



11

학습 목표

1. 스윙 컴포넌트 그리기와 `paintComponent()` 활용
2. `Graphics` 객체에 대한 이해
3. 도형 그리기와 칠하기
4. 이미지 그리기
5. `repaint()` 활용하기
6. 마우스와 그래픽 응용

스윙 컴포넌트 그리기, paintComponent()

3

- 스윙의 페인팅 기본
 - ▣ 모든 컴포넌트는 자신의 모양을 스스로 그린다.
 - ▣ 컨테이너는 자신을 그린 후 그 위에 자식 컴포넌트들에게 그리기 지시
 - ▣ 모든 스윙 컴포넌트는 자신의 모양을 그리는 paintComponent() 메소드 보유

- public void paintComponent(Graphics g)
 - ▣ 스윙 컴포넌트가 자신의 모양을 그리는 메소드
 - ▣ JComponent의 메소드 : 모든 스윙 컴포넌트가 이 메소드를 오버라이딩함
 - ▣ 언제 호출되는가?
 - 컴포넌트가 그려져야 하는 시점마다 호출
 - 크기가 변경되거나, 위치가 변경되거나, 컴포넌트가 가려졌던 것이 사라지는 등
 - 개발자가 직접 호출하면 안 됨
 - ▣ 매개변수인 Graphics 객체
 - 그래픽 컨텍스트 : 컴포넌트 그리기에 필요한 도구를 제공하는 객체
 - 자바 플랫폼에 의해 공급
 - 색 지정, 도형 그리기, 클리핑, 이미지 그리기 등의 메소드 제공

paintComponent()의 오버라이딩과 JPanel

4

- paintComponent(Graphics g)의 오버라이딩
 - ▣ 개발자가 JComponent를 상속받아 새로운 컴포넌트 설계
 - ▣ 기존 컴포넌트의 모양에 변화를 주고자 할 때

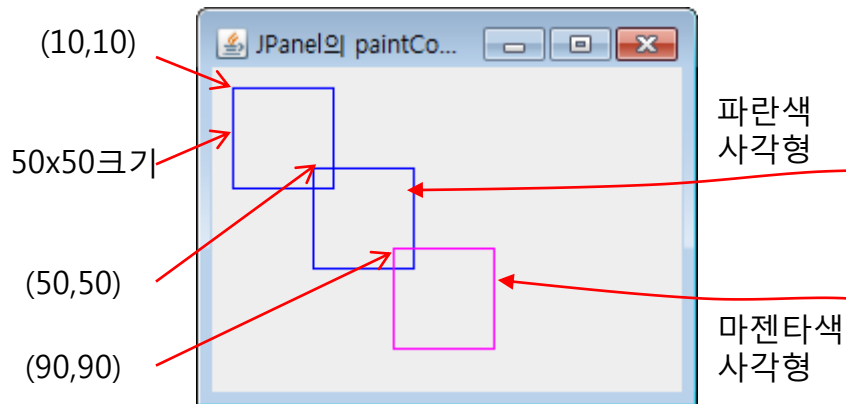
```
class MComponent extends JXXX {  
    public void paintComponent(Graphics g) {  
        super.paintComponent(g);  
        ... 필요한 그리기 코드 작성  
    }  
}
```

- JPanel
 - ▣ 비어 있는 컨테이너
 - ▣ 개발자가 다양한 GUI를 창출할 수 있는 캔버스로 적합
 - ▣ JPanel을 상속받아 개발자 임의의 모양을 가지는 패널로 많이 사용

예제 11-1 : JPanel을 상속받은 패널에 도형 그리기

5

JPanel을 상속받아 패널을 구성하고
이곳에 그림과 같은
3개의 도형을 그려라.



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

```
public class paintJPanelEx extends JFrame {
    paintJPanelEx() {
        setTitle("JPanel의 paintComponent() 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new MyPanel());
        setSize(250,200);
        setVisible(true);
    }
}
```

