제 12 장 그래픽

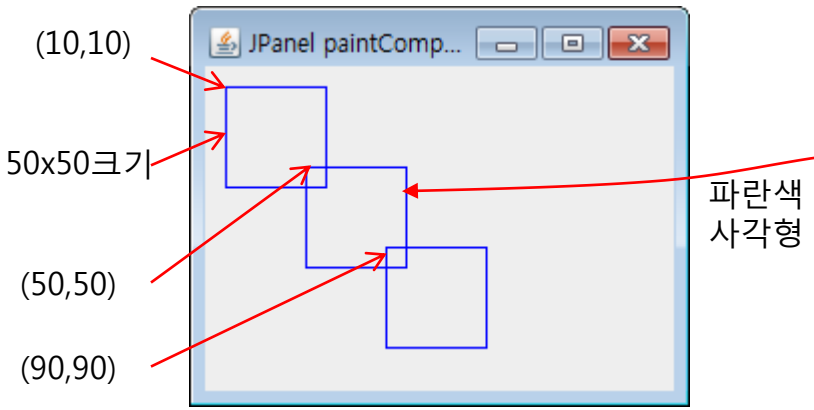
스윙 컴포넌트 그리기, paintComponent()

- 스윙의 기본 철학
  - 모든 컴포넌트는 자신의 모양을 스스로 그린다.
  - 컨테이너는 자신을 그린 후 자식들에게 그리기 지시
- public void paintComponent(Graphics g)
  - 스윙 컴포넌트가 자신의 모양을 그리는 메소드
  - JComponent의 메소드
    - 모든 스윙 컴포넌트가 이 메소드를 가지고 있음
  - 컴포넌트가 그려져야 하는 시점마다 호출
    - 크기가 변경되거나, 위치가 변경되거나 컴포넌트가 가려졌던 것이 사라지는 등
- Graphics 객체
  - java.awt.Graphics
  - 컴포넌트를 위한 그래픽 컨텍스트를 가지는 객체
    - 그리기에 필요한 모든 정보와 메소드 제공
    - 색 지정, 도형 그리기, 클리핑, 이미지 그리기 등을 위한 메소드 제공
- 사용자가 원하는 모양을 그리고자 할 때
  - paintComponent(Graphic g)를 오버라이딩하여 재작성

```
public void paintComponent(Graphics g) {
   super.paintComponent(g);
   ... 필요한 그리기 코드 작성
}
```

예제 12-1 : JPanel을 상속받아 도형 그리기

- JPanel의 용도
  - 사용자가 그래픽을 통해 다양한 UI를 창출하는 일종의 캔버스




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

public class paintJPanelEx extends JFrame {
   Container contentPane;
   paintJPanelEx() {
      setTitle("JPanel paintComponent 예제");

   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      contentPane = getContentPane();
      MyPanel panel = new MyPanel();
      contentPane.add(panel , BorderLayout.CENTER);
      setSize(250,200);
      setVisible(true);
   }

   class MyPanel extends JPanel {
      public void paintComponent(Graphics g) {
         super.paintComponent(g);
         g.setColor(Color.BLUE);
         g.drawRect(10,10, 50, 50);
         g.drawRect(50,50, 50, 50);
         g.drawRect(90,90, 50, 50);
      }
   }

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

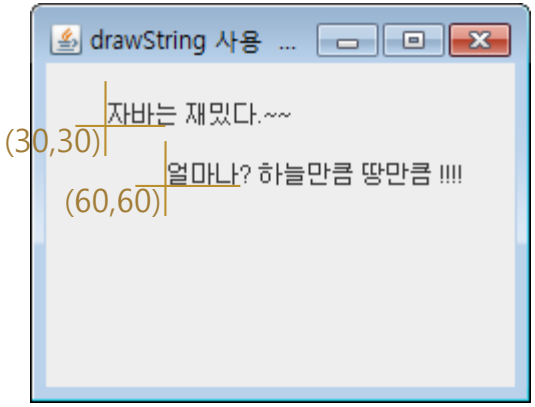
자바의 그래픽 좌표 시스템

Graphics

- Graphics의 기능
  - 색상 선택하기
  - 문자열 출력
  - 도형 그리기
  - 도형 칠하기
  - 이미지 출력
  - 클리핑
- 문자열 그리기를 위한 Graphics 메소드
  - void drawString(String str, int x, int y)
    - str 문자열을 (x,y) 영역에 출력한다. 이때 컨텍스트 내의 현재 색과 현재 폰트로 출력한다.

