**JAVA 프로그래밍 lab #13**

담당 교수님 : 천인국 교수님

이름 : 현 지 원

학과 : 컴퓨터공학과

학번 : 20174069

**1.** 간단한 애니메이션을 작성하여 보자. 우주선 이미지를 읽어서 화면에 표시한다. 동시에 타이머를 이용하여서 우주선을 움직여 보자. 우주선은 좌측 하단에서 우측 상단으로 움직인다.

**Source code #1**

**package** lab13;

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.swing.\*;

**public** **class** animation **extends** JFrame **implements** ActionListener {

Image img;

Timer timer;

**int** pos\_x = 0, pos\_y = 537;

**private** **int** xSpeed = 3, ySpeed = 3;

**private** **static** **final** **int** ***PERIOD*** = 10; // 시간

// 내부 클래스

**class** MyPanel **extends** JPanel {

// MyPanel 생성자

**public** MyPanel() {

setBackground(Color.***black***); // 배경을 검정색으로 설정

}

**public** **void** paintComponent(Graphics g) {

**super**.paintComponent(g);

g.drawImage(img, pos\_x, pos\_y, **this**); // 화면에 이미지 그리기

}

}

**public** **void** update() { // 좌표 이동

pos\_x += xSpeed;

pos\_y -= ySpeed;

}

**public** animation() {

setTitle("Animation");

setSize(700, 700);

setVisible(**true**);

add(**new** MyPanel());

ImageIcon icon = **new** ImageIcon("rrr.jpg");

img = icon.getImage();

add(**new** MyPanel());

setVisible(**true**);

timer = **new** Timer(***PERIOD***, **this**); // 10ms마다 타이머가 걸림

timer.start();

}

@Override

**public** **void** actionPerformed(ActionEvent arg0) { // ActionListener인터페이스 구현

update();

repaint();

}

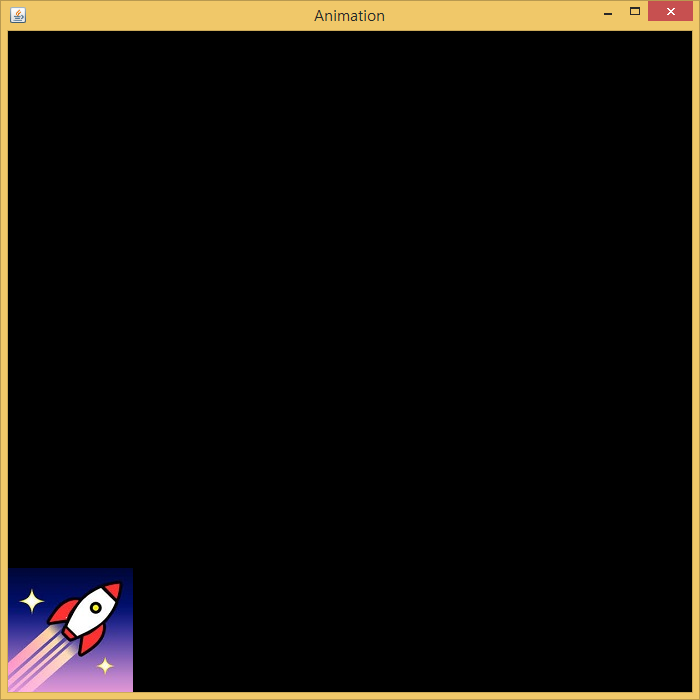
**public** **static** **void** main(String[] args) {

animation a = **new** animation();

}

}

**실행결과**



**해설**

먼저 JFrame을 상속받는 animation클래스를 만든다. 프레임 클래스로 이벤트를 처리하기 위해 ActionListener 인터페이스를 구현한다. 클래스에서 사용할 필드들을 작성한다. 자바에서는 프레임 위에 직접 그림을 그리는 것 보다 패널 위에 그리는 것이 좋은 방법이다. 패널은 그림을 그릴 수 있는 표면을 가지고 있고, 동시에 컨테이너의 역할도 한다. 일반적으로는 JPanel을 상속받은 자식 클래스를 작성한 후 사용한다. 그러므로 animation클래스 안에 JPanel을 상속받는 내부 클래스 MyPanel을 생성한다. MyPanel의 생성자에서 프레임의 배경화면을 검정색으로 설정해준다. paintComponent()에는 자신을 화면에 그리는 코드가 들어 있다. 따라서 어떤 컴포넌트 위에 그림을 그리기 위해서는 paintComponent() 메소드를 재정의하고 코드를 추가하여야 한다. paintComponent()는 Graphics 타입의 매개 변수를 가지고 있다. Graphics 클래스는 그림을 그리는데 필요한 모든 설정값과 메소드를 가지고 있다. paintComponent()메소드를 재정의 해주고 좌표를 이동시켜주는 메소드를 update()를 정의한다. animation의 생성자에서는 우주선 이미지를 읽어서 화면에 표시한다. 또한, 윈도우 창을 설정해주고 MyPanel() 객체를 추가해준다. 타이머는 10ms마다 타이머가 걸리도록 해주었다. 이벤트 처리는 actionPerformed() 메소드를 재정의로 해준다. update()메소드를 호출하고 repaint()메소드를 호출한다. repaint()는 적절한 시기에 paintComponent()를 호출한다. main() 메소드에서는 꼭 animation 객체를 선언해주어야 프로그램이 실행된다.

