Drawing

Drawing

- 그래픽
  - ✓ 자바에서 그래픽 도구는 그림을 그릴 수 있도록 해주는 펜, 붓, 폰트, 팔레트 등
  - ✓ 그래픽 도구들을 만든 후 계속해서 사용할 수 있도록 그래픽 도구의 정보를 저장하는데 이 정보를 저장하는 객체가 그래픽 컨텍스트(Graphic Context)
  - ✓ 자바에서 제공하는 GUI 컴포넌트는 paint(Graphics g) 메서드를 이용해서 화면에 출력
  - ✓ paint 메서드는 GUI 컴포넌트가 화면에 그려져야 할 때(처음 화면에 보여질 때, 사라졌다가 다시 화면에 나타날 때) 자동으로 호출되는 메서드
  - ✓ paint 메서드는 repaint 메서드나 paint 메서드 또는 paintComponent를 이용해서 강제로 호출 가능
  - ✓ AWT의 Frame에는 직접 그리기를 해도 되지만 JFrame에 그릴 때는 JPanel을 위에 부착해서 그리는 경우가 많음

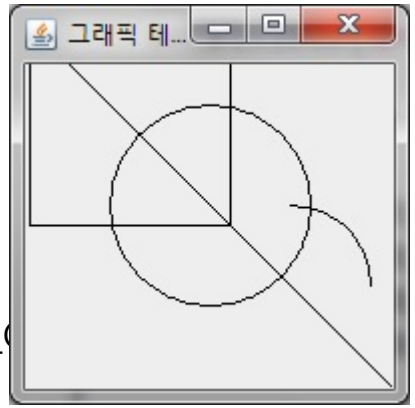
Drawing

❖ Graphics 클래스는 paint 메서드에 전달되는 객체의 클래스로 그래픽 컨텍스트(Graphic Context)라고도 함

반환형	메서드	설명
abstract void	drawString(String str, int x, int y)	그래픽 컨텍스트의 글꼴과 색상을 이용하여 지정한 문자를 지정한 위치에 그려준다.
	drawLine (int x1, int y1, int x2, int y2)	그래픽 컨텍스트 좌표 내에서 x1, y1과 x2, y2 사이에 현재 색상을 이용하여 선을 그려준다.
	drawRect(int x, int y, int width, int height)	지정한 위치에 폭과 높이가 있는 사각형을 그려준다.
	drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)	지정한 위치에 폭과 높이, 각도가 있는 모서리가 둥근 사각형을 그려준다.
	drawOval(int x, int y, int width, int height)	지정한 위치에 폭과 높이가 있는 타원을 그려준다.
abstract void	drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)	지정한 위치에 폭과 높이, 각도가 있는 원호를 그려준다.
	drawPolygon(int[ ] xPoints, int[ ] yPoints, int nPoints)	지정한 x 좌표와 y 좌표의 배열을 가지는 닫힌 다각형을 그려준다.
abstract boolean	drawImage(Image img, int x, int y, ImageObserver observer)	현재 이용할 수 있는 이미지를, x와 y를 시작 좌표로 기준하여 그려준다.
	drawImage(Image img, int x, int y, int width, int height, ImageObserver observer)	현재 이용할 수 있는 이미지를, x와 y를 시작 좌표로 기준하여 폭(width)과 높이(height)만큼 그려준다.
	drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, ImageObserver observer)	로딩된 이미지를 sx1, sy1을 시작 좌표로 하고 폭(sx2), 높이(sy2)만큼을 잘라서 출력시킬 화면의 dx1, dy1을 시작 좌표로 하여 폭(dx2)과 높이(dy2) 만큼의 공간에 그려준다.
void	setColor(Color c)	그래픽 컨텍스트의 현재 색상을 지정한 색상으로 설정한다.
	setFont(Font font)	그래픽 컨텍스트의 현재 글꼴을 지정한 글꼴로 설정한다.
FontMetrics	getFontMetrics( )	그래픽 컨텍스트의 현재 글꼴에 대한 FontMetrics 객체를 얻어온다.
	getFontMetrics(Font f)	지정한 글꼴에 대한 FontMetrics 객체를 얻어온다.

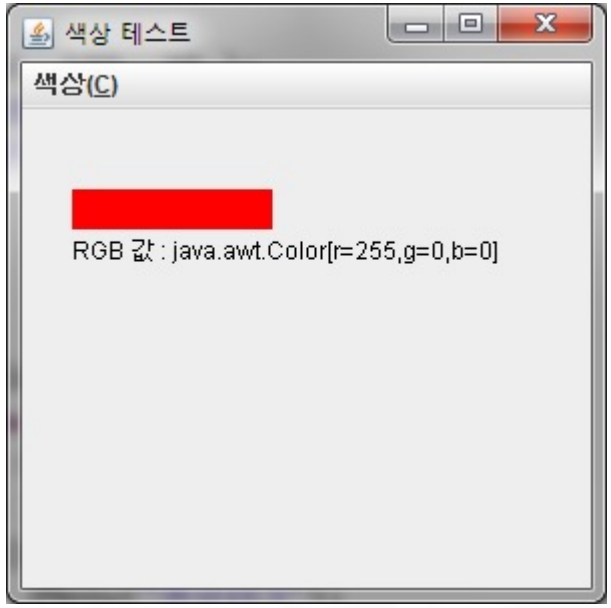
Paint1

```
import javax.swing.*;
import java.awt.*;
class GraphicTest extends JFrame
{
        public GraphicTest(String str)
        {
                super(str);
                setSize(200,200);
                setLocation(300,300);
                setDefaultCloseOperation(JFrame.EXIT_ON_
                setVisible(true);
        }
        public void paint(Graphics g)
        {
                g.drawLine(10, 10, 190,190);
                g.drawRect(10, 10, 100,100);
                g.drawOval(50, 50, 100,100);
                g.drawArc(100, 100, 80, 80, 0, 90);
        }
}
public class Paint1{
        public static void main(String[] args) {
                new GraphicTest("그래픽 테스트");
        }
}
```



