



# 응용 애플리케이션 개발을 위한 자바 프로그래밍

## 그래픽 프로그래밍



한국기술교육대학교  
온라인평생교육원

## 학습내용

- 그래픽 프로그래밍 개요
- 그래픽 프로그래밍 응용

## 학습목표

- 그래픽 처리 구조를 이해하고, 그래픽 프로그래밍을 할 수 있다.
- 그래픽 관련 클래스와 메소드를 이용하여 그래픽 프로그래밍을 할 수 있다.

# 그래픽 프로그래밍 개요

## 1 그래픽 처리 구조

### ① 컴포넌트의 그래픽 처리 방법

- 1 Component 클래스의 메소드를 통해 그래픽을 처리
- 2 제공되는 그래픽

색상, 글꼴, 도형 그리기, 텍스트 그리기, 이미지 그리기 등

그래픽 처리 메소드	설 명
<code>void paint(Graphics g)</code>	화면에 그래픽을 표시하는 메소드
<code>void update(Graphics g)</code>	화면을 지우는 메소드

## 1 그래픽 처리 구조

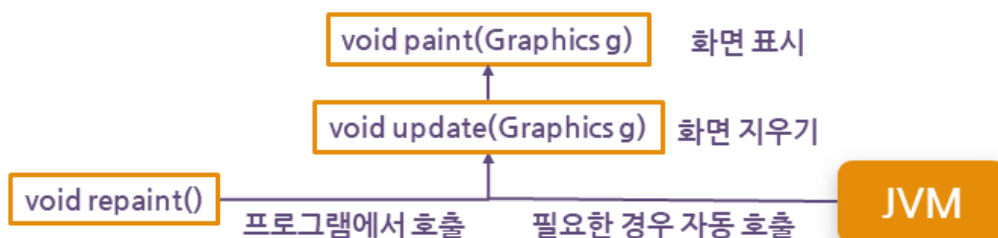
### ① 컴포넌트의 그래픽 처리 방법

- 3 그래픽이 화면에 표시되는 경우

프로그램에서 Component 클래스의 `repaint()` 메소드를 호출

JVM에서 화면에 그래픽을 표시할 필요가 있는 경우 자동 호출

호출순서: `update()` → `paint()` 메소드



# 그래픽 프로그래밍 개요

## 1 그래픽 처리 구조

### ② 그래픽 프로그램 작성 방법

#### 1 그래픽 처리를 위한 컴포넌트

컨테이너 : Frame, Panel 등

전용 컴포넌트 : Canvas

## 1 그래픽 처리 구조

### ② 그래픽 프로그램 작성 방법

#### 2 그래픽 프로그램 작성순서

가 컨테이너 또는 전용 컴포넌트를 상속

나 필요한 메소드를 오버라이딩

paint() : 반드시 오버라이딩이 필요함

update() : 필요한 경우 오버라이딩 함

(a) 오버라이딩 하면 : 화면 지우기 안됨

(b) 오버라이딩 안 하면 : 화면 지우기 동작

# 그래픽 프로그래밍 개요

## 1 그래픽 처리 구조

### ② 그래픽 프로그램 작성 방법

#### 2 그래픽 프로그램 작성순서

**다** paint() 메소드 내부에서 Graphics 클래스의 그래픽관련 메소드 호출

Graphics 객체 : paint(), update() 메소드의 매개 변수

**라** repaint() 메소드를 호출 : update() 메소드 → paint() 메소드 호출

## 1 그래픽 처리 구조

### ③ 그래픽 프로그램 예제

```
public class GraphicFrame extends Frame {
    public void update(Graphics g) {
        paint(g);
    }

    public void paint(Graphics g) {
        g.setColor(Color.RED);
        g.fillOval(x, y, 10, 10);
    }

    public void mouseClicked(MouseEvent e) {
        repaint();
    }
}
```

← Frame에 그래픽을 처리하기 위한 상속

# 그래픽 프로그래밍 개요

## 1 그래픽 처리 구조

### ③ 그래픽 프로그램 예제

```
public class GraphicFrame extends Frame {
    public void update(Graphics g) {
        paint(g);
    }
    public void paint(Graphics g) {
        g.setColor(Color.RED);
        g.fillOval(x, y, 10, 10);
    }
    public void mouseClicked(MouseEvent e) {
        repaint();
    }
}
```