// JPanel을 상속받는 새 패널 구현

```
class MyPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.BLUE); // 파란색 선택
        g.drawRect(10,10,50,50);
        g.drawRect(50,50,50,50);

        g.setColor(Color.MAGENTA); // 마젠타색 선택
        g.drawRect(90,90,50,50);
    }
}

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

패널 내에 이전에
그려진 잔상을
지우기 위해 호출

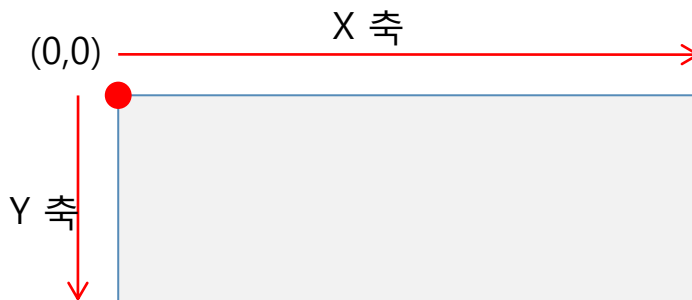
그래픽 기반 GUI 프로그래밍

6

□ 그래픽 기반 GUI 프로그래밍

- 스윙 컴포넌트에 의존하지 않고 선, 원 이미지 등을 이용하여 직접 화면을 구성하는 방법
- 그래픽 기반 GUI 프로그래밍의 학습이 필요한 이유
 - 컴포넌트의 한계를 극복하고 차트, 게임 등 자유로운 콘텐츠 표현
 - 그래픽은 컴포넌트에 비해 화면 출력 속도가 빠름
 - 스윙 컴포넌트들로 모두 그래픽으로 작성되어 있어, 그래픽에 대한 학습은 자바 GUI의 바탕 기술을 이해하는데 도움
 - 그래픽을 이용하여 개발자 자신만의 컴포넌트 개발

□ 자바의 그래픽(Graphics) 좌표 시스템



Graphics와 문자열 출력

7

- Graphics의 기능
 - ▣ 색상 선택하기
 - ▣ 문자열 그리기
 - ▣ 도형 그리기
 - ▣ 도형 칠하기
 - ▣ 이미지 그리기
 - ▣ 클리핑
- 문자열 출력을 위한 Graphics 메소드

```
void drawString(String str, int x, int y)
```

str 문자열을 (x,y) 영역에 그림. 현재 Graphics에 설정된 색과 폰트로 문자열 출력

```
Graphics g;  
g.drawString("자바는 재밌다.~~", 30,30); // (30, 30) 위치에 문자열 출력
```

그래픽의 색과 폰트

8

□ 색 : Color 클래스

- 자바의 색 : r(Red), g(Green), b(Blue) 성분으로 구성, 각 성분은 0~255(8비트) 범위의 정수

`Color(int r, int g, int b)` r, g, b 값으로 sRGB 색 생성

`Color(int rgb)` rgb는 32비트의 정수이지만 하위 24비트만 유효. 즉, `0x00rrggbb`로 표현. 각 바이트가 r, g, b의 색 성분

- 예) 빨간색 : `new Color(255, 0, 0)`, 초록색 : `new Color(0x0000ff00)`; 노란색 : `Color.YELLOW`

□ 폰트 : Font 클래스

`Font(String fontFace, int style, int size)`

- fontFace: "고딕체", "Arial" 등과 같은 폰트 이름
- style: `Font.BOLD`, `Font.ITALIC`, `Font.PLAIN` 중 한 값으로 문자의 스타일
- size: 픽셀 단위의 문자 크기

□ Graphics에 색과 폰트 설정

`void setColor(Color color)` 그래픽 색을 color로 설정. 그리기 시에 색으로 이용

`void setFont(Font font)` 그래픽 폰트를 font로 설정. 문자열 출력 시 폰트로 이용

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

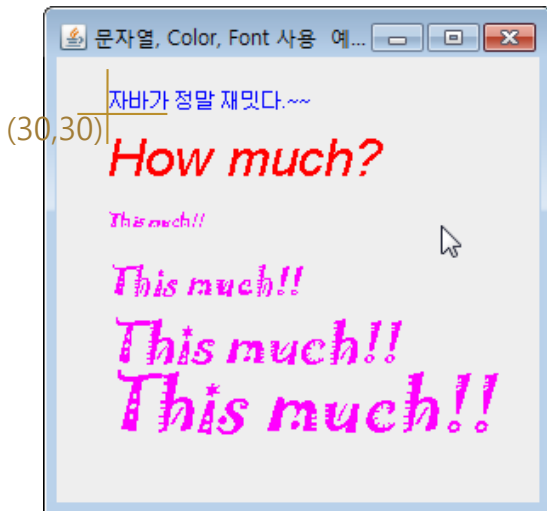
"Arial"체와 빨간색으로
"How much"를
(30, 30) 위치에 출력하는 사례

예제 11-2 : Color와 Font를 이용하여 문자열 그리기

9

Color와 Font를 이용하여 그림과 같이 문자열을 출력하라.

"How much?"는 "Arial" 체로,
"This much!!"는 Jokerman 체로 한다.
Jokerman 체는 아쉽게도
한글을 지원하지 않는다.



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