Paint2 – 그림을 그려주는 클래스

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class ColorTest extends JPanel
{
        Color c = new Color( 0, 0, 255 );
        public void paint( Graphics g )
        {
                g.setColor(c);
                g.fillRect( 25, 40, 100, 20 );
                g.setColor(Color.BLACK);
                g.drawString( "RGB 값 : " + c, 25,75 );
        }
}
```



Paint2 – 윈도우 클래스

```
class ColorFrame extends JFrame{
        public ColorFrame(){
                super("색상 테스트");
                ColorTest p = new ColorTest();
                JMenuBar mb = new JMenuBar();
                setJMenuBar(mb);
                JMenu color = new JMenu("색상(C)");
                color.setMnemonic('c');
                color.setToolTipText("색상 선택");
                mb.add(color);
                JMenuItem red = new JMenuItem("빨강");
                red.addActionListener(new MenuEvent(p));
                color.add(red);
                JMenuItem green = new JMenuItem("녹색");
                green.addActionListener(new MenuEvent(p));
                color.add(green);
                JMenuItem blue = new JMenuItem("파랑");
                blue.addActionListener(new MenuEvent(p));
                color.add(blue);
                add(p);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setSize(300, 300);
                setVisible(true);
        }
}
```

Paint2 – 메뉴 이벤트 핸들러

```
class MenuEvent implements ActionListener
{
		ColorTest pa;
		public MenuEvent(ColorTest arg)
		{
			pa = arg;
		}
		public void actionPerformed(ActionEvent e)
		{
			if(e.getActionCommand().equals("빨강")){
			pa.c = Color.red;
			}
			else if(e.getActionCommand().equals("녹색")){
				pa.c = Color.green;
				}
			else
				pa.c = Color.blue;
			pa.repaint();
		}
}
```

Paint2 – 실행 클래스

```
public class Paint2 {
		public static void main(String[] args) {
				ColorFrame frame = new ColorFrame();
		}
}
```

Font

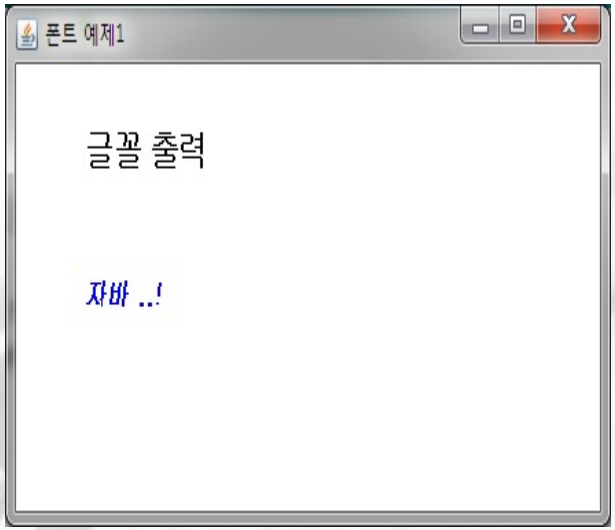
- Font 클래스
  - ✓ 그래픽 컨텍스트에 글꼴을 설정하기 위해 사용하는 클래스
  - ✓ Font 클래스 객체를 생성할 때 사용할 글꼴의 속성을 지정하여 그래픽 컨텍스트에 설정하면 그 이후의 모든 텍스트에 적용

자료형	필드명	설명
static int	BOLD	굵은 스타일 상수
	ITALIC	이탤릭 스타일 상수
	PLAIN	일반 스타일 상수

생성자	설명
Font(Map<? extends AttributedCharacterIterator.Attribute,?> attributes)	지정된 속성으로 새로운 폰트 객체를 생성한다.
Font(String name, int style, int size)	지정한 폰트 이름, 스타일, 크기를 통해 새로운 폰트 객체를 생성한다.

Paint3 – 글자를 출력하는 클래스

