(INTERMEDIATE)
JAVA PROGRAMMING

19. Swing – More Graphics Programming

Chapter 12

Reivew: 스윙의 페인팅 메카니즘

- 스윙 컴포넌트들이 그려지는 과정에 대한 이해 필요
  - 바탕 컨테이너부터 그려짐(다음 슬라이드 참고)
- 스윙의 페인팅에 관여되는 JComponent 메소드

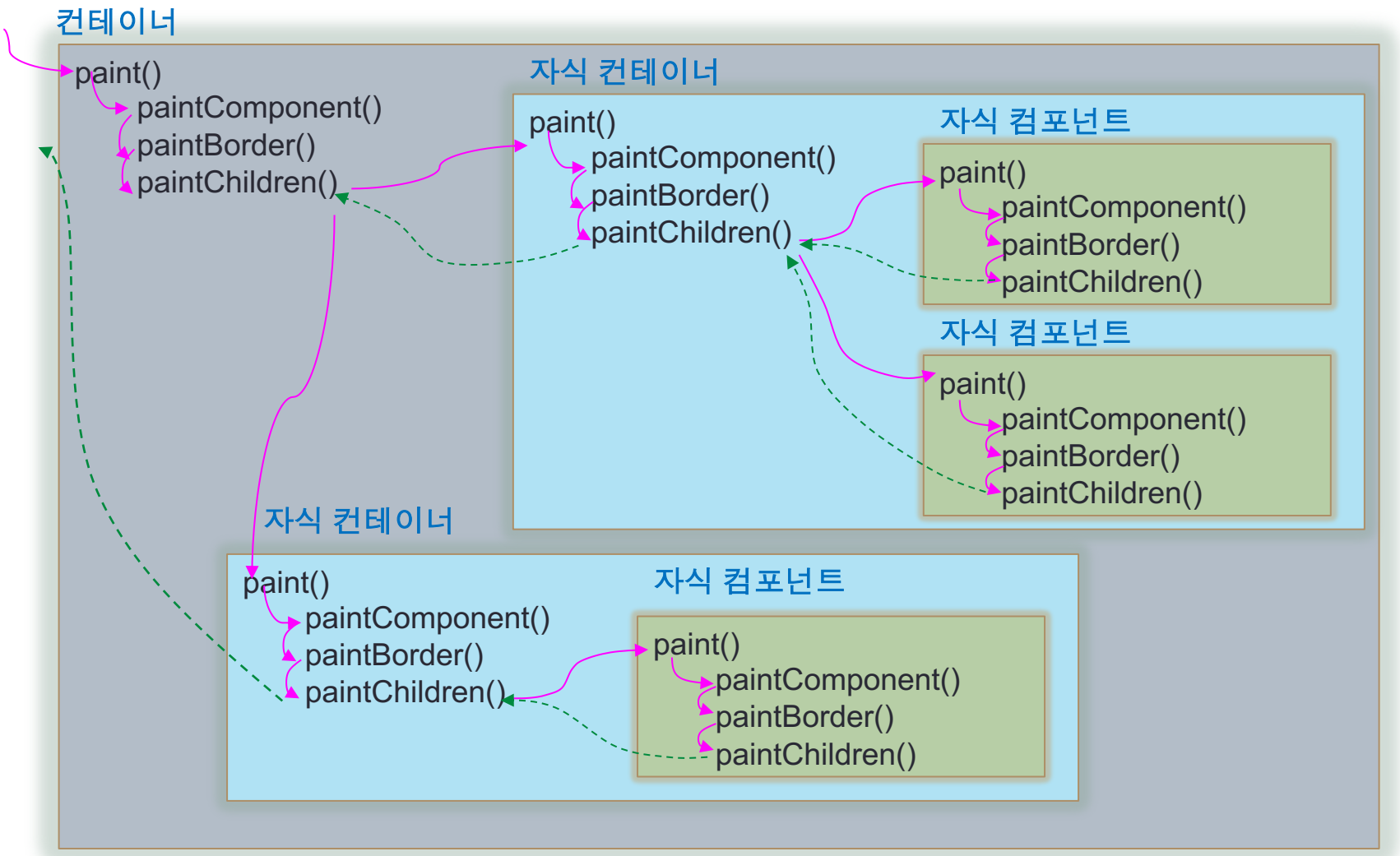
```
void paint(Graphics g) 컴포넌트 자신과 모든 자손 그리기
void paintComponent(Graphics g) 컴포넌트 자신의 내부 모양 그리기
void paintBorder(Graphics g) 컴포넌트의 외곽 그리기
void paintChildren(Graphics g) 컴포넌트의 자식들 그리기(컨테이너의 경우)
```

