Chapter 16

그래픽 프로그래밍

Contents

그래픽 요소

- 그래픽 주체
  - 그림을 그리도록 프로그래밍하는 개발자 혹은 JVM
- 그래픽 도구
  - 펜, 붓, 팔레트, 폰트 등을 의미. 자바는 Graphics 클래스로 그리기, 칠하기, 이미지 출력하기, 클리핑 등 프로그래밍에 필요한 모든 필드와 메서드를 제공
- 그래픽 대상
  - 그림을 그릴 수 있는 도화지 등을 의미한다. 자바에서는 AWT나 스윙의 모든 컴포넌트, 이미지가 그래픽 대상

컴포넌트의 렌더링

- 그래픽을 지원하는 대부분의 프로그래밍에서 컴포넌트는 GUI 시스템이 자신의 모양을 렌더링(Rendering: GUI 컴포넌트의 내부 데이터를 시각적 이미지로 변환하는 과정)

- 자바에서도 모든 스윙 컴포넌트는 JVM이 javax.swing.JComponent 클래스가 제공하는 paintComponent 메서드를 이용해 자신의 모양을 렌더링함

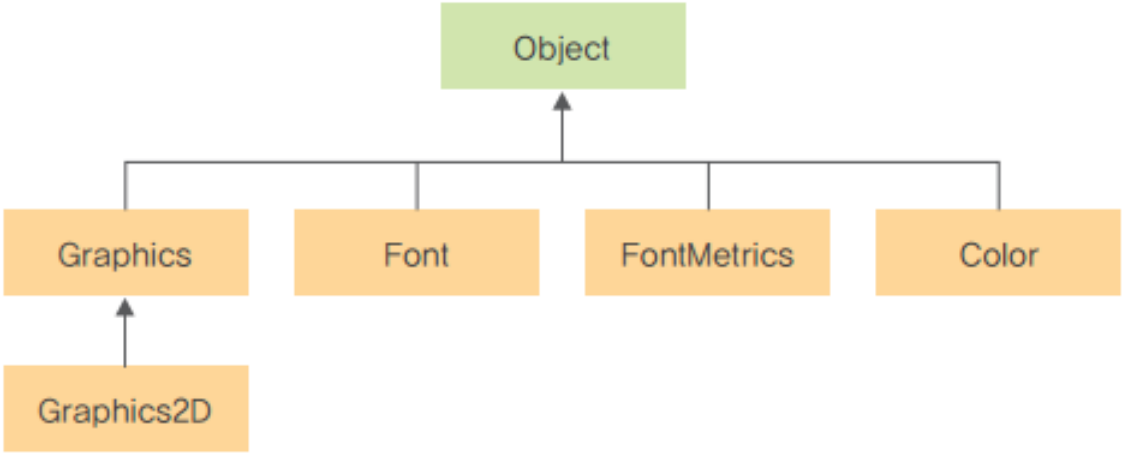
```
protected void paintComponent(Graphics g)
```

그래픽 컨텍스트

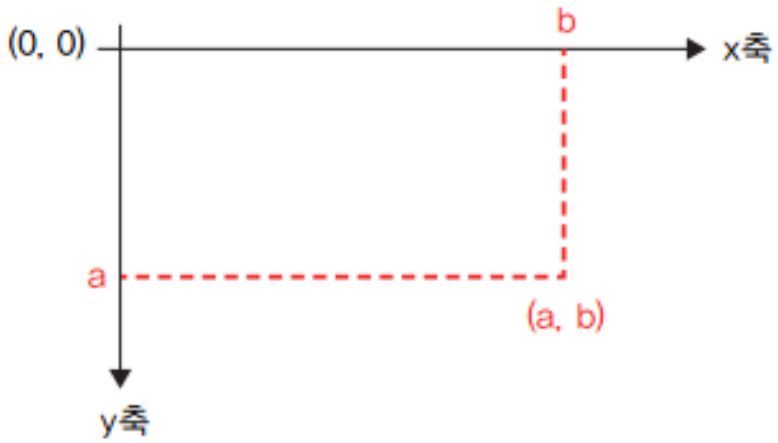
- 그래픽 객체를 그리는 데 필요한 정보

- Graphics 클래스는 자바에서 그리기 작업을 할 때 필요한 도구로 AWT뿐만 아니라 스윙에서도 사용. 자바에서는 색상 선택, 문자열 그리기, 도형 그리기, 도형 칠하기, 이미지 그리기, 클리핑 등을 지원하는 각종 메서드와 상수를 Graphics클래스로 제공

