12

그래픽

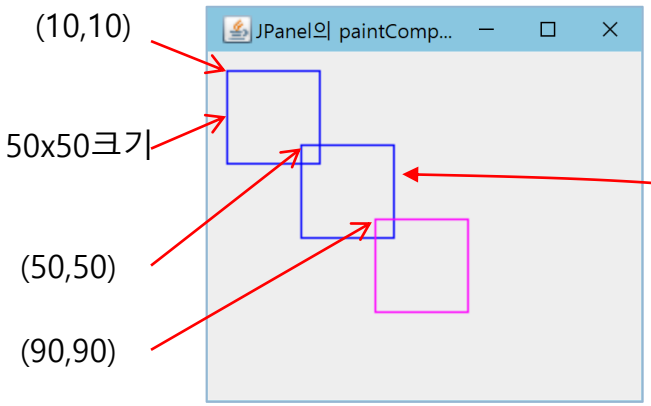
스윙 컴포넌트 그리기, paintComponent()

- 스윙의 그리기 기본 철학
  - 모든 컴포넌트는 자신의 모양을 스스로 그린다.
  - 컨테이너는 자신을 그린 후, 그 위에 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
  - 컴포넌트 그리기에 필요한 도구를 제공하는 객체
    - 색 지정, 도형 그리기, 클리핑, 이미지 그리기 등의 메소드 제공
- 사용자가 원하는 모양을 그리고자 할 때
  - paintComponent(Graphic g) 오버라이딩

```
class MyComponent extends JXXX { // JXXX는 기존의 스윙 컴포넌트
                ...
    public void paintComponent(Graphics g) { // 오버라이딩
        ... 필요한 코드 작성 ...
    }
}
```

예제 12-1 : JPanel을 상속받아 도형 그리기

- JPanel
  - 사용자가 그래픽을 통해 다양한 UI를 창출하는 일종의 캔버스



파란색 사각형

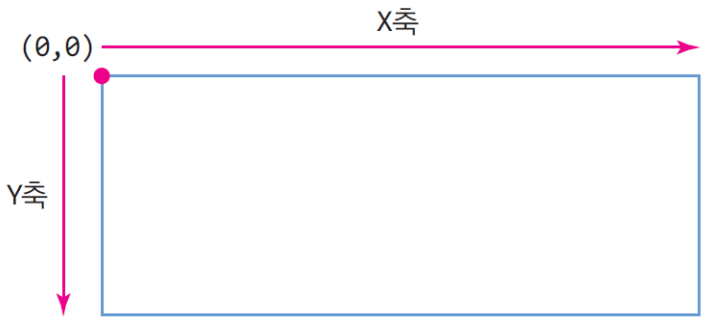
```
import javax.swing.*;
import java.awt.*;

public class paintJPanelEx extends JFrame {
    private MyPanel panel = new MyPanel();
        public paintJPanelEx() {
        setTitle("JPanel의 paintComponent() 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(panel);
        setSize(250,220);
        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.setColor(Color.MAGENTA);
            g.drawRect(90,90,50,50);
        }
    }
    public static void main(String [] args) {
        new paintJPanelEx();
    }
}
```

그래픽 기반 GUI 프로그래밍

- 그래픽 기반 GUI 프로그래밍
  - 스윙 컴포넌트를 사용하지 않고
  - 선, 원, 이미지 등을 직접 그려 GUI 화면을 구성하는 방식
- 장점
  - 스윙 컴포넌트로 만들 수 없는 자유로운 GUI 가능
    - 차트, 게임 등 자유로운 모양을 표현에 효과적
  - 그래픽 그리기는 컴포넌트 그리기보다 빠르다
  - 자바의 GUI 바탕 기술을 이해하는데 도움
  - 개발자 자신만의 컴포넌트를 창작