```
import java.awt.*;
import javax.swing.*;
class XCanvas extends Canvas
{
        Font font = null;
        public void paint(Graphics g)
        {
                font = new Font("Timesroman", Font.BOLD, 20);
                g.setFont(font);
                g.drawString("글꼴 출력", 50, 50);
                font = new Font("굴림", Font.ITALIC|Font.BOLD , 15);
                g.setFont(font);
                g.setColor( Color.blue);
                g.drawString("자바 ..!", 50, 120);
        }
}
```



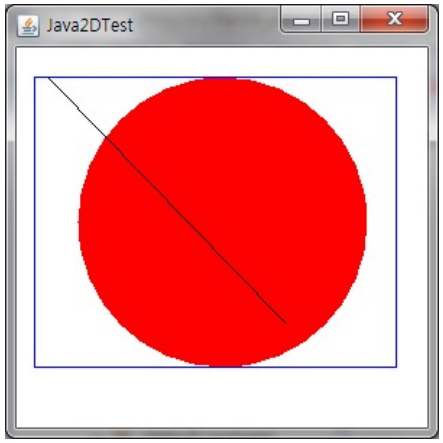
Paint3 – 실행 클래스

```
public class Paint3 {
		public static void main(String[] args)
		{
			XCanvas canvas = new XCanvas();
			JFrame f = new JFrame("폰트 예제");
			f.add(canvas);
			f.setSize(200, 200);
			f.setVisible(true);
			f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		}
}
```

Graphics2D

- Graphics2D
  - ✓ java.awt.geom 패키지에 존재하는 Graphics2D라는 이름으로 시작하는 클래스를 이용해서 드로잉 가능
  - ✓ Graphics 객체를 Graphics2D 객체로 형 변환 한 후 draw나 fill 메서드를 이용해서 위의 객체를 출력

Paint4 – 그림을 그리는 클래스




```
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;
class Java2DTest extends JFrame
{
        Rectangle2D.Double rect;
        Ellipse2D.Double circle;
        Line2D.Double line;
        public Java2DTest(String title)
        {
                super(title);
                rect = new Rectangle2D.Double(20, 50, 250, 200);
                circle = new Ellipse2D.Double(50, 50, 200, 200);
                line = new Line2D.Double(30.1, 51.3, 193.9, 219.1);
                setSize(300, 300);
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
```

Paint4 – 그림을 그리는 클래스

```
        public void paint(Graphics g)
        {
                Graphics2D g2 = (Graphics2D) g;
                g2.setColor(Color.blue);
                g2.draw(rect);
                g2.setColor(Color.red);
                g2.fill(circle);
                g2.setColor(Color.black);
                g2.draw(line);
        }
}
```

Paint4 – 실행 클래스

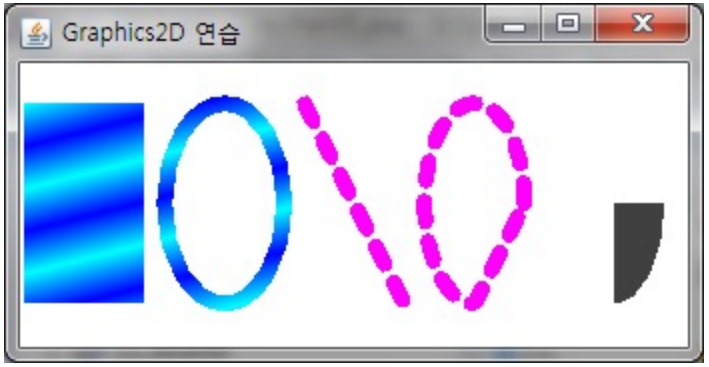
```
public class Paint4{
		public static void main(String[] args) {
				new Java2DTest("Java2DTest");
		}
}
```

Paint5 – 그림을 출력하는 클래스

```
mport java.awt.*;
import java.awt.geom.*;

import javax.swing.JFrame;

class Graphics2DTest1 extends JFrame {
        public Graphics2DTest1()
        {
                super( "Graphics2D 연습" );
                setSize( 350, 180 );
                setVisible(true);
        }
```



Paint5 – 그림을 출력하는 클래스

```
        public void paint( Graphics g )
        {
                Graphics2D g2d = ( Graphics2D ) g;
                g2d.setPaint(
                                new GradientPaint( 5, 30,Color.blue,10, 50,
Color.cyan,true ) );
                g2d.fill( new Rectangle2D.Double( 10, 50, 60, 100 ) );
                g2d.setStroke( new BasicStroke( 8 ) );
                g2d.draw( new Ellipse2D.Double( 80, 50, 60, 100 ) );
                float dashes[] = { 10 };
                g2d.setPaint( Color.magenta );
                g2d.setStroke(
                                new BasicStroke( 8,BasicStroke.CAP_ROUND,
BasicStroke.JOIN_ROUND, 10, dashes, 0 ) );

                g2d.draw( new Line2D.Double( 150, 50, 200, 150 ) );

                g2d.draw(new Arc2D.Double(210, 50, 50, 100, 0, 270, Arc2D.CHORD ) );

                g2d.setPaint( Color.darkGray );
                g2d.fill(new Arc2D.Double(280, 50, 50, 100, 0, -90, Arc2D.PIE ) );
        }
}
```

Paint5 – 실행 클래스