예제 12-2 : drawString() 메소드를 이용하여 문자열 출력하기

JPanel을 상속받아 paintComponent()를 오버라이딩하고 drawString() 메소드를 사용하여 다음 그림과 같이 패널 내의 (30, 30)과 (60, 60)에 각각 문자열을 출력하는 스윙 프로그램을 작성하라.




```
import javax.swing.*;
import java.awt.*;
public class GraphicsDrawStringEx extends JFrame {
    Container contentPane;
    GraphicsDrawStringEx() {
        setTitle("drawString 사용  예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        MyPanel panel = new MyPanel();
        contentPane.add(panel , BorderLayout.CENTER);
        setSize(250,200);
        setVisible(true);
    }

    class MyPanel extends JPanel {
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.drawString("자바는 재밌다.~~", 30,30);
            g.drawString("얼마나? 하늘만큼 땅만큼 !!!!", 60, 60);
        }
    }

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

Color와 Font 클래스

- Color
  - java.awt.Color,
  - 하나의 색을 표현하는 클래스
  - Red, Green, Blue 의 3 성분으로 구성되며 각 성분의 크기는 0-255(8비트)
- 생성자
  - Color(int r, int g, int b)
  - red(r), green(g), blue(b) 값, sRGB 색 생성
    - new Color(255, 0, 0) ;// 완전 빨강색
  - Color(int rgb)
    - rgb 정수 값은 총 32비트 중 하위 24 비트만이 유효하고 0x00rrggbb로 표현된다.
    - 하위 8비트는 blue, 그 다음 상위 8 비트는 green, 그 다음 8 비트는 blue 성분을 표시한다.
    - new Color(0x0000ff00); // 완전 초록
- 다른 생성 방법
  - Color.BLUE 등의 static 상수 활용

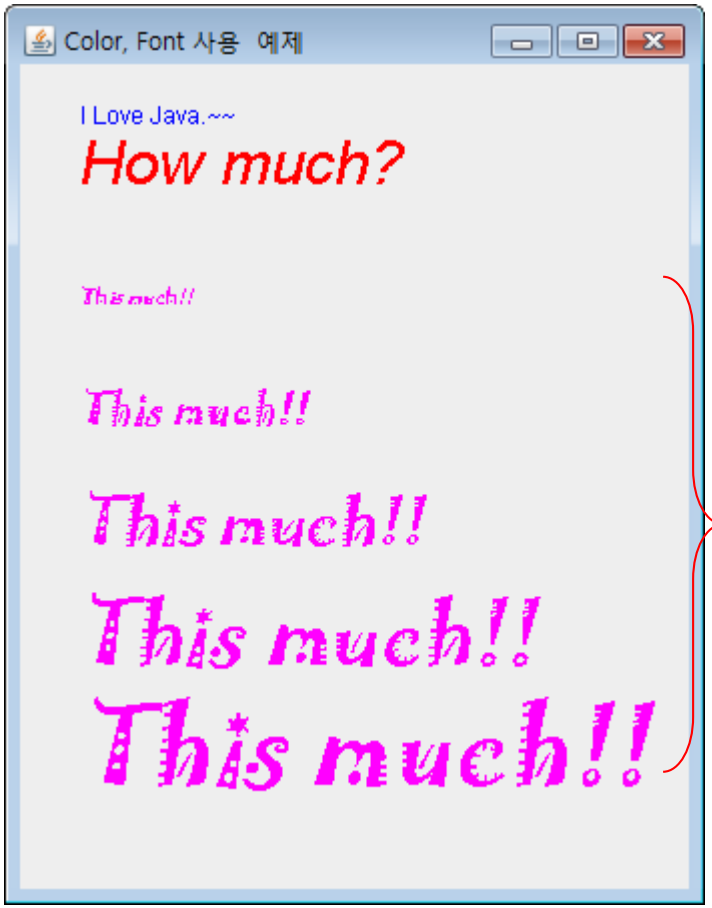
- Font
  - java.awt.Font, 폰트를 표현하는 클래스
- 생성자
  - Font(String fontFace, int style, int size)
    - fontFace는 "고딕체", "Arial" 등
    - style은 Font.BOLD, Font.ITALIC , Font.PLAIN 셋 중 하나
    - size는 픽셀 단위의 크기
- Graphics 객체에서 색상과 폰트 설정
  - void setColor(Color color)
    - 칠할 색을 color로 지정
  - void setFont(Font font)
    - 폰트를 font로 지정