Graphics

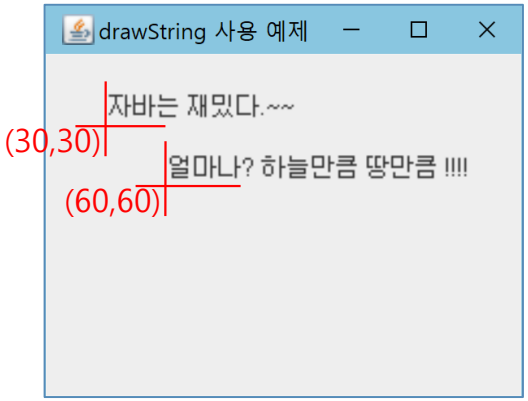
- Graphics의 좌표 체계



- Graphics의 기능
  - 색상 선택하기
  - 문자열 출력
  - 도형 그리기
  - 도형 칠하기
  - 이미지 출력
  - 클리핑
- 문자열 그리기
  - void drawString(String str, int x, int y)
    - (x,y) 영역에 str 문자열 그리기
    - 현재 색과 현재 폰트로 출력

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 {
   private MyPanel panel = new MyPanel();

   public GraphicsDrawStringEx() {
      setTitle("drawString 사용 예제");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setContentPane(panel);
      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
  - 하나의 색을 표현하는 클래스
    - Red, Green, Blue 의 3 성분으로 구성
    - 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 표현
    - 하위 8비트는 blue, 그 다음 상위 8 비트는 green, 그 다음 8 비트는 blue 성분
    - new Color(0x0000ff00); // 완전 초록
- 색을 사용하는 다른 방법
  - Color.BLUE 등의 static 상수 활용

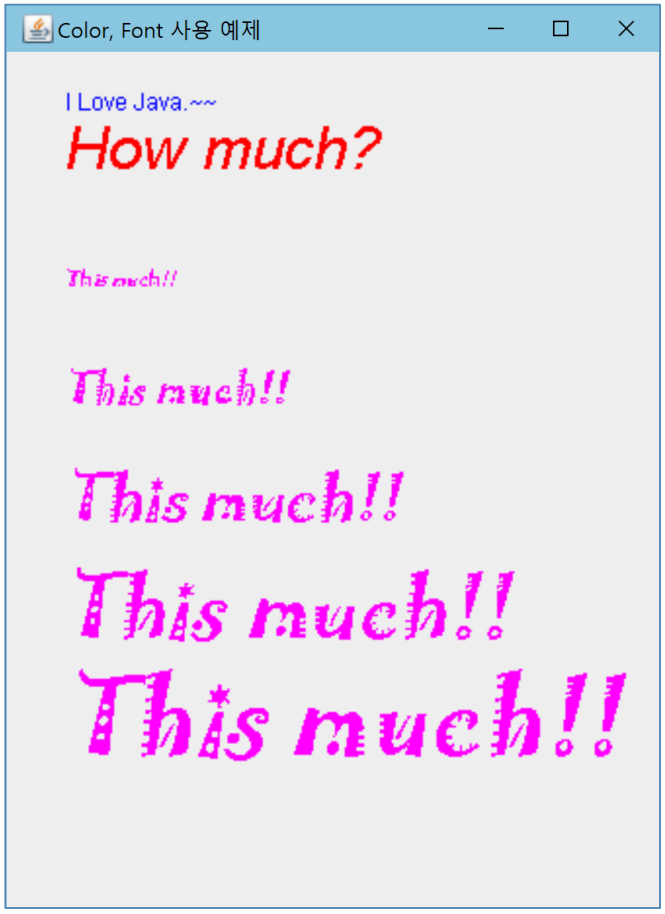
```
Graphics g;
g.setColor(new Color(255, 0, 0));          // 빨간색
g.setColor(new Color(0x0000ff00));         // 초록색
g.setColor(Color.YELLOW);                  // 노란색
```

- Font
  - 폰트를 표현하는 클래스
- 생성자
  - Font(String fontFace, int style, int size)
    - fontFace는 "고딕체", "Arial" 등
    - style은 Font.BOLD, Font.ITALIC , Font.PLAIN 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;
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 이용하여 문자열 그리기

Color와 Font 클래스를 이용하여 그림과 같이 출력되는 패널을 작성하라.