```
public class Paint5{
        public static void main(String[] args) {
                new Graphics2DTest1();
        }
}
```

Image 처리

- 이미지 출력
  - ✓ 이미지를 불러 올 수 있는 메서드로는 getImage() 메서드를 이용하고 이 메서드는 Toolkit 클래스와 Applet 클래스의 메서드
  - ✓ Toolkit 클래스의 getImage() 메서드는 일반적인 어플리케이션 프로그램에서 사용하는 메서드고 Applet 클래스의 getImage() 메서드는 애플릿 프로그램에서 사용하는 메서드
  - ✓ getImage() 메서드는 이미지를 불러오면서 이미지 객체를 생성해주는 메서드
  - ✓ 이미지 객체를 관리하기 위해서 자바에서는 java.awt 패키지의 Image 클래스를 사용

Image 처리

- 이미지 출력 메서드
  - ✓ 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)

Image 처리

- 이미지 출력
  - ✓ 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);
}
```

Image 처리

- 이미지 출력
  - ✓ 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);
}
```

Image 처리

OriginalImage – 이미지 출력

```
import java.awt.*;
import java.awt.event.*;
class ImageTest extends Frame{
        Image img ;
        String s = "aespa.jpeg" ;
        public ImageTest(String title)
        {
                super(title);
                img = Toolkit.getDefaultToolkit().getImage(s) ;
                addWindowListener(new WindowHandler());
                setSize(800,600);
                setVisible(true);
        }
        public void paint(Graphics g)
        {
                g.drawImage(img, 0, 0 , this);
        }
        class WindowHandler extends WindowAdapter
        {
                public void windowClosing(WindowEvent e)
                {
                        System.exit(0);
                }
        }
}
```

OriginalImage – 실행 클래스

```
public class OriginalImage {
		public static void main(String[] args) {
				new ImageTest("태연");
		}

}
```

EditImage

EditImage – 작업을 위한 ENUM

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;

import javax.swing.*;
import javax.swing.filechooser.*;

enum ImageTask{
        BASIC,ZOOMIN,ZOOMOUT,HORIZONTAL,VERTICAL,REVERSE;
}
```

EditImage - 윈도우

```
class ImageEditing extends Frame
{
		Image img ;
		EditCanvas xcanvas;
		Button[] btn = new Button[7];
		String[] btn_str = {"기본","확대","축소","좌우","상하","반전", "이미지 불러오기"};
		TextField txt ;

		ImageTask message = ImageTask.BASIC ;

		String s = "aespa.jpeg" ; // 초기 이미지
```

EditImage - 윈도우

```
public ImageEditing(String title){
        super(title);
        img = Toolkit.getDefaultToolkit().getImage(s) ;
        xcanvas = new EditCanvas(this);
        xcanvas.setSize(550,450);
        Panel panel1 = new Panel();
        Panel panel2 = new Panel();
        ActionHandler handler = new ActionHandler(this);
        for( int i = 0 ; i < 6 ; i++ ){
                btn[i] = new Button(btn_str[i]);
                btn[i].addActionListener(handler);
                panel1.add(btn[i]);
        }
        panel2.add(new Label("파일 불러오기 :"));
        txt = new TextField(30);
        panel2.add(txt);
        btn[6] = new Button(btn_str[6]);
        btn[6].addActionListener(handler);
        panel2.add(btn[6]);
```

EditImage - 윈도우

```
                add("North", panel2);
                add("Center",xcanvas);
                add("South", panel1);
                addWindowListener(new WindowHandler());
                setSize(550,500);
                setVisible(true);
        }
}
```

EditImage – 이미지 편집

```
class EditCanvas extends Canvas {
		ImageEditing imageTest;

		public EditCanvas(ImageEditing imageTest) {
			this.imageTest = imageTest;
		}

		public void paint(Graphics g) {
			switch(imageTest.message ){
			case BASIC:
				g.drawImage(imageTest.img, 0, 0, this);
				break;
			case ZOOMIN:
				g.drawImage(imageTest.img, 0, 0, imageTest.img.getWidth(this)*2,
this.getHeight() + imageTest.img.getHeight(this)*2, 0 , 0, imageTest.img.getWidth(this),
imageTest.img.getHeight(this), this);
				break;
			case ZOOMOUT:
				g.drawImage(imageTest.img, 0, 0, imageTest.img.getWidth(this)/2,
imageTest.img.getHeight(this)/2, 0 , 0, imageTest.img.getWidth(this), imageTest.img.getHeight(this),
this);
				break;
```

EditImage – 이미지 편집

```
                        case HORIZONTAL:
                                g.drawImage(imageTest.img, imageTest.img.getWidth(this), 0, 0,
imageTest.img.getHeight(this) ,0,0,imageTest.img.getWidth(this), imageTest.img.getHeight(this), this);
                                break;
                        case VERTICAL:
                                g.drawImage(imageTest.img, 0,
imageTest.img.getHeight(this) ,imageTest.img.getWidth(this), 0, 0, 0, imageTest.img.getWidth(this),
imageTest.img.getHeight(this), this);
                                break;
                        case REVERSE:
                                g.drawImage(imageTest.img, imageTest.img.getWidth(this),
imageTest.img.getHeight(this) ,0, 0, 0, 0, imageTest.img.getWidth(this),
imageTest.img.getHeight(this),this);
                                break;
                        }
                }
}
```