```
Graphics g;
g.setColor(new Color(255, 0, 0));          // 빨간색을 그래픽 색으로 설정
g.setColor(new Color(0x0000ff00));         // 초록색을 그래픽 색으로 설정
g.setColor(Color.YELLOW);                  // 노란색을 그래픽 색으로 설정
```

```
Graphics g;
Font f = new Font("Arial", Font.ITALIC, 30);
g.setFont(f);
g.setColor(Color.RED);
g.drawString("How much", 30,30);
```

예제 12-3 : Color와 Font를 활용한 문자열 그리기

Color와 Font 클래스를 이용하여 그림과 같이 출력되는 패널을 작성하라.

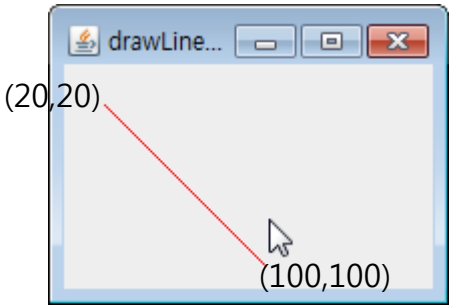



```
import javax.swing.*;
import java.awt.*;

public class GraphicsColorFontEx extends JFrame {
    Container contentPane;
    GraphicsColorFontEx() {
        setTitle("Color, Font 사용  예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        MyPanel panel = new MyPanel();
        contentPane.add(panel , BorderLayout.CENTER);
        setSize(350, 450);
        setVisible(true);
    }

    class MyPanel extends JPanel {
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.BLUE);
            g.drawString("I Love Java.~~", 30,30);
            g.setColor(new Color(255, 0, 0));
            g.setFont(new Font("Arial", Font.ITALIC, 30));
            g.drawString("How much?", 30, 60);
            g.setColor(new Color(0x00ff00ff));
            for(int i=1; i<=5; i++) {
                g.setFont(new Font("Jokerman", Font.ITALIC, i*10));
                g.drawString("This much!!", 30, 60+i*60);
            }
        }
    }

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

Graphics의 도형 그리기 메소드

- void drawLine(int x1, int y1, int x2, int y2)
  - (x1,y1)에서 (x2,y2) 까지 선을 그린다.
- void drawOval(int x1, int y1, int w, int h)
  - (x1,y1)에서 wxh 크기의 사각형에 내접하는 타원 그린다.
- void drawRect(int x1, int y1, int w, int h)
  - (x1,y1)에서 wxh 크기의 사각형을 그린다.
- void drawRoundRect(int x1, int y1, int w, int h, int arcWidth, int arcHeight)
  - (x1,y1)에서 wxh 크기의 사각형을 그리고, 4 개의 모서리는 원으로 처리
  - arcWidth는 모서리의 원 수평 반지름, arcHeight는 수직 반지름

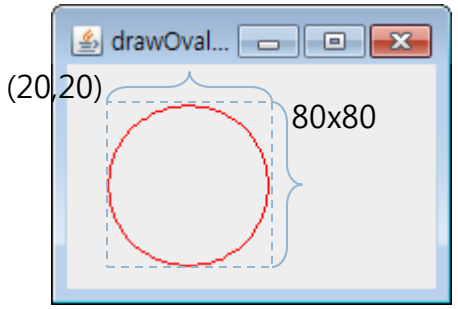
선 그리기 사례



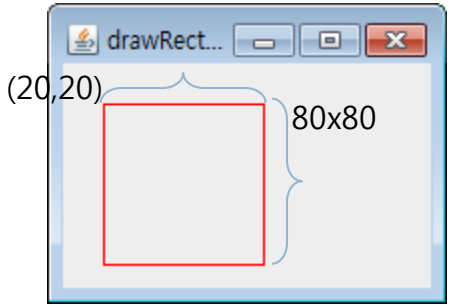

```
import javax.swing.*;
import java.awt.*;

public class GraphicsDrawLineEx extends JFrame {
   Container contentPane;
   GraphicsDrawLineEx() {
      setTitle("drawLine 사용  예제");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      contentPane = getContentPane();
      MyPanel panel = new MyPanel();
      contentPane.add(panel, BorderLayout.CENTER);
      setSize(200, 150);
      setVisible(true);
   }

   class MyPanel extends JPanel {
      public void paintComponent(Graphics g) {
         super.paintComponent(g);
         g.setColor(Color.RED);
         g.drawLine(20,20, 100, 100);
      }
   }

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

다른 도형 그리기 사례

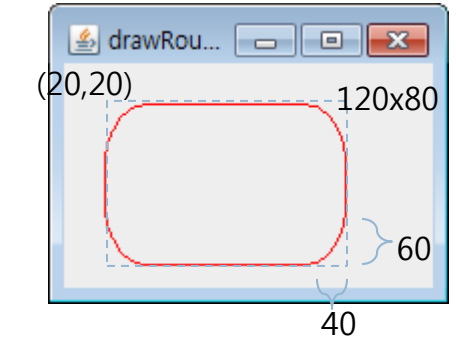
```
class MyPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.RED);
        g.drawOval(20,20,80,80);
    }
}
```




