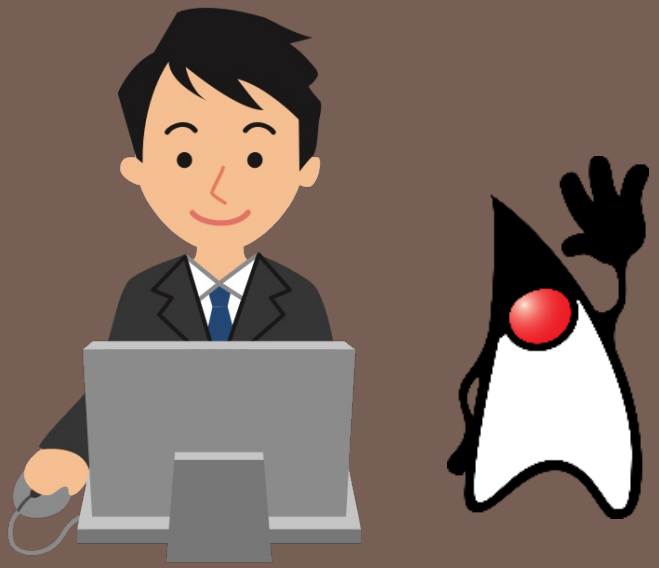
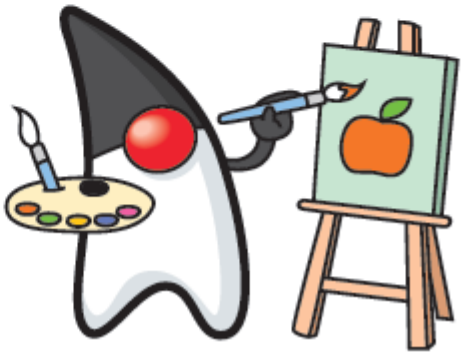
파워자바(개정3판)



12장 자바 그래픽

12장의 목표

1. 화면에 도형들을 그릴 수 있나요?
2. 화면에 이미지를 그릴 수 있나요?
3. 버튼을 눌러서 그림을 변경할 수 있나요?
4. 마우스로부터 좌표를 받아서 그림을 그릴 수 있나요?



자바 그래픽

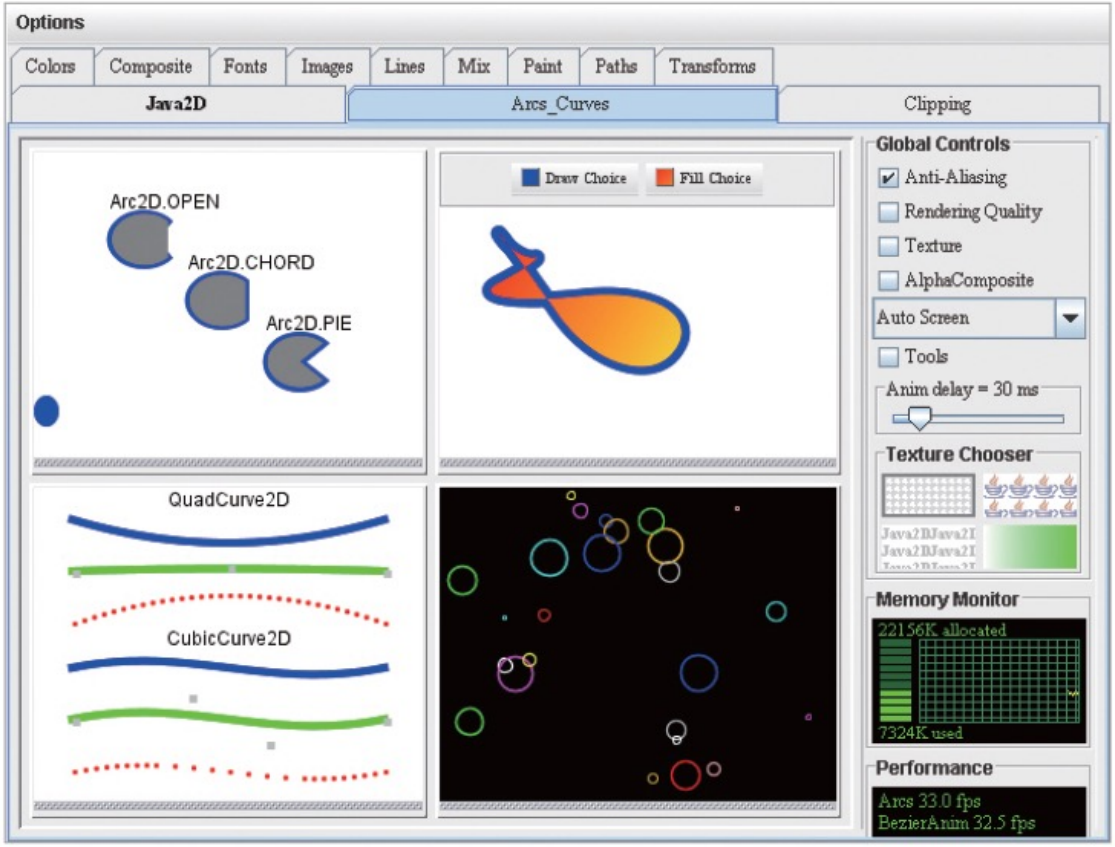


그림 12.1 자바 그래픽 데모 예제(java.sun.com)

어디에 그릴까?

- 우리는 Jpanel 위에 그리도록 하자. JPanel은 그림을 그릴 수 있는 화면을 가지고 있고, 동시에 컨테이너의 역할도 한다.

```
class MyPanel extends JPanel
{
   ...
}
public class MyFrame extends JFrame
{
   public MyFrame(){
      MyPanel panel = new MyPanel();
      add(panel);
   }
}
```

프레임 안에 패널을 추가하고 패널 위에 그림을 그리자.

어떻게 그리는가?

- 컴포넌트 위에 그림을 그리기 위해서는 컴포넌트가 가지고 있는 paintComponent() 메소드를 재정의하여야 한다.
- paintComponent()는 컴포넌트가 자신의 모습을 화면에 그리는 메소드라고 생각하면 된다. 모든 컴포넌트가 paintComponent()를 가지고 있다

언제 paintComponent()가 호출되는 것일까?

- paintComponent()는 컴포넌트를 다시 그릴 필요가 있을 때마다 자바 시스템에 의하여 호출된다.

