(INTERMEDIATE) JAVA PROGRAMMING

21. Swing – More Graphics Programming

Chapter 12

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)

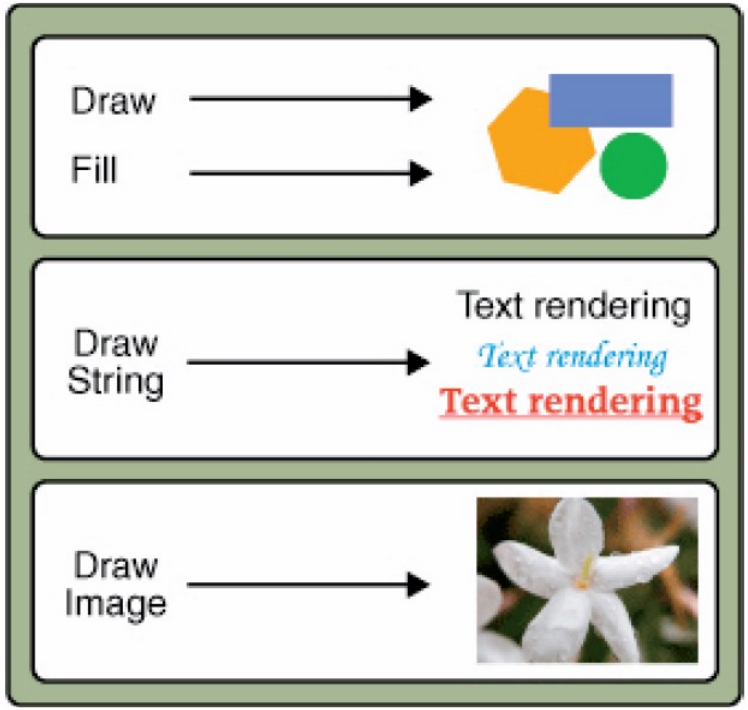
Java 2D를 사용하려면?

```
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

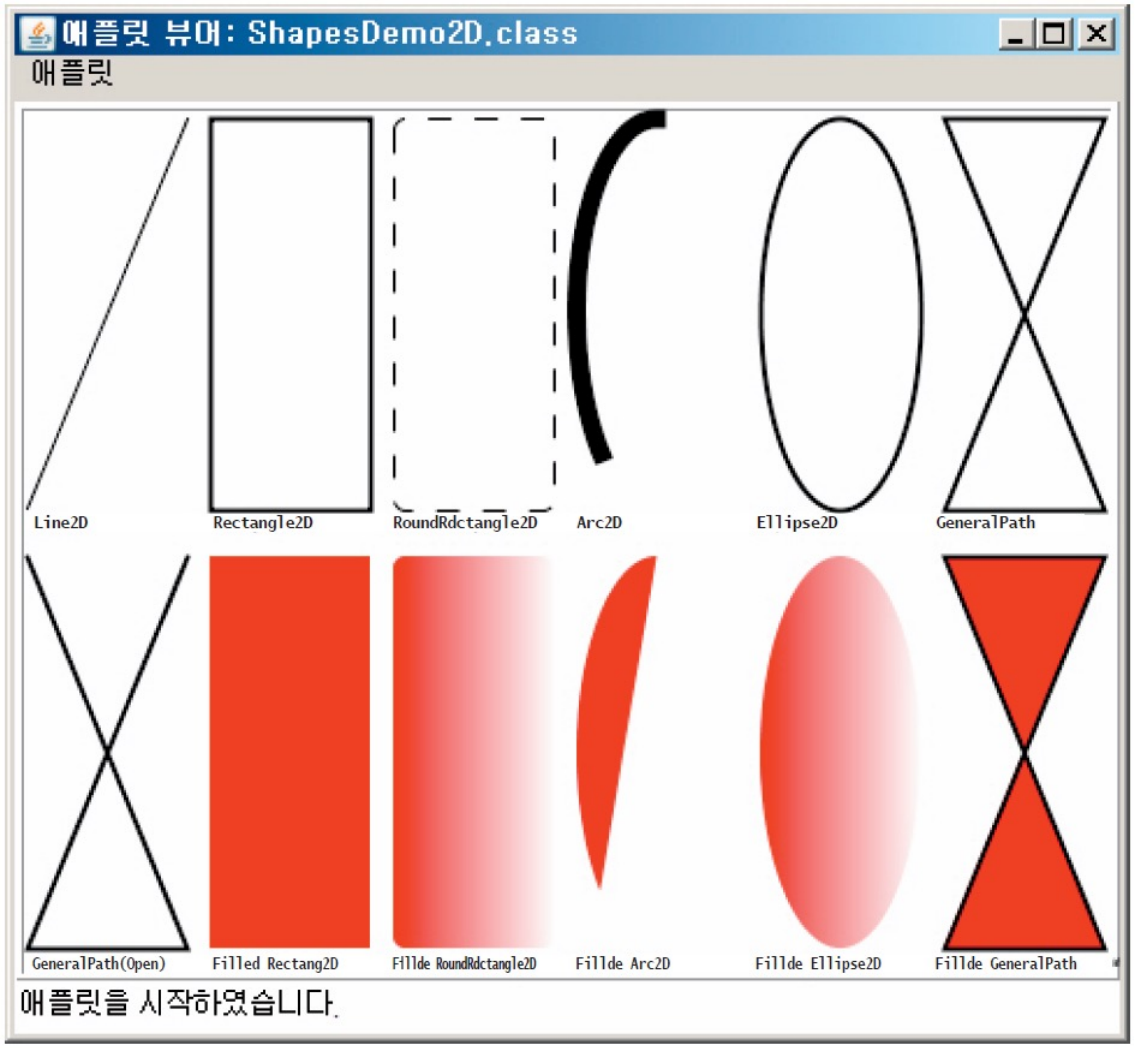