**2.** 프로그램이 시작되면 화면에 사각형이 하나 그려진다. 사용자가 사각형 안에서 마우스를 드래그하면 사각형이 움직이도록 프로그램을 작성해보자.

**Source code#2**

**package** asdf;

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.swing.\*;

**public** **class** animation **extends** JFrame {

**int** x = 0, y = 0;

// 내부 클래스

**class** MyPanel **extends** JPanel {

// 사각형 그리기

**protected** **void** paintComponent(Graphics g) {

**super**.paintComponent(g);

g.setColor(**new** Color(117, 0, 252));

g.fillRect(x, y, 100, 100);

}

**public** MyPanel() {

addMouseMotionListener(**new** MouseMotionAdapter() { // 마우스 이벤트 처리

**public** **void** mouseDragged(MouseEvent e) {

**if** (e.getX() > 0 && e.getY() > 0 && e.getX() < x + 100 && e.getY() < y + 100) { // 도형 내 좌표 값 범위

x = e.getX() - 50; // 사각형 중앙으로 커서 옮김

y = e.getY() - 50;

repaint();

}

}

});

}

}

// 생성자

**public** animation() {

setTitle("그래픽 기초 프로그램");

setSize(500, 500);

add(**new** MyPanel());

setVisible(**true**);

}

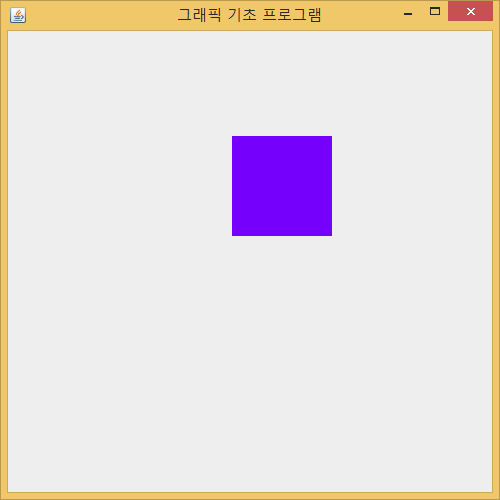
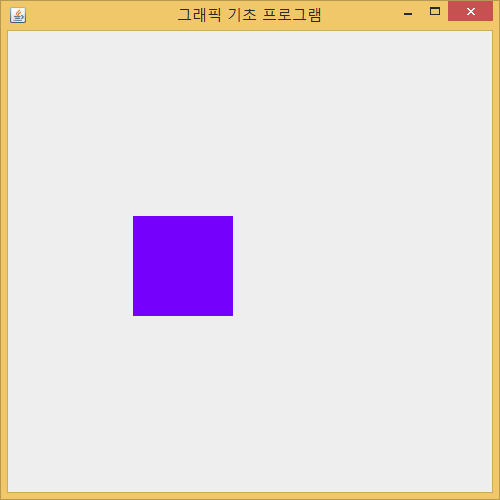
**public** **static** **void** main(String[] args) {

animation a = **new** animation();

}

}

**실행결과**



**해설**

프로그램이 시작되면 화면에 사각형이 하나 그려지고 사용자가 사각형 안에서 마우스를 드래그하면 사각형이 움직이도록 하는 프로그램이다. JFrame을 상속받는 animation 클래스를 만들어준다. 자바에서는 프레임에 바로 그림을 그리는 것은 좋지 않기 때문에 패널에 그림을 그릴 수 있도록 JPanel을 상속받는 Mypanel 클래스를 만들어준다. MyPanel과 같이 클래스 안에 있는 클래스를 내부 클래스라고 한다. 그림을 그리기 위해서는 컴포넌트의 paintComponent() 메소드를 재정의해 주어야 한다. super.paintComponent(g)를 해주는 이유는 부모 클래스가 그려야 될 부분도 있기 때문이다. paintComponent() 메소드에서 사각형의 색을 정해주고 x,y 좌표에 가로 세로 길이가 100인 사각형을 그려주는 코드를 작성한다. MyPanel() 생성자에서 addMouseMotionListener(new MouseMotionAdapter())로 마우스 이벤트 처리를 해준다. 사용자가 사각형을 마우스로 드래그하면 사각형이 움직이는 것이므로 mouseDragged() 메소드를 사용한다. x와 y의 위치를 얻어와 사각형의 안의 좌표를 클릭하면 드래그가 가능하도록 조건을 주었다. 사각형의 모서리 부분을 클릭하여도 중앙으로 커서를 옮기도록 하였다. 그 후 repaint()로 paintComponent()를 적절한 시기에 호출하도록 한다. animation() 생성자에서는 윈도우창을 설정해주고 MyPanel()객체를 생성해 프레임에 추가 해준다.