```
class MyPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.RED);
        g.drawRect(20,20,80,80);
    }
}
```




```
class MyPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.RED);
        g.drawRoundRect(20,20,120,80,40,60);
    }
}
```

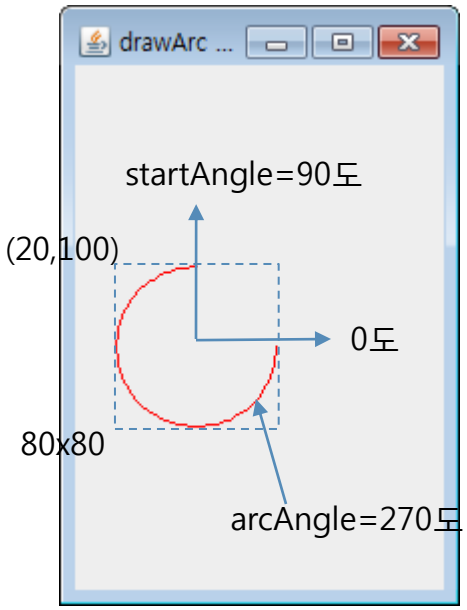


Graphics의 원호와 폐다각형 그리기 메소드

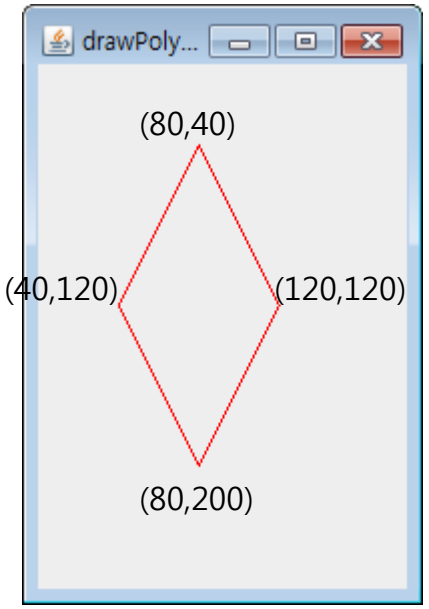
- void drawArc(int x, int y, int w, int h, int startAngle, int arcAngle)
  - (x1,y1)에서 wxh 크기의 사각형에 내접하는 원호를 그린다.
  - 원호의 시작 각도는 startAngle, 원호 각도는 arcAngle
  - 원호는 3시 방향을 0도의 기점에서 시작
  - arcAngle이 양수이면 반시계방향, 음수이면 시계방향으로 그리기
- void drawPolygon(int []x, int []y, int n)
  - 연결된 폐다각형을 그리며, 다각형의 점들은 x, y 배열에서 지정됨.
  - (x[0], y[0]),,(x[n-1], y[n-1])의 총 n 개의 점을 연결함

원호와 폐다각형 그리기 사례

```
class MyPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.RED);
        g.drawArc(20,100,80,80,90,270);
    }
}
```