메소드 오버라이딩

## 1 그래픽 처리 구조

### ③ 그래픽 프로그램 예제

```
public class GraphicFrame extends Frame {
    public void update(Graphics g) {
        paint(g);
    }
    public void paint(Graphics g) {
        g.setColor(Color.RED);
        g.fillOval(x, y, 10, 10);
    }
    public void mouseClicked(MouseEvent e) {
        repaint();
    }
}
```

paint(g) 메소드의 직접 호출은 update() 메소드 내부에서만 가능

# 그래픽 프로그래밍 개요

## 1 그래픽 처리 구조

### ③ 그래픽 프로그램 예제

```
public class GraphicFrame extends Frame {
    public void update(Graphics g) {
        paint(g);
    }
    public void paint(Graphics g) { ← 매개변수로 넘어오는 Graphics 객체
        g.setColor(Color.RED);
        g.fillOval(x, y, 10, 10);
    }
    public void mouseClicked(MouseEvent e) {
        repaint();
    }
}
```

## 1 그래픽 처리 구조

### ③ 그래픽 프로그램 예제

```
public class GraphicFrame extends Frame {
    public void update(Graphics g) {
        paint(g);
    }
    public void paint(Graphics g) {
        g.setColor(Color.RED);
        g.fillOval(x, y, 10, 10); ← Graphics 클래스의 그래픽 관련 메소드 사용
    }
    public void mouseClicked(MouseEvent e) {
        repaint();
    }
}
```

# 그래픽 프로그래밍 개요

## 1 그래픽 처리 구조

### ③ 그래픽 프로그램 예제

```
public class GraphicFrame extends Frame {
    public void update(Graphics g) {
        paint(g);
    }

    public void paint(Graphics g) {
        g.setColor(Color.RED);
        g.fillOval(x, y, 10, 10);
    }

    public void mouseClicked(MouseEvent e) {
        repaint(); ← repaint() 메소드 호출
    }
}
```

## 1 그래픽 처리 구조

### ③ 그래픽 프로그램 예제

```
public class GraphicFrame extends Frame {
    public void update(Graphics g) {
        paint(g);
    }

    public void paint(Graphics g) {
        g.setColor(Color.RED);
        g.fillOval(x, y, 10, 10);
    }

    public void mouseClicked(MouseEvent e) {
        repaint(); ← update() 메소드 호출
    }
}
```



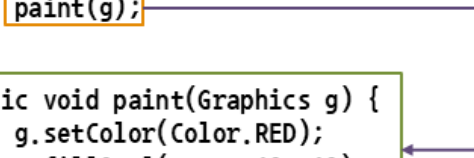
# 그래픽 프로그래밍 개요

## 1 그래픽 처리 구조

### ③ 그래픽 프로그램 예제

```
public class GraphicFrame extends Frame {  
    public void update(Graphics g) {  
        paint(g);  
    }  
    public void paint(Graphics g) {  
        g.setColor(Color.RED);  
        g.fillOval(x, y, 10, 10);  
    }  
    public void mouseClicked(MouseEvent e) {  
        repaint();  
    }  
}
```

paint() 메소드 호출

A diagram with a purple line starting from the `paint(g);` line in the `update` method, extending to the right, then turning down and left to point at the `paint` method definition.

# 그래픽 프로그래밍 개요



그래픽 프로그래밍

그래픽 프로그래밍의 개요



## 실습하기



그래픽 프로그래밍

그래픽 프로그래밍의 개요

### 실습순서

1. update() 메소드와 paint() 메소드 프로그래밍 실습



### 유의사항

- JDK와 이클립스를 설치한 후 실습이 가능함
- 본인이 원하는 작업 폴더를 미리 정해 놓은 다음 실습하기
- 작업 폴더는 C드라이브에 지정하기 보다는 D드라이브나 외장하드디스크를 활용하는 것을 추천함



※ 제공되는 실습 코드를 다운받아 실습해보시기 바랍니다.

## 그래픽 프로그래밍 개요

### 2 그래픽 관련 클래스

#### ① Canvas

- 1 java.awt.Component 클래스의 하위 클래스
- 2 그래픽을 처리할 수 있는 빈 사각형 영역
- 3 Canvas 클래스를 상속한 후 paint(), update() 메소드 오버라이딩

### 2 그래픽 관련 클래스

#### ① Canvas

##### 4 관련 메소드

이벤트 관련 메소드	설 명
Canvas()	생성자 메소드
void update(Graphics g)	Canvas 객체의 update() 메소드
void paint(Graphics g)	Canvas 객체의 paint() 메소드

# 그래픽 프로그래밍 개요

## 2 그래픽 관련 클래스

### ② Font

- 1 java.awt.Component 클래스의 하위 클래스
- 2 텍스트를 화면에 표시하기 위한 글꼴
- 3 Graphics 클래스의 setFont() 메소드로 지정
- 4 관련 메소드