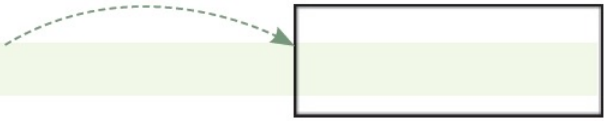


그림15-11. Java 2D를 이용한 형상 그리기(출저: java.sun.com)

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

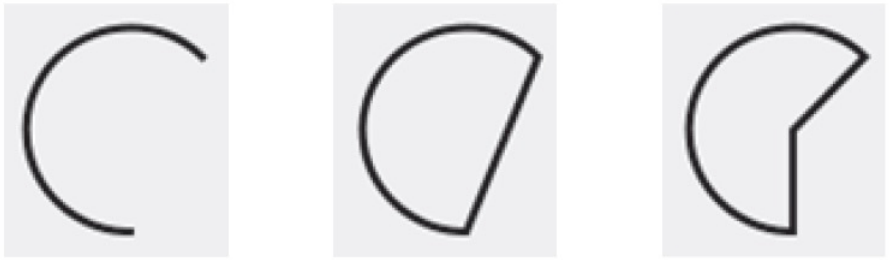
```
// 직선 객체를 생성하고 직선을 그린다.
g2.draw(new Line2D.Double(x1, y1, x2, y2));
```

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);
```