```
class MyPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.RED);

        int []x = {80,40,80,120};
        int []y = {40,120,200,120};
        g.drawPolygon(x, y, 4);
    }
}
```



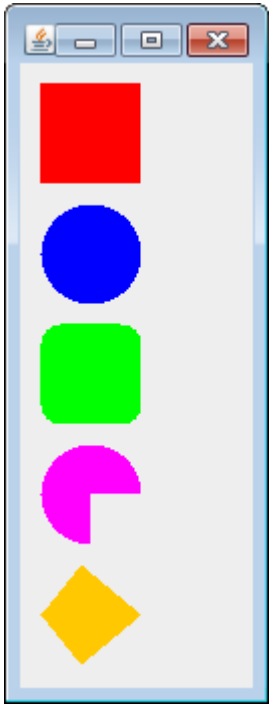
Graphics의 도형 칠하기

- 도형 칠하기
  - 도형을 그리고 그 내부를 칠하는 기능
  - 도형의 외곽선과 내부를 따로 칠하는 기능은 없다.
  - 도형 칠하기를 위한 메소드는 도형 그리기 메소드 명에서 draw 를 fill로 대치하면 된다. 인자는 동일함
    - 예) drawRect() -> fillRect(), drawArc() -> fillArc()

- 칠하기 메소드
  - void fillOval(int x1, int y1, int w, int h)
  - void fillRect(int x1, int y1, int w, int h)
  - void fillRoundRect(int x1, int y1, int w, int h, int arcWidth, int arcHeight)
  - void fillArc(int x, int y, int w, int h, int startAngle, int arcAngle)
  - void fillPolygon(int []x, int []y, int n)

예제 12-4 : 도형 칠하기 예

Graphics의 칠하기 메소드를 이용하여 그림과 같은 패널을 작성하라.



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

public class GraphicsFillEx extends JFrame {
    Container contentPane;
    GraphicsFillEx() {
        setTitle("fillXXX 사용  예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        MyPanel panel = new MyPanel();
        contentPane.add(panel, BorderLayout.CENTER);
        setSize(100, 350);
        setVisible(true);
    }
    class MyPanel extends JPanel {
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.RED);
            g.fillRect(10,10,50,50);
            g.setColor(Color.BLUE);
            g.fillOval(10,70,50,50);
            g.setColor(Color.GREEN);
            g.fillRoundRect(10,130,50,50, 20,20);
            g.setColor(Color.MAGENTA);
            g.fillArc(10, 190, 50, 50, 0, 270);
            g.setColor(Color.ORANGE);
            int []x ={30,10,30,60};
            int []y ={250,275,300,275};
            g.fillPolygon(x, y, 4);
        }
    }
    public static void main(String [] args) {
        new GraphicsFillEx();
    }
}
```

스윙에서 이미지를 그리는 2 가지 방법

1. JLabel 컴포넌트를 이용한 이미지 출력
   - JLabel 컴포넌트가 이미지를 자신의 영역에 그린다.

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

   - 장점 : 이미지 그리기가 간편하고 쉬운 장점
   - 단점 : 이미지의 원본 크기대로 그리므로 이미지의 크기 조절 불가능
2. JPanel에 Graphics 메소드를 이용한 이미지 출력
   - 장점 : 이미지의 원본 크기와 다르게 그리기 가능
   - 단점 : Graphics.drawImage()를 호출하여 개발자가 직접 이미지 그리기 필요

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)

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);
}
```

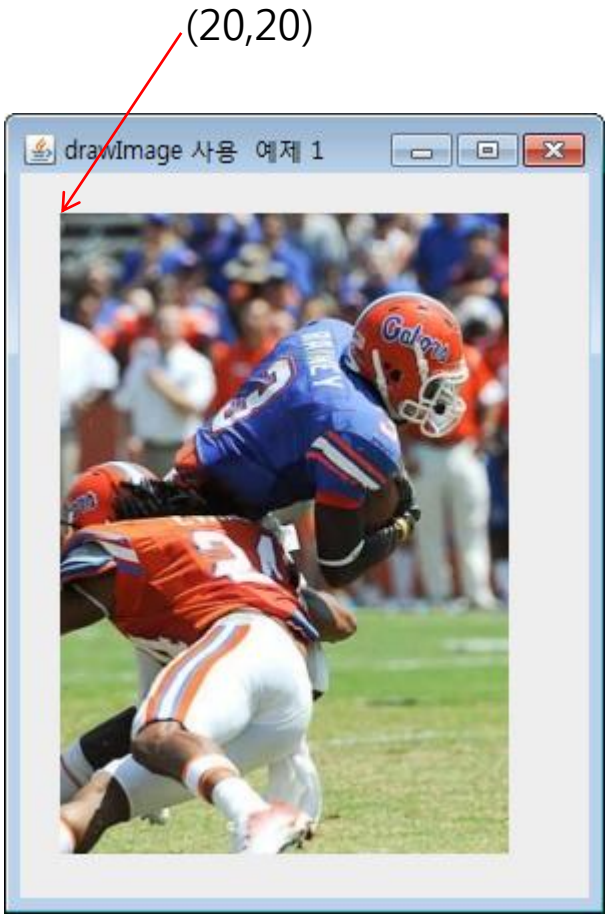
- 원본 이미지를 패널에 꽉차도록 그리기
  - JPanel의 크기로 조절하여 그리기
  - 가변 크기임
    - JPanel의 크기가 변할 때마다 이미지의 크기도 따라서 변함