- JComponent.paint()의 코드 구조

```
public void paint(Graphics g) { // g가 아래 3개의 메소드에 그대로 전달된다.
    ...
    paintComponent(g); // ① 컴포넌트 자신의 내부 모양 그리기
    paintBorder(g); // ② 컴포넌트 자신의 외곽 그리기
    paintChildren(g); // ③ 컴포넌트의 자식들 그리기
    ...
}
```

- 개발자가 paintComponent()를 직접 호출하면 안됨
  - paintComponent()는 페인팅 메카니즘에 의해 자동으로 호출됨

Review: 스윙 컴포넌트가 그려지는 과정



Review: repaint() 메소드

- 강제로 컴포넌트의 다시 그리기 지시하는 메소드

```
component.repaint();
```

  - 자바 플랫폼에게 지금 당장 컴포넌트를 다시 그리도록 지시
    - 컴포넌트의 페인팅 과정 진행
- repaint()가 필요한 경우
  - 프로그램 내에서 컴포넌트의 모양과 위치를 변경한 경우
    - repaint()를 호출하면 자바 플랫폼에 의해 컴포넌트의 paintComponent()가 호출됨
- 부모 컴포넌트부터 다시 그리는 것이 좋음
  - 만일 컴포넌트의 위치가 변경된 경우
    - repaint()가 불려지면 이 컴포넌트는 새로운 위치에 다시 그려지지만 이전 위치에 있던 자신의 모양이 남아 있기 때문에 부모 컴포넌트의 repaint()를 호출하는 것이 좋음

```
component.getParent().repaint();
```

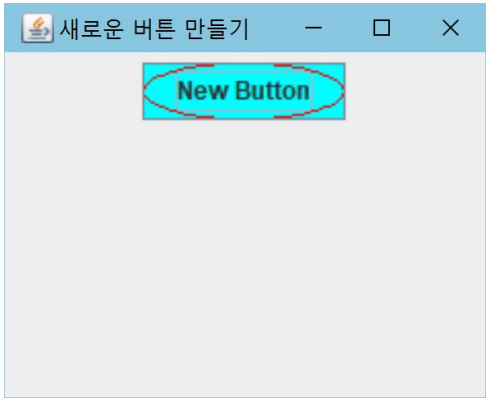
JButton을 상속받아 새로운 버튼 생성 예

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

public class paintComponentEx extends JFrame {
    public paintComponentEx() {
        setTitle("새로운 버튼 만들기");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container c = getContentPane();
        c.setLayout(new FlowLayout());
        MyButton b = new MyButton("New Button");
        b.setOpaque(true);
        b.setBackground(Color.CYAN);
        c.add(b);
        setSize(250,200);
        setVisible(true);
    }
```

```
    class MyButton extends JButton {
        MyButton(String s) {
            super(s);
        }
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.RED);
            g.drawOval(0,0,this.getWidth()-1,
                           this.getHeight()-1);
        }
    }

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



JAVA:
MORE GRAPHICS – JAVA2D

Java 2D

- 광범위한 그래픽 객체를 그릴 수 있다.
- 도형의 내부를 그라디언트(gradient)나 무늬로 채울 수 있다.
- 문자열을 출력할 때 폰트와 렌더링 과정을 세밀하게 조정할 수 있다
- 이미지를 그릴 수 있고 필터링 연산을 적용할 수 있다.
- 그래픽 객체들의 충돌을 감지할 수 있는 메커니즘을 제공한다.
- 렌더링 중간에 객체들을 조합하거나 변형할 수 있다.
- 화면과 프린터에 같은 방법으로 그릴 수 있다.