```
public class GraphicsColorFontEx extends JFrame {
    GraphicsColorFontEx() {
        setTitle("문자열, Color, Font 사용 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new MyPanel());
        setSize(300, 300);
        setVisible(true);
    }

    class MyPanel extends JPanel {
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.BLUE); // 파란색 지정
            g.drawString("자바가 정말 재밌다.~~", 30,30);
            g.setColor(new Color(255, 0, 0)); // 빨간색 지정
            g.setFont(new Font("Arial", Font.ITALIC, 30));
            g.drawString("How much?", 30, 70);
            g.setColor(new Color(0x00ff00ff));
            for(int i=1; i<=4; i++) {
                g.setFont(new Font("Jokerman", Font.ITALIC, i*10));
                g.drawString("This much!!", 30, 60+i*40);
            }
        }
    }

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

도형 그리기와 칠하기

10

□ 도형 그리기

- ▣ 선, 타원, 사각형, 둥근 모서리 사각형, 원호, 폐 다각형 그리기
- ▣ 선의 굵기 조절할 수 없음

```
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 이용하여 원호로 그린다.

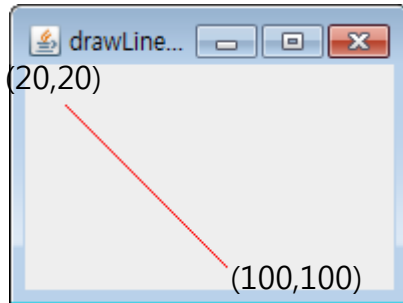
□ 도형 칠하기

- ▣ 도형을 그리고 내부를 칠하는 기능
 - 도형의 외곽선과 내부를 따로 칠하는 기능 없음
- ▣ 도형 칠하기를 위한 메소드
 - 그리기 메소드 명에서 draw 대신 fill로 이름 대체하면 됨. fillRect(), fillOval() 등

예제 11-3 : 선 그리기

11

Graphics의 drawLine()을 이용하여 콘텐츠 팬에 (20, 20)에서 (100, 100)까지 빨간선을 그리는 프로그램을 작성하라.



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

public class GraphicsDrawLineEx extends JFrame {
    GraphicsDrawLineEx() {
        setTitle("drawLine 사용 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new MyPanel());
        setSize(200, 150);
        setVisible(true);
    }

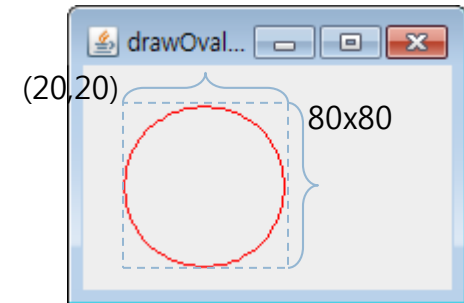
    class MyPanel extends JPanel {
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            g.setColor(Color.RED); // 빨간색을 선택한다.
            g.drawLine(20, 20, 100, 100);
        }
    }

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

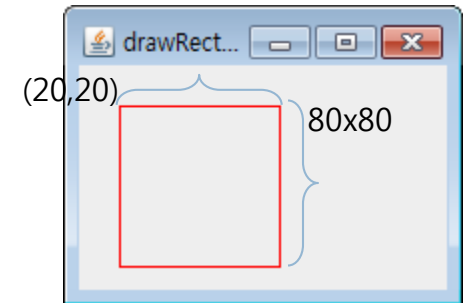
다른 도형 그리기 사례

12

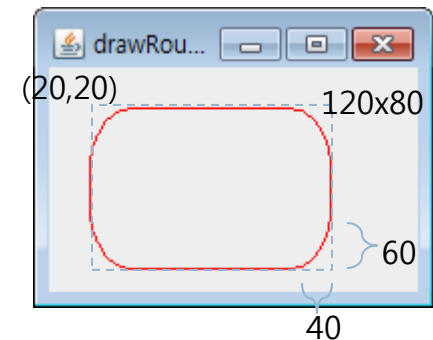
```
class MyPanel extends JPanel {  
    public void paintComponent(Graphics g) {  
        super.paintComponent(g);  
        g.setColor(Color.RED);  
        g.drawOval(20,20,80,80);  
    }  
}
```



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



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



Graphics의 원호와 폐다각형 그리기 메소드

13

```
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 0도의 기점이다. *startAngle* 지점에서 *arcAngle* 각도만큼 원호를 그린다. *arcAngle*이 양수이면 반시계 방향, 음수이면 시계 방향으로 그린다.