생성자 메소드	설 명
Font(String name, int style, int size)	<ul style="list-style-type: none"> <li>• name : 글꼴의 이름을 지정</li> <li>• style : 글꼴의 스타일을 지정, 상수를 이용</li> <li>• size : 글꼴의 크기를 지정</li> </ul>

## 2 그래픽 관련 클래스

### ② Font

- 4 관련 메소드

스타일 상수	설 명
Font.PLAIN	기본 스타일
Font.BOLD	굵은 글씨체
Font.ITALIC	이탤릭 글씨체(기울인 글씨체)
Font.BOLD   Font.ITALIC	굵은 글씨체 + 이탤릭 글씨체

## 그래픽 프로그래밍 개요

### 2 그래픽 관련 클래스

#### ③ Graphics

##### 3 관련 메소드

그래픽 관련 메소드	설 명
<code>void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)</code>	내부가 색으로 채워져 있는 원호를 그림
<code>void drawLine(int x1, int y1, int x2, int y2)</code>	x1, y1 좌표에서 x2, y2 좌표까지 선을 그림

### 2 그래픽 관련 클래스

#### ③ Graphics

그래픽 관련 메소드	설 명
<code>void drawOval(int x, int y, int width, int height)</code>	내부가 색이 없는 타원을 그림 • x, y : x, y 좌표 • width, height : 폭과 높이의 크기
<code>void fillOval(int x, int y, int width, int height)</code>	내부가 색으로 채워져 있는 타원을 그림
<code>void drawPolygon(int[] x, int[] y, int nPoints)</code>	내부가 색이 없는 다각형 그림 • x, y : x, y 좌표 배열 • nPoints : 다각형 포인트의 개수를 지정
<code>void fillPolygon(int[] x, int[] y, int nPoints)</code>	내부가 색으로 채워져 있는 다각형을 그림

## 그래픽 프로그래밍 개요

### 2 그래픽 관련 클래스

#### ③ Graphics

그래픽 관련 메소드	설 명
<code>void drawRect(int x, int y, int width, int height)</code>	내부가 색이 없는 사각형을 그림 <ul style="list-style-type: none"> <li>• x, y : x, y 좌표</li> <li>• width, height : 폭과 높이의 크기</li> </ul>
<code>void fillRect(int x, int y, int width, int height)</code>	내부가 색으로 채워져 있는 사각형을 그림
<code>void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)</code>	내부가 색이 없고, 모서리가 둥근 사각형 그림 <ul style="list-style-type: none"> <li>• x, y : x, y 좌표</li> <li>• width, height : 폭과 높이의 크기</li> <li>• arcWidth, arcHeight : 둥근 모서리의 크기</li> </ul>

### 2 그래픽 관련 클래스

#### ③ Graphics

그래픽 관련 메소드	설 명
<code>void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)</code>	내부가 색으로 채워져 있고, 모서리가 둥근 사각형을 그림
<code>void drawString(String str, int x, int y)</code>	문자열을 x, y 좌표에 그림

# 그래픽 프로그래밍 개요

## 2 그래픽 관련 클래스

### ③ Graphics

이벤트 관련 메소드	설 명
<code>void setColor(Color c)</code>	현재 그래픽 색상을 지정
<code>void setFont(Font font)</code>	현재 글꼴을 지정
<code>Color getColor()</code>	현재 그래픽 색상을 반환
<code>Font getFont()</code>	현재 글꼴을 반환



# 그래픽 프로그래밍 개요



그래픽 프로그래밍

그래픽 프로그래밍의 개요



## 실습하기



그래픽 프로그래밍

그래픽 프로그래밍의 개요

### 실습순서

1. Canvas와 그래픽 관련 메소드를 이용한 프로그래밍 실습
  - 1) Canvas를 이용한 프로그래밍
  - 2) 그래픽 관련 메소드를 이용한 프로그래밍



### 유의사항

- JDK와 이클립스를 설치한 후 실습이 가능함
- 본인이 원하는 작업 폴더를 미리 정해 놓은 다음 실습하기
- 작업 폴더는 C드라이브에 지정하기 보다는 D드라이브나 외장하드디스크를 활용하는 것을 추천함

※ 제공되는 실습 코드를 다운받아 실습해보시기 바랍니다.