EditImage – 버튼의 이벤트 처리

```
class ActionHandler implements ActionListener{
        ImageEditing imageTest;


        public ActionHandler(ImageEditing imageTest) {
                super();
                this.imageTest = imageTest;
        }

        public void actionPerformed(ActionEvent e){
                Button obj = (Button)e.getSource();
                if( obj == imageTest.btn[0] ){
                        imageTest.message = ImageTask.BASIC ;
                }
                else if( obj == imageTest.btn[1] ){
                        imageTest.message = ImageTask.ZOOMIN ;
                }
                else if( obj == imageTest.btn[2] ){
                        imageTest.message = ImageTask.ZOOMOUT ;
                }
                else if( obj == imageTest.btn[3] ){
                        imageTest.message = ImageTask.HORIZONTAL ;
                }
```

EditImage – 버튼의 이벤트 처리

```
                else if( obj == imageTest.btn[4] ){
                        imageTest.message = ImageTask.VERTICAL ;
                }
                else if( obj == imageTest.btn[5] ){
                        imageTest.message = ImageTask.REVERSE ;
                }
                else{
                        JFileChooser fc = new JFileChooser();
                        FileNameExtensionFilter filter =
                                        new FileNameExtensionFilter("ImageFile",
"jpg", "gif", "png");
                        fc.setFileFilter(filter);
                        int result = fc.showOpenDialog(null);
                        if(result ==JFileChooser.APPROVE_OPTION){
                                File f = fc.getSelectedFile();
                                imageTest.txt.setText(f.getAbsolutePath());
                                imageTest.img =
Toolkit.getDefaultToolkit().getImage(f.getAbsolutePath()) ;
                        }
                }
                imageTest.xcanvas.repaint();
        }
}
```

EditImage – 윈도우의 이벤트 처리

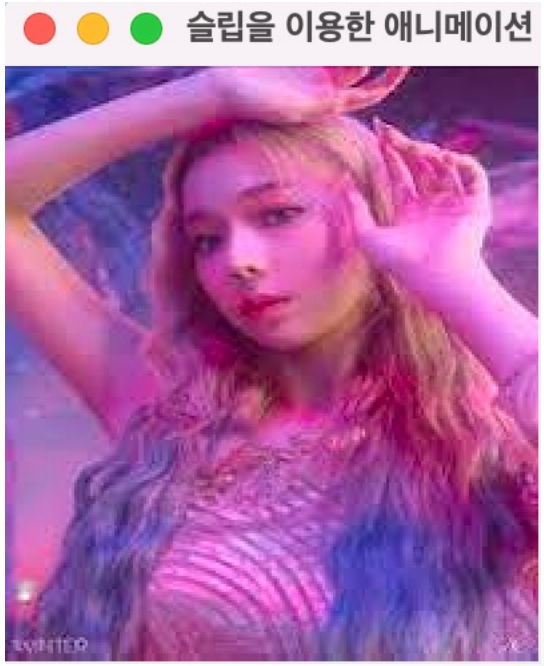
```
class WindowHandler extends WindowAdapter
{
		public void windowClosing(WindowEvent e)
		{
			System.exit(0);
		}
}
```

EditImage – 실행

```
public class EditImage {
		public static void main(String[] args) {
				new ImageEditing("이미지 확대축소");
		}

}
```

애니메이션



애니메이션 – 이미지 출력

```
import java.awt.*;
import java.awt.event.*;
class ImageAnimationTest extends Frame {
        Image img[] = new Image[10];
        int num = 0 ;
        public ImageAnimationTest(String title){
                super(title);
                for(int i = 0 ; i < 4 ; i++){
                        img[i] = Toolkit.getDefaultToolkit().getImage("aespa"+(i)+".jpeg");
                }
                addWindowListener(new WindowHandler());
                setSize(201,251);
                setVisible(true);
        }
        public void paint(Graphics g){
                if( num > 3 ) num = 0 ;
                g.drawImage(img[num++], 0, 0, this);
        }
        class WindowHandler extends WindowAdapter{
                public void windowClosing(WindowEvent e)
                {
                        System.exit(0);
                }
        }
}
```

애니메이션 – 실행

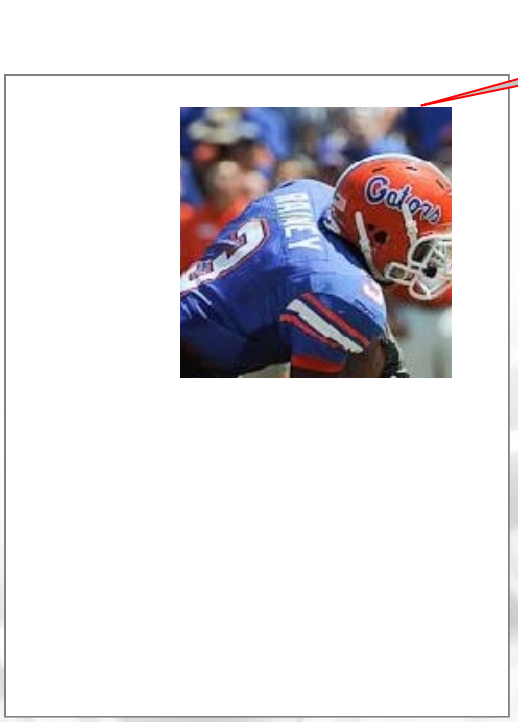
```
public class AnimationMain {
	public static void main(String[] args) {
		ImageAnimationTest ani = new ImageAnimationTest("슬립을 이용한 애니메이
션");
		while(true)
		{
			try
			{
				Thread.sleep(500);
				ani.repaint();
			}
			catch(InterruptedException e)
			{
				System.out.println(e.getMessage());
				break;
			}
		}
	}

}
```