- 그래픽 프로그래밍과 관련된 클래스의 계층구조



Graphics 클래스가 사용하는 좌표 체계



- 단위는 pixel

문자열 그리기

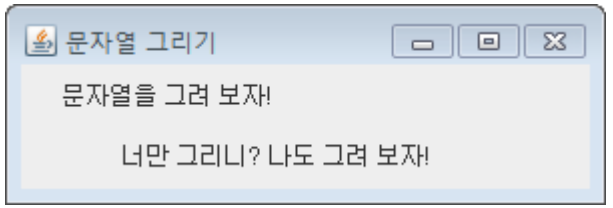
- 현재 Graphics에 설정된 색상과 폰트를 사용해 지정된 좌표에 문자열을 그림

- 형식

```
void drawString(String str, int x, int y)
```

- 예제 : [예제 16-1]

```java
package sec02;

import java.awt.Graphics;

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

            g.drawString("문자열을 그려 보자!", 20, 20);
            g.drawString("너만 그리니? 나도 그려 보자!", 50, 50);
        }
    }

    StringDemo() {
        setTitle("문자열 그리기");

        add(new MyPanel());

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 100);
        setVisible(true);
    }

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



색상 지정 : Color 클래스(1)

- 생성자

```
Color(float r, float g, float b)
Color(float r, float g, float b, float a)
Color(int r, int g, int b)
Color(int r, int g, int b, int a)
Color(int rgb)
Color(int rgba, boolean hasalpha)
```

- Color 클래스가 제공하는 대표적인 상수

상수	RGB 값
static Color black(또는 BLACK)	(0, 0, 0)
static Color blue(또는 BLUE)	(0, 0, 255)
static Color darkgray(또는 DARK_GRAY)	(64, 64, 64)
static Color lightGray(또는 LIGHT_GRAY)	(192, 192, 192)
static Color green(또는 GREEN)	(0, 255, 0)
static Color red(또는 RED)	(255, 0, 0)
static Color white(또는 WHITE)	(255, 255, 255)

색상 지정 : Color 클래스(2)

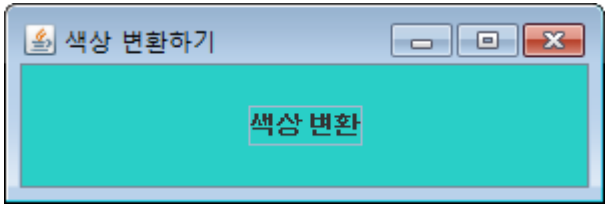
- 컴포넌트 객체의 전경색이나 배경색을 설정할 수 있도록 Component 클래스가 메서드 제공

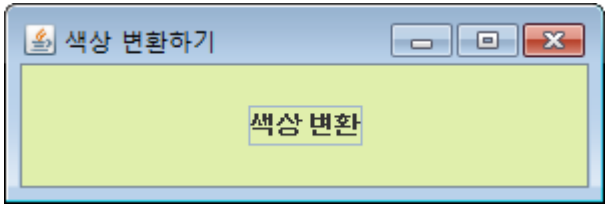
```
void setBackground(Color c)
void setForeground(Color c)
```

- 컴포넌트 객체 위에서 문자나 도형 등을 그리기 전에 Graphics 클래스의 메서드를 이용해 색상 설정

```
void setColor(Color c)
```

- 예제 : [예제 16-2]






```java
package sec02;

import java.awt.Color;

public class ColorDemo extends JFrame {
    ColorDemo() {
        setTitle("색상 변환하기");

        JButton b = new JButton("색상 변환");
        add(b);
        b.addActionListener(e -> {
            Color color = new Color((int) (Math.random() * 255.0),
                                    (int) (Math.random() * 255.0),
                                    (int) (Math.random() * 255.0));
            b.setBackground(color);
        });

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 100);
        setVisible(true);
    }