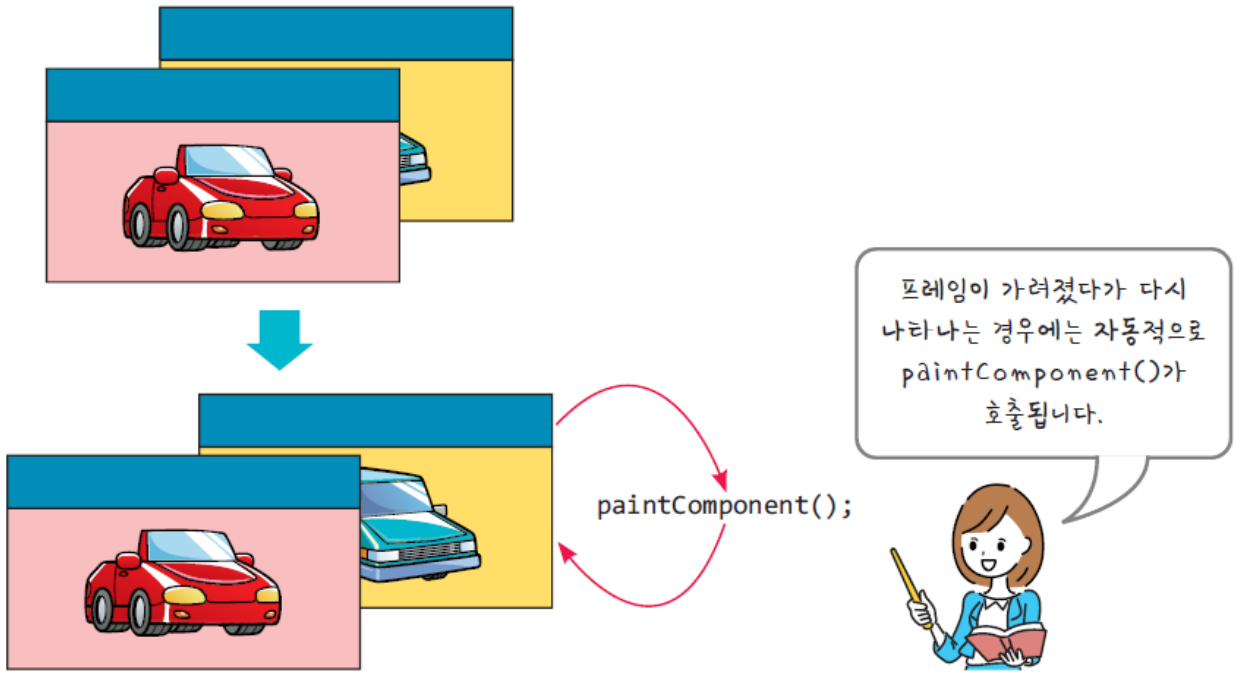


그림 12.2 paintComponent() 메소드

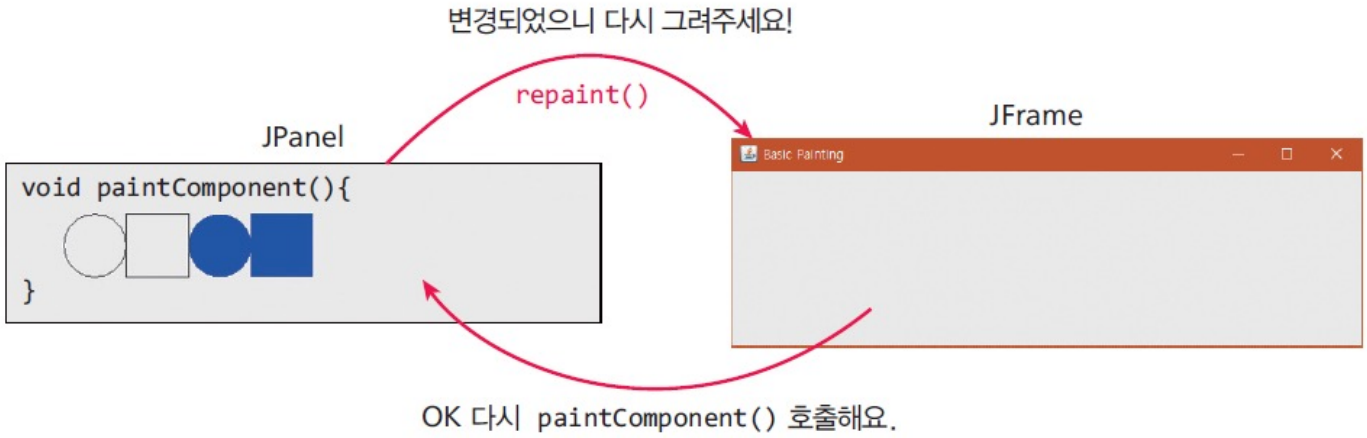
그래픽 구조

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

paintComponent()를 재정의한다.

repaint() 메소드

- 사용자가 paintComponent()를 직접 호출하면 안 된다. 이 메소드는 반드시 자동으로 호출되어야 한다. 그런데 만약 사용자가 화면을 다시 그리고 싶으면 어떻게 해야 하는가? 이 경우에는 repaint()를 호출하면 된다. repaint()가 적절한 시기에 paintComponent()를 호출한다.



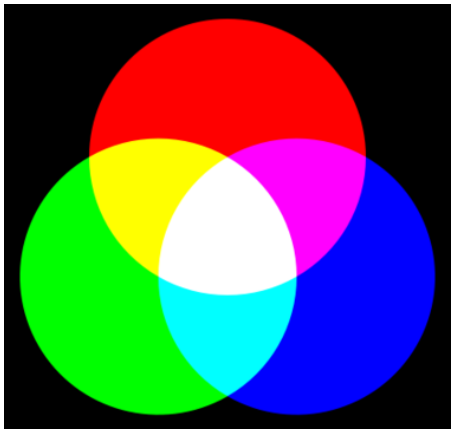
예제:

- 다음과 같은 화면을 생성하는 예제를 작성해보고 지나가자.
  paintComponent() 메소드를 오버라이드하면 된다.

색상 변경하기

- Color c = Color.magenta;
- Color c = **new** Color (255, 0, 255);

- setBackground(Color c) - 컴포넌트 객체에서 배경색을 설정한다.
- setColor(Color c) - 전경색을 설정한다.
- Color getColor() - 현재의 전경색을 반환한다.



윈도우에 문자열 출력

- 윈도우에 텍스트를 출력하려면 drawString() 메소드를 사용하여야 한다.

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

폰트 객체 생성

- 폰트를 지정하기 위해서는 Font 클래스를 사용한다. 각 Font 객체는 폰트 이름(Courier, Helvetica,..)과 스타일(plain, bold, italic,...), 크기(12포인트,...)의 3가지 속성을 가지고 있다.

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

논리적인 폰트

논리적인 폰트	설명
"Serif"	삐침(serif)를 갖는 가변폭 글꼴, 대표적으로 TimesRoman이 있다.
"SansSerif"	삐침(serif)를 갖지않는 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

폰트 수식자

- 생성자의 두 번째 매개 변수는 볼드(bold)나 이탤릭(italic)과 같은 수식자이다. 폰트의 스타일에는 다음의 3가지가 있다.

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

폰트 설정 방법

- 컴포넌트나 Graphics 객체에서 폰트를 지정하기 위해서는 setFont() 메소드를 사용한다.

```
public void paint(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
```

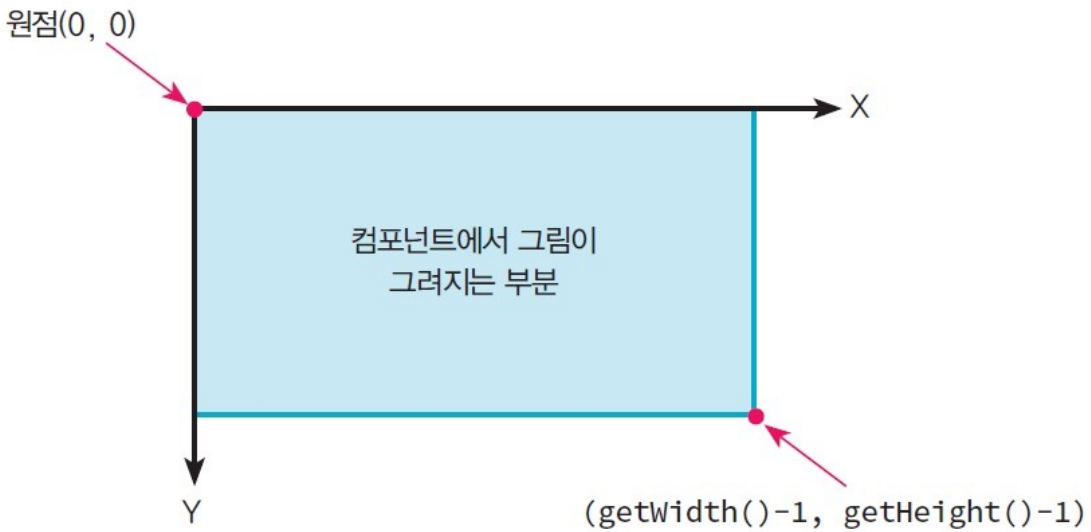
예제:

- 논리적인 폰트들을 생성하고 각각의 폰트를 사용하여서 문자열을 출력하여 보자.