Clipping

- 클리핑(Clipping)
  - ✓ 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

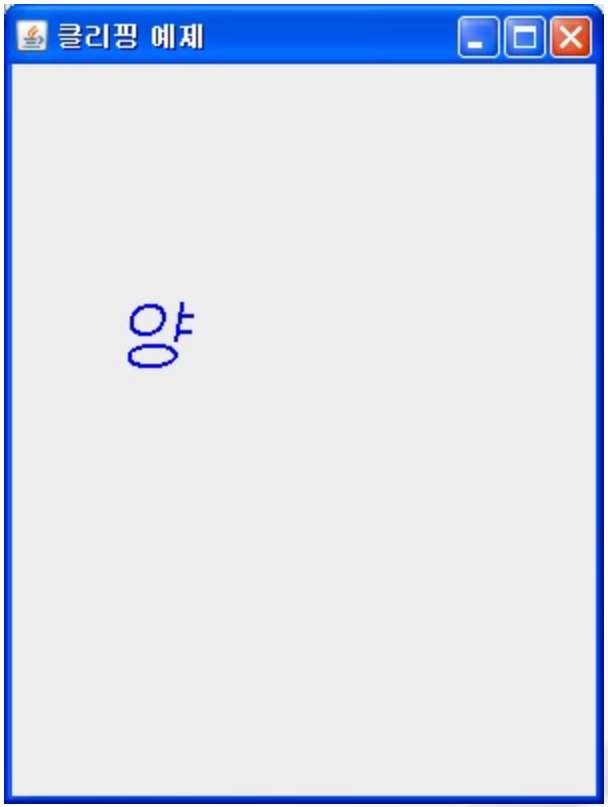


특정 사각형 영역을 클리핑 영역으로 설정된 경우

Clipping

- Graphics의 클리핑 메서드
  - ✓ 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

Clipping



Clipping – 출력 클래스

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

class ClippingTest extends JFrame {
        Container contentPane;
        public ClippingTest() {
                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 {
                public void paintComponent(Graphics g) {
                        super.paintComponent(g);
                        g.setClip(50, 20, 150, 150);
                        g.setColor(Color.BLUE);
                        g.setFont(new Font("SanSerif", Font.ITALIC, 40));