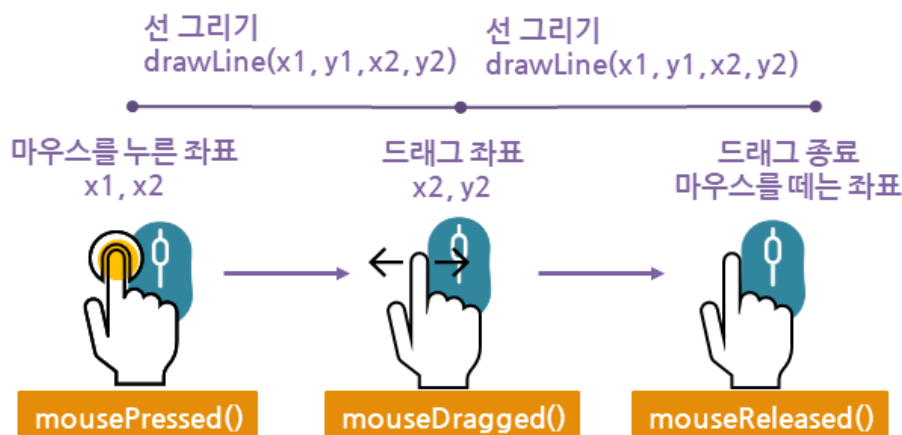


# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### ③ 마우스를 누른 좌표와 드래그 좌표에 선을 그림



## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### ④ 이벤트 핸들러 메소드 구현

```
int x1=-1, y1=-1, x2=-1, y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    if(x2 != -1 && y2 != -1) {
        x1 = x2;
        y1 = y2;
    }
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1, y1=-1, x2=-1, y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    if(x2 != -1 && y2 != -1) {
        x1 = x2;
        y1 = y2;
    }
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

선을 그리는 시작 좌표(x1, y1)와  
종료 좌표(x2, y2)의 초기값을 -1로 지정

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1, y1=-1, x2=-1, y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    if(x2 != -1 && y2 != -1) {
        x1 = x2;
        y1 = y2;
    }
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

마우스 드래그 이벤트 핸들러 메소드

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1, y1=-1, x2=-1, y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY(); ← 마우스를 누른 좌표값을 x1, y1에 지정
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    if(x2 != -1 && y2 != -1) {
        x1 = x2;
        y1 = y2;
    }
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1, y1=-1, x2=-1, y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1; ← x2, y2 초기화
}
public void mouseDragged(MouseEvent e) {
    if(x2 != -1 && y2 != -1) {
        x1 = x2;
        y1 = y2;
    }
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1, y1=-1, x2=-1, y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    if(x2 != -1 && y2 != -1) {
        x1 = x2;
        y1 = y2;
    }
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

← 마우스 드래그 이벤트 핸들러 메소드

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1, y1=-1, x2=-1, y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    if(x2 != -1 && y2 != -1) {
        x1 = x2;
        y1 = y2;
    }
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

← 처음 마우스 드래그 이벤트 발생이 아니라면

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1, y1=-1, x2=-1, y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    if(x2 != -1 && y2 != -1) {
        x1 = x2;
        y1 = y2;
    }
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

종료 좌표 x2, y2를 시작 좌표 x1, y1에 지정

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1, y1=-1, x2=-1, y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    if(x2 != -1 && y2 != -1) {
        x1 = x2;
        y1 = y2;
    }
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

마우스가 드래그된 좌표값을 x2, y2에 지정

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1, y1=-1, x2=-1, y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    if(x2 != -1 && y2 != -1) {
        x1 = x2;
        y1 = y2;
    }
    x2 = e.getX();
    y2 = e.getY();
    repaint(); ← repaint() 메소드 호출
}
```

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드

```
public void mouseReleased(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint(); ← 마우스를 떼는 경우 호출되는 이벤트 핸들러 메소드
}
public void update(Graphics g) {
    paint(g);
}
public void paint(Graphics g) {
    g.drawLine(x1, y1, x2, y2);
}
```

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드

```
public void mouseReleased(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
public void update(Graphics g) {
    paint(g);
}
public void paint(Graphics g) {
    g.drawLine(x1, y1, x2, y2);
}
```

← 마우스를 댄 좌표 값을 x2, y2에 지정

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드

```
public void mouseReleased(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
public void update(Graphics g) {
    paint(g);
}
public void paint(Graphics g) {
    g.drawLine(x1, y1, x2, y2);
}
```

← repaint() 메소드 호출

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드

```
public void mouseReleased(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
public void update(Graphics g) {
    paint(g);
}
public void paint(Graphics g) {
    g.drawLine(x1, y1, x2, y2);
}
```

← update() 메소드 오버라이딩  
: 화면 지우는 동작을 하지 않음

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ① 자유 곡선 그리기

#### 4 이벤트 핸들러 메소드

```
public void mouseReleased(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
public void update(Graphics g) {
    paint(g);
}
public void paint(Graphics g) {
    g.drawLine(x1, y1, x2, y2);
}
```