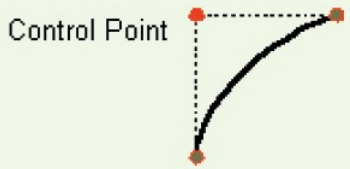


(a) Arc2D.OPEN (b) Arc2D.CHORD (c) Arc2D.PIE

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

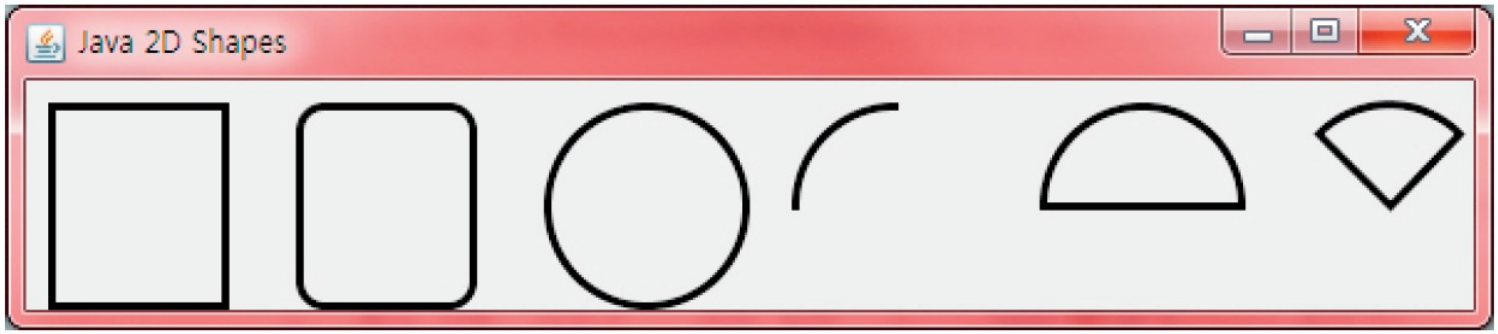
```
44
45          g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
46                RenderingHints.VALUE_ANTIALIAS_ON);
47
48          g2.setColor(Color.BLACK);
49          g2.setStroke(new BasicStroke(3));
50          for (Shape s : shapeArray)
51             g2.draw(s);
52       }
53    }
```

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);
```

Stroke and fill 참고문서: https://docs.oracle.com/javase/tutorial/2d/geometry/strokeandfill.html

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        ...
```

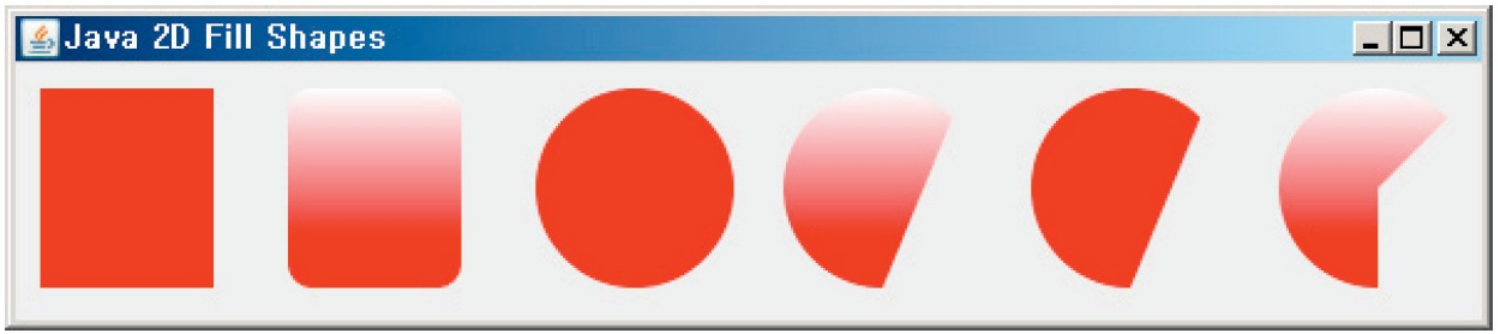
GradientPaint 객체를 생성한다. → 13

GradientPaint 객체로 채우는 색상을 지정 → 19

예제

```
22        }
23    }
```

실행결과



JAVA:
MORE GRAPHICS – IMAGE

스윙에서 이미지를 그리는 2 가지 방법

1. JLabel 컴포넌트로 이미지 그리기

```
ImageIcon image = new ImageIcon("images/apple.jpg");
JLabel label = new JLabel(image);
panel.add(label);
```

프로젝트폴더를 기준으로 상대위치

- 장점
  - 이미지 그리기가 간편하고 쉬운 장점
  - 이미지가 컴포넌트이므로 이벤트 발생
    - 이미지에 마우스 클릭하면 이벤트 받을 수 있음
- 단점
  - 이미지 크기 조절 불가 : 원본 크기로만 그리기

