

Lecture 4

GUI: 그래픽 프로그래밍

2018년도 2학기

컴퓨터프로그래밍2

김 영 국
충남대학교 컴퓨터공학과

이번 주에 학습할 내용

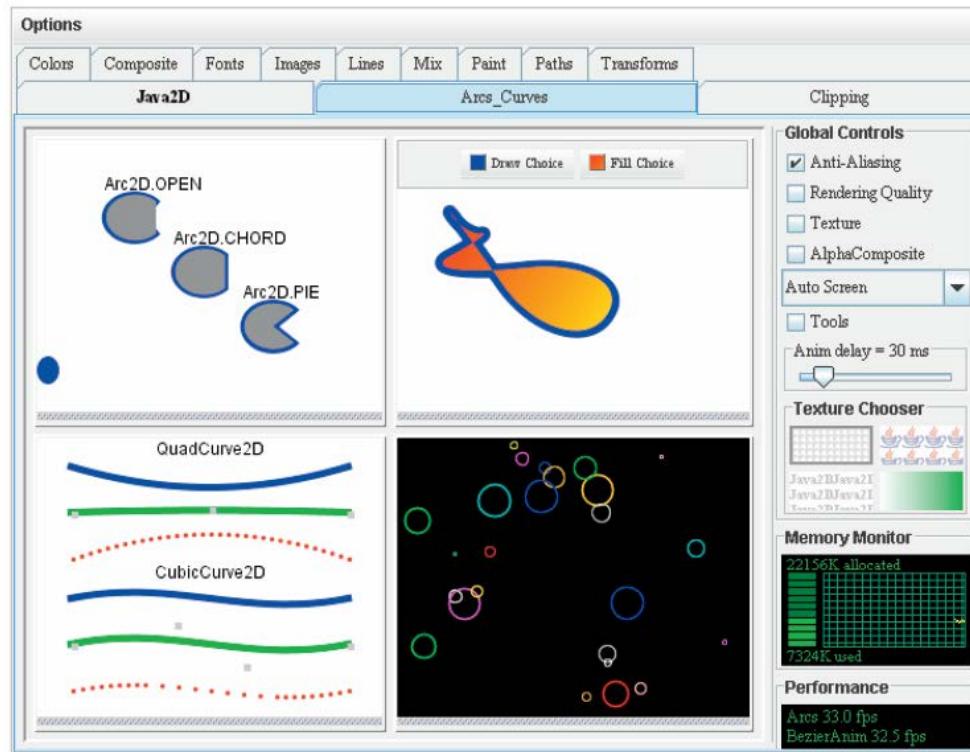


- 자바에서의 그래픽
- 기초 사항
- 기초 도형 그리기
- 색상
- 폰트
- Java 2D
- Java 2D를 이용한 그리기
- Java 2D를 이용한 채우기
- 도형 회전과 평행 이동

자바를
이용하여서
화면에 그림을
그려봅시다.