← paint() 메소드  
: x1,y1을 시작으로 x2,y2로 선을 그림



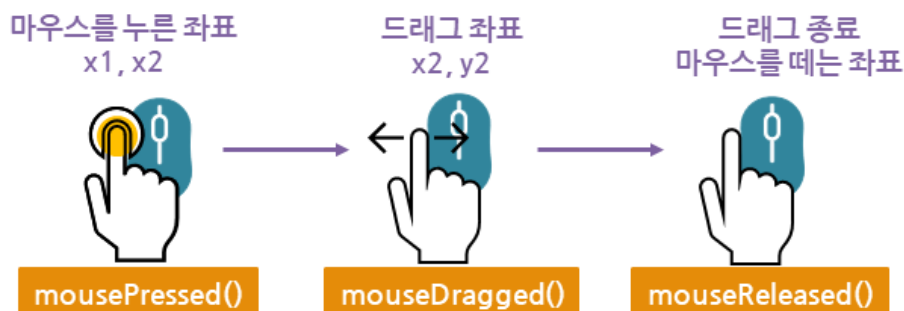
# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### ③ 마우스를 누른 좌표와 드래그 좌표에 도형을 그림

도형의 폭과 넓이 계산, 도형의 좌측 위 좌표 계산 필요



## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### ④ 이벤트 핸들러 메소드 구현

```
int x1=-1,y1=-1,x2=-1,y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1,y1=-1,x2=-1,y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

도형을 그리는 시작 좌표(x1,y1)와  
종료 좌표(x2,y2)의 초기값을 -1로 지정

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1,y1=-1,x2=-1,y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

← 마우스를 누르는 경우 호출되는 이벤트 핸들러 메소드

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1,y1=-1,x2=-1,y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

마우스를 누른 좌표값을 x1, y1에 지정하고, x2, y2 초기화

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1,y1=-1,x2=-1,y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

마우스 드래그 이벤트 핸들러 메소드

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### 4 이벤트 핸들러 메소드 구현

```
int x1=-1,y1=-1,x2=-1,y2=-1;
public void mousePressed(MouseEvent e) {
    x1 = e.getX();
    y1 = e.getY();
    x2 = y2 = -1;
}
public void mouseDragged(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
```

마우스가 드래그된 좌표값을 x2, y2에 지정  
repaint() 메소드 호출

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### 4 이벤트 핸들러 메소드 구현

```
public void mouseReleased(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
public void update(Graphics g) {
    paint(g);
}
public void paint(Graphics g) {
    int x = Math.min(x1, x2);
    int y = Math.min(y1, y2);
    int width = Math.abs(x2 - x1);
    int height = Math.abs(y2 - y1);
    g.drawRect(x, y, width, height);
}
```

도형을 그리는 시작 좌표(x1, y1)와  
종료 좌표(x2, y2)의 초기값을 -1로 지정

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### 4 이벤트 핸들러 메소드 구현

```
public void mouseReleased(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
public void update(Graphics g) {
    paint(g);
}
public void paint(Graphics g) {
    int x = Math.min(x1, x2);
    int y = Math.min(y1, y2);
    int width = Math.abs(x2 - x1);
    int height = Math.abs(y2 - y1);

    g.drawRect(x, y, width, height);
}
```

마우스를 댄 좌표 값을 x2, y2에 지정  
repaint() 메소드 호출

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### 4 이벤트 핸들러 메소드 구현

```
public void mouseReleased(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
public void update(Graphics g) {
    paint(g);
}
public void paint(Graphics g) {
    int x = Math.min(x1, x2);
    int y = Math.min(y1, y2);
    int width = Math.abs(x2 - x1);
    int height = Math.abs(y2 - y1);

    g.drawRect(x, y, width, height);
}
```

update() 메소드 오버라이딩하지 않음  
: 화면 지우는 동작을 적용

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### 4 이벤트 핸들러 메소드 구현

```
public void mouseReleased(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}

public void update(Graphics g) {
    paint(g);
}

public void paint(Graphics g) {
    int x = Math.min(x1, x2);
    int y = Math.min(y1, y2);
    int width = Math.abs(x2 - x1);
    int height = Math.abs(y2 - y1);

    g.drawRect(x, y, width, height);
}
```

← paint() 메소드

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### 4 이벤트 핸들러 메소드 구현

```
public void mouseReleased(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}

public void update(Graphics g) {
    paint(g);
}

public void paint(Graphics g) {
    int x = Math.min(x1, x2);
    int y = Math.min(y1, y2);
    int width = Math.abs(x2 - x1);
    int height = Math.abs(y2 - y1);

    g.drawRect(x, y, width, height);
}
```

← 도형의 좌측 위 좌표 계산 (좌표 값이 작은 것)

# 그래픽 프로그래밍 응용

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### 4 이벤트 핸들러 메소드 구현