Graphics 클래스

- 텍스트 문자열: drawString() 메소드를 통해 텍스트를 화면에 그릴 수 있다.
- 기초 도형들: drawXxx()과 fillXxx() 형식의 메소드를 통하여 직선, 사각형, 타원 등을 화면에 그릴 수 있다.
- 이미지 : drawImage() 메소드를 통해 화면에 이미지를 그릴 수 있다.

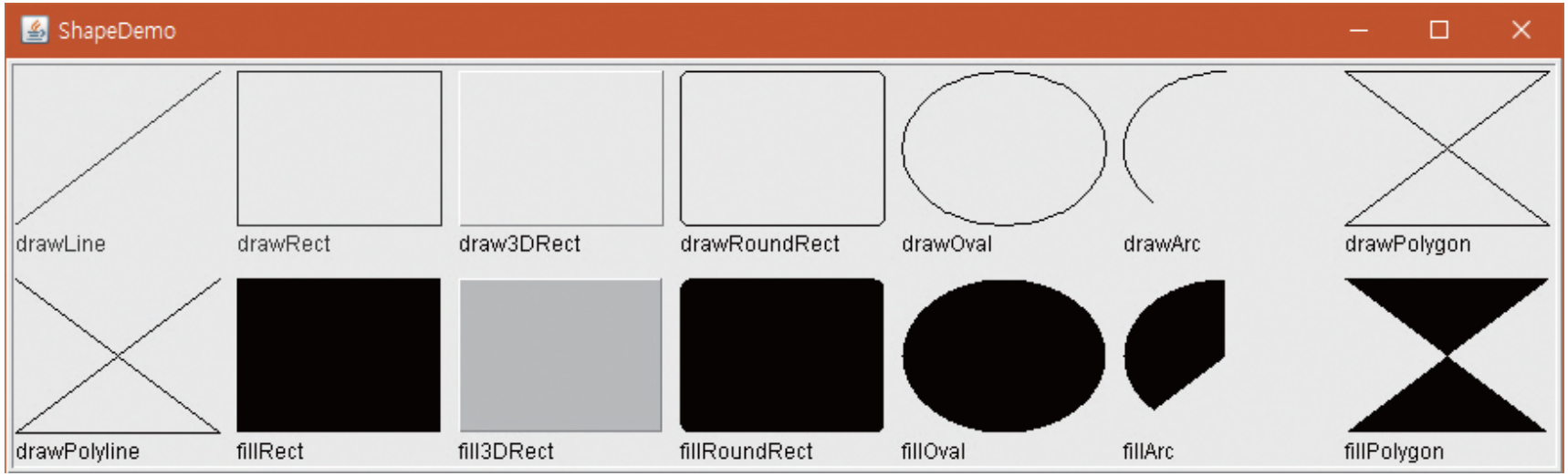
그래픽 좌표계



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()

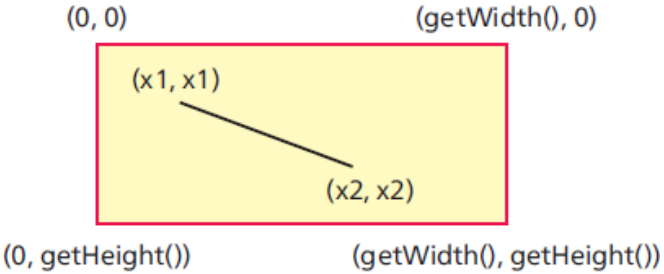
그리기 메소드


ShapeDemo
drawLine
drawRect
draw3DRect
drawRoundRect
drawOval
drawArc
drawPolygon
drawPolyline
fillRect
fill3DRect
fillRoundRect
fillOval
fillArc
fillPolygon

그리기 메소드

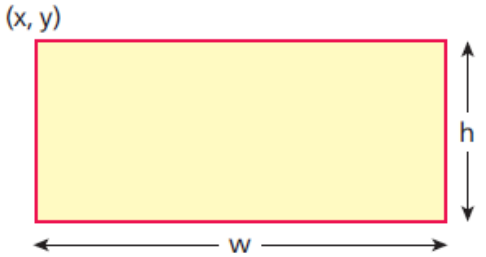
drawLine(int x1, int y1, int x2, int y2)

좌표 (x1,y1)에서 좌표 (x2,y2) 까지 직선을 그린다.



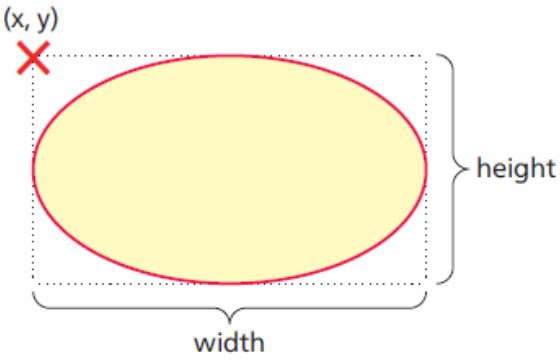
drawRect(int x, int y, int width, int height)

좌표 (x1,y1)에서 좌표 (x2,y2) 까지 직선을 그린다.



drawOval(int x, int y, int width, int height)

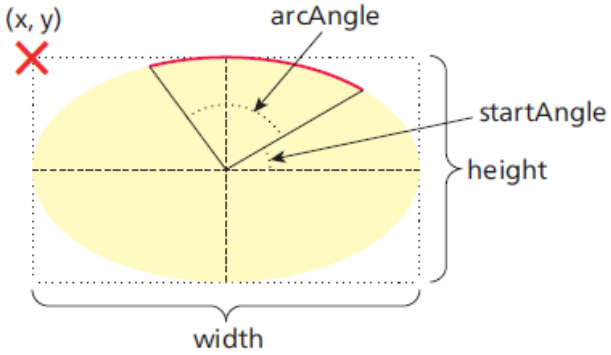
좌측 상단의 좌표가 x,y이며 폭 width, 높이 height인 타원을 그린다.



그리기 메소드

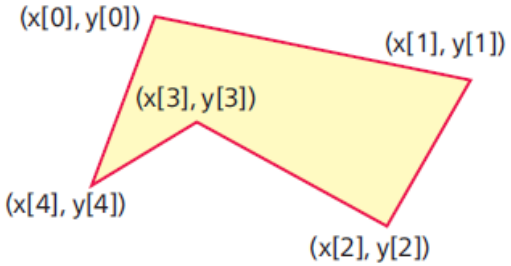
drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)

좌측 상단의 좌표가 x, y이며 폭 width, 높이 height의 사각형 안에 내접하는 타원을 startAngle을 시작 각도로 하여 arcAngle의 각도만큼의 호를 그린다.