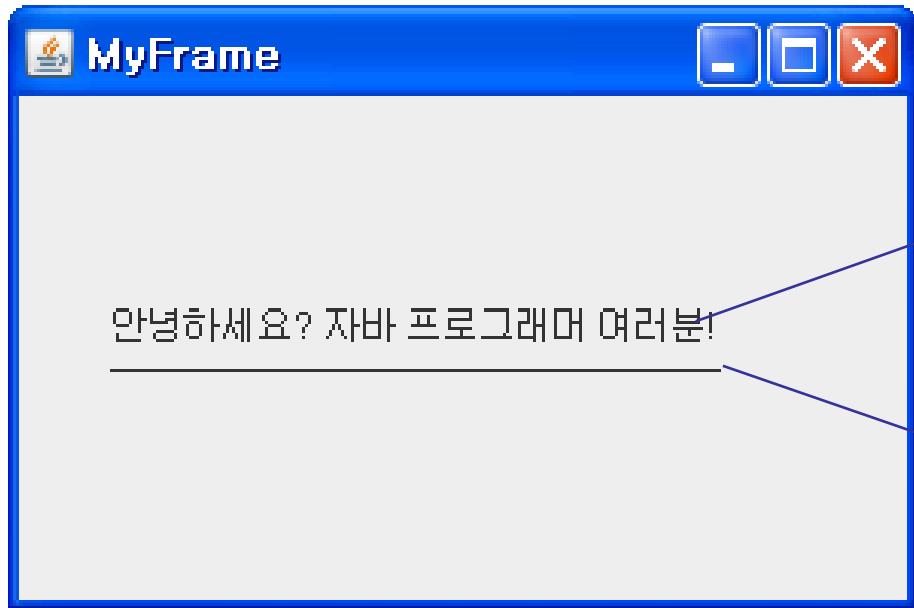


자바에서의 그래픽



< 자바 그래픽 데모 예제(java.sun.com) >

간단한 예제

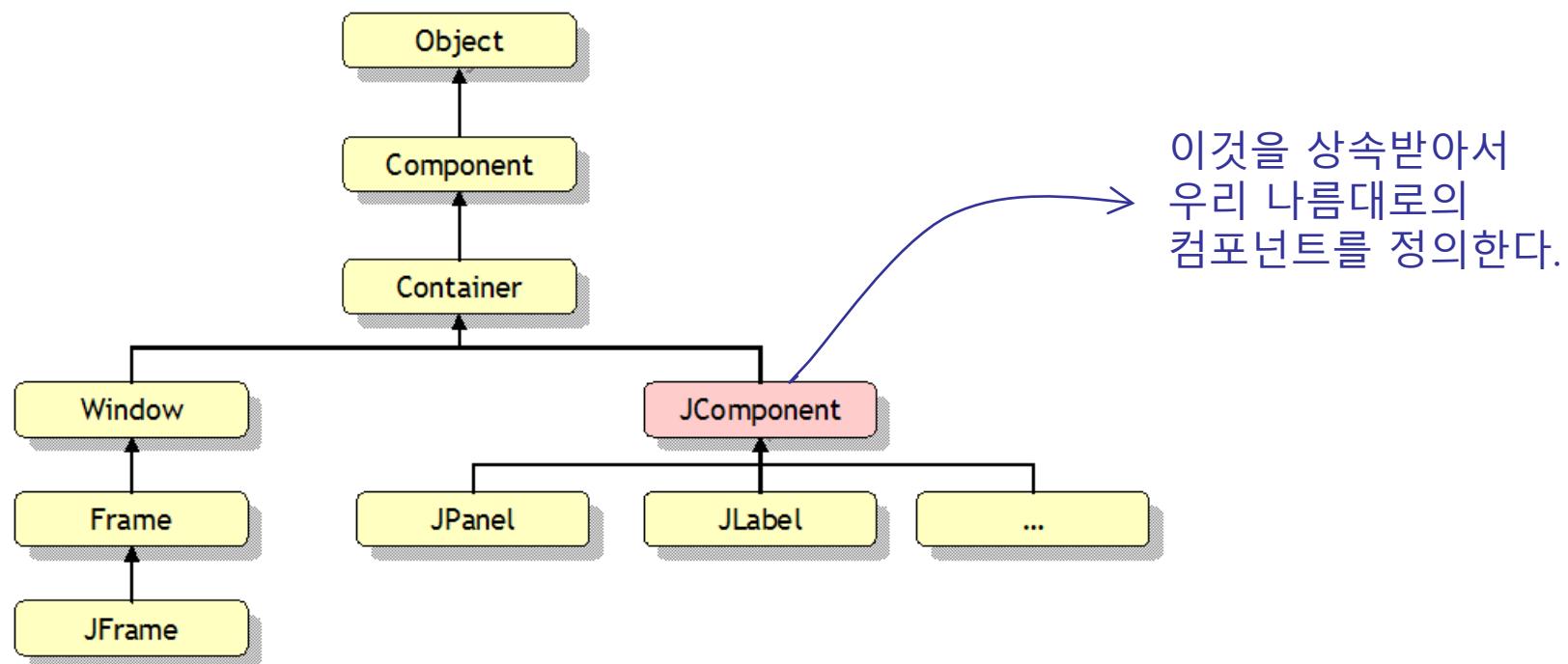


문자열 출력

직선

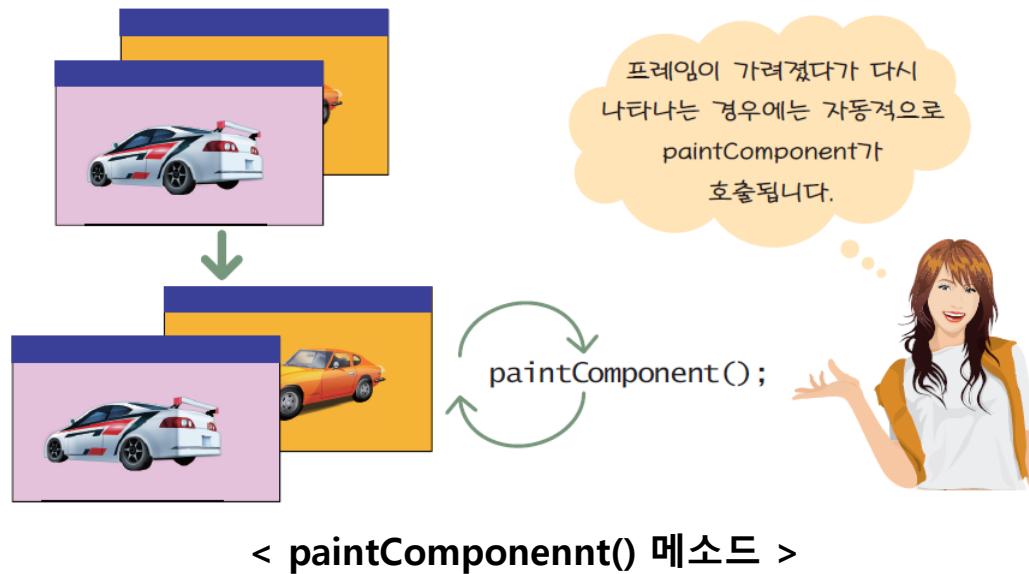
어디에 그릴 것인가?

- JFrame이나 JPanel에도 그릴 수 있지만 우리 나름대로의 컴포넌트를 정의하여 보자.



어떻게 그릴 것인가?

- 컴포넌트에 무언가를 그리려면 paintComponent() 메소드를 중복 정의한다.
- paintComponent() 메소드는 컴포넌트가 화면에 그려질 때 호출된다.

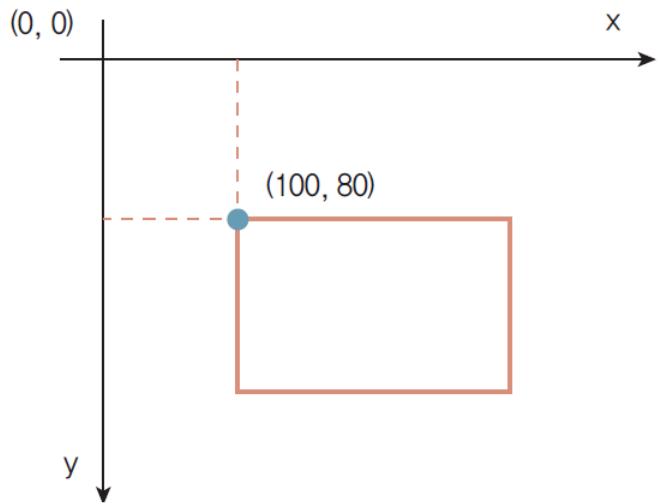


일반적인 코드의 형태

```
class MyPanel extends JPanel
{
    public void paintComponent(Graphics g)
        // 여기에 그림을 그리는 코드를 넣는다.
    }
}
```

paintComponent()를 재정의한다.

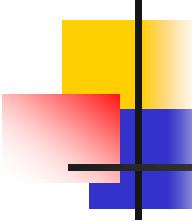
그래픽 좌표계



이 좌표는 아래로
갈수록 증가합니다.



< 자바의 좌표계 >



어떤 메소드를 사용하여야 하는가?

```
g.drawLine(100, 80, 200, 160);
```

```
g.drawString("텍스트", 100, 80); // (100, 80)에 텍스트를 출력
```

예제

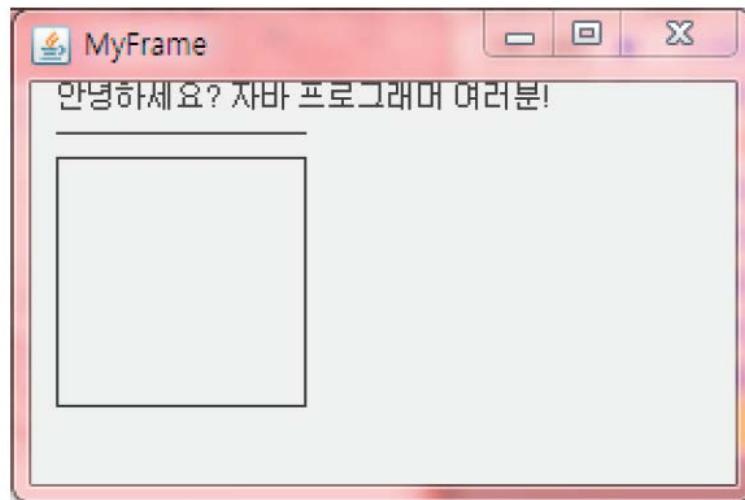
MyFrame.java

```
01 import java.awt.*;
02 import javax.swing.*;
03
04 class MyPanel extends JPanel { // JPanel을 상속받는 MyPanel 정의
05     public void paintComponent(Graphics g) {
06         super.paintComponent(g); // 반드시 부모 클래스의 paintComponent()를 호출
07         g.drawString("안녕하세요? 자바 프로그래머 여러분!", 10, 10);
08         g.drawLine(10, 20, 110, 20); // 텍스트, 직선, 사각형을 그린다.
09         g.drawRect(10, 30, 100, 100);
10     }
11 }
12
13 public class MyFrame extends JFrame {
14     public MyFrame() {
15         setTitle("MyFrame");
16         setSize(300, 200);
17         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18         MyPanel p = new MyPanel(); // MyPanel 객체를 생성
19         setVisible(true);
20         add(p); // MyPanel 객체를 프레임에 추가
21     }
22 }
```