Java 2D

Using 2D Graphics API to display complex charts

Image Blur Sharpen

Using image-filtering operations

그림15-10. Java 2D를 이용한 그래픽의 예(출저:java.sun.com)

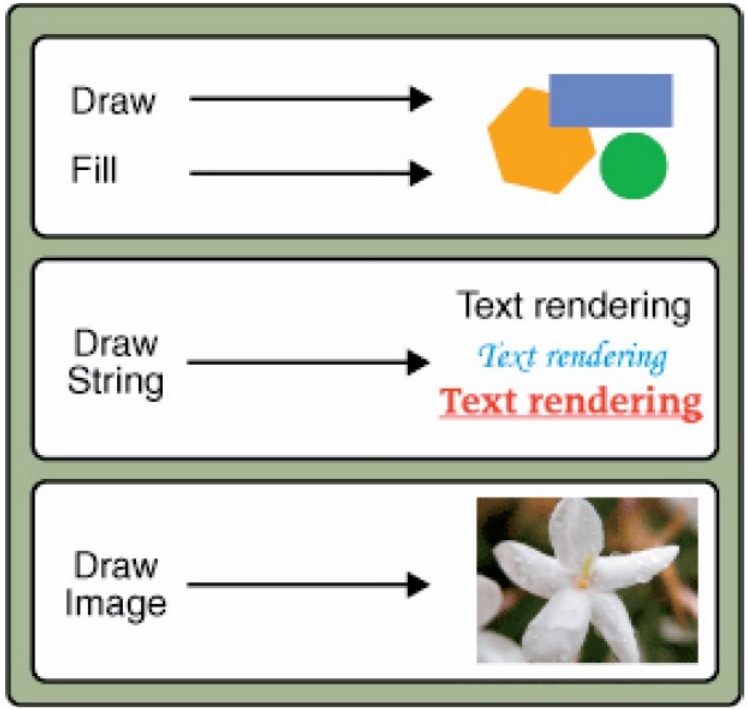
Java 2D를 사용하려면?

```
public void paintComponent(Graphics g)
{
    Graphics2D g2 = (Graphics2D) g;
    ...
}
```

Java2D를 사용하려면 단순히 Graphics 객체 변수를 Graphics2D 타입으로 형변환한다.

Java 2D의 메소드

```
public void paintComponent(Graphics g)
{
   Graphics2D    g2 = (Graphics2D) g;
   g2.drawLine(100, 100, 300, 300);
   g2.drawRect(10, 10, 100, 100);
   ...
}
```