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

폰트 지정 : Font 클래스

- 생성자

```
// 지정한 폰트 이름, 폰트 스타일, 폰트 크기를 사용해 폰트 객체를 생성한다.
Font(String name, int style, int size)
```

- 폰트 설정 메서드

```
void setFont(Font font)
```

- Font 클래스가 제공하는 상수

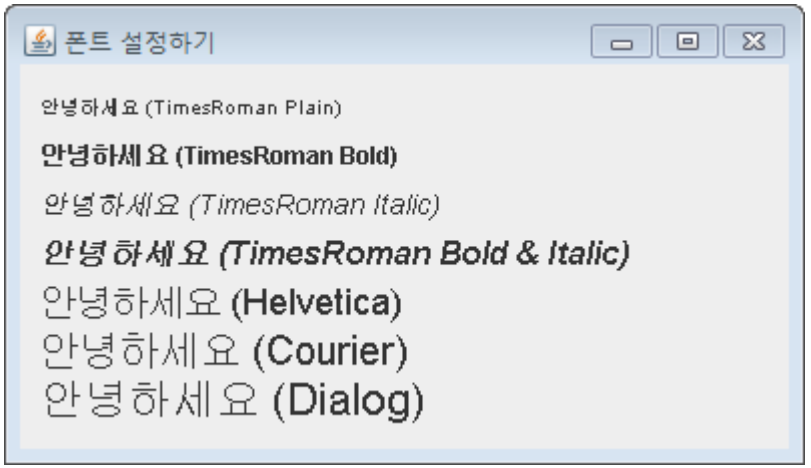
상수		설명
폰트 이름	String DIALOG	대화상자에서 주로 사용하는 폰트이다.
	String MONOSPACED	고정 폭을 가지는 폰트이다.
	String SERIF	삐침이 있는 가변 폭의 폰트이다.
	String SANS_SERIF	삐침이 없는 가변 폭의 폰트이다.
폰트	int BOLD	굵은체이다.
스타일	int ITALIC	이탤릭체이다.
	int PLAIN	일반 폰트이다.

- 예제 : [예제 16-3]

```java
package sec02;

import java.awt.GraphicsEnvironment;

public class Font1Demo {
    public static void main(String[] args) {
        GraphicsEnvironment e =
                GraphicsEnvironment.getLocalGraphicsEnvironment();

        String[] fontNames = e.getAvailableFontFamilyNames();

        for (String s : fontNames)
            System.out.println(s);
    }
}
```

```
Wingdings 3
굴림
굴림체
궁서
궁서체
나눔고딕
나눔고딕 ExtraBold
나눔고딕 Light
```

- 예제 : [예제 16-4]




```
1  package sec02;
2
3  import java.awt.Font;
4  import java.awt.Graphics;
5  import javax.swing.JFrame;
6  import javax.swing.JPanel;
7
8  public class Font2Demo extends JFrame {
9      Font2Demo() {
10         setTitle("폰트 설정하기");
11         add(new MyPanel());
12         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13         setSize(400, 230);
14         setVisible(true);
15     }
16
17     public static void main(String[] args) {
18         new Font2Demo();
19     }
20 }
21
22 class MyPanel extends JPanel {
23     protected void paintComponent(Graphics g) {
24         super.paintComponent(g);
25
26         Font f1 = new Font("TimesRoman", Font.PLAIN, 10);
27         Font f2 = new Font("TimesRoman", Font.BOLD, 12);
28         Font f3 = new Font("TimesRoman", Font.ITALIC, 14);
29         Font f4 = new Font("TimesRoman", Font.BOLD + Font.ITALIC, 16);
30         Font f5 = new Font("Helvetica", Font.PLAIN, 18);
31         Font f6 = new Font("Courier", Font.PLAIN, 20);
32         Font f7 = new Font("Dialog", Font.PLAIN, 22);
```