**3.** 이번 장에서 학습한 내용을 바탕으로 버튼을 누르면 2차 함수가 화면에 그려지는 프로그램을 작성하여 보자.

**Source code #3**

**package** asdf;

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** javax.swing.\*;

**public** **class** animation **extends** JFrame {

**private** JPanel panel, panel1;

**private** JButton draw;

**private** JTextField a, b, c;

**int** A, B, C;

**int** x = 200, y = 150;

**int** x1 = 0, y1 = 0;

// 내부 클래스

**class** MyPanel **extends** JPanel {

// 점 그리기

**protected** **void** paintComponent(Graphics g) {

**super**.paintComponent(g);

g.drawLine(0, 150, 500, 150);

g.drawLine(200, 0, 200, 500);

g.setColor(Color.***BLUE***);

// -10 <= x1 < 10 에 대한 y값에 표시

**for** (x1 = -10; x1 < 10; x1++) {

y1 = (A \* x1 \* x1) + (B \* x1) + C; // y = a\*x^2 + b\*x + c

g.fillOval(x + x1, y - y1, 5, 5);

}

}

}

// 생성자

**public** animation() {

setTitle("y = ax^2 + bx + c");

setSize(500, 300);

panel = **new** JPanel();

draw = **new** JButton("DRAW");

a = **new** JTextField(7);

b = **new** JTextField(7);

c = **new** JTextField(7);

panel.add(a);

panel.add(b);

panel.add(c);

panel.add(draw);

add(panel, BorderLayout.***NORTH***);

setVisible(**true**);

// 이벤트 처리 - 람다식 사용

draw.addActionListener(e -> {

A = Integer.*parseInt*(a.getText()); // 스트링형을 정수형으로 형 변환

B = Integer.*parseInt*(b.getText());

C = Integer.*parseInt*(c.getText());

add(**new** MyPanel(), BorderLayout.***CENTER***); // 프레임 중앙에 배치

setVisible(**true**);

});

}

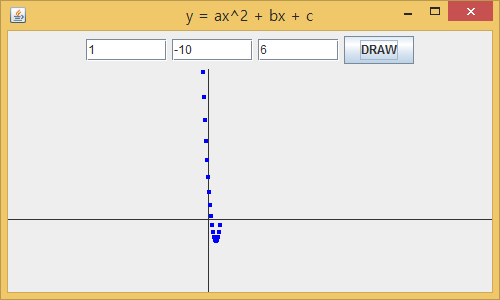
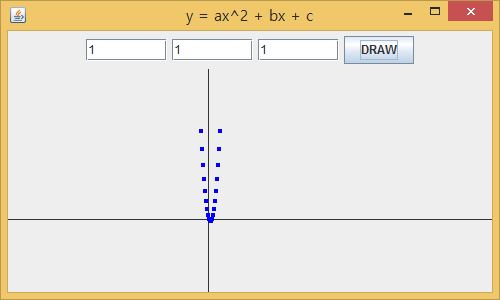
**public** **static** **void** main(String[] args) {

animation a = **new** animation();

}

}

**실행결과**



**해설**

DRAW 버튼을 누르면 2차 함수가 화면에 그려지는 프로그램이다. JFrame을 상속받는 animation 클래스를 만들어준다. 먼저 프로그램에 필요한 컴포넌트들이 어떤 것들인지 생각하고 필드로 정의해 주었다. 그림을 그려야 하기 때문에 JPanel을 상속받는 MyPanel 내부 클래스를 작성하여 함수를 그래프를 그릴 수 있도록 한다. 그림을 그리기 위해서는 paintComponent() 메소드를 재정의 해주어야 한다. super.paintComponent()를 해주는 이유는 부모클래스가 그려야 할 부분도 있기 때문이다. x와 y축을 패널에 그려주고 10 x < 10에 대한 y값을 표시할 수 있도록 for문 안에 방적식을 작성한 후 그 결과에 맞게 타원을 그리도록 하였다. animation() 생성자에서는 윈도우창을 설정해주고 panel을 만들어서 사용자에게 a,b,c 값을 입력 받을 수 있도록 textfield 세개와 DRAW 버튼을 만들어 패널에 추가한다. 배치관리자로 BorderLayout을 사용하여 북쪽에 패널을 배치시킨다. 버튼을 눌렀을 때 이벤트 처리는 람다식으로 사용하였고 textfield는 문자열을 입력 받기 때문에 방정식을 계산하기 위해 문자열을 정수형으로 변환할 수 있도록 하였다. MyPanel을 프레임의 CENTER로 배치해 준다. main에서 animation 객체를 생성하여 프로그램이 실행 될 수 있도록 한다.