Java 2D를 이용한 그리기

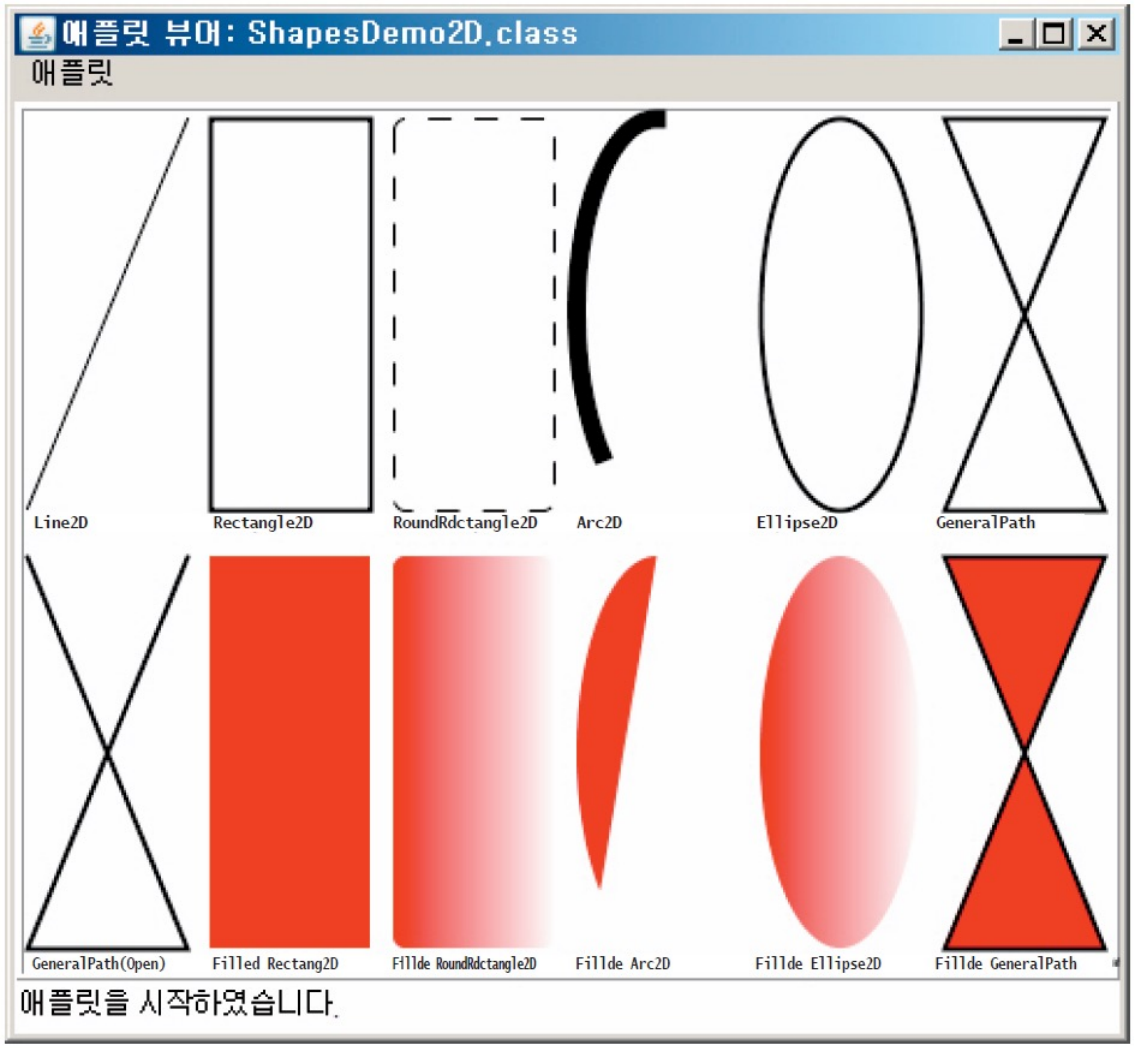
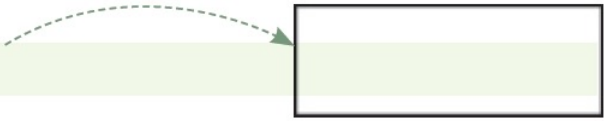


그림15-11. Java 2D를 이용한 형상 그리기(출저: java.sun.com)

Java 2D를 이용한 도형 그리기

```
Shape rect = new Rectangle2D.Float(7, 8, 100, 200);
```



```
g2.draw(rect);      // 사각형을 그린다.
```

```
g2.fill(rect);
```

Java 2D를 이용한 도형 그리기

점생성

```
// 점을 생성한다.
Point2D.Double point = new Point2D.Double(x, y);
```

직선생성

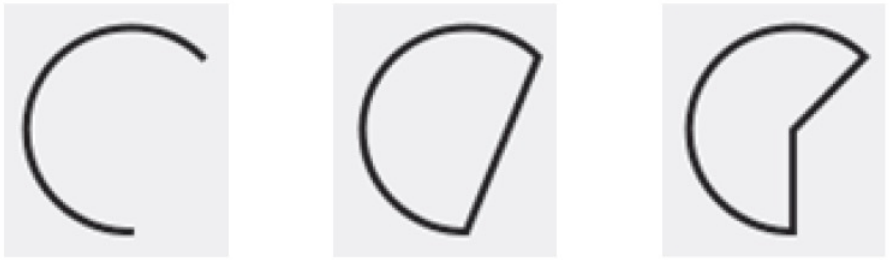
```
// 직선 객체를 생성하고 직선을 그린다.
g2.draw(new Line2D.Double(x1, y1, x2, y2));
```