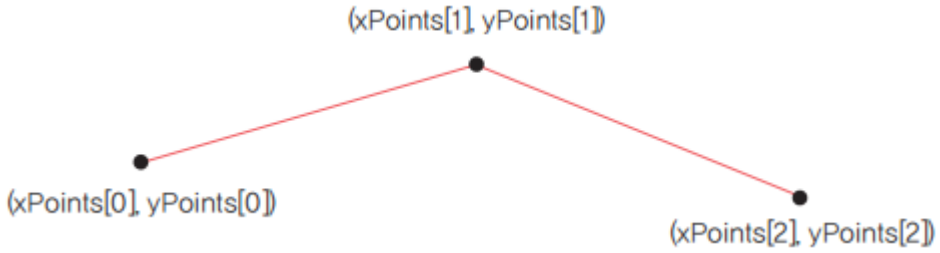
```
        g.setFont(f1);
        g.drawString("안녕하세요 (TimesRoman Plain)", 10, 25);
        g.setFont(f2);
        g.drawString("안녕하세요 (TimesRoman Bold)", 10, 50);
        g.setFont(f3);
        g.drawString("안녕하세요 (TimesRoman Italic)", 10, 75);
        g.setFont(f4);
        g.drawString("안녕하세요 (TimesRoman Bold & Italic)", 10, 100);
        g.setFont(f5);
        g.drawString("안녕하세요 (Helvetica)", 10, 125);
        g.setFont(f6);
        g.drawString("안녕하세요 (Courier)", 10, 150);
        g.setFont(f7);
        g.drawString("안녕하세요 (Dialog)", 10, 175);
    }
}
```

도형 그리기(1)

- Graphics 클래스가 제공하는 직선을 그리는 메서드

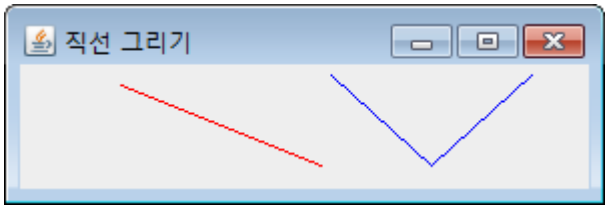
```
void drawLine(int x1, int y1, int x2, int y2)
void drawPolyline(int[] xPoints, int[] yPoints, int nPoints)
```

  - drawPolyLine() 메서드를 사용해 직선 그리기



- 예제 : [예제 16-5]

```
1  package sec02;
2
3  import java.awt.Color;
7
8  public class LineDemo extends JFrame {
9      int[] x = { 155, 205, 255 };
10     int[] y = { 5, 50, 5 };
11
12     LineDemo() {
13         setTitle("직선 그리기");
14
15         class MyPanel extends JPanel {
16             protected void paintComponent(Graphics g) {
17                 super.paintComponent(g);
18
19                 g.setColor(Color.RED);
20                 g.drawLine(50, 10, 150, 50);
21                 g.setColor(Color.BLUE);
22                 g.drawPolyline(x, y, 3);
23             }
24         }
25
26         add(new MyPanel());
27
28         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
29         setSize(300, 100);
30         setVisible(true);
31     }
32
33     public static void main(String[] args) {
34         new LineDemo();
35     }
36 }
```



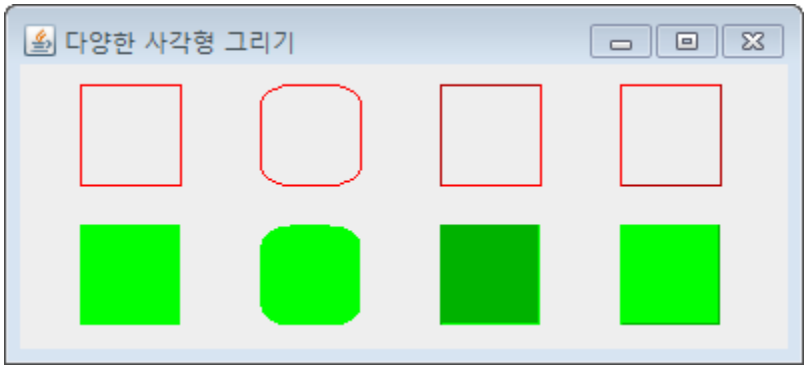
도형 그리기(2)

- 사각형을 그릴 수 있는 메서드

```
void drawRect(int x, int y, int width, int height)
void drawRoundRect(int x, int y, int width, int height, int arcWidth,
                   int arcHeight)
void draw3DRect(int x, int y, int width, int height, boolean raised)

void fillRect(int x, int y, int width, int height)
void fillRoundRect(int x, int y, int width, int height, int arcWidth,
                   int arcHeight)
void fill3DRect(int x, int y, int width, int height, boolean raised)
```