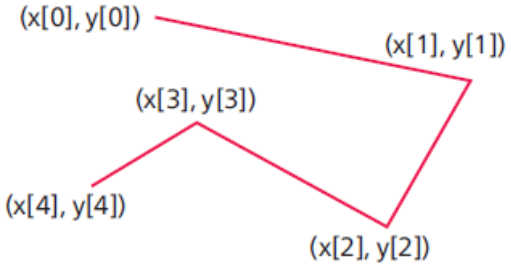
drawPolygon(int[] x, int y[], length)

배열 x[]와 배열 y[]을 가지고 여러 개의 직선을 그린다.



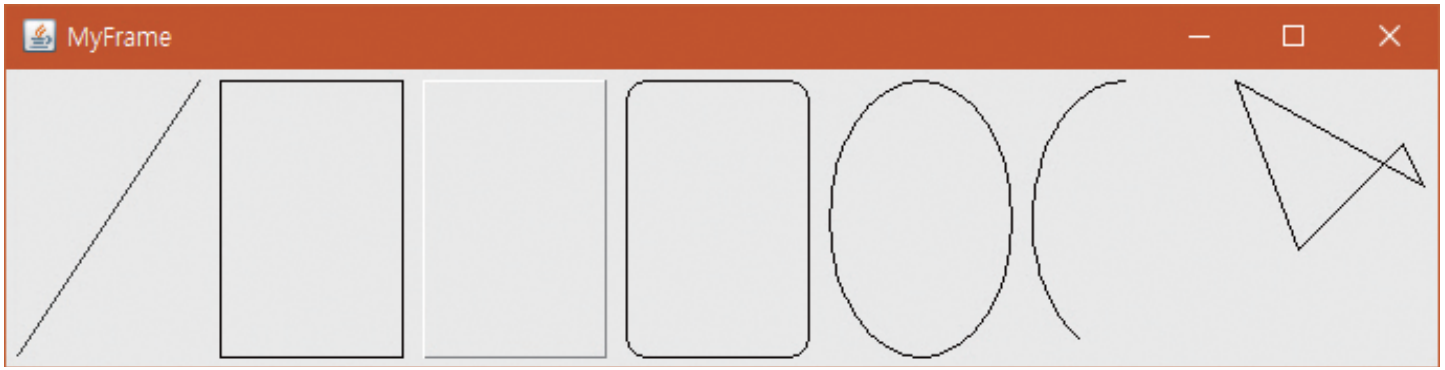
drawPolyline(int[] x, int y[], length)

배열 x[]와 배열 y[]을 가지고 여러 개의 직선을 그린다.



예제: 그래픽 기본 예제

- 패널을 하나 생성하고 여기에 여러 가지 도형을 그려보자

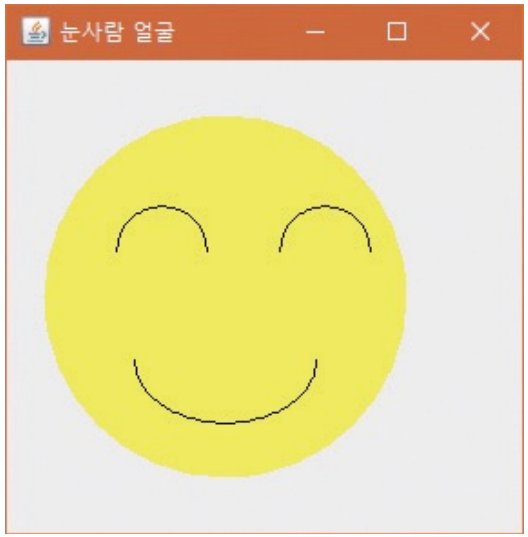


예제: 채워진 도형 그리기

- 앞의 예제에서 drawXXX()를 fillXXX()로 변경하면 색상으로 채워진 도형이 그려진다.

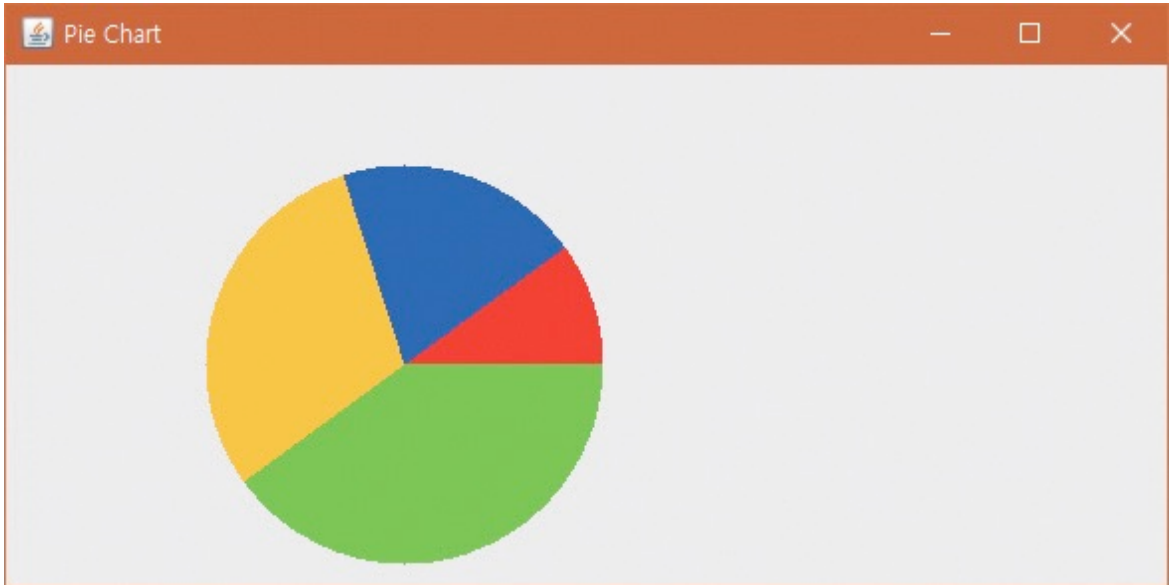
예제: 눈사람 얼굴 그리기

- 여기서는 지금까지 학습한 내용을 바탕으로 눈사람 얼굴을 직선, 타원, 사각형 등을 이용하여 그려보자.



예제: 파이 차트 그리기

- 앞에서 학습한 원호 그리기 메소드와 색상을 이용하여 파이 차트를 그려보자.



이미지 출력 및 처리

- 자바 그래픽을 사용하면 직접 화면에 이미지를 그릴 수 있다.

```
ImageIcon icon = new ImageIcon("d://car.png");
img = icon.getImage();
```

```
public void paintComponent(Graphics g) {
   super.paintComponent(g);
   g.drawImage(img, x, y, null);
}
```

예제:

- 하드디스크에 저장된 자동차 이미지를 화면에 그려보자.

예제:

```
public class DrawImageFrame extends JFrame {
        Image img;

        public DrawImageFrame() {
                ImageIcon icon = new ImageIcon("car.jpg");
                img = icon.getImage();              // 이미지 아이콘 객체에서 이미지를
추출한다.
                setSize(500, 200);
                add(new MyPanel());
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }

        class MyPanel extends JPanel {
                public void paintComponent(Graphics g) {
                        super.paintComponent(g);
                        g.drawImage(img, 0, 0,  null);      // 이미지를 화면의 원점에
그린다.
                }
        }

        public static void main(String[] args) {
                DrawImageFrame f = new DrawImageFrame();
        }
}
```

예제:

- 이번에는 자동차를 나타내는 클래스 Car 안에서 자동차를 그려보자. 어떤 점이 이전 예제와 달라지는지를 관찰해보자.