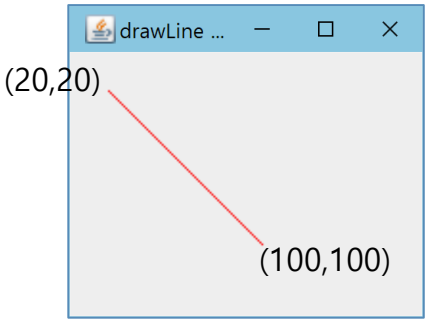

```
import javax.swing.*;
import java.awt.*;

public class GraphicsColorFontEx extends JFrame {
    private MyPanel panel = new MyPanel();
    public GraphicsColorFontEx() {
        setTitle("Color, Font 사용 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(panel);
        setSize(350, 470);
        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();
    }
}
```

도형 그리기와 칠하기

- 도형 그리기
  - 선, 타원, 사각형, 둥근 모서리 사각형, 원호, 폐 다각형
- Graphics의 메소드

```
void drawLine(int x1, int y1, int x2, int y2)
    (x1, y1)에서 (x2, y2)까지 선을 그린다.
void drawOval(int x, int y, int w, int h)
    (x, y)에서 w x h 크기의 사각형에 내접하는 타원을 그린다.
void drawRect(int x, int y, int w, int h)
    (x, y)에서 w x h 크기의 사각형을 그린다.
void drawRoundRect(int x, int y, int w, int h, int arcWidth, int arcHeight)
• arcWidth: 모서리 원의 수평 반지름
• arcHeight: 모서리 원의 수직 반지름
    (x, y)에서 w x h 크기의 사각형을 그리되, 4개의 모서리는 arcWidth와 arcHeight를 이용하
    여 원호로 그린다.
```

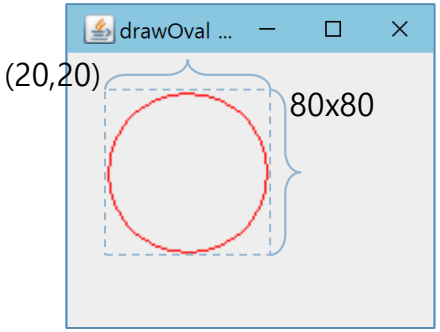
예제 12-4 : Graphics의 drawLine() 메소드로 선 그리기



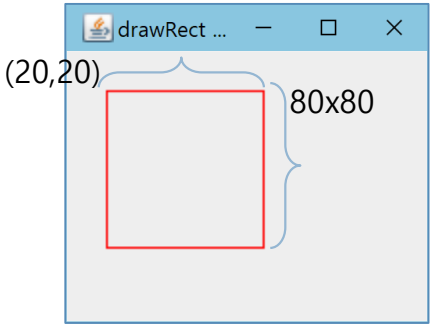

```
import javax.swing.*;
import java.awt.*;

public class GraphicsDrawLineEx extends JFrame {
   private MyPanel panel = new MyPanel();
   public GraphicsDrawLineEx() {
      setTitle("drawLine 사용 예제");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setContentPane(panel);
      setSize(200, 170);
      setVisible(true);
   }

   class MyPanel extends JPanel {
      public void paintComponent(Graphics g) {
         super.paintComponent(g);
         g.setColor(Color.RED); // 빨간색 선택
         g.drawLine(20, 20, 100, 100); // 선그리기
      }
   }
   public static void main(String [] args) {
      new GraphicsDrawLineEx();
   }
}
```