- 예제 : [예제 16-6]

```
 9 public class RectDemo extends JFrame {
10     RectDemo() {
11         setTitle("다양한 사각형 그리기");
12 
13         add(new JPanel() {
14             protected void paintComponent(Graphics g) {
15                 super.paintComponent(g);
16 
17                 g.setColor(Color.RED);
18                 g.drawRect(30, 10, 50, 50);
19                 g.drawRoundRect(120, 10, 50, 50, 30, 20);
20                 g.draw3DRect(210, 10, 50, 50, false);
21                 g.draw3DRect(300, 10, 50, 50, true);
22 
23                 g.setColor(Color.GREEN);
24                 g.fillRect(30, 80, 50, 50);
25                 g.fillRoundRect(120, 80, 50, 50, 30, 20);
26                 g.fill3DRect(210, 80, 50, 50, false);
27                 g.fill3DRect(300, 80, 50, 50, true);
28             }
29         });
30 
31         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
32         setSize(400, 180);
33         setVisible(true);
34     }
35 
36     public static void main(String[] args) {
37         new RectDemo();
38     }
39 }
```



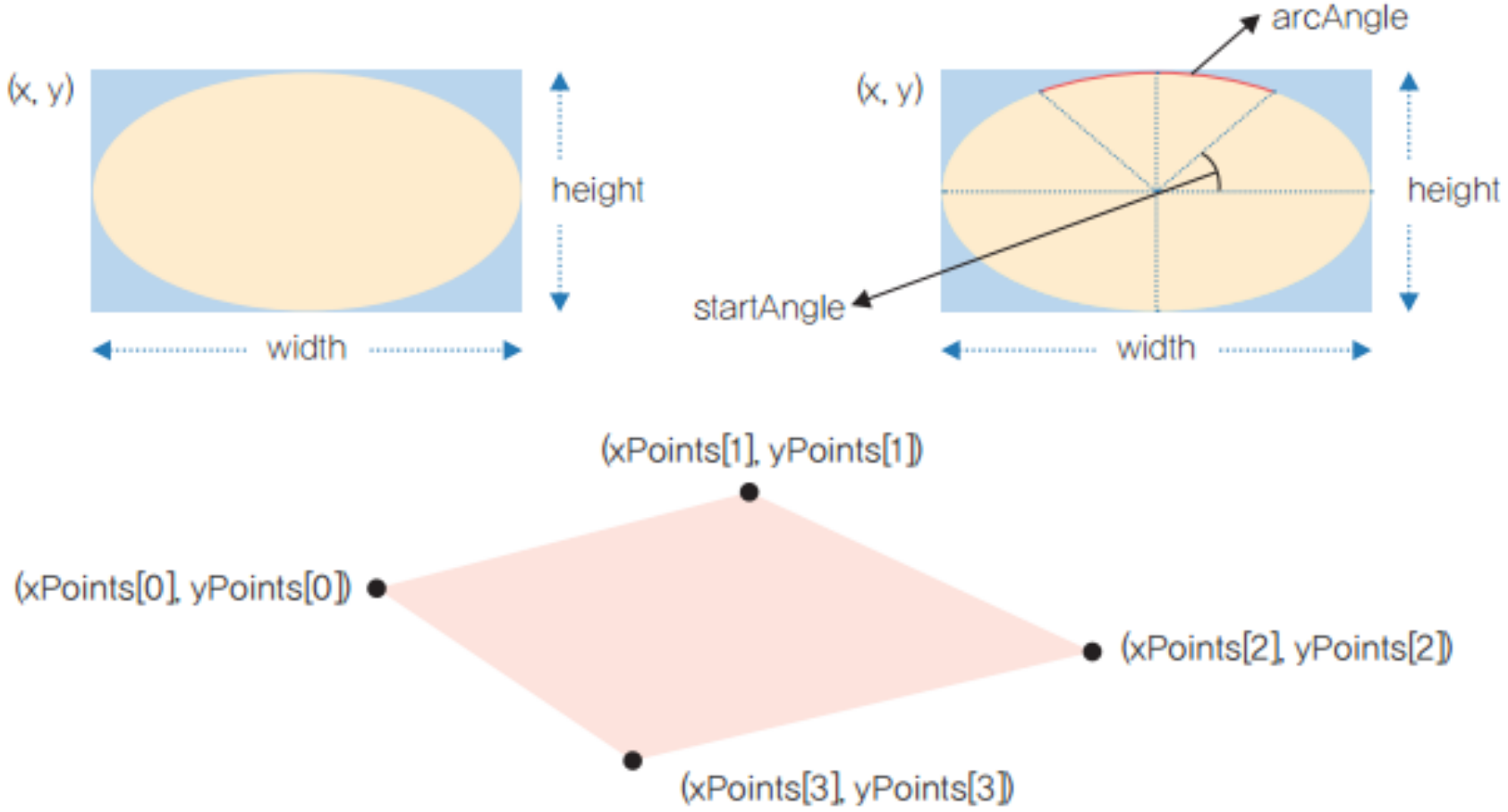
도형 그리기(3)