2. JPanel에 Graphics 메소드로 이미지 그리기

- 장점
  - 이미지의 원본 크기와 다르게, 이미지 일부분 등 그리기 가능
- 단점
  - 컴포넌트로 관리 되지 않음
    - 개발자가 상황에 따라 이미지의 위치나 크기 등을 적절히 조절해야 함
    - 이미지가 마우스를 클릭해도 이미지에 이벤트 발생하지 않음

Graphics로 이미지 그리기

- 총 6 개의 메소드
  - 원본 크기로 그리기
    - void drawImage(Image img, int x, int y, Color bgColor, ImageObserver observer)
    - void drawImage(Image img, int x, int y, ImageObserver observer)
  - 크기 조절하여 그리기
    - void drawImage(Image img, int x, int y, int width, int height, Color bgColor, ImageObserver observer)
    - void drawImage(Image img, int x, int y, int width, int height, ImageObserver observer)
  - 원본의 일부분을 크기 조절하여 그리기
    - void drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, Color bgColor, ImageObserver observer)
    - void drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, ImageObserver observer)

* ImageObserver는 이미지가 다 그려졌을 때, 통보를 받는 객체를 지정하는 매개변수
이미지는 경우에 따라 디코딩 등으로 인해 시간이 오래 걸릴 수 있기 때문에,
이미지 그리기가 완료되었는지 통보 받을 때 사용.
보통의 경우 this를 주거나 null을 주어 통보를 받지 않을 수 있음

이미지 그리기 샘플 코드

- 이미지 로딩 : Image 객체 생성

```
//그리고자 하는 이미지가 “image/image0.jpg”인 경우

ImageIcon icon = new ImageIcon(“image/image0.jpg”);
Image img = icon.getImage();
```

- 원본 이미지를 (20,20) 위치에 원본 크기로 그리기
  - 고정 크기임

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawImage(img, 20, 20, this);
}
```

- 원본 이미지를 100x100 크기로 조절하여 그리기
  - 고정 크기임

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawImage(img, 20, 20, 100, 100, this);
}
```

- 원본 이미지를 패널에 꽉 차도록 그리기
  - JPanel의 크기로 조절하여 그리기
  - 가변 크기임
    - JPanel의 크기가 변할 때마다 이미지의 크기도 따라서 변함

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawImage(img, 0, 0, getWidth(), getHeight(), this);
}
```

- 원본 이미지의 (50, 0)에서 (150,150) 사각형 부분을 JPanel의 (20,20)에서 (250,100) 영역에 그리기
  - 고정 크기임

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawImage(img, 20,20,250,100,50,0,150,150, this);
}
```

예제 12-6 : 이미지 그리기

JPanel을 상속받아 MyPanel을 만들고, 아래의 그림과 같이 "images/image0.jpg" 파일의 이미지를 패널의 (20, 20) 위치에 원본 크기로 그리는 프로그램을 작성하라.