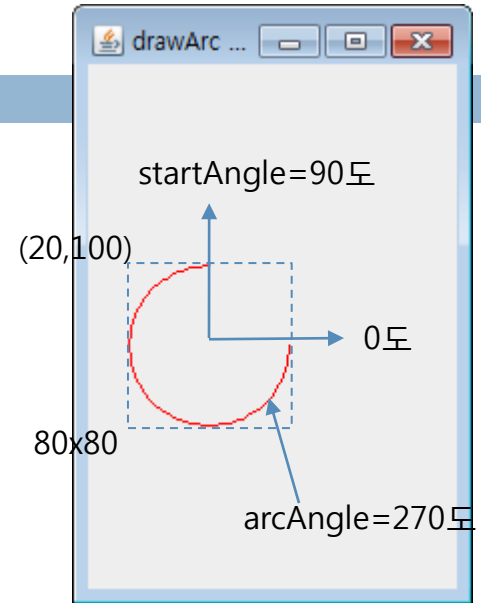
```
void drawPolygon(int []x, int []y, int n)
```

x, *y* 배열에 저장된 점들 중 *n*개를 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.

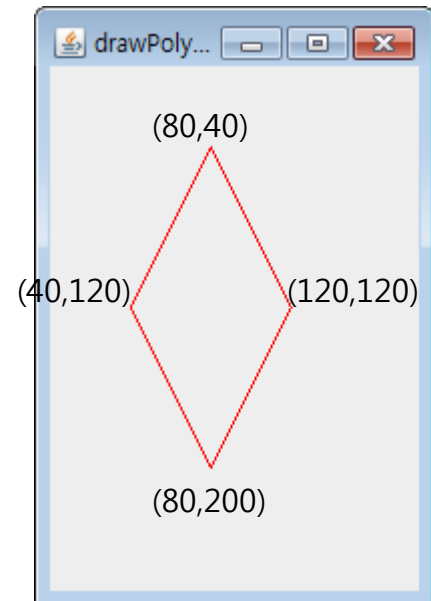
원호와 폐다각형 그리기 사례

14

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



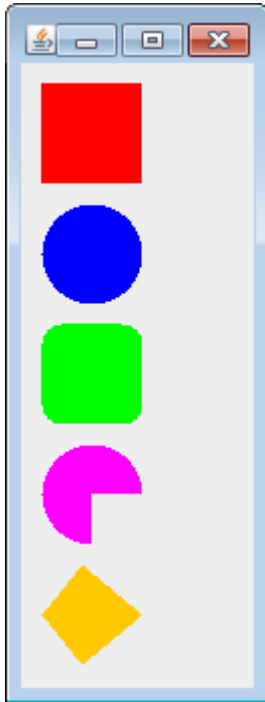
```
class MyPanel extends JPanel {  
    public void paintComponent(Graphics g) {  
        super.paintComponent(g);  
        g.setColor(Color.RED);  
  
        int []x = {80,40,80,120};  
        int []y = {40,120,200,120};  
        g.drawPolygon(x, y, 4);  
    }  
}
```



예제 11-4 : 도형 칠하기

15

Graphics의 칠하기 메소드를 이용하여
그림과 같은 패널을 작성하라.



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

public class GraphicsFillEx extends JFrame {
    GraphicsFillEx() {
        setTitle("fillXXX 사용 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new MyPanel());
        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

16

1. JLabel을 이용한 이미지 그리기

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

- 장점 : 이미지 그리기 간편 용이
- 단점 : 이미지의 원본 크기대로 그리므로 이미지 크기 조절 불가

2. Graphics의 drawImage()로 이미지 출력

- 장점 : 이미지 일부분 등 이미지의 원본 크기와 다르게 그리기 가능
- 단점 : 컴포넌트로 관리할 수 없음
이미지의 위치나 크기 등을 적절히 조절하는 코딩 필요

Graphics의 drawImage() 메소드

17

▣ 원본 크기로 그리기

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

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

- *img* : 이미지 객체
- *x*, *y* : 이미지가 그려질 좌표
- *bgColor* : 이미지가 투명한 부분을 가지고 있을 때 투명한 부분에 칠해지는 색상
- *observer* : 이미지 그리기의 완료를 통보받는 객체

*img*를 그래픽 영역의 (*x*, *y*) 위치에 *img*의 원본 크기로 그린다.