예제

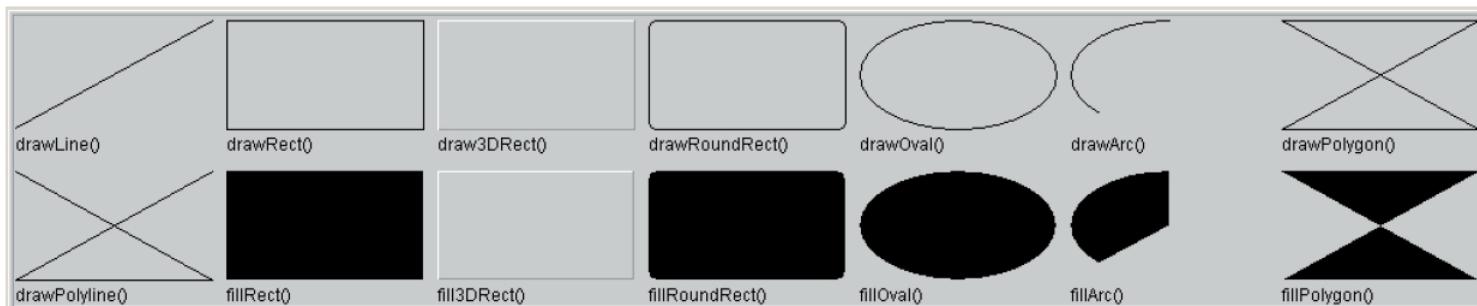
```
23     public static void main(String[] args) {  
24         MyFrame frame = new MyFrame();  
25     }  
26 }
```

실행결과

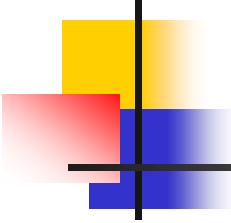


기초 도형 그리기

기초 도형	관련된 메소드
직선	drawLine(), drawPolyline()
사각형	drawRect(), fillRect(), clearRect()
3차원 사각형	draw3DRect(), fill3DRect()
둥근 사각형	drawRoundRect(), fillRoundRect()
타원	drawOval(), fillOval()
호	drawArc(), fillArc()
다각형	drawPolygon(), fillPolygon()

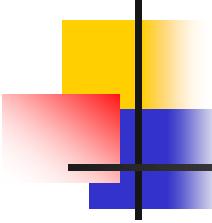


< 그리기 메소드 >



직선 그리기

메소드	설명
<code>drawLine(int x1, int y1, int x2, int y2)</code>	좌표(x1, y1)에서 좌표(x2, y2) 까지 직선을 그린다.
<code>drawPolyline(int[] xpoints, int[] ypoints, int numpoints)</code>	배열 xpoints[]와 배열 ypoints[]을 가지고 여러 개의 직선을 그린다. polygon과 다른 점은 첫 번째 점과 마지막 점이 연결되지 않는다.



사각형 그리기

메소드 및 설명

```
drawRect(int x, int y, int width, int height) // 왼쪽 상단 좌표 (x, y)
```

```
fillRect(int x, int y, int width, int height) // 채워진 사각형
```

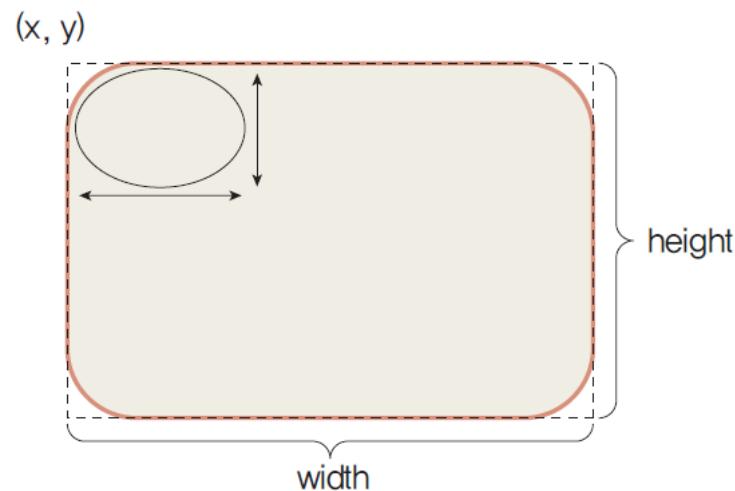
```
draw3DRect(int x, int y, int width, int height, boolean raised) // 3D 사각형
```

```
fill3DRect(int x, int y, int width, int height, boolean raised) // 채워진 3D 사각형
```

```
drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)
```

```
fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)
```

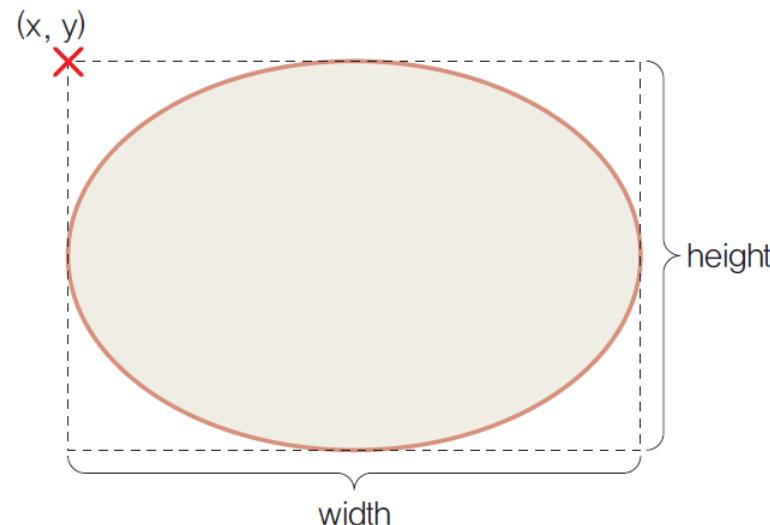
drawRoundRect()



< drawRoundRect()의 매개변수의 의미 >

타원 그리기

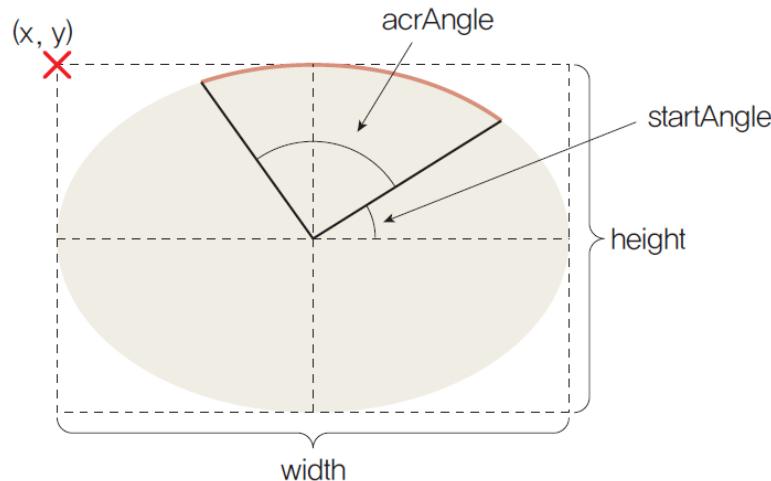
메소드	설명
<code>drawOval(int x, int y, int width, int height)</code>	좌측 상단의 좌표가 x, y이며, 폭 width, 높이 height의 사각형 안에 내접하는 타원을 그린다.
<code>fillOval(int x, int y, int width, int height)</code>	채워진 타원을 그린다.