예제:

```
class Car {
        public Car() {
                super();
                ImageIcon icon = new ImageIcon("car.png");
                image = icon.getImage();          // 이미지 아이콘 객체에서 이미지를
추출한다.
        }
        public void draw(Graphics g) {
                g.drawImage(image, x, y, null);   // 이미지를 화면의 원점에 그린다.
        }
        int x=0, y=0;
        Image image;
}
```

예제:

```
public class DrawImageFrame extends JFrame {
        Car car;
        public DrawImageFrame() {
                car = new Car();
                setSize(500, 200);
                add(new MyPanel());
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }

        class MyPanel extends JPanel {
                public void paintComponent(Graphics g) {
                        super.paintComponent(g);
                        car.draw(g);
                }
        }

        public static void main(String[] args) {
                DrawImageFrame f = new DrawImageFrame();
        }
}
```

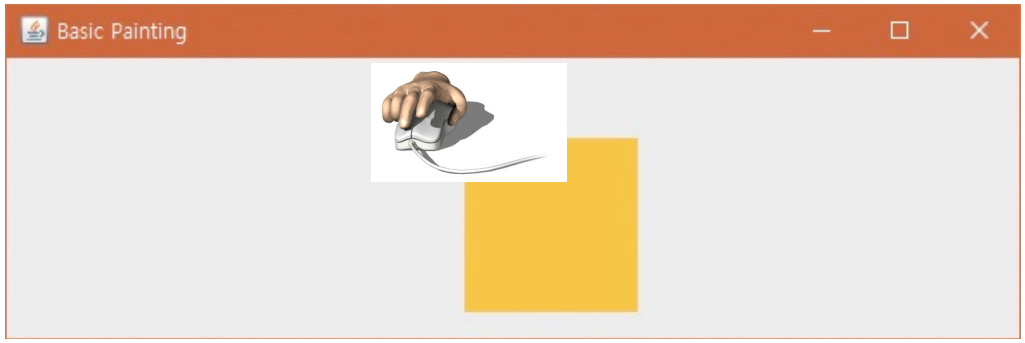
중간점검

중간점검

1. 이미지를 화면에 그리는 메소드 이름은?
2. 하드 디스크 d:에 있는 "a.jpg" 이미지를 읽어서 객체로 생성하는 문장을 작성해보자.

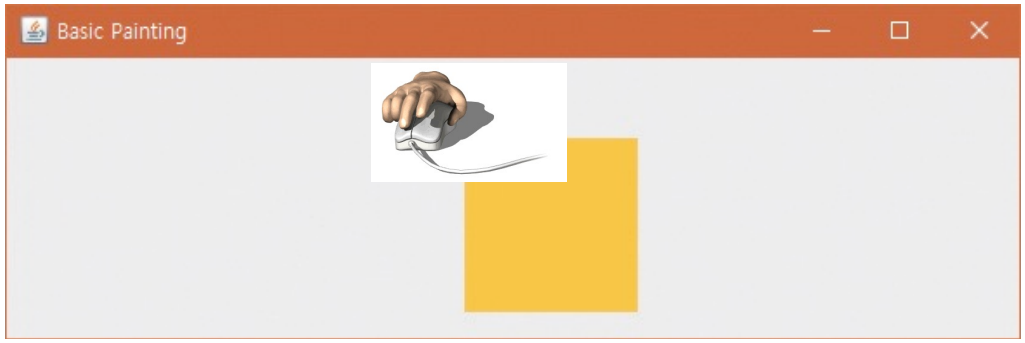
그래픽과 이벤트의 결합

- 응용 프로그램에서는 그래픽만 단독으로 사용되는 경우도 있지만, 일반적으로 그래픽은 이벤트처리와 결합되어서 사용되는 경우가 많다.



에제: 이벤트와 그래픽의 결합

- 사용자가 화면을 클릭하면 그 위치에 사각형을 그리도록 프로그램을 작성하여 보자. 마우스의 버튼이 눌려지면 위치를 저장했다가 화면에 사각형을 그리면 된다. 사각형을 직접 그리는 것이 아니고 repaint()만 호출해주고 사각형은 paintComponent()에서 그려야 한다.



에제: 이벤트와 그래픽의 결합

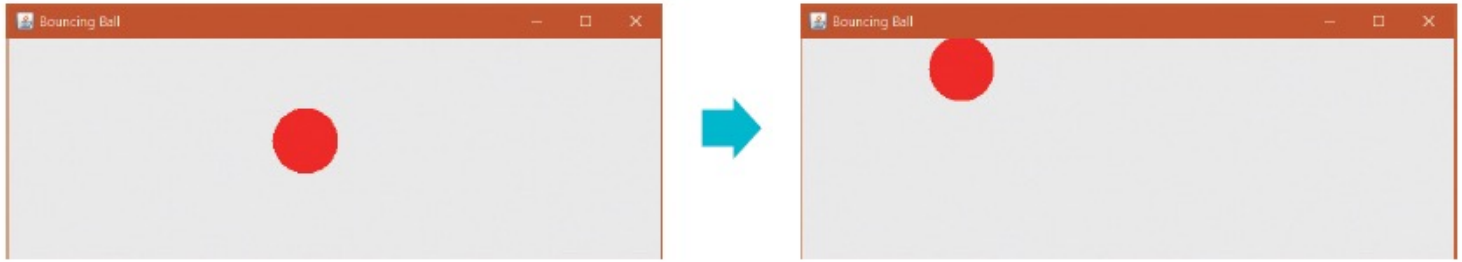
```
public class MyFrame extends JFrame {
    int x, y;
    class MyPanel extends JPanel {
        public MyPanel()            {


            addMouseListener(new MouseAdapter() {
                public void mousePressed(MouseEvent e) {
                    x = e.getX();
                    y = e.getY();
                    repaint();
                }
            });
        }
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.ORANGE);
            g.fillRect(x, y, 100, 100);
        }
    }
```

에제: 이벤트와 그래픽의 결합

```
    public MyFrame() {
        setTitle("Basic Painting");
        setSize(600, 200);
        add(new MyPanel());
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

    public static void main(String[] args) {
        MyFrame f = new MyFrame();
    }
}
```

예제: 바운싱 볼 애니메이션

- 그래픽과 밀접한 관계가 있는 것이 애니메이션이다. 여기서는 튀어 오르는 공을 애니메이션으로 만들어보자. 공의 현재 위치는 변수 x,y로, 공의 속도는 xSpeed, ySpeed로 표현된다. 공은 바닥에서는 반사되어야 한다.