- 타원, 호, 다각형을 위한 메서드

```
void drawOval(int x, int y, int width, int height)
void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)
void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)

void fillOval(int x, int y, int width, int height)
void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)
void fillPolygon(int[] xPoints, int[] yPoints, int nPoints)
```

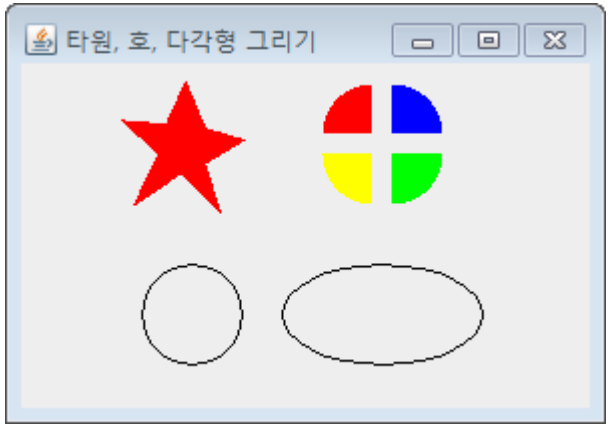
도형 그리기(4)

- 타원, 호, 다각형 그리기에서 매개변수의 의미



• 예제 : [예제 16-7]

```
public class StarDemo extends JFrame {
    StarDemo() {
        setTitle("타원, 호, 다각형 그리기");

        add(new JPanel() {
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);

                g.setColor(Color.RED);
                int x[] = { 82, 92, 112, 92, 100, 80, 55, 68, 49, 72, 82 };
                int y[] = { 8, 32, 38, 50, 75, 55, 72, 45, 28, 30, 8 };

                g.fillPolygon(x, y, x.length);
                g.fillArc(150, 10, 50, 50, 90, 90);
                g.setColor(Color.BLUE);
                g.fillArc(160, 10, 50, 50, 0, 90);
                g.setColor(Color.YELLOW);
                g.fillArc(150, 20, 50, 50, 180, 90);
                g.setColor(Color.GREEN);
                g.fillArc(160, 20, 50, 50, 270, 90);
                g.setColor(Color.BLACK);
                g.drawOval(60, 100, 50, 50);
                g.drawOval(130, 100, 100, 50);
            }
        });

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 210);
        setVisible(true);
    }

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



이미지 그리기(1)

- 주요 클래스
  - java.awt.image 클래스: 그래픽 이미지를 사각형 픽셀 이미지로 표현.
  - java.awt.image.BufferedImage 클래스 : Image의 자식 클래스로 애플리케이션이 직접 BufferedImage 객체를 생성할 수 있고, 이미지 데이터도 조작 가능

```
BufferedImage img;
try {
   img = ImageIO.read(new File("strawberry.jpg"));
} catch (IOException e) {
}
```

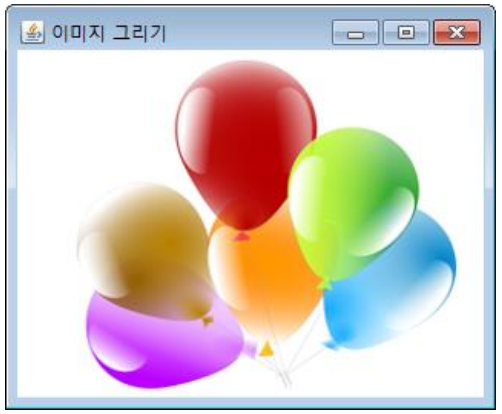
Image의 자식 클래스이다.

이미지 그리기(2)

- Graphics 클래스가 제공하는 이미지 그리기 메서드