< **drawOval()** 매개변수의 의미 >

호(Arc)그리기

메소드	설명
<code>drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)</code>	좌측 상단의 좌표가 x, y이며, 폭 width, 높이 height의 사각형 안에 내접하는 타원을 startAngle을 시작 각도로 하여 arcAngle의 각도 만큼의 호를 그린다.
<code>fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)</code>	좌측 상단의 좌표가 x, y이며, 폭 width, 높이 height의 사각형 안에 내접하는 타원을 startAngle을 시작 각도로 하여 arcAngle의 각도 만큼의 채워진 호를 그린다.



< drawArc() 매개변수의 의미 >

예제

- 아래 그림과 비슷한 얼굴을 그려보자.



SnowManFace.java

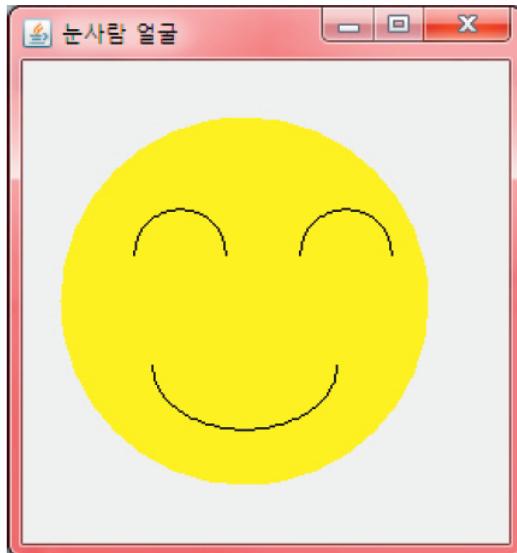
```
01 ...
02
03 class MyPanel extends JPanel {
04
05     public void paintComponent(Graphics g) {
06         super.paintComponent(g);
07         g.setColor(Color.YELLOW);           ← 전경색을 노란색으로 변경한다.
08         g.fillOval(20, 30, 200, 200);      ← 채워진 타원을 그린다.
09         g.setColor(Color.BLACK);          ← 전경색을 검정색으로 변경한다.
10         g.drawArc(60, 80, 50, 50, 180, -180); // 왼쪽 눈
11         g.drawArc(150, 80, 50, 50, 180, -180); // 오른쪽 눈
```

예제

```
12         g.drawArc(70, 130, 100, 70, 180, 180); // 입
13     }
14 }
15
16 public class SnowManFace extends JFrame {
17     public SnowManFace() {
18         setSize(280, 300);
19         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20         setTitle("눈사람 얼굴");
21         setVisible(true);
22         add(new MyPanel());
23     }
24
25     public static void main(String[] args) {
26         SnowManFace s = new SnowManFace();
27     }
28 }
```

실행 결과

실행결과

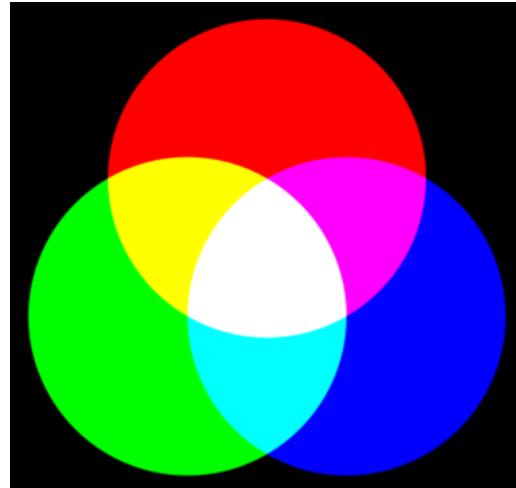


중간점검

1. SnowManFace에 버튼을 추가하고 이 버튼이 눌리면 징그린 얼굴로 변경되도록 소스를 수정하라.
2. 얼굴의 디테일을 좀 더 추가하여 보자.

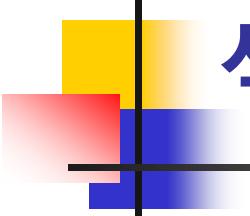
색상

- java.awt 패키지의 일부인 Color 클래스를 사용
- 빛의 3원색인 Red 성분, Green 성분, Blue 성분이 얼마나 함유되어 있는지를 0에서 255까지의 수를 사용하여 나타낸다.



색상

클래스 변수 이름	RGB 값
Color.black	(0,0,0)
Color.blue	(0,0,255)
Color.cyan	(0,255,255)
Color.gray	(128,128,128)
Color.darkGray	(64,64,64)
Color.lightGray	(192,192,192)
Color.green	(0,255,0)
Color.magenta	(255,0,255)
Color.orange	(255,200,0)
Color.pink	(255,175,175)
Color.red	(255,0,0)
Color.white	(255,255,255)
Color.yellow	(255,255,0)



색상 설정 방법

생성자	설명
<code>setBackground(Color c)</code>	컴포넌트 객체에서 배경색을 설정한다.
<code>setColor(Color c)</code>	전경색을 설정한다.
<code>Color getColor()</code>	현재의 전경색을 반환한다.

ColorTest.java

```
01 ...
02
03 class MyPanel extends JPanel implements ActionListener {
04     JButton button;
05     Color color = new Color(0, 0, 0);
06
07     public MyPanel() {
08         setLayout(new BorderLayout());
09         button = new JButton("색상 변경");
10         button.addActionListener(this);
11         add(button, BorderLayout.SOUTH);
12     }
13
14     public void paintComponent(Graphics g) {
15         super.paintComponent(g);
16         g.setColor(color);
17         g.fillRect(10, 10, 210, 220);
18     }
19
20     public void actionPerformed(ActionEvent e) {
21         color = new Color((int)(Math.random() * 255.0),
22                           (int)(Math.random() * 255.0), (int)(Math.random() * 255.0));
23         repaint();
24     }
25 }
```

setLayout()은 컨테이너의 배치 관리자를 설정하는 메소드이다.

버튼에 이벤트 처리기를 붙인다 (다음 장에서 학습한다).

버튼이 눌려지면 호출된다 (다음 장에서 학습한다).