```
public void mouseReleased(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
public void update(Graphics g) {
    paint(g);
}
public void paint(Graphics g) {
    int x = Math.min(x1, x2);
    int y = Math.min(y1, y2);
    int width = Math.abs(x2 - x1);
    int height = Math.abs(y2 - y1);
    g.drawRect(x, y, width, height);
}
```

← 도형의 폭과 높이 계산(절대값으로 계산)

## 1 마우스 이벤트를 이용한 그래픽 프로그래밍

### ② 도형 그리기

#### 4 이벤트 핸들러 메소드 구현

```
public void mouseReleased(MouseEvent e) {
    x2 = e.getX();
    y2 = e.getY();
    repaint();
}
public void update(Graphics g) {
    paint(g);
}
public void paint(Graphics g) {
    int x = Math.min(x1, x2);
    int y = Math.min(y1, y2);
    int width = Math.abs(x2 - x1);
    int height = Math.abs(y2 - y1);
    g.drawRect(x, y, width, height);
}
```

← 도형 그리기



# 그래픽 프로그래밍 응용



그래픽 프로그래밍

그래픽 프로그래밍의 응용



## 실습하기



그래픽 프로그래밍

그래픽 프로그래밍의 응용

### 실습순서

1. 마우스 이벤트를 이용한 그래픽 프로그래밍
  - 1) 자유 곡선을 그리는 그래픽 프로그래밍
  - 2) 도형을 그리는 그래픽 프로그래밍



### 유의사항

- JDK와 이클립스를 설치한 후 실습이 가능함
- 본인이 원하는 작업 폴더를 미리 정해 놓은 다음 실습하기
- 작업 폴더는 C드라이브에 지정하기 보다는 D드라이브나 외장하드디스크를 활용하는 것을 추천함



※ 제공되는 실습 코드를 다운받아 실습해보시기 바랍니다.



## 그래픽 프로그래밍 응용



### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ① 임의의 위치와 색상 이용하기

- 1 임의의 위치와 색상을 적용한 도형을 화면에 표시
- 2 임의의 값 : Random
- 3 색상 : Color
- 4 멀티스레드 : Thread 클래스, Runnable 인터페이스

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ① 임의의 위치와 색상 이용하기

##### 5 관련 메소드

이벤트 관련 메소드	설 명
Dimension getSize()	컴포넌트의 크기를 Dimension 객체로 반환
int nextInt(int bound)	<ul style="list-style-type: none"> <li>정수형 난수 값을 반환( 0 ~ bound-1 사이의 값을 반환 )</li> <li>java.util.Random 클래스의 메소드</li> </ul>
Color(int r, int g, int b)	<ul style="list-style-type: none"> <li>Color 클래스의 생성자 메소드</li> <li>r, g, b 값을 지정하여 Color 객체 생성</li> <li>r, g, b 값은 0~255 사이의 값</li> </ul>

## 그래픽 프로그래밍 응용

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ① 임의의 위치와 색상 사용하기

##### 5 관련 메소드

이벤트 관련 메소드	설 명
<code>void run()</code>	멀티스레드 코드가 정의된 메소드 Thread 클래스를 상속하거나, Runnable 인터페이스를 구현한 경우 오버라이딩하여야 함
<code>void start()</code>	<code>run()</code> 메소드 호출
<code>static void sleep(long millis)</code>	지정한 시간 동안 스레드를 멈춤 ( 1/1000초 단위로 지정 )

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ① 임의의 위치와 색상 사용하기

##### 6 이벤트 핸들러 메소드 구현

```
ThreadRandom tr = new ThreadRandom();
tr.start();
```

## 그래픽 프로그래밍 응용



### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ① 임의의 위치와 색상 이용하기

##### 6 이벤트 핸들러 메소드 구현

```
ThreadRandom tr = new ThreadRandom();
tr.start();
```

← 스레드 객체를 생성

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ① 임의의 위치와 색상 이용하기

##### 6 이벤트 핸들러 메소드 구현

```
ThreadRandom tr = new ThreadRandom();
tr.start();
```

← 스레드 시작

## 그래픽 프로그래밍 응용

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ① 임의의 위치와 색상 이용하기

##### 6 이벤트 핸들러 메소드 구현

```
public void run() {
    for(int i = 0; i < 100; i++) {
        x = random.nextInt(width+1);
        y = random.nextInt(height+1);
        red = random.nextInt(256);
        green = random.nextInt(256);
        blue = random.nextInt(256);
        try {
            Thread.sleep(100);
        } catch (Exception e) {
            System.out.println("Thread Error!!");
        }
        repaint();
    }
}
```

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ① 임의의 위치와 색상 이용하기

##### 6 이벤트 핸들러 메소드 구현