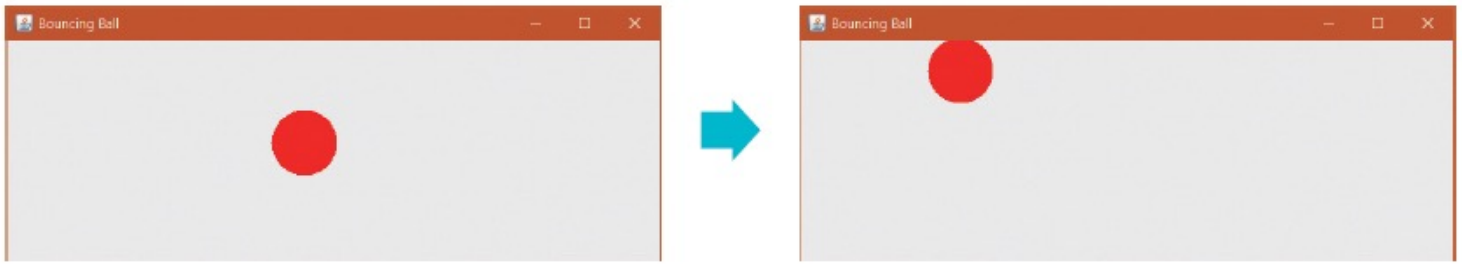
예제: 바운싱 볼 애니메이션

```
public class BouncingBall extends JFrame implements ActionListener {
    static final int WIDTH = 600;
    static final int HEIGHT = 200;
    private static final int PERIOD = 10;

    class MyPanel extends JPanel {
        int x=0, y=0, xInc=3, yInc=3, diameter=60;

        public void paintComponent(Graphics g) {
            super.paintComponent(g);
                if (x < 0 || x > (BouncingBall.WIDTH - diameter))
                    xInc = -xInc;
                if (y < 0 || y > (BouncingBall.HEIGHT - diameter))
                    yInc = -yInc;

                x += xInc;
                y += yInc;
                g.setColor(Color.RED);
                g.fillOval(x, y, diameter, diameter);
        }
    }
```

예제: 바운싱 볼 애니메이션

```
        public BouncingBall() {
                MyPanel panel = new MyPanel();

                panel.setPreferredSize(new Dimension(WIDTH, HEIGHT));
                add(panel);
                pack();
                setTitle("Bouncing Ball");
                Timer timer = new Timer(PERIOD, this);
                timer.start();

                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
        @Override
        public void actionPerformed(ActionEvent evt) {
                repaint();
        }
        public static void main(String[] args) {
                BouncingBall f = new BouncingBall();
        }

}
```

예제: 바운싱 볼 애니메이션



예제: 버튼이 눌리면 이미지를 이동

- 화면의 버튼이 눌리면 자동차 이미지가 왼쪽이나 오른쪽으로 이동하는 프로그램을 작성해보자.

예제: 버튼이 눌리면 이미지를 이동

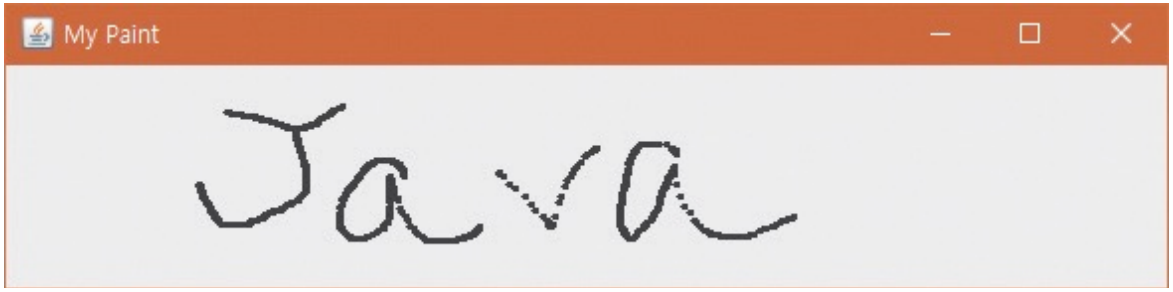
```
public class DrawImageFrame2 extends JFrame {

        Image img;
        int pos_x = 100, pos_y = 0;

        public DrawImageFrame2() {
                ImageIcon icon = new ImageIcon("d://car.jpg");
                img = icon.getImage();

                setSize(500, 200);

                add(new MyPanel(), BorderLayout.CENTER);
                JPanel panel = new JPanel();
                Button b1 = new Button("왼쪽으로 이동");
                Button b2 = new Button("오른쪽으로 이동");
                b1.addActionListener(e -> {
                        pos_x -= 10;
                        repaint();
                });
                b2.addActionListener(e -> {
                        pos_x += 10;
                        repaint();
                });
                panel.add(b1);
                panel.add(b2);
```

예제: 버튼이 눌리면 이미지를 이동

```
                add(panel, BorderLayout.SOUTH);
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }

        class MyPanel extends JPanel {
                public void paintComponent(Graphics g) {
                        super.paintComponent(g);
                        g.drawImage(img, pos_x, pos_y, this);
                }

        }

        public static void main(String[] args) {
                DrawImageFrame2 f = new DrawImageFrame2();
        }
}
```

예제: 그림판 프로그램 작성

- 마우스로 그림을 그릴 수 있는 프로그램을 작성해보자.



예제: 그림판 프로그램 작성

```
class Point {
    int x, y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
public class MyPaintApp extends JFrame {
    int x, y;
    Vector<Point> list = new Vector<>();

    class MyPanel extends JPanel {
        public MyPanel() {
            addMouseMotionListener(new MouseMotionAdapter() {
                public void mouseDragged(MouseEvent event) {
                    x = event.getX();
                    y = event.getY();
                    list.add(new Point(x, y));
                    repaint();
                }
            });
        }
```

예제: 그림판 프로그램 작성

```
                public void paintComponent(Graphics g) {
                        super.paintComponent(g);
                        for (Point p : list)
                                g.fillOval(p.x, p.y, 4, 4);
                }
        }

        public MyPaintApp() {
                setSize(600, 150);
                setTitle("My Paint");
                add(new MyPanel());
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }

        public static void main(String[] args) {
                MyPaintApp f = new MyPaintApp();
        }
}
```

Lab: 반사되는 공 애니메이션

- 앞의 예제에서 하나의 공이 움직이는 프로그램을 작성하였다. 이번 실습에서는 색상이나 크기가 다른 여러 개의 공을 생성하여 동시에 화면에서 움직이게 하자.

Sol: 반사되는 공 애니메이션

```
class Ball {
        int x, y, xInc, yInc, diameter;
        final Random r = new Random();
        Color color;

        public Ball(int d) { // (1)
                this.diameter = d;

                x = (int) (Math.random() * (BouncingBall.WIDTH - d) + 3);
                y = (int) (Math.random() * (BouncingBall.HEIGHT - d) + 3);
                xInc = (int) (Math.random() * 5 + 1);
                yInc = (int) (Math.random() * 5 + 1);
                color = new Color(r.nextInt(256), r.nextInt(256), r.nextInt(256));
        }
        public void paint(Graphics g) { // (2)
                if (x < 0 || x > (BouncingBall.WIDTH - diameter))
                        xInc = -xInc;
                if (y < 0 || y > (BouncingBall.HEIGHT - diameter))
                        yInc = -yInc;
                x += xInc;
                y += yInc;
                g.setColor(color);
                g.fillOval(x, y, diameter, diameter);
        }
}
```