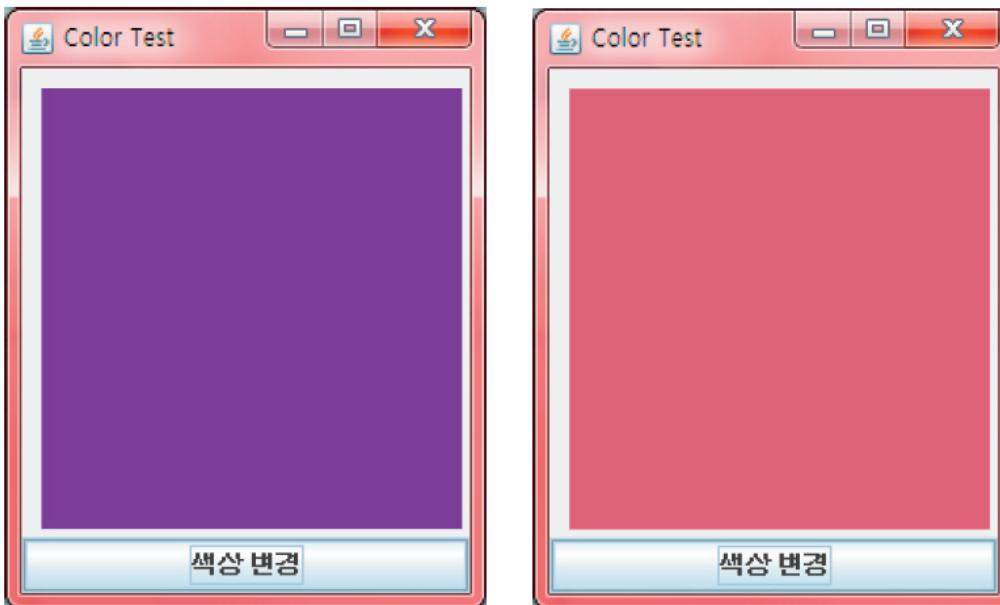
색상을 난수로 변경한다.

repaint()는 현재 컨테이너를 다시 그리게 하는 메소드이다. 절대로 paint()를 직접 호출하면 안된다.

예제

```
24      }
25  }
26
27 public class ColorTest extends JFrame {
28     public ColorTest() {
29         setSize(240, 300);
30         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
31         setTitle("Color Test");
32         setVisible(true);
33         JPanel panel = new MyPanel();
34         add(panel);
35     }
36
37     public static void main(String[] args) {
38         ColorTest s = new ColorTest();
39     }
40
41 }
```

실행 결과



색상 선택기



< 색상 선택기 테스트 프로그램 >

예제

```
JColorChooser colorChooser = new JColorChooser(Color.RED);
```

ColorChooserTest.java

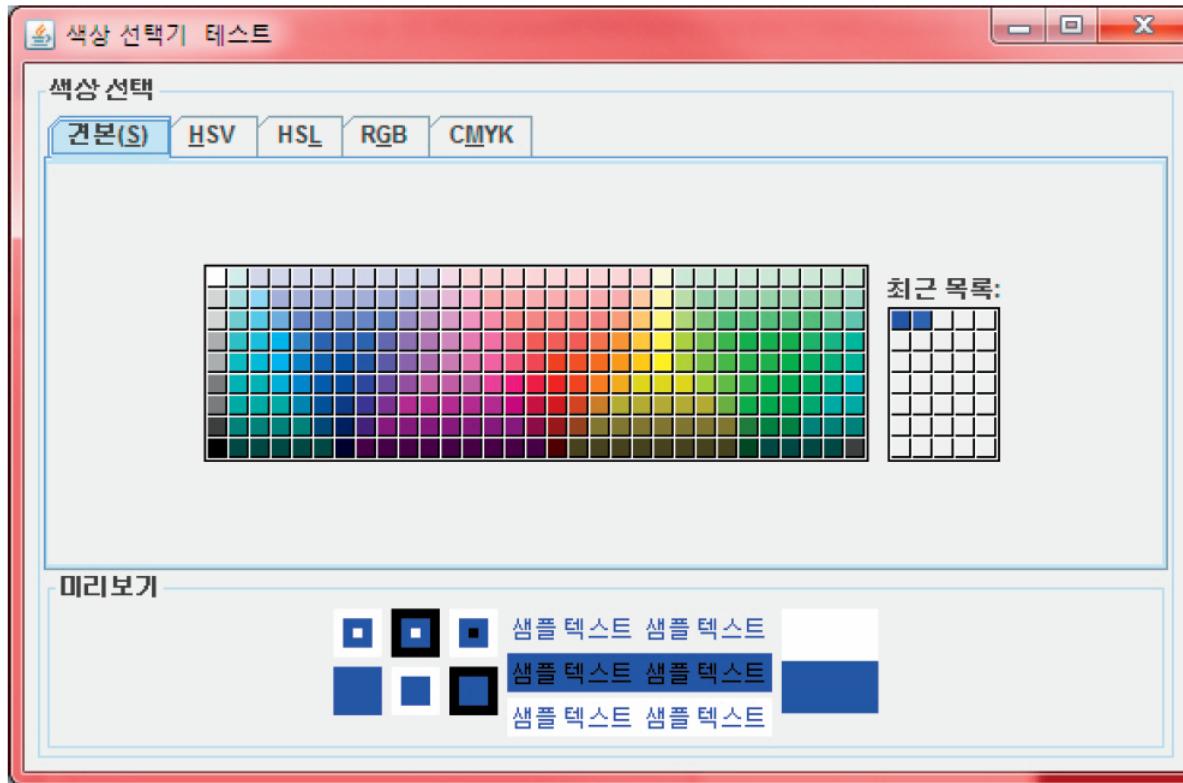
```
01  ...
02
03  public class ColorChooserTest extends JFrame implements ChangeListener {
04
05      protected JColorChooser color
06      public ColorChooserTest() {
07          setTitle("색상 선택기 테스트");
08          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
09
10          color = new JColorChooser(); // 생성자 호출
11          color.getSelectionModel().addChangeListener(this); // 리스너 등록
12          color.setBorder(BorderFactory.createTitledBorder("색상 선택"));
13
14          JPanel panel = new JPanel();
15          panel.add(color); <----- 패널에 색상 선택기 객체를 추가한다.
16          add(panel);
17
18          pack(); <----- 프레임의 크기를 입축한다.
19          this.setVisible(true);
20
```

예제

```
21     }
22     public void stateChanged(ChangeEvent e) {
23         Color newColor = color.getColor();
24         System.out.println(newColor);
25     }
26     public static void main(String[] args) {
27         new ColorChooserTest();
28     }
29 }
```

←----- 사용자가 색상을 선택하면 호출된다.