타원생성

```
// 타원 객체를 생성하고 타원을 그린다.
g2.draw(new Ellipse2D.Double(x, y, rectwidth, rectheight));
```

원호 생성

```
Shape arc1 = new Arc2D.Float(10, 10, 90, 90, 90, 60, Arc2D.OPEN);
```

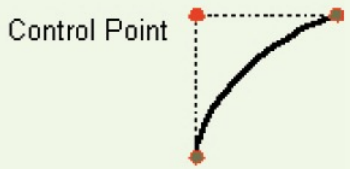


(a) Arc2D.OPEN (b) Arc2D.CHORD (c) Arc2D.PIE

그림15-12. 원호의 종류

2차 곡선과 3차 곡선(Quadratic and Cubic Curves)

```
// QuadCurve2D.Float 객체를 생성한다.
QuadCurve2D q = new QuadCurve2D.Float();
// QuadCurve2D.Float 객체의 값을 설정하고 화면에 그린다.
q.setCurve(x1, y1, ctrlx, ctrly, x2, y2);
g2.draw(q);
```




```
// CubicCurve2D.Double 객체를 생성한다.
CubicCurve2D c = new CubicCurve2D.Double();
// CubicCurve2D.Double 객체에 값을 설정하고 화면에 그린다.
c.setCurve(x1, y1, ctrlx1, ctrly1, ctrlx2, ctrly2, x2, y2);
g2.draw(c);
```

임의의 형상 - PolyLine

```
// GeneralPath 객체를 생성한다.
int x2Points[] = {0, 100, 0, 100};
int y2Points[] = {0, 50, 50, 0};
GeneralPath polyline =
   new GeneralPath(GeneralPath.WIND_EVEN_ODD, x2Points.length);
polyline.moveTo (x2Points[0], y2Points[0]);
for (int index = 1; index < x2Points.length; index++) {
   polyline.lineTo(x2Points[index], y2Points[index]);
};
g2.draw(polyline);
```

예제

MoreShapes.java

```
01  ...
02
03  public class MoreShapes extends JFrame {
04     public MoreShapes() {
05        setSize(600, 130);
06        setTitle("Java 2D Shapes");
07        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
08        JPanel panel = new MyPanel();
09        add(panel);
10        setVisible(true);
11     }
12     public static void main(String[] args) {
13        new MoreShapes();
14     }
15  }
16  class MyPanel extends JPanel {
17     ArrayList<Shape> shapeArray = new ArrayList<Shape>();
18
19     public MyPanel() {
20        Shape s;
21
```

향상된 배열인 컬렉션의 일종인 ArrayList를 사용한다. Shape 객체들을 저장한다 (22장에서 학습한다).

예제

```
22          s = new Rectangle2D.Float(10, 10, 70, 80);
23          shapeArray.add(s);
24
25          s = new RoundRectangle2D.Float(110, 10, 70, 80, 20, 20);
26          shapeArray.add(s);
27
28          s = new Ellipse2D.Float(210, 10, 80, 80);
29          shapeArray.add(s);
30
31          s = new Arc2D.Float(310, 10, 80, 80, 90, 90, Arc2D.OPEN);
32          shapeArray.add(s);
33
34          s = new Arc2D.Float(410, 10, 80, 80, 0, 180, Arc2D.CHORD);
35          shapeArray.add(s);
36
37          s = new Arc2D.Float(510, 10, 80, 80, 45, 90, Arc2D.PIE);
38          shapeArray.add(s);
39      }
40
41      public void paintComponent(Graphics g) {
42          super.paintComponent(g);
43          Graphics2D g2 = (Graphics2D) g;
```