Sol: 반사되는 공 애니메이션

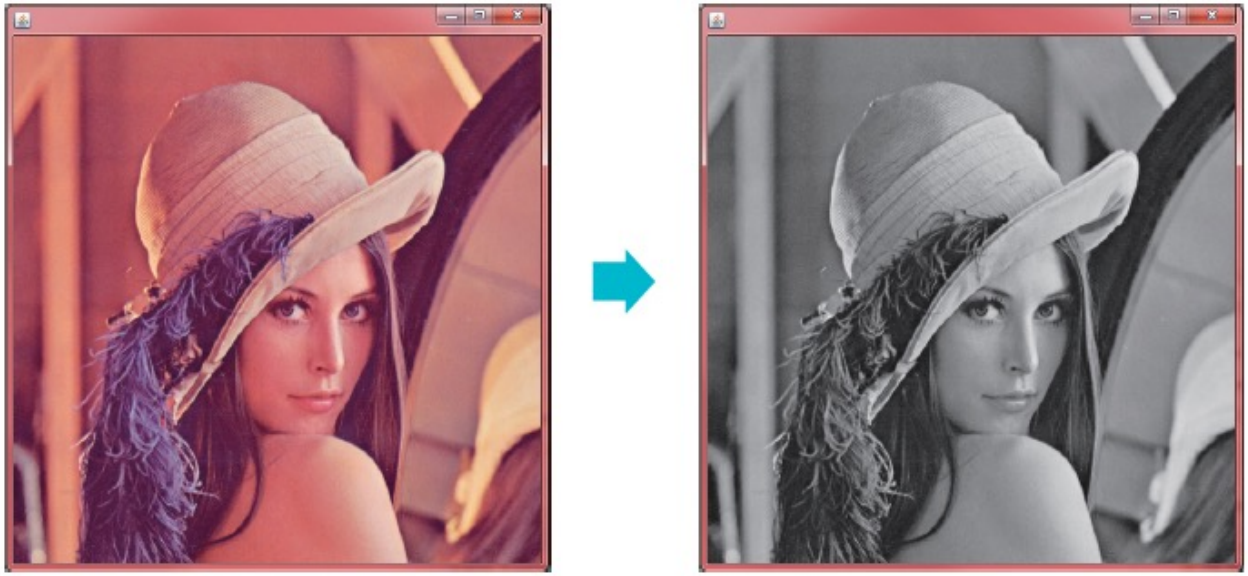
```
public class BouncingBall extends JFrame implements ActionListener {
    static final int WIDTH = 600;
    static final int HEIGHT = 200;
    private static final int PERIOD = 10;

    class MyPanel extends JPanel {
        public Ball basket[] = new Ball[10]; // (3)

        public MyPanel() {
            for (int i = 0; i < 10; i++)
                basket[i] = new Ball((int) (30 + 30 * Math.random()));

        }
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            for (Ball b : basket) {
                b.paint(g);
            }
        }
    }
```

Sol: 반사되는 공 애니메이션

```
        public BouncingBall() {
                MyPanel panel = new MyPanel();

                panel.setPreferredSize(new Dimension(WIDTH, HEIGHT));
                add(panel);
                pack();

                setTitle("Bouncing Ball");
                Timer timer = new Timer(PERIOD, this);
                timer.start();
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }

        @Override
        public void actionPerformed(ActionEvent evt) {
                repaint();
        }

        public static void main(String[] args) {
                BouncingBall f = new BouncingBall();
        }
}
```

Lab: 영상 처리

- 영상 처리(image processing)는 이미지를 읽어서 여러 가지 처리를 하는 학문 분야이다. 예를 들어서 화질이 나쁜 이미지의 화질을 향상시키는 것도 영상 처리의 일종이다. 간단한 영상 처리를 학습하여 보자. 컬러 이미지를 읽어서 흑백 이미지로 만드는 프로그램을 작성하여 보자



Sol: 영상 처리

Sol: 영상 처리

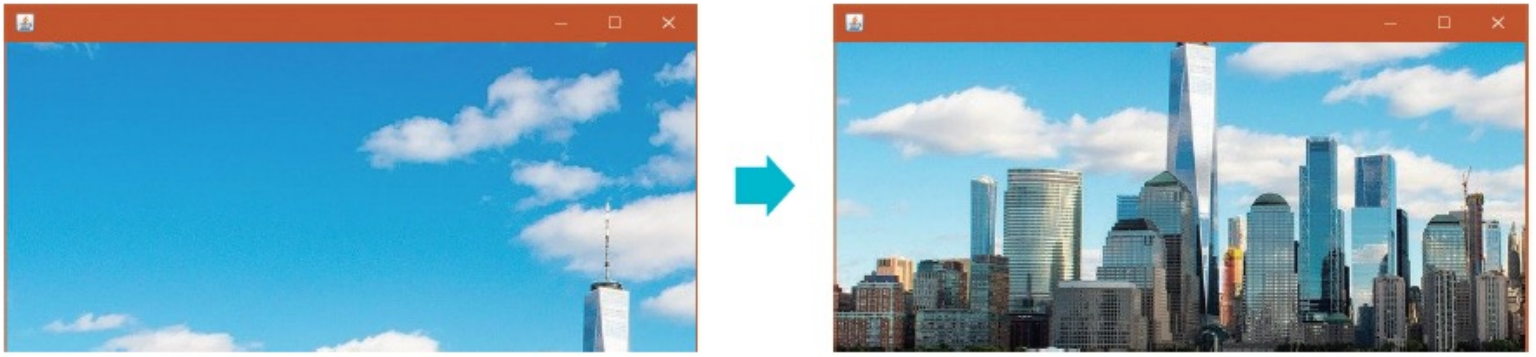
```
public class GrayScaleImage extends JFrame {
    BufferedImage image;
    int width;
    int height;
    public GrayScaleImage() {
        try {
            File input = new File("Lenna.png");
            image = ImageIO.read(input);
            width = image.getWidth();
            height = image.getHeight();

            for (int r = 0; r < height; r++) {
                for (int c = 0; c < width; c++) {
                    Color color = new Color(image.getRGB(r, c));
                    int red = (int) (color.getRed());
                    int green = (int) (color.getGreen());
                    int blue = (int) (color.getBlue());
                    int avg = (red + green + blue) / 3;
                    Color newColor = new Color(avg, avg, avg);
                    image.setRGB(r, c, newColor.getRGB());
                }
            }
```

Sol: 영상 처리

```
                File ouptut = new File("output.png");
                ImageIO.write(image, "png", ouptut);
                add(new MyPanel());
                pack();
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            } catch (Exception e) {
                System.out.println("이미지 읽기 실패!");
            }
        }
        class MyPanel extends JPanel {
            public void paintComponent(Graphics g) {
                g.drawImage(image, 0, 0, null);
            }
            public Dimension getPreferredSize() {
                if (image == null) return new Dimension(100, 100);
                else return new Dimension(width, height);
            }
        }
        static public void main(String args[]) throws Exception {
            GrayScaleImage obj = new GrayScaleImage();
        }
}
```

Lab: 움직이는 사진

- 유튜브를 보다 보면 큰 사진을 화면에서 천천히 움직여서 움직이는 영상처럼 느끼게 하는 기법을 많이 본다. 이 효과를 자바로 구현하여보자. 화면보다 큰 이미지를 읽어서 이미지가 시작되는 위치를 변경하여 이미지가 움직이는 효과를 주자. g.drawImage(image, x, y)에서 x와 y의 위치를 변경한다.