실행 결과



문자열 출력과 폰트

- 문자열 출력 방법

// (x, y) 위치에 문자열을 출력하려면

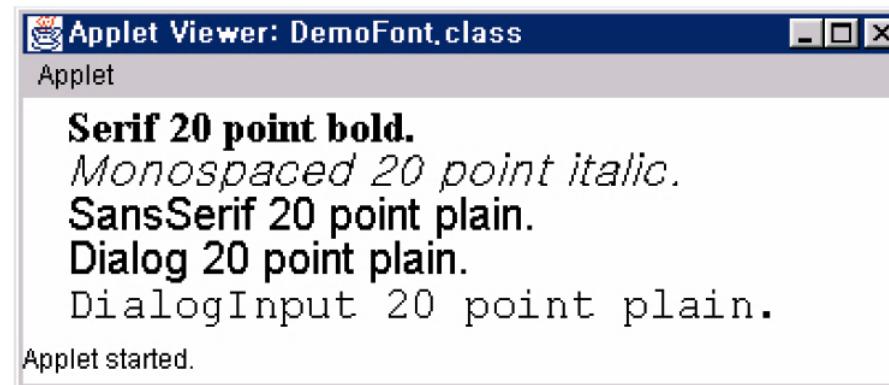
```
g.drawString("Hello World!", x, y);
```

- 폰트를 지정하기 위해서는 Font 클래스를 사용
- Font 객체는 폰트 이름(Courier, Helvetica,...)과 스타일 (plain, bold, italic,...), 크기(12포인트,...)의 3가지 속성

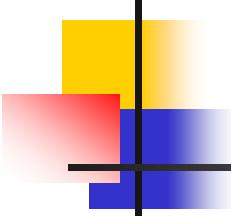
```
Font font = new Font("Courier", Font.PLAIN, 10); // plain 형식이고 크기는 10포인트
```

폰트의 종류

논리적인 폰트	설명
"Serif"	빼침(serif)을 갖는 가변폭 글꼴, 대표적으로 TimesRoman이 있다.
"SansSerif"	빼침(serif)을 갖지 않는 가변폭 글꼴, 대표적으로 Helvetica가 있다.
"Monospaced"	고정폭을 가지는 글꼴, 대표적으로 Courier가 있다.
"Dialog"	대화상자에서 텍스트 출력을 위하여 사용되는 글꼴
"DialogInput"	대화상자에서 텍스트 입력을 위하여 사용되는 글꼴



< FontTest 예제 프로그램 실행 화면 >



폰트의 종류

폰트 스타일	설명
Font.BOLD	볼드체
Font.ITALIC	이탤릭체
Font.PLAIN	표준체

폰트의 지정

```
public void paintComponent(Graphics g)
{
    Font f = new Font("Serif", Font.BOLD | Font.ITALIC, 12);
    g.setFont(f);
    ...
}
```

```
JLabel myLabel = new JLabel("폰트 색상");
Font f = new Font("Dialog", Font.ITALIC, 10); // ①
myLabel.setFont(f); // ②
```



중간점검

1. 만약 폰트 참조 변수 f가 더 이상 필요하지 않은 경우에 ①과 ②를 합쳐서 하나의 문장으로 작성하여 보라.

폰트의 나열

FontLister.java

```
01 // 폰트 목록 출력
02 import java.awt.*;
03
04 public class FontLister {
05     public static void main(String[] args){
06         String[] font_list;
07         GraphicsEnvironment g;
08         g = GraphicsEnvironment.getLocalGraphicsEnvironment();
09         font_list = g.getAvailableFontFamilyNames();
10         for(int i=0;i<font_list.length;i++)
11             System.out.println(font_list[i]);
12     }
13 }
```

실행결과

Agency FB
Arial
Arial Black
...

예제

FontTest.java

```
01 class MyPanel extends JPanel {  
02  
03     Font f1, f2, f3, f4, f5;  
04  
05     public MyPanel() {  
06         f1 = new Font("Serif", Font.PLAIN, 20);  
07         f2 = new Font("San Serif", Font.BOLD, 20);  
08         f3 = new Font("Monospaced", Font.ITALIC, 20);  
09         f4 = new Font("Dialog", Font.BOLD | Font.ITALIC, 20);  
10         f5 = new Font("DialogInput", Font.BOLD, 20);  
11     }  
12  
13     public void paintComponent(Graphics g) {  
14         super.paintComponent(g);  
15         g.setFont(f1); ←————— 폰트를 변경한다.  
16         g.drawString("Serif 20 points PLAIN", 10, 50);  
17         g.setFont(f2);  
18         g.drawString("SanSerif 20 points BOLD", 10, 70);  
19         g.setFont(f3);  
20         g.drawString("Monospaced 20 points ITALIC", 10, 90);  
21         g.setFont(f4);  
22         g.drawString("Dialog 20 points BOLD + ITALIC", 10, 110);  
}
```

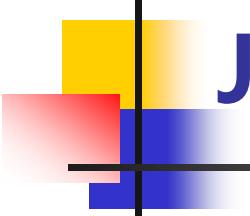
생성자에서 논리적인
폰트를 생성한다.

예제

```
20         g.drawString("Monospaced 20 points ITALIC", 10, 90);
21         g.setFont(f4);
22         g.drawString("Dialog 20 points BOLD + ITALIC", 10, 110);
23         g.setFont(f5);
24         g.drawString("DialogInput 20 points BOLD", 10, 130);
25     }
26 }
27
28 public class FontTest extends JFrame {
29     public FontTest() {
30         setSize(500, 200);
31         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
32         setTitle("Font Test");
33         JPanel panel = new MyPanel();
34         add(panel);
35         setVisible(true);
36     }
37
38     public static void main(String[] args) {
39         FontTest s = new FontTest();
40     }
41 }
```

실행 결과

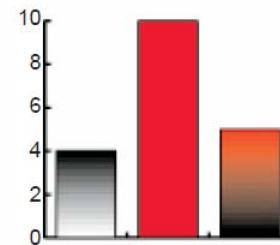
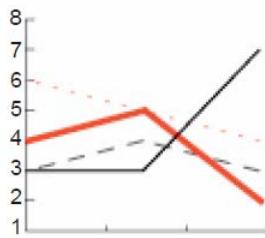




Java 2D

- 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



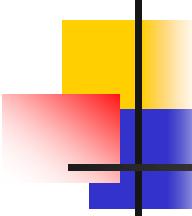
Blur



Sharpen

Using image-filtering operations

< Java 2D를 이용한 그래픽의 예 (출처:java.sun.com) >



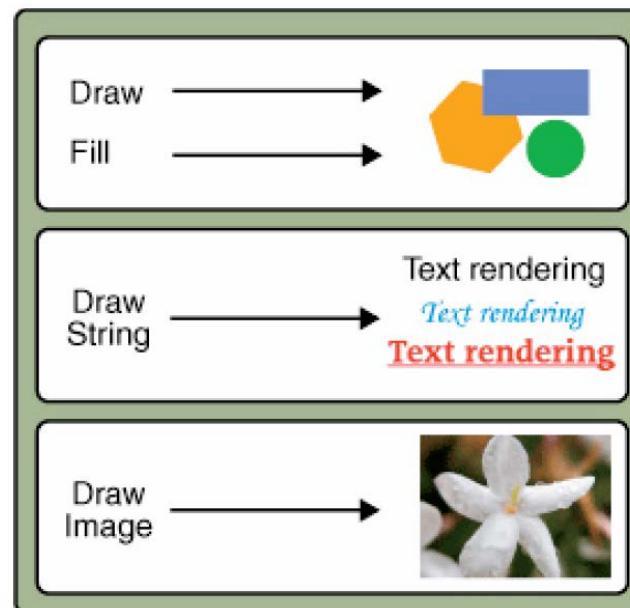
Java 2D를 사용하려면?

```
public void paintComponent(Graphics g)
{
    Graphics2D g2 = (Graphics2D) g; ←
    ...
}
```