다른 도형 그리기 사례

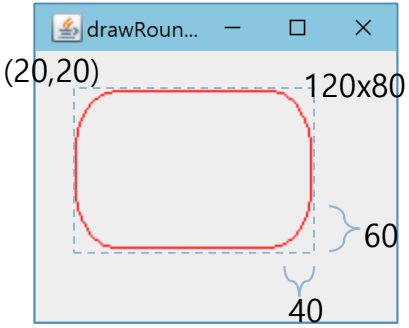

```
class MyPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.RED);
        g.drawOval(20,20,80,80);
    }
}
```




```
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);
    }
}
```



원호와 폐다각형 그리기

- 원호와 폐다각형 그리는 Graphics 메소드

void drawArc(int x, int y, int w, int h, int startAngle, int arcAngle)

- startAngle: 원호의 시작 각도
- arcAngle: 원호 각도

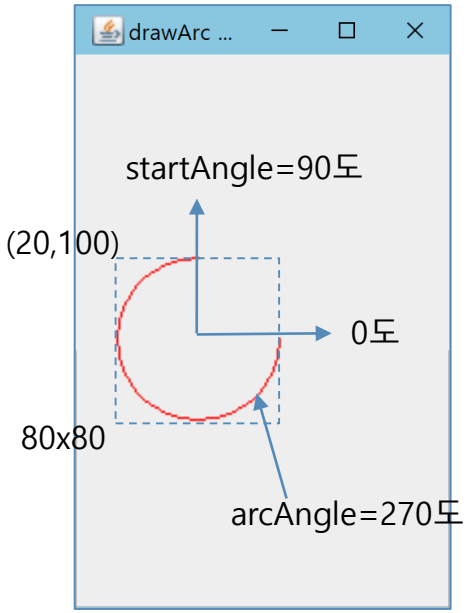
(x, y)에서 w x h 크기의 사각형에 내접하는 원호를 그린다. 3시 방향이 0도 기점이다. startAngle 지점에서 arcAngle 각도만큼 원호를 그린다. arcAngle이 양수이면 반시계 방향, 음수이면 시계 방향으로 그린다.

void drawPolygon(int [] x, int [] y, int n)

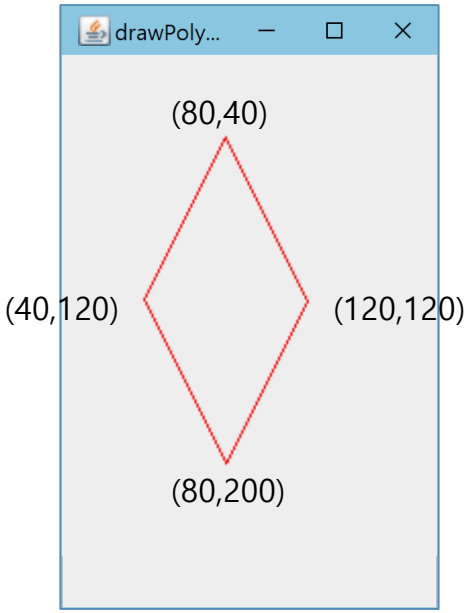
x, y 배열에 저장된 점들 중 n개를 연결하는 폐다각형을 그린다. (x[0], y[0]), (x[1], y[1]), ... , (x[n-1], y[n-1]), (x[0], y[0])의 점들을 순서대로 연결한다.