                        g.drawString("석양", 10, 150);
                }
        }
}
```

Clipping – 실행 클래스

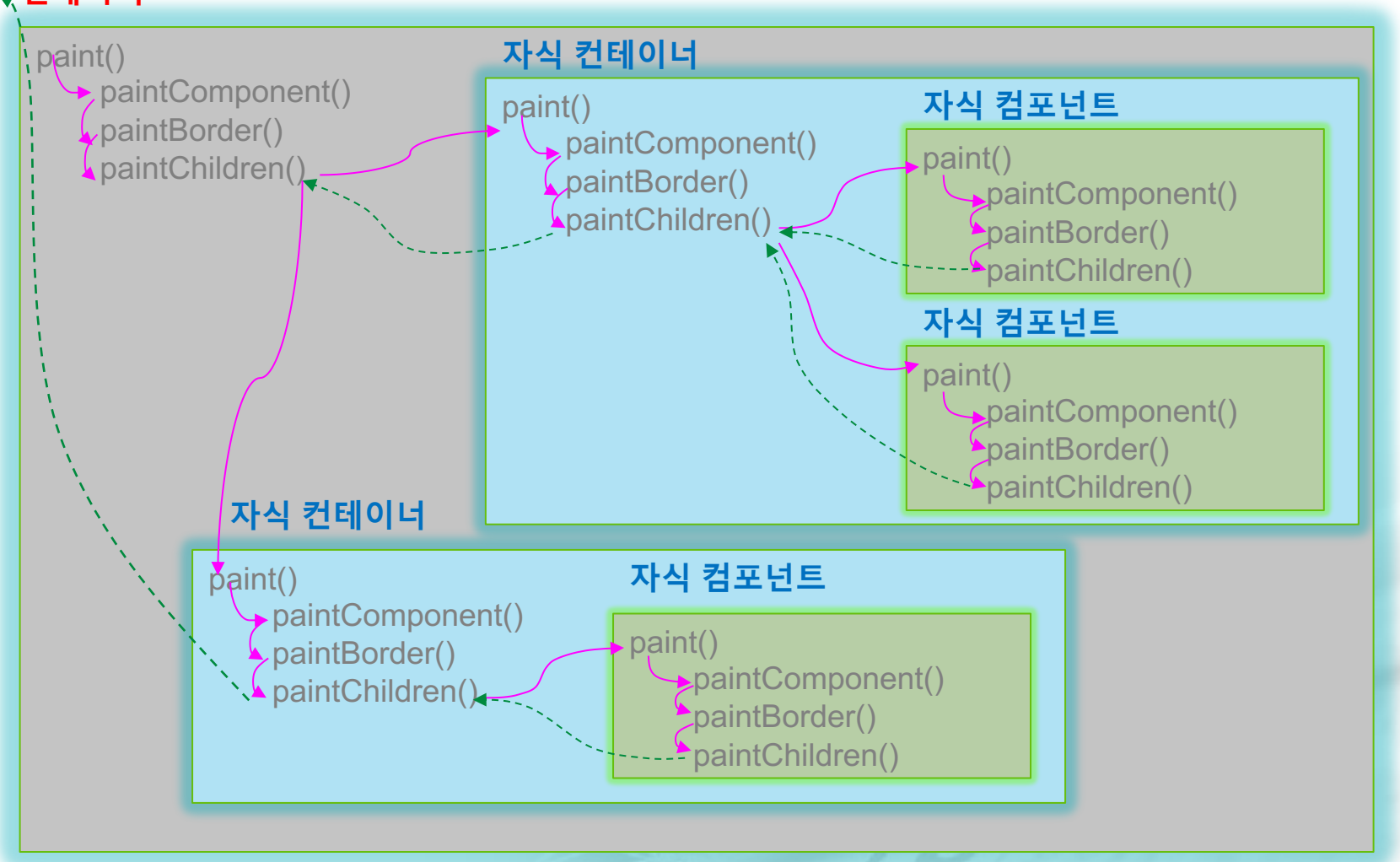
```
public class ClippingMain {
        public static void main(String [] args) {
                new ClippingTest();
        }
}
```

Swing의 출력 시스템

- JComponent.paint()
  - ✓ 컴포넌트 자신과 모든 자손 그리기
- JComponent.paint()는 다음 메서드를 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

Swing의 출력 시스템

❖ 컨테이너



Swing의 출력 시스템

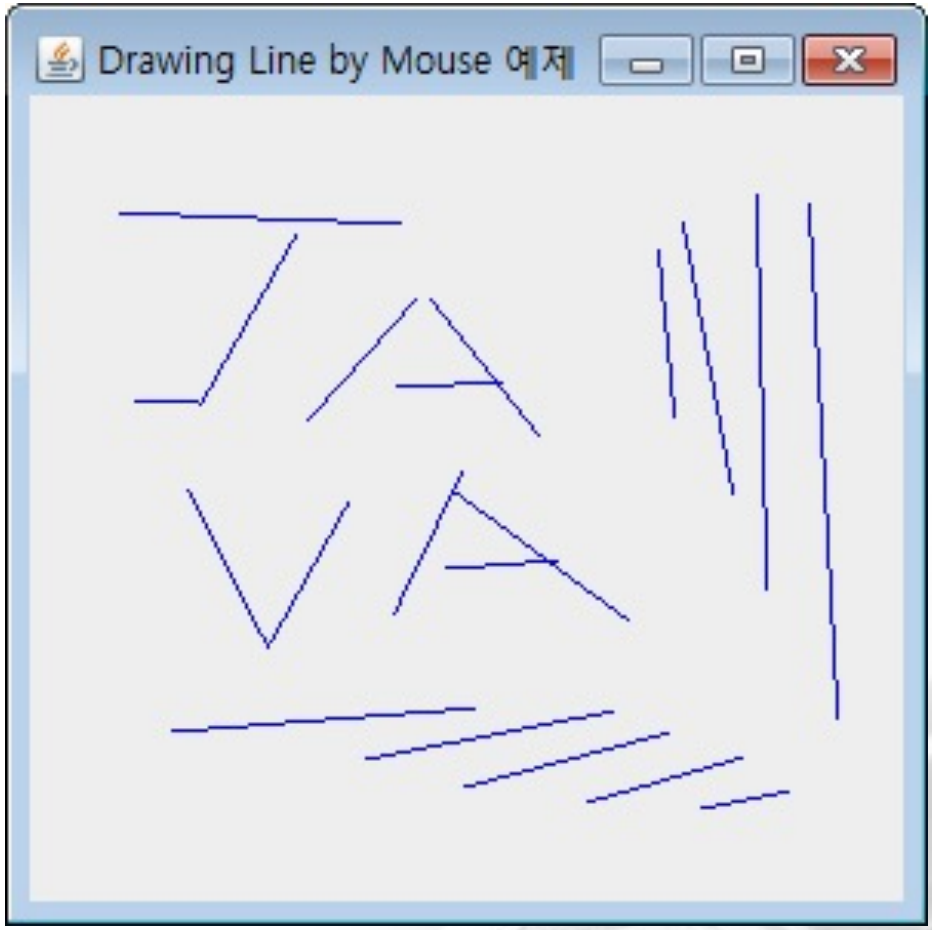
- Component의 메서드
  - ✓ repaint() 를 호출해야 하는 경우
    - ❑ 개발자가 컴포넌트를 다시 그리고자 하는 경우
      - ○ 프로그램 내에서 컴포넌트의 모양과 위치를 변경하였지만 바로 화면에 반영되지 않는다.
      - ○ 이 이유는 이 컴포넌트가 다시 그려져야 그 때 변경된 위치에 변경된 모양으로 출력됨
      - ○ repaint()는 지금 당장 컴포넌트를 다시 그리도록 지시함