```
// 원본 이미지와 동일한 크기로 그리기
boolean drawImage(Image img, int x, int y, Color bgcolor,
                  ImageObserver observer)
boolean drawImage(Image img, int x, int y, ImageObserver observer)
```

- 예제 : [예제 16-8]




```
12 public class ImageDemo extends JFrame {
13     ImageDemo() {
14         setTitle("이미지 그리기");
15 
16         class MyPanel extends JPanel {
17             BufferedImage img;
18 
19             public MyPanel() {
20                 try {
21                     img = ImageIO.read(new File("images/balloons.png"));
22                 } catch (IOException e) {
23                 }
24             }
25             //Image img = (new ImageIcon("images/balloons.png")).getImage();
26 
27             public void paintComponent(Graphics g) {
28                 super.paintComponent(g);
29                 g.drawImage(img, 0, 0, null);
30             }
31         }
32 
33         add(new MyPanel());
34         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
35         setSize(320, 265);
36         setVisible(true);
37     }
38 
39     public static void main(String[] args) {
40         new ImageDemo();
41     }
42 }
```

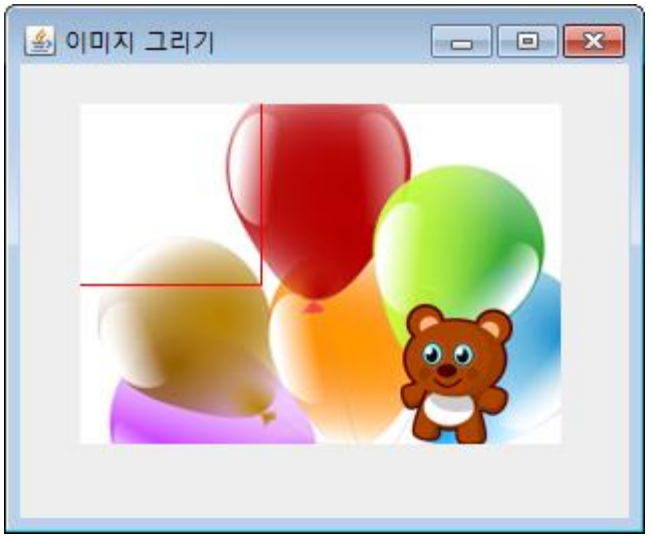
클리핑

- 실제 이미지의 필요한 부분을 꺼내기 위해 잘라내는 것
- 클리핑은 Graphics 클래스로 그리기를 할 때 지정된 특정 부분만 보이도록 하는 기능
- 클리핑을 위한 사각형 영역을 클리핑 영역이라고 부름
- 영역을 따로 설정하지 않으면 컴포넌트의 전체 영역이 클리핑 영역

메서드	설명
void clipRect(int x, int y, int w, int h)	기존 클리핑 영역과 (x, y) 좌표에서 w×h 사각형 영역의 교집합을 새로운 클리핑 영역으로 지정한다.
Shape getClip( )	클리핑 영역을 반환한다.
void setClip(int x, int y, int w, int h)	(x, y) 좌표에서 w×h 사각형 영역을 클리핑 영역으로 지정한다.

- 예제 : [예제 16-9]