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);
    }
}
```



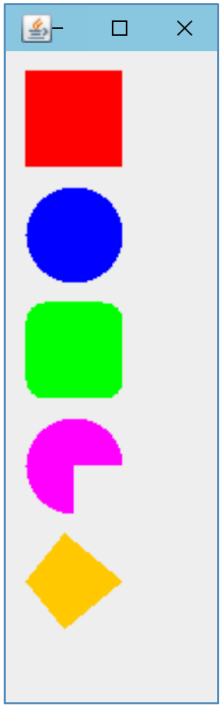
도형 칠하기

- 도형 칠하기
  - 도형을 그리고 내부를 칠하는 기능
  - 도형의 외곽선과 내부를 따로 칠하는 기능은 없다.
  - 도형 칠하기를 위한 메소드
    - 도형 그리기 메소드 명에서 draw 를 fill로 대치하면 된다. 인자는 동일
    - 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 12-5 : 도형 칠하기 예

Graphics의 칠하기 메소드를 이용하여 그림과 같은 패널을 작성하라.



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

public class GraphicsFillEx extends JFrame {
    private MyPanel panel = new MyPanel();
    public GraphicsFillEx() {
        setTitle("fillXXX 사용 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(panel);
        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 컴포넌트로 이미지 그리기

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

- 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.

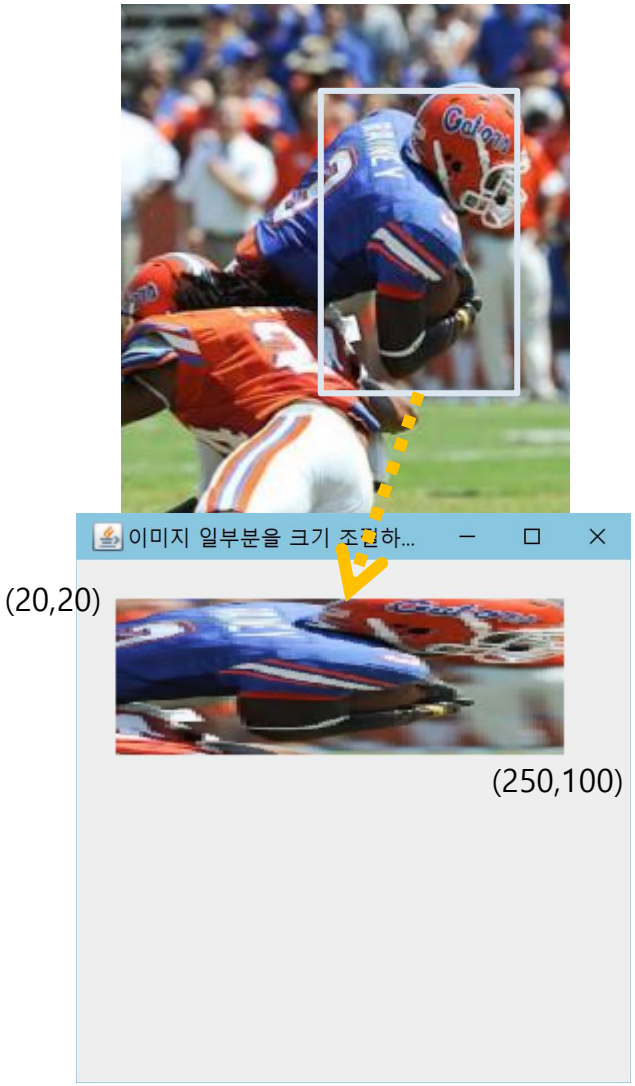

```
import javax.swing.*;
import java.awt.*;

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();
    }
}
```

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

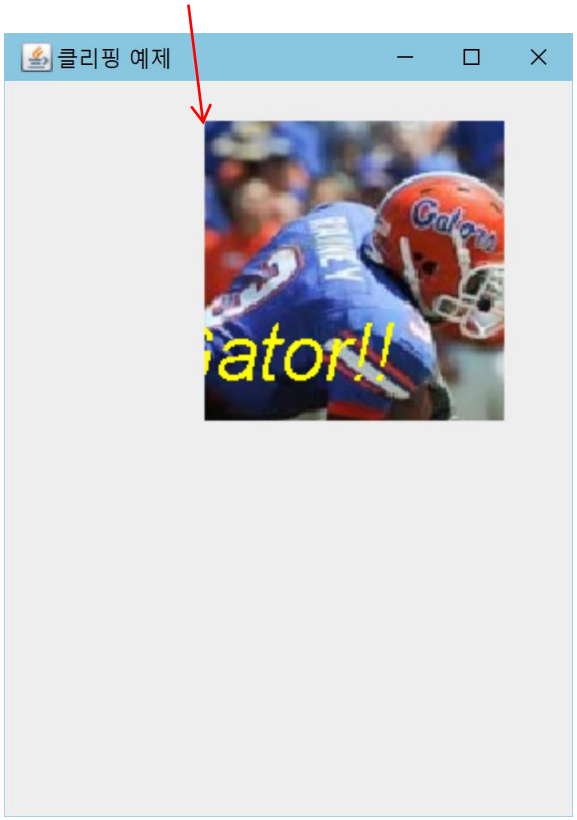
클리핑 영역 설정

- 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();
    }
}
```