```
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 12-5 : 원본 이미지 그리기




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

public class GraphicsDrawImageEx1 extends JFrame {
    Container contentPane;
    GraphicsDrawImageEx1() {
        setTitle("drawImage 사용  예제 1");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        MyPanel panel = new MyPanel();
        contentPane.add(panel, BorderLayout.CENTER);
        setSize(300, 400);
        setVisible(true);
    }

    class MyPanel extends JPanel {
       ImageIcon imageIcon = new ImageIcon("images/image0.jpg");
        Image image = imageIcon.getImage();

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

    public static void main(String [] args) {
        new GraphicsDrawImageEx1();
    }
}
```

예제 12-6 : JPanel 크기로 이미지 그리기


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

public class GraphicsDrawImageEx2 extends JFrame {
    Container contentPane;
    GraphicsDrawImageEx2() {
        setTitle("drawImage 사용  예제 2");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        MyPanel panel = new MyPanel();
        contentPane.add(panel, BorderLayout.CENTER);
        setSize(300, 400);
        setVisible(true);
    }

    class MyPanel extends JPanel {
        ImageIcon imageIcon = new  ImageIcon("images/image0.jpg");
        Image image = imageIcon.getImage();

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

    public static void main(String [] args) {
        new GraphicsDrawImageEx2();
    }
}
```

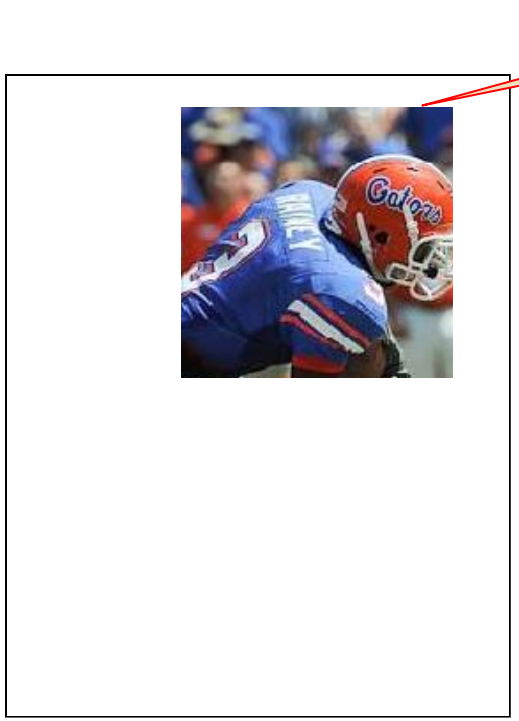
예제 12-7 : 이미지의 일부분을 크기 조절하여 그리기


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

public class GraphicsDrawImageEx3 extends JFrame {
    Container contentPane;
    GraphicsDrawImageEx3() {
        setTitle("drawImage 사용  예제 3");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        MyPanel panel = new MyPanel();
        contentPane.add(panel, BorderLayout.CENTER);
        setSize(300, 400);
        setVisible(true);
    }