▣ 크기 조절하여 그리기

```
boolean drawImage(Image img, int x, int y, int width, int height, Color bgColor,  
                  ImageObserver observer)
```

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

- *width* : 그려지는 폭으로서 픽셀 단위
- *height* : 그려지는 높이로서 픽셀 단위

*img*를 그래픽 영역의 (*x*, *y*) 위치에 *width* x *height* 크기로 조절하여 그린다.

* *ImageObserver*는 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 *this*를 주거나 *null*을 주어 통보를 받지 않을 수 있음

이미지 그리기 샘플 코드

18

- 이미지 로딩 : Image 객체 생성

```
ImageIcon icon = new ImageIcon("image/image0.jpg");  
Image img = icon.getImage();
```

- (20,20) 위치에 원본 크기로 그리기
 - ▣ 고정 크기임

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

- (20, 20) 위치에 100x100 크기로 그리기
 - ▣ 고정 크기임

```
public void paintComponent(Graphics g) {  
    super.paintComponent(g);  
    g.drawImage(img, 20, 20, 100, 100, this);  
}
```

- 이미지를 패널에 꼭 차도록 그리기
 - ▣ JPanel의 크기로 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);  
}
```

예제 11-5 : 원본 크기로 이미지 그리기

19



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

```
public class GraphicsDrawImageEx1 extends JFrame {
    Container contentPane;
    GraphicsDrawImageEx1() {
        setTitle("원본 크기로 원하는 위치에 이미지 그리기");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new MyPanel());

        setSize(300, 400);
        setVisible(true);
    }

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

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

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

예제 11-6 : JPanel 크기에 맞추어 이미지 그리기

20



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

```
public class GraphicsDrawImageEx2 extends JFrame {
```

```
    GraphicsDrawImageEx2() {
        setTitle("패널의 크기에 맞추어 이미지 그리기");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new MyPanel());
    }
```

```
    setSize(200, 300);
    setVisible(true);
}
```

```
class MyPanel extends JPanel {
```

```
    ImageIcon icon = new ImageIcon("images/image0.jpg");
    Image img = icon.getImage();
```

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

패널의 폭과 높이

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

repaint()

21

□ repaint()

- 모든 컴포넌트가 가지고 있는 메소드
- 자바 플랫폼에게 컴포넌트 그리기를 강제 지시하는 메소드
- repaint()를 호출하면, 자바 플랫폼이 컴포넌트의 paintComponent() 호출

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

□ repaint()의 호출이 필요한 경우

- 개발자가 컴포넌트를 다시 그리고자 하는 경우
 - 프로그램에서 컴포넌트의 모양과 위치를 변경하고 바로 화면에 반영시키고자 하는 경우
 - 컴포넌트가 다시 그려져야 그 때 변경된 위치에 변경된 모양으로 출력됨
 - repaint()는 자바 플랫폼에게 지금 당장 컴포넌트를 다시 그리도록 지시함

□ 부모 컴포넌트부터 다시 그리는 것이 좋음

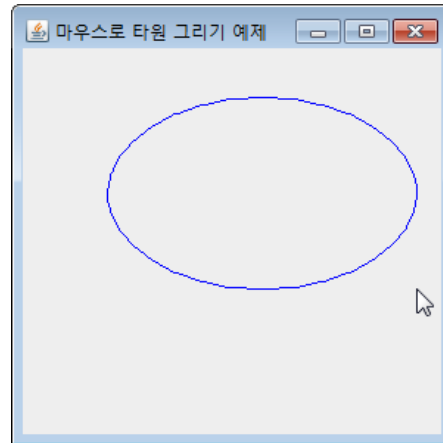
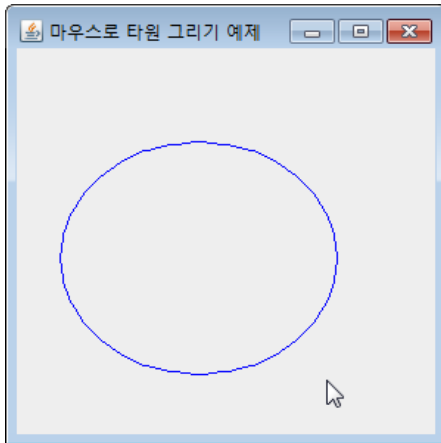
- 컴포넌트 repaint()가 불려지면
 - 이 컴포넌트는 새로운 위치에 다시 그려지지만 이전의 위치에 있던 자신의 모양이 남아 있음
- 부모 컴포넌트의 repaint()를 호출하면
 - 부모 컨테이너의 모든 내용을 지우고 자식을 다시 그리기 때문에 컴포넌트의 이전 모양이 지워지고 새로 변경된 크기나 위치에 그려짐