Lab: 움직이는 사진

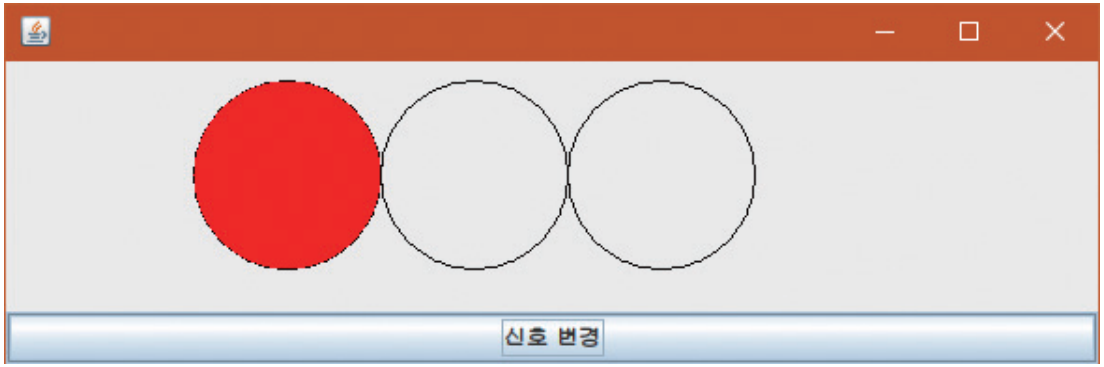
```
ublic class MyFrame extends JFrame implements ActionListener {
        static final int MAX_FRAME = 500;
        int frameNumber;
        Image background;
        private Thread aThread;

        class MyPanel extends JPanel {
                public void paintComponent(Graphics g) {
                        super.paintComponents(g);
                        g.drawImage(background, -frameNumber, -frameNumber, this);
                }
        }
        public MyFrame() {
                ImageIcon icon2 = new ImageIcon("back.jpg");
                background = icon2.getImage();
                setSize(600, 300);
                Timer timer = new Timer(10, this);
                timer.start();
                add(new MyPanel());
                setVisible(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
```

Lab: 움직이는 사진

```
        @Override
        public void actionPerformed(ActionEvent evt) {
                frameNumber = (++frameNumber) % MAX_FRAME;
                repaint();
        }

        public static void main(String[] args) {
                MyFrame f = new MyFrame();
        }
}
```

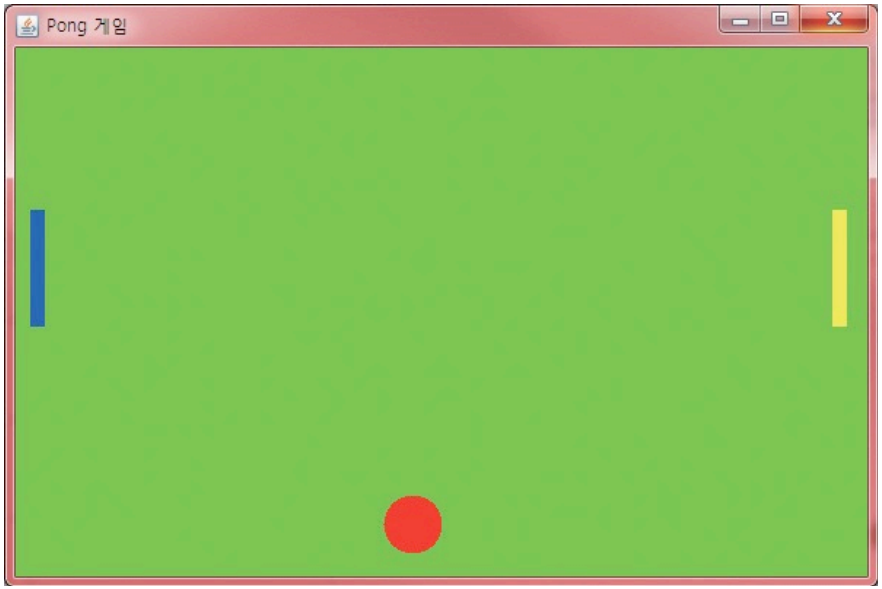
Mini Project: 신호등 프로그램

- 신호등을 나타내는 프로그램을 작성해보자. 버튼을 하나 만들어서 신호등의 하단에 추가한다. 버튼을 누르면 신호등이 차례대로 바뀌어야 한다.



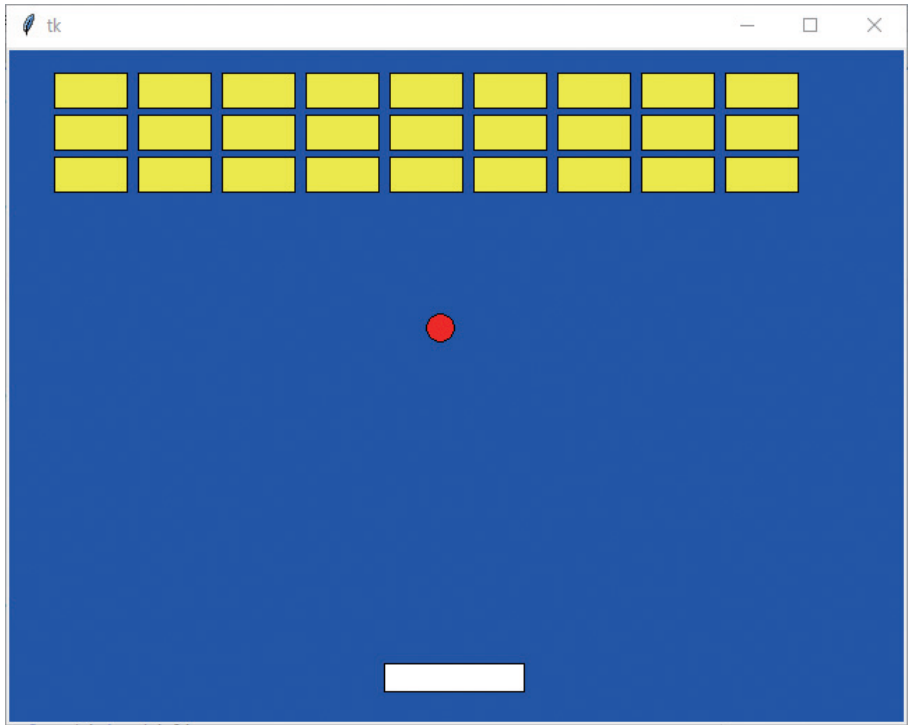
Mini Project: 탁구 게임

- 다음과 같이 2사람이 탁구 게임을 할 수 있는 프로그램을 작성하여 보자.



Mini Project: 벽돌깨기 게임

- 다음과 같이 공을 이용해 벽돌을 깨는 고전 게임을 작성해보자. 이번 장에서는 상속을 이용하여 프로그램을 작성해본다.



Mini Project: 갤러그 게임

- "갤러그"와 유사한 게임을 제작해보자. 우리는 자바 그래픽을 이용하여서 다음과 같은 "갤러그" 유사 게임을 제작할 것이다.

Summary

- paintComponent()는 모든 컴포넌트의 모양을 그리는 메소드이다. 우리는 이것을 오버라이드하여서 컴포넌트 위에 우리가 원하는 그림을 그릴 수 있다.
- 컴포넌트가 다시 그리기를 요청하려면 repaint()를 호출하면 된다. repaint()는 결국 컴포넌트의 paintComponent()를 호출하게 된다. 우리가 paintComponent()를 직접 호출하면 안 된다.
- JPanel은 그래픽 응용 프로그램에서 캔버스의 역할을 할 수 있다.
- Graphics 객체에는 색상과 폰트를 변경할 수 있는 메소드들을 제공한다.
- 색상은 Color 객체로 나타낸다.
- 폰트는 Font 객체로 나타낸다.
- 화면에 문자열을 그리려면 drawString()을 사용한다.
- 도형을 그리기 위하여 Graphics 객체는 drawRect(), drawLine(), drawOval() 등의 메소드를 제공한다.
- 내부가 칠해진 도형을 그리기 위하여 Graphics 객체는 fillRect(), fillOval(), fillArc() 등의 메소드를 제공한다.
- 화면에 이미지를 그리려면 drawImage()를 사용한다.

Q & A