    class MyPanel extends JPanel {
        ImageIcon imageIcon = new  ImageIcon("images/image0.jpg");
        Image image = imageIcon.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();
    }
}
```

클리핑

- 클리핑(Clipping)이란?
  - 그리기(그리기, 칠하기, 이미지 그리기, 문자열 출력 등)에 의해 그래픽 대상 컴포넌트 내 일정 영역에 있는 부분만 보이도록 하는 기능
  - 그래픽 대상 컴포넌트 내 클리핑 영역에서만 그리기 연산 진행
  - 클리핑 영역 : 하나의 사각형 영역

클리핑이 설정되지 않아서, 전체 영역이 클리핑 영역으로 설정된 경우



특정 사각형 영역을 클리핑 영역으로 설정된 경우

클리핑 영역 설정 메소드

- 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 12-8 : 클리핑 예제

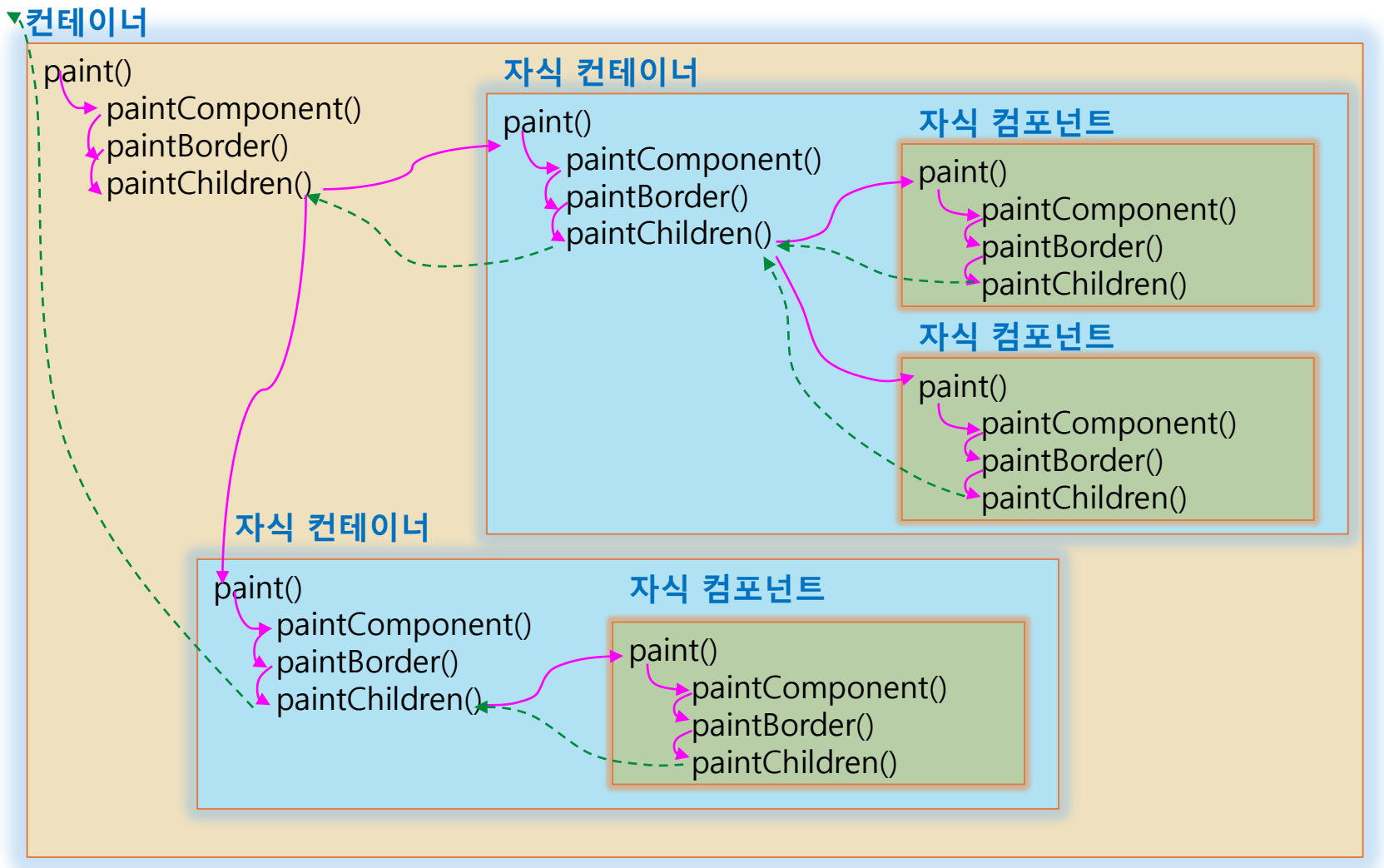
클리핑 영역 : (50,20)에서 150x150 사각형 영역


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

public class GraphicsClipEx extends JFrame {
    Container contentPane;
    GraphicsClipEx() {
        setTitle("클리핑 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        MyPanel panel = new MyPanel();
        contentPane.add(panel, BorderLayout.CENTER);
        setSize(300, 400);
        setVisible(true);
    }

    class MyPanel extends JPanel {
        ImageIcon icon = new ImageIcon("images/image0.jpg");
        Image img = icon.getImage();

        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setClip(50, 20, 150, 150);
            g.drawImage(img, getWidth(), getHeignt(), this);
            g.setColor(Color.BLUE);
            g.setFont(new Font("SanSerif", Font.ITALIC, 40));
            g.drawString("Ji Sung Park", 10, 150);
        }
    }

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

스윙의 페인팅

- JComponent.paint()
  - 컴포넌트 자신과 모든 자손 그리기
- JComponent.paint()는 다음 메소드를 순서대로 호출
  - JComponent.paintComponent()
    - 컴포넌트 자신의 모양 그리기
  - JComponent.paintBorder()
    - 컴포넌트의 외곽 그리기
  - JComponent.paintChildren()
    - 컴포넌트의 자식들 그리기
- 개발자가 paintComponent()를 직접 호출하면 안됨
  - paintComponent()는 페인팅 메카니즘에 의해 자동으로 호출됨

스윙 컴포넌트가 그려지는 과정



repaint() 메소드

- Component의 메소드
- repaint() 를 호출해야 하는 경우
  - 개발자가 컴포넌트를 다시 그리고자 하는 경우
    - 프로그램 내에서 컴포넌트의 모양과 위치를 변경하였지만 바로 화면에 반영되지 않는다.
    - 이 이유는 이 컴포넌트가 다시 그려져야 그 때 변경된 위치에 변경된 모양으로 출력됨
    - repaint()는 지금 당장 컴포넌트를 다시 그리도록 지시함

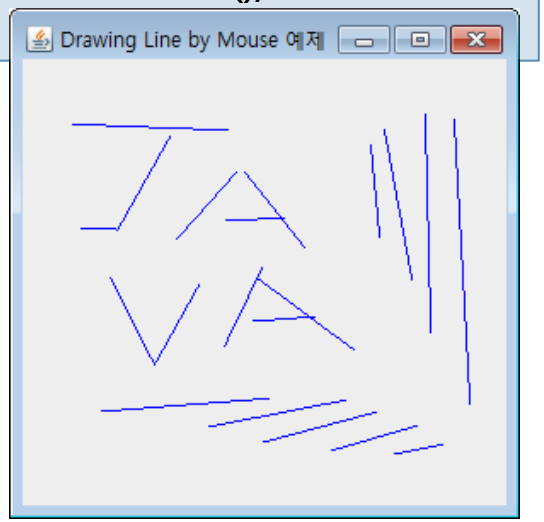
```
component.repaint();
```

- 부모 컴포넌트부터 다시 그리는 것이 좋음