예제

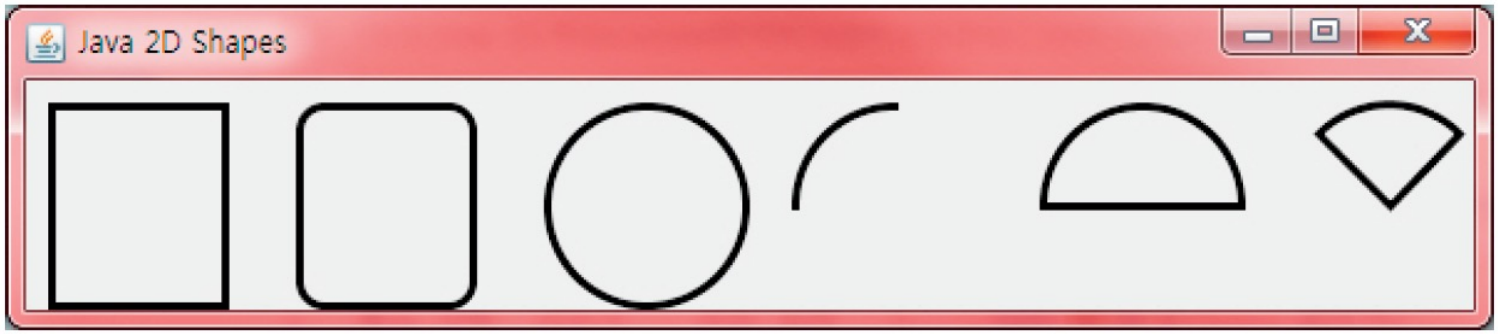
```
44
45          g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
46                RenderingHints.VALUE_ANTIALIAS_ON);
47
48          g2.setColor(Color.BLACK);
49          g2.setStroke(new BasicStroke(3));
50          for (Shape s : shapeArray)
51             g2.draw(s);
52       }
53    }
```

앤티에일리어싱은 도형을 매끄럽게 그리기 위하여 설정한다. 연산 시간은 조금 더 걸리지만 그만큼 그래픽의 품질이 좋아진다.

setStroke() 메소드를 이용하여서 도형을 그리는 두께를 설정할 수 있다.

shapeArray에 저장된 Shape 객체들을 꺼내서 화면에 그려준다.
Java 2D의 도형들은 모두 Shape 인터페이스를 구현하기 때문에 Shape 타입으로 생각할 수 있다.

실행결과



Java 2D를 이용한 도형 채우기

- Java 2D를 이용하여 도형을 채우는 방법에 대하여 살펴보자. 단일색으로 도형을 채우려면 먼저 setColor()를 호출하여서 채우는 색상을 설정한 후에 fill() 메소드를 호출하면 된다

```
g2.setColor(Color.BLUE);
g2.fill(ellipse);
```

투명하게 그리기

```
g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.50F));
```

그라디언트 칠하기

```
GradientPaint gp = new GradientPaint(0, 0, Color.WHITE, 0, 100, Color.RED);
```

예제

FillShapes.java

```
01  ...
02
03  class MyComponent extends JComponent {
04
05     public void paint(Graphics g) {
06        Graphics2D g2 = (Graphics2D) g;
07
08        // 앤티 에일리어싱을 설정한다.
09        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
10              RenderingHints.VALUE_ANTIALIAS_ON);
11
12        g2.setStroke(new BasicStroke(3));
13        GradientPaint gp = new GradientPaint(0, 10, Color.WHITE, 0, 70,
14                          Color.RED);
15        // 사각형
16        g2.setPaint(Color.RED);
17        g2.fill(new Rectangle2D.Float(10, 10, 70, 80));
18        // 둥근 사각형
19        g2.setPaint(gp);
20        g2.fill(new RoundRectangle2D.Float(110, 10, 70, 80, 20, 20));
21        ...
```

GradientPaint 객체를 생성한다. → 13

GradientPaint 객체로 채우는 색상을 지정 → 19

예제

```
22        }
23    }
```

실행결과