```
public void run() {
    for(int i = 0; i < 100; i++) { ← 100번 반복
        x = random.nextInt(width+1);
        y = random.nextInt(height+1);
        red = random.nextInt(256);
        green = random.nextInt(256);
        blue = random.nextInt(256);
        try {
            Thread.sleep(100);
        } catch (Exception e) {
            System.out.println("Thread Error!!");
        }
        repaint();
    }
}
```

# 그래픽 프로그래밍 응용

## 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

### ① 임의의 위치와 색상 이용하기

#### 6 이벤트 핸들러 메소드 구현

```
public void run() {
    for(int i = 0; i < 100; i++) {
        x = random.nextInt(width+1);
        y = random.nextInt(height+1); ← 임의의 x, y 좌표 값 지정
        red = random.nextInt(256);
        green = random.nextInt(256);
        blue = random.nextInt(256);
        try {
            Thread.sleep(100);
        } catch (Exception e) {
            System.out.println("Thread Error!!");
        }
        repaint();
    }
}
```

## 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

### ① 임의의 위치와 색상 이용하기

#### 6 이벤트 핸들러 메소드 구현

```
public void run() {
    for(int i = 0; i < 100; i++) {
        x = random.nextInt(width+1);
        y = random.nextInt(height+1);
        red = random.nextInt(256);
        green = random.nextInt(256); ← 임의의 r, g, b 값 지정
        blue = random.nextInt(256);
        try {
            Thread.sleep(100);
        } catch (Exception e) {
            System.out.println("Thread Error!!");
        }
        repaint();
    }
}
```

# 그래픽 프로그래밍 응용

## 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

### ① 임의의 위치와 색상 이용하기

#### 6 이벤트 핸들러 메소드 구현

```
public void run() {
    for(int i = 0; i < 100; i++) {
        x = random.nextInt(width+1);
        y = random.nextInt(height+1);
        red = random.nextInt(256);
        green = random.nextInt(256);
        blue = random.nextInt(256);
        try {
            Thread.sleep(100); ← 0.1초 동안 스레드를 지연(0.1초간 멈춤)
        } catch (Exception e) {
            System.out.println("Thread Error!!");
        }
        repaint();
    }
}
```

## 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

### ① 임의의 위치와 색상 이용하기

#### 6 이벤트 핸들러 메소드 구현

```
public void run() {
    for(int i = 0; i < 100; i++) {
        x = random.nextInt(width+1);
        y = random.nextInt(height+1);
        red = random.nextInt(256);
        green = random.nextInt(256);
        blue = random.nextInt(256);
        try {
            Thread.sleep(100);
        } catch (Exception e) {
            System.out.println("Thread Error!!");
        }
        repaint(); ← repaint() 메소드 호출
    }
}
```

## 그래픽 프로그래밍 응용

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ① 임의의 위치와 색상 이용하기

##### 6 이벤트 핸들러 메소드 구현

```
public void update(Graphics g) {
    paint(g);
}

public void paint(Graphics g) {
    if( x != -1 && y != -1 ) {
        g.setColor(new Color(red, green, blue));
        g.fillOval(x, y, 10, 10);
    }
}
```

← update() 메소드 오버라이딩  
: 화면 지우는 동작을 하지 않음

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ① 임의의 위치와 색상 이용하기

##### 6 이벤트 핸들러 메소드 구현

```
public void update(Graphics g) {
    paint(g);
}

public void paint(Graphics g) {
    if( x != -1 && y != -1 ) {
        g.setColor(new Color(red, green, blue));
        g.fillOval(x, y, 10, 10);
    }
}
```

← 임의의 값을 가지고 있는 r, g, b를 이용하여  
Color 객체를 생성한 후 Canvas에 색상을 지정

## 그래픽 프로그래밍 응용

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ① 임의의 위치와 색상 사용하기

##### ⑥ 이벤트 핸들러 메소드 구현

```
public void update(Graphics g) {
    paint(g);
}
public void paint(Graphics g) {
    if( x != -1 && y != -1 ) {
        g.setColor(new Color(red, green, blue));
        g.fillOval(x, y, 10, 10);
    }
}
```

← 임의의 값을 가지고 있는 x, y 좌표에 타원을 그림

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ② 이미지 사용하기

- ① 10장의 이미지를 차례대로 화면에 표시
- ② 이미지 관리 클래스 : Image
- ③ 관련 메소드

관련 메소드	설 명
Image getImage("파일이름")	"파일이름"에 해당하는 이미지 파일을 불러와서 Image 객체로 반환 • 호출방법 : Toolkit.getDefaultToolkit().getImage("파일이름")



## 그래픽 프로그래밍 응용



### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ② 이미지 이용하기

##### 4 이벤트 핸들러 메소드 구현

