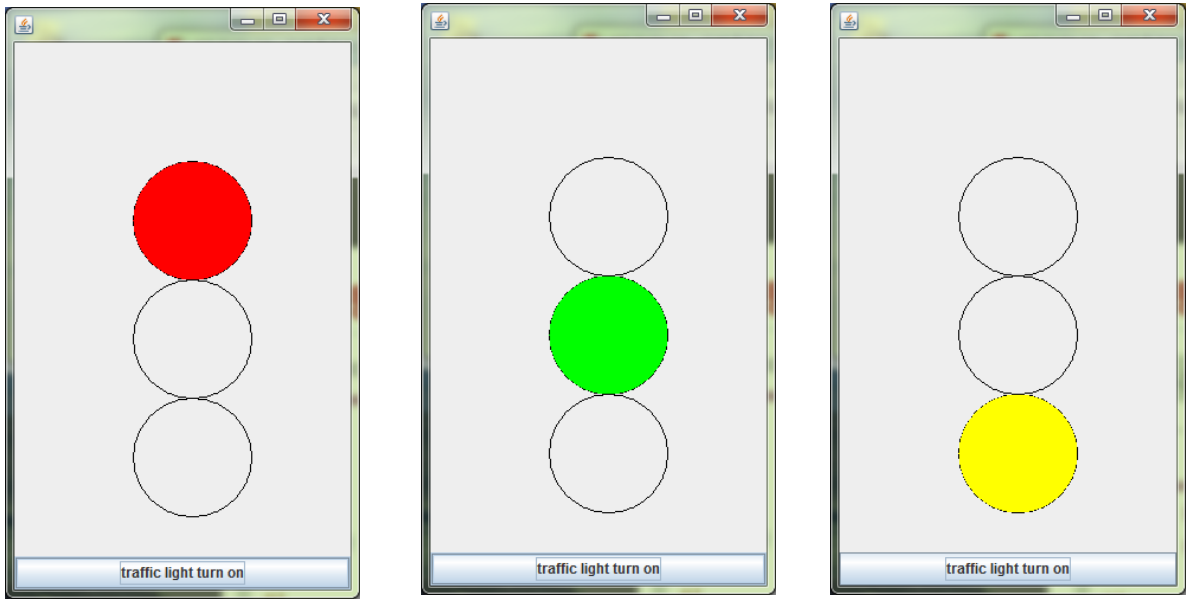
```
for(Circle c :  v){
    // x 계산
    // y 계산
    g.drawOval(x, y, c.radius*2,
c.radius*2);
}
```

# 실습 4> 추가 (선택)

- 앞의 예제를 마우스 누를 때와 뗄 때만 구현하게 되면 드래그 하는 동안의 원 모양을 볼 수가 없습니다.
- 드래그 하는 동안에도 원 모양을 볼 수 있도록 수정해 보시오.

# 실습 5> 신호등 예제(TrafficTest.java)

- 아래 그림과 같은 모양의 신호등을 그리시오. 그리고 그림처럼 아래쪽에 버튼을 하나 부착하시오.
- 그림처럼 버튼을 누를 때 마다 신호등이 켜지는 위치와 색깔이 달라지도록 설정하시오.

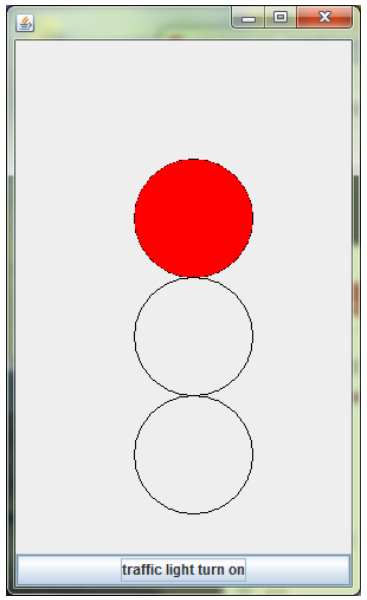



초기 상태: red 만 보이는 상태

버튼을 누를 때 마다 red → green → yellow 의 순서대로 보이도록 하면 됩니다.

# 실습 5> 신호등 예제(TrafficTest.java)

- 교재 슬라이드 38 “버튼이 눌러지면 이미지를 이동” 예제 참고




fillOval 3개와 drawOval 3개 그리기
fillOval은 상태변수를 하나 두고 그 변수값에 따라 3중의 하나만 그려지도록 함.

그렇다면 상태값의 변화는 누가?
버튼을 누를 때마다 상태값 변화 (간단하게 정수 변수를 두고 1씩 증가하도록 하면 됨)
그리고 repaint() 호출

paintComponent에서는 변경된 상태값에 따라 해당 fillOval이 실행되도록 구성하면 됨.
조건과 상관없이 drawOval은 항상 그려짐.