  - 특히 컴포넌트의 위치가 변경된 경우 repaint()가 불려지면 이 컴포넌트는 새로운 위치에 다시 그려지지만 이전의 위치에 있던 자신의 모양이 남아 있기 때문에 부모 컴포넌트의 repaint()를 호출하는 것이 좋음

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

예제 12-9 : 마우스를 이용한 선 그리기(repaint() 사용)

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

public class GraphicsDrawLineMouseEx extends JFrame
{
    Container contentPane;
    GraphicsDrawLineMouseEx() {
        setTitle("Drawing Line by Mouse 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        MyPanel panel = new MyPanel();
        contentPane.add(panel, BorderLayout.CENTER);
        setSize(300, 300);
        setVisible(true);
    }

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




```
    class MyPanel extends JPanel {
        Vector<Point> vs = new Vector<Point>();
        Vector<Point> ve = new Vector<Point>();

        Point startP = null;
        Point endP = null;

        public MyPanel() {
            addMouseListener(new MouseAdapter(){
                public void mousePressed(MouseEvent e) {
                    startP = e.getPoint();
                }
                public void mouseReleased(MouseEvent e) {
                    endP = e.getPoint();
                    vs.add(startP);
                    ve.add(endP);
                    repaint();
                }
            });
        }
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.BLUE);
                for(int i=0; i<vs.size(); i++) {
                    Point s = vs.elementAt(i);
                    Point e = ve.elementAt(i);
                    g.drawLine((int)s.getX(), (int)s.getY(),
                                (int)e.getX(), (int)e.getY());
                }
        }
    }
}
```

JButton을 상속받아 새로운 버튼 생성 예


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

public class paintComponentEx extends JFrame {
    Container contentPane;
    paintComponentEx() {
        setTitle("paintComponent 사용  예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        contentPane.setLayout(new FlowLayout());
        MyButton b = new MyButton("New Button");
        b.setOpaque(true);
        b.setBackground(Color.CYAN);
        contentPane.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();
    }
}
```