```
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();
```

마우스를 이용한 선 그리기



마우스를 이용한 선 그리기

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

class PaintMain extends JFrame {
        Container contentPane;
        public PaintMain() {
                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);
        }
```

마우스를 이용한 선 그리기

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

        Point startP = null;
        Point endP = null;

        public MyPanel() {
                addMouseListener(new MouseAdapter(){
                        public void mousePressed(MouseEvent e) {
                                startP = e.getPoint();
                        }

                });
```

마우스를 이용한 선 그리기

```
        addMouseMotionListener(new MouseMotionListener() {
                @Override
                public void mouseDragged(MouseEvent e) {
                        endP = e.getPoint();
                        vs.add(startP);
                        ve.add(endP);
                        repaint();
                }

                @Override
                public void mouseMoved(MouseEvent e) {
                        // TODO Auto-generated method stub

                }
        });
}
```

마우스를 이용한 선 그리기

```
                public void paintComponent(Graphics g) {
                        super.paintComponent(g);
                        g.setColor(Color.BLUE);
                        for(int i=0; i<vs.size(); i++) {
                                Point s = vs.get(i);
                                Point e = ve.get(i);
                                g.drawLine((int)s.getX(), (int)s.getY(),
                                                (int)e.getX(), (int)e.getY());
                        }
                }
        }
}
```

마우스를 이용한 선 그리기

```
public class MouseDrawing {
        public static void main(String [] args) {
                new PaintMain();
        }
}
```

사운드 재생

- wav 파일 재생
  - ✓ javax.sound.sampled.AudioInputStream, javax.sound.sampled.AudioSystem;, javax.sound.sampled.Clip 클래스 이용
  - ✓ AudioSystem 클래스 와 File 클래스를 이용해서 wav 파일의 내용을 읽을 수 있는 AudioInputStream 클래스의 인스턴스를 만들고 AudioInputStream 클래스의 인스턴스를 이용해서 Clip 클래스의 인스턴스를 만들고 open 과 start 메서드를 호출하면 재생됨

사운드 재생

- wav 파일 재생
  - ✓ 재생
    - ❑ 프로젝트에 dingdong.wav 파일을 복사
    - ❑ main 메서드를 소유한 클래스를 생성하고 작성 – WavMain.java

```
import java.io.File;

import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.swing.JFrame;

public class WavMain {

	public static void main(String[] args) {
		JFrame f = new JFrame("메인윈도우");
		f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		f.setLocationRelativeTo(null);
		f.setSize(200,200);
		f.setVisible(true);
```

사운드 재생

- wav 파일 재생
  - ✓ 재생
    - ❑ main 메서드를 소유한 클래스를 생성하고 작성 – WavMain.java

```
                File file = new File("./dingdong.wav");
                System.out.println(file.exists()); //true
                try {
                        AudioInputStream stream =
AudioSystem.getAudioInputStream(file);
                        Clip clip = AudioSystem.getClip();
                        clip.open(stream);
                        clip.start();
                } catch(Exception e) {
                        e.printStackTrace();
                }
        }
}
```

사운드 재생

- MP3 파일 재생
  - ✓ Java의 기본 라이브러리에서는 MP3 형식의 오디오를 재생할 수 없으므로 외부 라이브러리를 사용해야 함
  - ✓ JLayer를 사용하여 MP3 파일 재생
    - ❑ 재생하고자 하는 MP3 파일의 InputStream 인스턴스를 생성하고 Player 클래스의 인스턴스를 생성하고 play 메서드를 호출하면 됨
    - ❑ 한 가지 주의할 점은 오디오가 재생되는 스레드는 오디오 처리가 완료될 때까지 해당 스레드가 차단된다는 것으로 메인 스레드를 차단하고 싶지 않으면 새로운 스레드를 만들고 그 안에서 run() 메소드를 호출해서 재생해야 함

사운드 재생

- MP3 파일 재생
  - ✓ JLayer를 사용하여 MP3 파일 재생
    - ❑ 프로젝트에 Jlayer.jar 파일 과 재생할 mp3 파일을 복사
    - ❑ jar 파일을 선택하고 마우스 오른쪽을 클릭해서 [Configure Build Path] – [Add to Build Path]를 클릭

사운드 재생

- MP3 파일 재생
  - ✓ JLayer를 사용하여 MP3 파일 재생
    - ❑ Main 메서드를 소유한 클래스를 만들고 코드 작성

```
import java.io.BufferedInputStream;
import java.io.FileInputStream;
import javazoom.jl.player.Player;
public class MP3Main {
        public static void main(String[] args) {
         new Thread() {
             public void run() {
                 try {
                       FileInputStream fileInputStream = new
FileInputStream("./ttl.mp3");
                     BufferedInputStream bufferedInputStream = new
BufferedInputStream(fileInputStream);
                     Player player = new Player(bufferedInputStream);
                       player.play();
                 } catch (Exception e) {
                     System.out.println(e.getMessage());
                 }
             }
         }.start();
        }
}
```