```
13 public class ClipDemo extends JFrame {
14     ClipDemo() {
15         setTitle("이미지 그리기");
16 
17         class MyPanel extends JPanel {
18             BufferedImage balloons, bear;
19 
20             public MyPanel() {
21                 try {
22                     balloons = ImageIO.read(new File("images/balloons.png"));
23                     bear = ImageIO.read(new File("images/bear.png"));
24                 } catch (IOException e) { }
25             }
26 
27             public void paintComponent(Graphics g) {
28                 super.paintComponent(g);
29 
30                 g.setClip(30, 20, 240, 170);
31                 g.drawImage(balloons, 0, 0, null);
32                 g.setColor(Color.RED);
33                 g.drawRect(20, 10, 100, 100);
34                 g.drawImage(bear, 190, 120, null);
35             }
36         }
37 
38         add(new MyPanel());
39         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
40         setSize(320, 265);
41         setVisible(true);
42     }
43 
44     public static void main(String[] args) {
45         new ClipDemo();
46     }
47 }
```

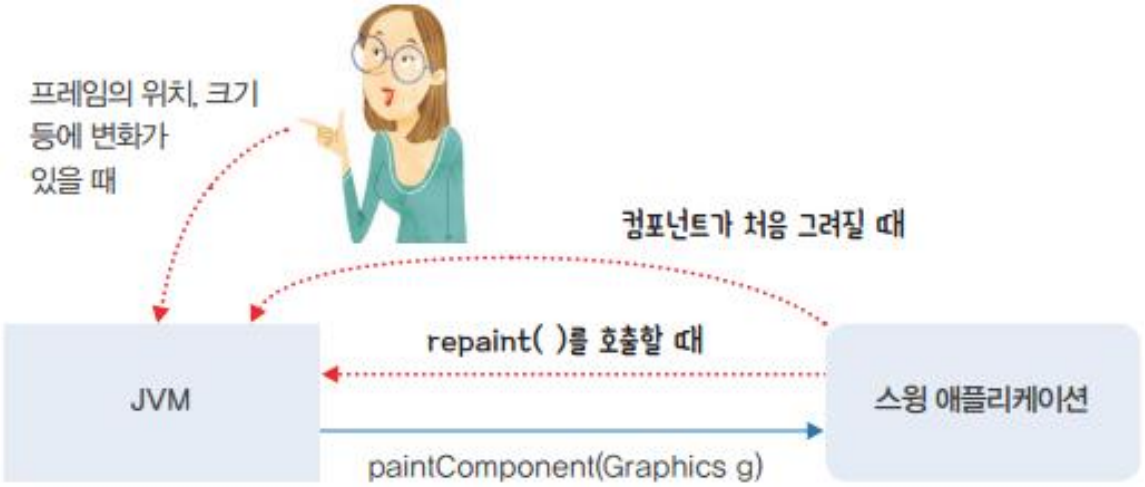


스윙의 그리기 과정

- 스윙 컴포넌트는 모두 JComponent 클래스의 자식 객체로 JComponent 클래스의 JVM이 paintComponent( ) 메서드를 호출해서 그린다.

- paintComponent( ) 메서드는 직접 호출할 수 없고, repaint() 메서드를 통해 JVM에 호출을 요청해야 함

- paintComponent( ) 메서드 요청



- 예제 : [예제 16-10]

```
13 public class RepaintDemo extends JFrame {
14     Random r = new Random();
15     List<Rectangle> list = new ArrayList<>();
16     MouseEvent e;
17 
18     public RepaintDemo() {
19         setTitle("클릭할 때마다 임의의 사각형 그리기");
20         add(new MousePanel());
21 
22         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23         setSize(400, 200);
24         setVisible(true);
25     }
26 
27     class MousePanel extends JPanel {
28         protected void paintComponent(Graphics g) {
29             super.paintComponent(g);
30 
31             addMouseListener(new MouseAdapter() {
32                 public void mousePressed(MouseEvent e) {
33                     if (RepaintDemo.this.e != null) {
34                         if (RepaintDemo.this.e.equals(e))
35                             return;
36                     }
37 
38                     int w = r.nextInt(20) + 5;
39                     int x = r.nextInt(350);
40                     int y = r.nextInt(150);
41                     list.add(new Rectangle(x, y, w, w));
42 
43                     repaint();
44 
45                     RepaintDemo.this.e = e;
46                 }
47             });
```

```
48
49              for (int i = 0; i < list.size() - 1; i++) {
50                  Rectangle r = list.get(i);
51                  int x = (int) r.getX();
52                  int y = (int) r.getY();
53                  int l = (int) r.getWidth();
54                  g.drawRect(x, y, l, l);
55              }
56          }
57      }
58
59      public static void main(String[] argv) {
60          new RepaintDemo();
61      }
62  }
```