스윙의 페인팅 메카니즘

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

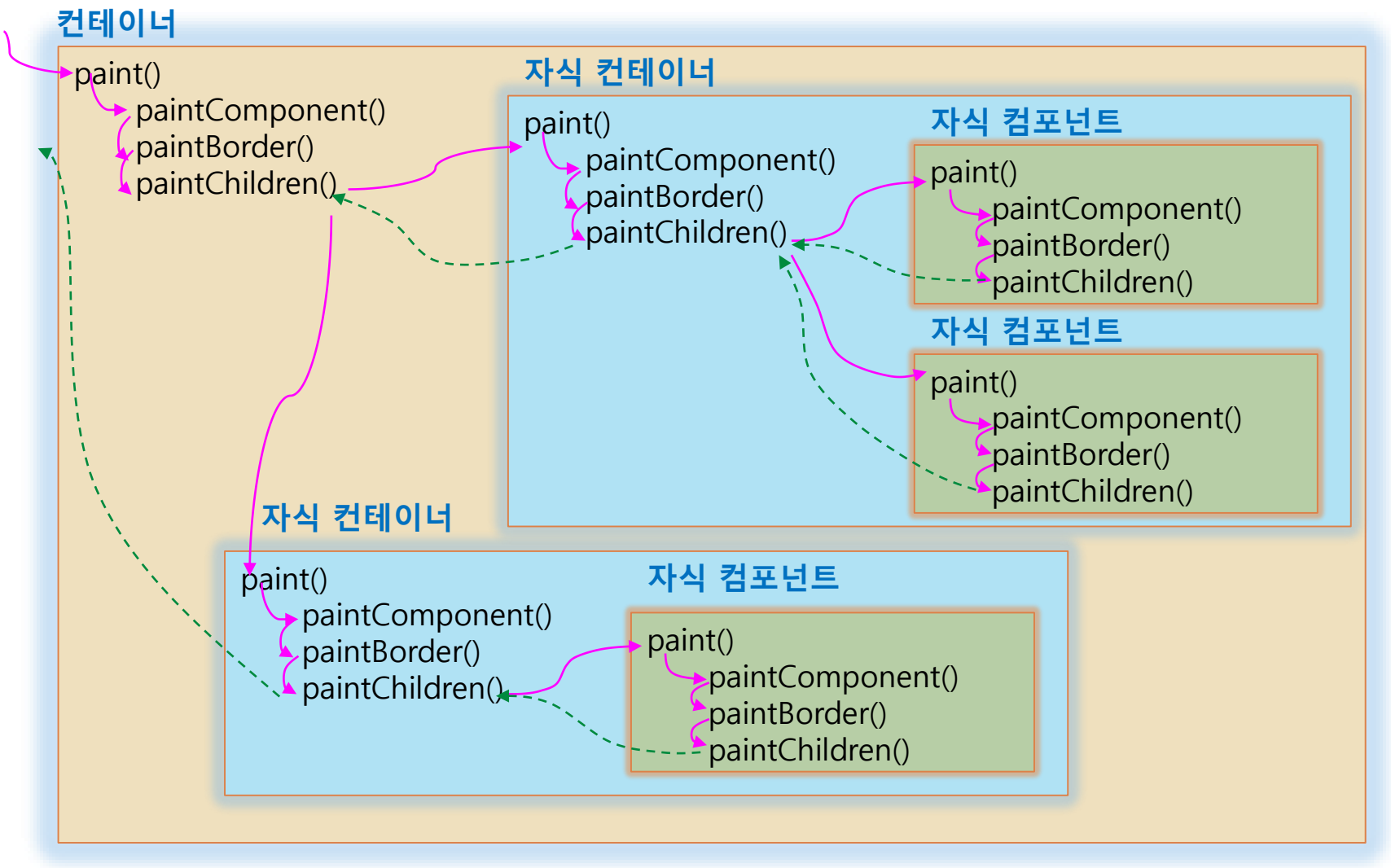
```
void paint(Graphics g) 컴포넌트 자신과 모든 자손 그리기
void paintComponent(Graphics g) 컴포넌트 자신의 내부 모양 그리기
void paintBorder(Graphics g) 컴포넌트의 외곽 그리기
void paintChildren(Graphics g) 컴포넌트의 자식들 그리기(컨테이너의 경우)
```