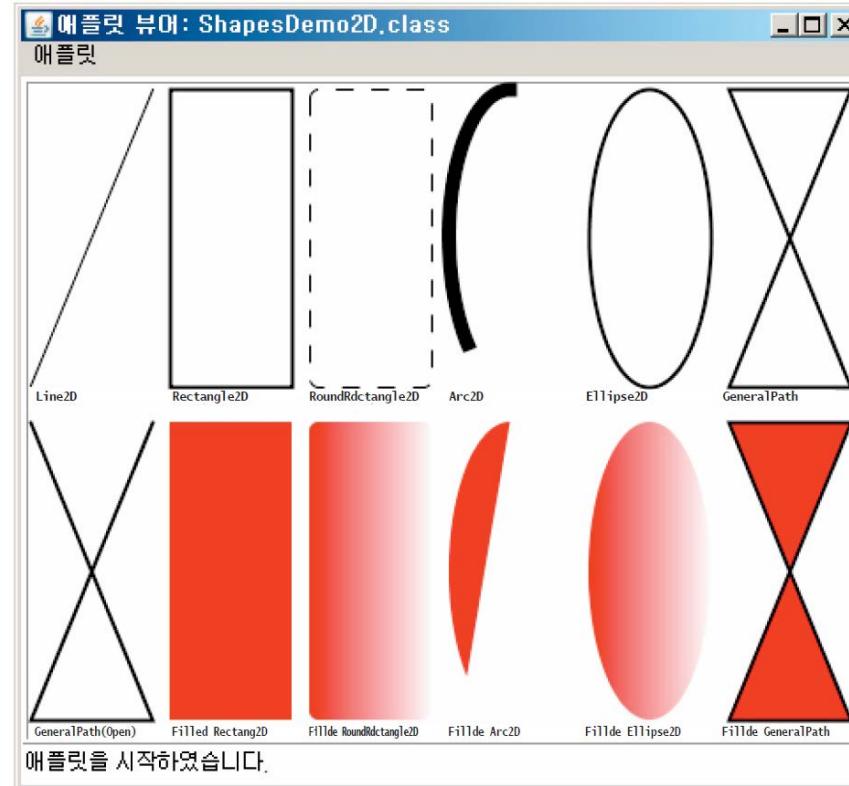
Java2D를 사용하려면 단순히
Graphics 객체 변수를 Graphics2D
타입으로 형변환한다.

Java 2D의 메소드

```
public void paintComponent(Graphics g)
{
    Graphics2D     g2 = (Graphics2D) g;
    g2.drawLine(100, 100, 300, 300);
    g2.drawRect(10, 10, 100, 100);
    ...
}
```



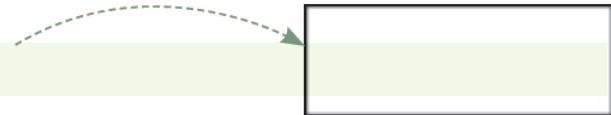
Java 2D를 이용한 그리기



< Java 2D를 이용한 형상 그리기 (출처: 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));
```



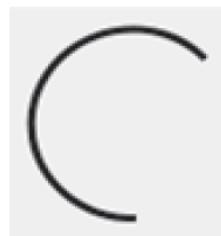
타원생성

```
// 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

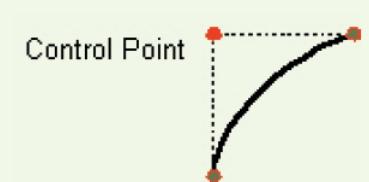


(c) Arc2D.PIE

< 원호의 종류 >

2차/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

```
// 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>(); ← 향상된 배열인 컬렉션의 일종인 ArrayList를
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

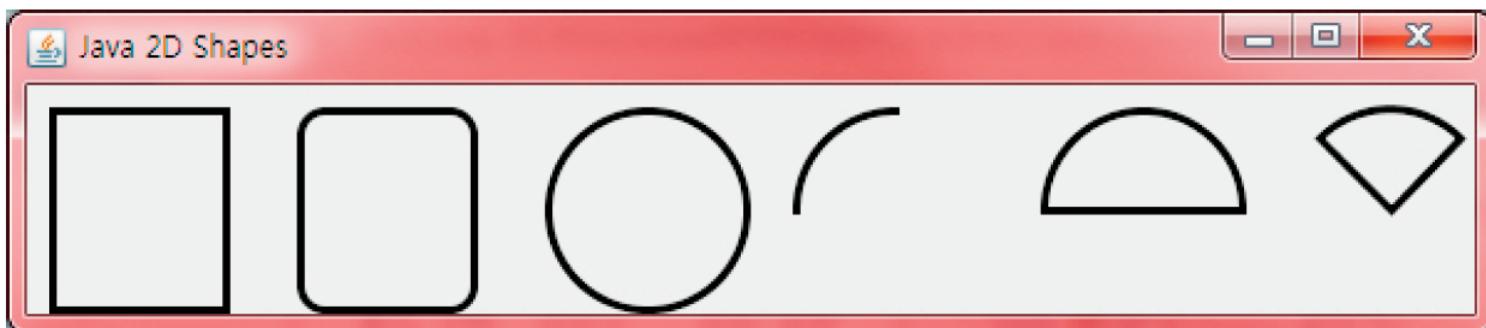
```
44  
45     g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,  
46                           RenderingHints.VALUE_ANTIALIAS_ON);  
47  
48     g2.setColor(Color.BLACK);  
49     g2.setStroke(new BasicStroke(3));  
50     for (Shape s : shapeArray)  
51         g2.draw(s);  
52     }  
53 }
```

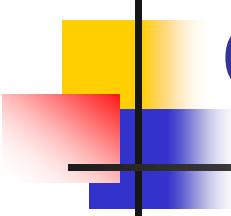
44
45 g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
46 RenderingHints.VALUE_ANTIALIAS_ON); ← 앤티에일리어싱은 도형을 매끄럽게 그리기
47
48 g2.setColor(Color.BLACK);
49 g2.setStroke(new BasicStroke(3)); ← 위하여 설정한다. 연산 시간은 조금 더 걸리지
50 for (Shape s : shapeArray) ← 만 그만큼 그래픽의 품질이 좋아진다.
51 g2.draw(s);
52 }
53 }

setStroke() 메소드를 이용하여서 도
형을 그리는 두께를 설정할 수 있다.

shapeArray에 저장된 Shape 객체들을 꺼내서 화
면에 그려준다.
Java 2D의 도형들은 모두 Shape 인터페이스를 구
현하기 때문에 Shape 타입으로 생각할 수 있다.

실행결과





이미지 출력

- 자바는 GIF, PNG JPEG 타입의 이미지를 화면에 그릴 수 있다.

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

예제

MyImageFrame.java