```
for(int i=0; i< name.length; i++)
    image[i] = Toolkit.getDefaultToolkit().getImage(name[i]);
Thread t = new Thread(this);
t.start();
```

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ② 이미지 이용하기

##### 4 이벤트 핸들러 메소드 구현

```
for(int i=0; i< name.length; i++)
    image[i] = Toolkit.getDefaultToolkit().getImage(name[i]);
Thread t = new Thread(this);
t.start();
```

↑ 이미지를 가져와 Image 객체 생성

## 그래픽 프로그래밍 응용

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ② 이미지 이용하기

##### 4 이벤트 핸들러 메소드 구현

```
for(int i=0; i< name.length; i++)
    image[i] = Toolkit.getDefaultToolkit().getImage(name[i]);
Thread t = new Thread(this);
t.start();
```

← 스레드 객체 생성 후 시작

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ② 이미지 이용하기

##### 4 이벤트 핸들러 메소드 구현

```
public void run() {
    while(true) { ← 무한 반복
        index++;
        if( index == image.length ) index = 0;
        try {
            Thread.sleep(1000);
        } catch(Exception e) {
            System.out.println("Thread Error!!");
        }
        repaint();
    }
}
```

## 그래픽 프로그래밍 응용

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ② 이미지 이용하기

##### 4 이벤트 핸들러 메소드 구현

```
public void run() {
    while(true) {
        index++;
        if( index == image.length ) index = 0;
        try {
            Thread.sleep(1000);
        } catch(Exception e) {
            System.out.println("Thread Error!!");
        }
        repaint();
    }
}
```

← index 변수의 값 : 0~9까지 무한 반복

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ② 이미지 이용하기

##### 4 이벤트 핸들러 메소드 구현

```
public void run() {
    while(true) {
        index++;
        if( index == image.length ) index = 0;
        try {
            Thread.sleep(1000);
        } catch(Exception e) {
            System.out.println("Thread Error!!");
        }
        repaint();
    }
}
```

← 1초 동안 스레드 지연

## 그래픽 프로그래밍 응용

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ② 이미지 이용하기

##### 4 이벤트 핸들러 메소드 구현

```
public void run() {
    while(true) {
        index++;
        if( index == image.length ) index = 0;
        try {
            Thread.sleep(1000);
        } catch(Exception e) {
            System.out.println("Thread Error!!");
        }
        repaint(); ← repaint() 메소드 호출
    }
}
```

### 2 멀티스레드(Multi-Thread)를 이용한 그래픽 프로그래밍

#### ② 이미지 이용하기

##### 4 이벤트 핸들러 메소드 구현

```
public void paint(Graphics g) {
    if( index != -1 )
        g.drawImage(image[index], 0, 0, this); ← 이미지를 화면에 그림
}
```

# 그래픽 프로그래밍 응용



그래픽 프로그래밍

그래픽 프로그래밍의 응용



## 실습하기



그래픽 프로그래밍

그래픽 프로그래밍의 응용

### 실습순서

1. 멀티스레드를 이용한 그래픽 프로그래밍
  - 1) 임의의 위치와 색상을 이용한 그래픽 프로그래밍
  - 2) 이미지를 처리하는 그래픽 프로그래밍



### 유의사항

- JDK와 이클립스를 설치한 후 실습이 가능함
- 본인이 원하는 작업 폴더를 미리 정해 놓은 다음 실습하기
- 작업 폴더는 C드라이브에 지정하기 보다는 D드라이브나 외장하드디스크를 활용하는 것을 추천함

※ 제공되는 실습 코드를 다운받아 실습해보시기 바랍니다.

# 응용문제

그래픽 프로그래밍 응용문제

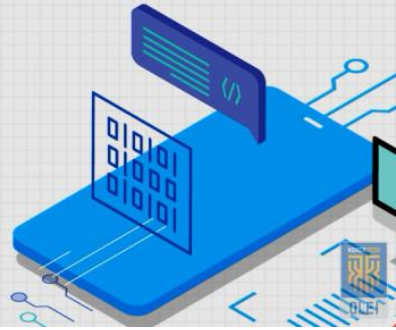
## 다음 실행화면과 조건에 맞게 프로그램을 작성하시오.

### 조건

- 1 도형 선택 : 사각형, 모서리가 둥근 사각형, 타원
- 2 색상 선택 : Red, Green, Blue
- 3 채우기 선택 : true or false
- 4 3가지를 선택한 사항에 맞게 마우스를 이용하여 도형 그리기

### 클래스명 : GraphicFigure

제공되는 실습 소스코드를 다운받아 실습해보시기 바랍니다.



그래픽 프로그래밍 응용문제

### 실행화면