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

public class GraphicsDrawImageEx1 extends JFrame {
    private MyPanel panel = new MyPanel();
    public GraphicsDrawImageEx1() {
        setTitle("원본 크기로 원하는 위치에 이미지 그리기");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(panel);
        setSize(300, 420);
        setVisible(true);
    }
    class MyPanel extends JPanel {
        private ImageIcon icon = new ImageIcon("images/image0.jpg");
        private Image img = icon.getImage(); // 이미지 객체
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.drawImage(img, 20, 20, this);
        }
    }
    public static void main(String [] args) {
        new GraphicsDrawImageEx1();
    }
}
```

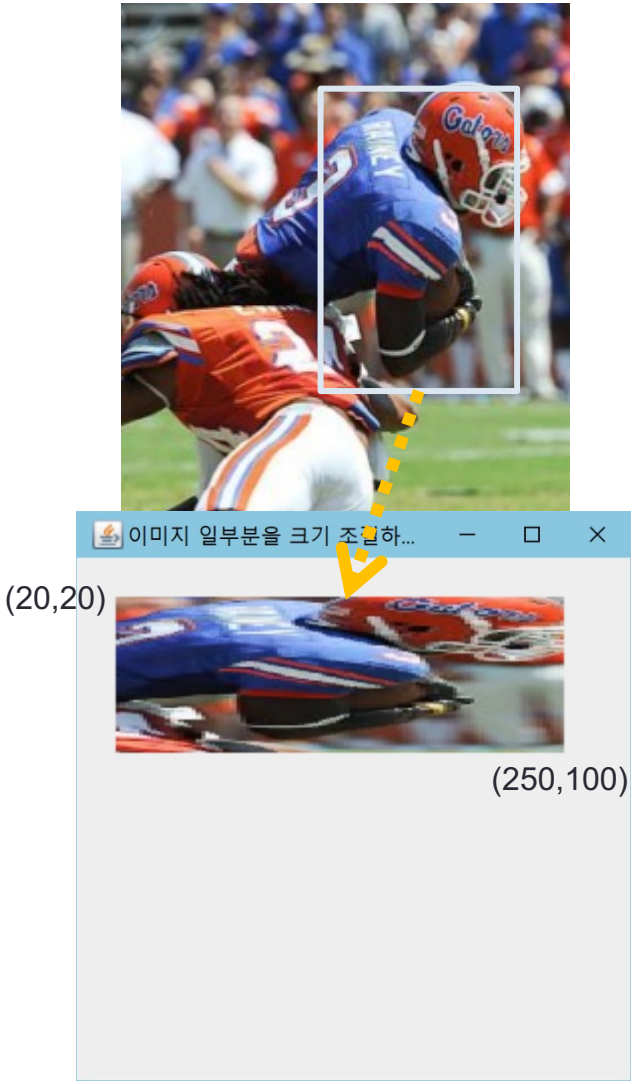
예제 12-7 : JPanel로 만든 패널에 꽉차도록 이미지 그리기

JPanel을 상속받아 MyPanel을 만들고, 아래의 그림과 같이 "images/image0.jpg" 파일의 이미지를 패널에 꽉 차도록 그리는 프로그램을 작성하라.


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

public class GraphicsDrawImageEx2 extends JFrame {
    private MyPanel panel = new MyPanel();
    public GraphicsDrawImageEx2() {
        setTitle("패널의 크기에 맞추어 이미지 그리기");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(panel);
        setSize(200, 300);
        setVisible(true);
    }
    class MyPanel extends JPanel {
        private ImageIcon icon = new ImageIcon("images/image0.jpg");
        private Image img = icon.getImage(); // 이미지 객체
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.drawImage(img, 0, 0, getWidth(), getHeight(), this);
        }
    }
    public static void main(String [] args) {
        new GraphicsDrawImageEx2();
    }
}
```

예제 12-7 : 이미지의 일부분 크기조절하여 그리기




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

public class GraphicsDrawImageEx3 extends JFrame {
    private MyPanel panel = new MyPanel();

    public GraphicsDrawImageEx3() {
        setTitle("이미지 일부분을 크기 조절하여 그리기");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(panel);
        setSize(300, 300);
        setVisible(true);
    }
    class MyPanel extends JPanel {
        private ImageIcon icon = new ImageIcon("images/image0.jpg");
        private Image img = icon.getImage();
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.drawImage(img, 20, 20, 250, 100, 100, 50, 200, 200, this);
        }
    }
    public static void main(String [] args) {
        new GraphicsDrawImageEx3();
    }
}
```

참고 : Image file 바이트코드의 상대경로로 읽기

- bin 폴더아래 java class 파일이 위치한 곳에 image파일을 가져다 놓고, 다음과 같은 방법으로 이미지를 로드 할 수 있다.