```
01 ...
02
03 public class MyImageFrame extends JFrame {
04
05     BufferedImage img=null;
06     public MyImageFrame() {
07         setTitle("Image Load Test");
08
09         try {
10             img = ImageIO.read(new File("cat.jpg")); ← 이미지를 읽는다.
11         } catch (IOException e) {
12             System.out.println(e.getMessage()); ← 오류가 발생하면 프로그램을
13             System.exit(0); 종료한다.
14         }
15
16         add(new MyPanel());
17         pack();
18         setVisible(true);
19     }
20 }
```

예제

```
20  
21     class MyPanel extends JPanel {  
22         public void paint(Graphics g) {  
23             g.drawImage(img, 0, 0, null);  
24         }  
25  
26         public Dimension getPreferredSize() {  
27             if (img == null) {  
28                 return new Dimension(100, 100);  
29             } else {  
30                 return new Dimension(img.getWidth(null), img.getHeight(null));  
31             }  
32         }  
33     }  
34  
35     public static void main(String[] args) {  
36         new MyImageFrame();  
37     }  
38 }
```

필드 img를 사용하기 위해서
MyPanel을 내부 클래스로 선언

이미지의 크기에 패널의 크기를 맞춘다.
getPreferredSize()는 컴포넌트가 원하는 크기를
배치 관리자에게 알려는 메소드이다.
현재 일은 이미지의 크기로 다시 설정한다.

실행 결과



이미지 그리기

```
boolean drawImage(Image img,  
                  int x, int y,  
                  ImageObserver observer);
```

```
boolean drawImage(Image img,  
                  int dstx1, int dsty1, int dstx2, int dsty2,  
                  int srcx1, int srcy1, int srcx2, int srcy2,  
                  ImageObserver observer);
```



예제

MyImageFrame1.java

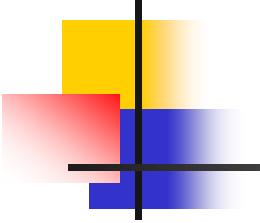
```
01  public class MyImageFrame1 extends JFrame implements ActionListener {  
02  
03      BufferedImage img;  
04      private int pieces = 4;  
05      private int totalPieces = pieces * pieces;  
06      private int[] pieceNumber;  
07  
08      public MyImageFrame1() {  
09          setTitle("Image Draw Test");  
10         try {  
11             img = ImageIO.read(new File("space.jpg"));  
12         } catch (IOException e) {  
13             System.out.println(e.getMessage());  
14             System.exit(0);  
15         }  
16     }  
17     pieceNumber = new int[totalPieces];  
18     for (int i = 0; i < totalPieces; i++) {  
19         pieceNumber[i] = i;  
20     }
```

pieceNumber라는 이름의 배열을 생성한다. pieceNumber 안에는 각 조각들의 번호가 기록된다. 처음에는 첫 번째 조각의 번호가 1, 두 번째 조각의 번호는 2, ... 이런 식으로 순차 번호가 기록된다.

이미지를 읽는다.

예제

```
21     add(new MyPanel(), BorderLayout.CENTER);
22     JButton button = new JButton("DIVIDE");
23     button.addActionListener(this); ← 버튼에 이벤트 처리기를 붙인다
24     add(button, BorderLayout.SOUTH);
25     setSize(600, 600);
26     setVisible(true);
27 }
28
29 void divide() {
30     Random rand = new Random();
31     int ri;
32     for (int i = 0; i < totalPieces; i++) {
33         ri = rand.nextInt(totalPieces);
34         int tmp = pieceNumber[i];
35         pieceNumber[i] = pieceNumber[ri];
36         pieceNumber[ri] = tmp;
37     }
38 }
39
40 class MyPanel extends JPanel {
41     public void paintComponent(Graphics g) {
42         super.paintComponent(g);
43         int pieceWidth = img.getWidth(null) / pieces;
44         int pieceHeight = img.getHeight(null) / pieces;
45         for (int x = 0; x < pieces; x++) { ← divide()에서는 조각들의
                                번호를 난수로 만든다.
```



```
46     int sx = x * pieceWidth;
47     for (int y = 0; y < pieces; y++) {
48         int sy = y * pieceHeight;
49         int number = pieceNumber[x * pieces + y];
50         int dx = (number / pieces) * pieceWidth;
51         int dy = (number % pieces) * pieceHeight;
52         g.drawImage(img, dx, dy, dx + pieceWidth, dy + pieceHeight,
53                     sx, sy, sx + pieceWidth, sy + pieceHeight, null);
54     }
55 }
56 }
57 }
58 }
59 }
60 }
61 public static void main(String[] args) {
62     new MyFrame();
63 }
64 }
65 public void actionPerformed(ActionEvent e) {
66     divide();
67     repaint(); ←
68 }
69 }
70 }
```

MyPanel 클래스의 paint() 메소드에서 각 조각의 위치를 계산하여서 그 위치에 각 조각을 그린다.

버튼이 클릭될 때마다 다시 나누고 각 조각들을 다시 그린다.

실행 결과



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);
```

예제

FillShapes.java

GradientPaint 객체를
생성한다.

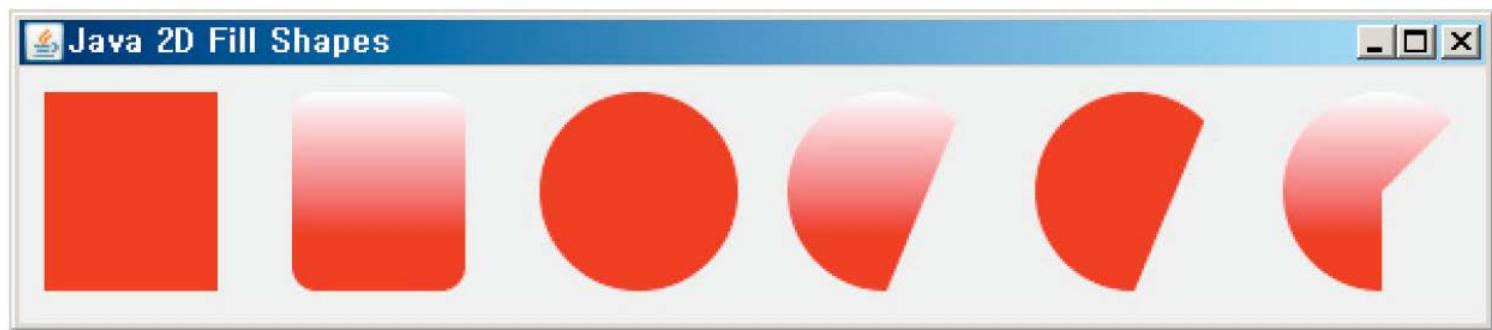
GradientPaint 객체로
채우는 색상을 지정

```
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      ...
}
```

예제

```
22     }  
23 }
```

실행결과



Q & A