- 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

스윙 컴포넌트가 그려지는 과정



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

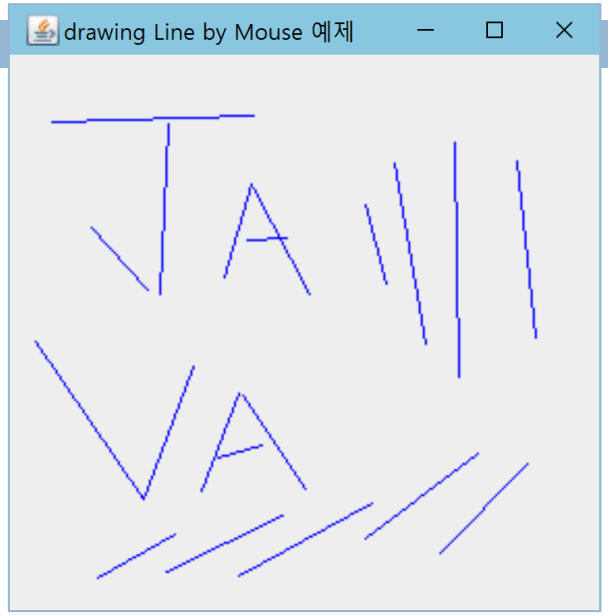
```
component.getParent().repaint();
```

revalidate()

- 컨테이너의 배치관리자에게 자식 컴포넌트들을 다시 배치 하도록 지시하는 메소드
- revalidate()가 필요한 경우
  - 컨테이너에 변화가 생겨 다시 그려야할 때
    - 프로그램에서 컨테이너에 컴포넌트 새로 삽입, 삭제

```
container.revalidate(); // 컨테이너에 부착된 컴포넌트의 재배치 지시
container.repaint(); // 컨테이너 다시 그리기 지시
```

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 {
    private MyPanel panel = new MyPanel();
    public GraphicsDrawLineMouseEx() {
        setTitle("drawing Line by Mouse 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(panel);
        setSize(300, 300);
        setVisible(true);
    }
    public static void main(String [] args) {
        new GraphicsDrawLineMouseEx();
    }

    class MyPanel extends JPanel {
        private Vector<Point> vStart = new Vector<Point>();
        private Vector<Point> vEnd = new Vector<Point>();
        public MyPanel() {
            addMouseListener(new MouseAdapter(){
                public void mousePressed(MouseEvent e) {
                    Point startP = e.getPoint();
                    vStart.add(startP);
                }
                public void mouseReleased(MouseEvent e) {
                    Point endP = e.getPoint();
                    vEnd.add(endP);
                    repaint();
                }
            });
        }

        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.BLUE);
            for(int i=0; i<vStart.size(); i++) {
                Point s = vStart.elementAt(i);
                Point e = vEnd.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 {
    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();
    }
}
```