```
URL url = getClass().getResource("Image.jpg");
BufferedImage image = ImageIO.read(url);
```

<경로의 위치는 byte code(.class) 위치를 기준으로 함>

숙제 제출시 반드시 url을 써서 바이트 코드 상대 경로로 입력할 것 (JAR 파일 제출 시 필수!!!)

클리핑

- 클리핑(Clipping)이란?
  - 클리핑 영역에서만 그래픽이 이루어지도록 하는 기능
    - 클리핑 영역 : 하나의 사각형 영역
  - 클리핑이 작동하는 그래픽 기능
    - 그리기, 칠하기, 이미지 그리기, 문자열 출력 등에서 모드 클리핑 작동

클리핑이 설정되지 않아서, 전체 영역에 그려지는 경우(전체가 클릭핑 영역)

특정 사각형 영역을 클리핑 영역으로 설정한 경우

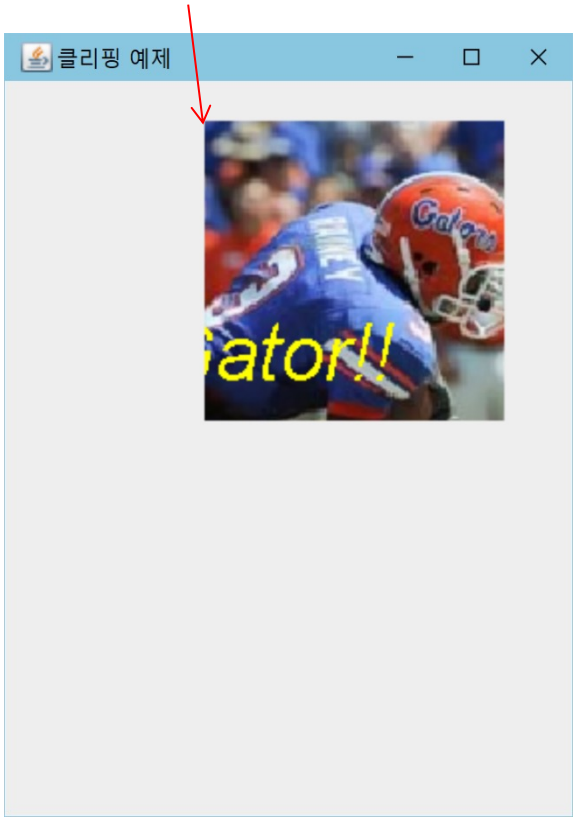
클리핑 영역 설정

- Graphics의 클리핑 메소드
  - void setClip(int x, in y, int w, int h)
    - 그래픽 대상 컴포넌트의 (x, y) 위치에서 wxh 의 사각형 영역을 클리핑 영역으로 지정
  - void clipRect(int x, in y, int w, int h)
    - 기존 클리핑 영역과 지정된 사각형 영역((x,y)에서 wxh의 영역)의 교집합 영역을 새로운 클리핑 영역으로 설정
    - clipRect()이 계속 불리게 되면 클리핑 영역을 계속 줄어들게 됨

예제 12-9 : 클리핑 영역에 그리기

패널에 (100, 20)에서 150×150 크기로 클리핑 영역을 설정하고 문자열과 이미지를 출력하여 클리핑 여부를 확인하여보라.

클리핑 영역 : (50,20)에서 150x150 사각형 영역




```
import javax.swing.*;
import java.awt.*;
public class GraphicsClipEx extends JFrame {
    private MyPanel panel = new MyPanel();
    public GraphicsClipEx() {
        setTitle("클리핑 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(panel);
        setSize(300, 400);
        setVisible(true);
    }
    class MyPanel extends JPanel {
        private ImageIcon icon = new ImageIcon("images/image0.jpg");
        private Image img = icon.getImage(); // 이미지 객체
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setClip(100, 20, 150, 150);
            g.drawImage(img, 0, 0, getWidth(), getHeight(), this);
            g.setColor(Color.YELLOW);
            g.setFont(new Font("Arial", Font.ITALIC, 40));
            g.drawString("Go Gator!!", 10, 150);
        }
    }
    public static void main(String [] args) {
        new GraphicsClipEx();
    }
}
```