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

예제 11-7 : repaint()와 마우스를 이용한 타원 그리기

22

마우스를 드래킹하여 타원을 그리는 프로그램을 작성하라.
마우스로 한 점을 짚고 드래킹을 하면 타원이 그려진다. 드래킹하는 동안 타원 모양을 보기 위해서는 `mouseDragged()`에서 `repaint()`를 호출해야 한다.



예제 11-7 정답

23

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

```
public class GraphicsDrawOvalMouseEx extends JFrame {
    GraphicsDrawOvalMouseEx() {
        setTitle("마우스 드래깅으로 타원 그리기 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new MyPanel());
        setSize(300, 300);
        setVisible(true);
    }
}
```

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

// 타원을 그릴 패널 작성. 이 패널에 마우스 리스너 구현

```
class MyPanel extends JPanel {
    Point start=null, end=null; // 마우스의 시작점과 끝점
    public MyPanel() {
        MyMouseListener listener = new MyMouseListener();

        // listener를 아래 두 리스너로 공통으로 등록해야 한다.
        addMouseListener(listener);
        addMouseMotionListener(listener);
    }
}
```

repaint()가 호출되면, 자바 플랫폼에 의해 MyPanel의 paintComponent()가 호출된다. 여기서 start와 end 사이의 타원을 그린다.

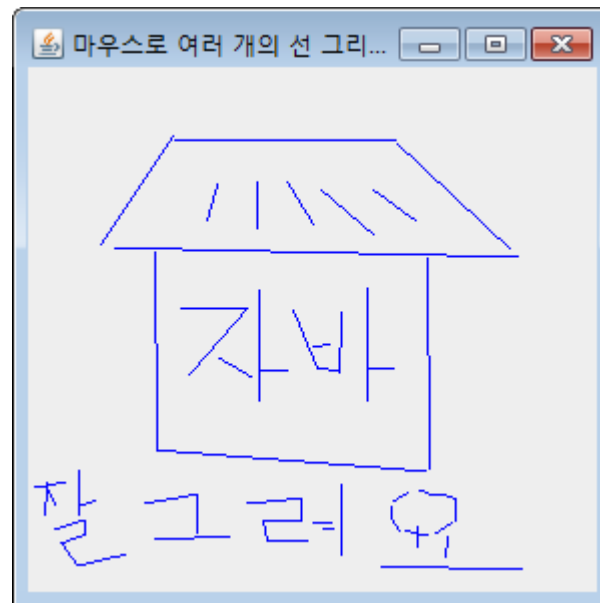
```
class MyMouseListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        start = e.getPoint();
    }
    public void mouseDragged(MouseEvent e) {
        end = e.getPoint();
        repaint(); // 패널의 그리기 요청 주목
    }
}
```

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    if(start == null) // 타원이 만들어지지 않았음
        return;
    g.setColor(Color.BLUE); // 파란색 선택
    int x = Math.min(start.x, end.x);
    int y = Math.min(start.y, end.y);
    int width = Math.abs(start.x - end.x);
    int height = Math.abs(start.y - end.y);
    g.drawOval(x, y, width, height); // 타원 그리기
}
}
```

예제 11-8 : repaint()와 마우스를 이용한 여러 개의 선 그리기

24

그림과 같이 마우스를 이용하여 여러 개의 선을 그리는 프로그램을 작성하라. 마우스를 누르고 드래킹하여 놓으면 선이 그려진다. 여러 개의 선을 그리기 위해 각 선의 위치를 기억하는 벡터를 사용한다. 그린 선이 보이게 하기 위해서는 `mouseReleased()`에서 `repaint()`를 호출한다.



예제 11-8 정답

25

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

public class GraphicsDrawLineMouseEx extends JFrame {
    GraphicsDrawLineMouseEx() {
        setTitle("마우스로 여러 개의 선 그리기 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new MyPanel());
        setSize(300, 300);
        setVisible(true);
    }

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

    class MyPanel extends JPanel {
        Vector<Point> vStart = new Vector<Point>();
        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);

            // 패널의 다시 그리기를 요청한다.
            repaint(); // 주목
        }
    });
}

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());
    }
}
}
```