< JAVA PROGRAMMING LAB -10장: 그래픽 이벤트 처리>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과



**1. 화면의 끝에서 끝으로 로켓 이미지가 움직이는 프로그램을 만들어라**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

**import** javax.swing.Timer;

**class** Rocket **extends** JPanel{

 **int** x,y;

 **public** Rocket(**int** x,**int** y) {

 **this**.x=x;

 **this**.y=y;

 }

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

 g.drawImage(window.*roc*,x,y,100,100,**this**);

 }

 **public** **void** update(){

 **if**(y<=0) {}

 **else** {

 x+=30;y-=10;}

 }

}

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

 **static** **final** **int** ***speed***=30;

 Timer timer;

 ImageIcon img;

 **static** Image *roc*;

 Rocket r;

 **public** window() {

 setSize(500,300);

 setTitle("Animation");

 img=**new** ImageIcon("e://rocket.jpg");

 *roc*=img.getImage();

 r=**new** Rocket(10,100);

 setVisible(**true**);

 **class** Mypanel **extends** JPanel{

 **public** Mypanel() {

 setBackground(Color.***WHITE***);}

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

 **super**.paintComponent(g);

 r.paint(g);

 }

 }

 add(**new** Mypanel());

 timer=**new** Timer(***speed***,**this**);

 timer.start();

 }

 @Override

 **public** **void** actionPerformed(ActionEvent e) {

 // **TODO** Auto-generated method stub

 r.update();

 repaint();

 }

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

 // **TODO** Auto-generated method stub

 window f=**new** window();

 }

}

◇ 제목

◇ 제목

◇ 프로그램 소스

◇ 제목

◇ 제목

◇ 프로그램 소스

윈도우 클래스부터 살펴보겠다. 타이머 주기를 담당할 speed상수와timer,imageicon,image,rocket참조변수가 필드로 쓰인다. window생성자가 호출되면 윈도우의 크기를 지정하고 타이틀을 지정한다. 이후 이미지 아이콘 객체를 만들어서 e드라이브에 있는 사진을 객체에 담는다. 다음은 이미지 아이콘 객체가 담은 이미지를 얻어서 이미지 객체가 가지게 하였다. 로켓이라는 객체를 만들었다. 로켓 클래스를 살펴보겠다. 자바는 객체 지향인지라 개체 따위를 클래스로 지정하는게 상당히 편하다. 그래서 로켓 문제이니 로켓 클래스를 만든 것이다. 로켓 클래스는 위치에 해당하는 x,y필드를 가진다. 생성자에서는 이 필드들을 초기화해주고 paint메소드에서는 이미지 객체가 가진 이미지를 통해 패널에 로켓을 나타낸다. 로켓 클래스에서 JPanel을 상속하지 않으면 그림을 어디다가 그려야 되는지 몰라서 오류가 뜬다고 생각하자. 그리고 window클래스와 다른 클래스임에도 불구하고 window의 패널에 그림이 그려지는 이유는, 넘겨받은 그래픽 객체가 window안의 패널 클래스에서 super.paintComponent(g)<부모 클래스 메소드 거침>되었기 때문이다. update메소드는 x,y값을 바꾸는 메소드로 이 메소드와 함께 repaint(바뀐 변수를 바탕으로 다시 그리는 함수)가 호출되면 로켓이 움직인 것처럼 보인다. 다시 window로 돌아오겠다. MyPanel클래스가 보이는데 JPanel을 상속받고 paintcomponent를 재정의해서 그래픽 처리를 한다. 그래픽 처리는 이처럼 메소드 재정의와 부모 클래스의 paintcomponent에 그래픽 객체를 넘겨주면 가능하다. 그리고 이전에 로켓 객체에서 만든 paint클래스를 호출해서 그림을 그리게 된다. 이후 윈도우에서는 마이 패널을 인스턴스화한 뒤 윈도우에 추가한다. 그럼 setVisible이 동작해서 paintcomponent가 동작하고 그림이 자연스레 그려질 것이다. 이후 타이머 객체가 나오는데 이는 두 개의 인수를 가진다. speed와 this라는 인수가 쓰였는데 다음과 같이 해석된다. ( speed를 주기로 액션 이벤트를 한다. 이벤트 처리는 이 객체가 담당한다.) 이 타이머는 start라는 메소드가 호출되야 동작한다. 즉 speed가 30이니, 0.03초를 기준으로 actionPerfoemed가 된 다는 것이다. 이 액션 이벤트는 update와 repaint이니 0.03초마다 그림이 바뀐다. 사용자는 이를 로켓이 움직이는 것처럼 볼 것이다.

◇ 제목

◇ 제목

◇ 프로그램 설명

< JAVA PROGRAMMING LAB -10장: 그래픽 이벤트 처리>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과



**2. 속이 찬 사각형을 드래그하면 드래그한 방향대로 움직이게 프로그램을 만들어라**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseMotionListener;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

**import** javax.swing.Timer;

**class** Rect **extends** JPanel{

 **int** x,y;

 **public** Rect(**int** x,**int** y) {

 **this**.x=x;

 **this**.y=y;

 }

 **public** **void** setXandY(**int** x,**int** y) {

 **this**.x = x;

 **this**.y=y;

 }

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

 g.fillRect(x,y,40,40);

 }

}

**public** **class** window **extends** JFrame{

 Rect r;

 **public** window() {

 setSize(500,300);

 setTitle("Animation");

 r=**new** Rect(10,100);

 setVisible(**true**);

 **class** Mypanel **extends** JPanel **implements** MouseMotionListener{

 **public** Mypanel() {

 setBackground(Color.***WHITE***);

 addMouseMotionListener(**this**);

 }

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

 **super**.paintComponent(g);

 r.paint(g);

 }

 @Override

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

 // **TODO** Auto-generated method stub

 **if**(e.getX()>=r.x && e.getX()<=r.x+40&&e.getY()>=r.y && e.getY()<=r.y+40) {

 r.setXandY(e.getX()-20, e.getY()-20);

 repaint();} }

 @Override

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

 // **TODO** Auto-generated method stub

 }

 }

 add(**new** Mypanel());

 }

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

 // **TODO** Auto-generated method stub

 window f=**new** window();

 }

}

◇ 제목

◇ 제목

◇ 프로그램 소스

◇ 제목

◇ 제목

◇ 프로그램 소스

이 문제에서는 사각형이 개체이니 사각형에 대한 객체 Rect를 만들면 편하다. 그래서 Rect클래스를 만들었다. 이 클래스는 1번 문제와 별 다를게 없다. 다만 x,y필드를 수정할 설정자가 추가됐고 paint에서는 fillRect가 쓰여서 이미지 객체가 필요하지 않다는 점이다. 이젠 윈도우 클래스로 와보겠다. 사이즈와 타이틀을 정하고 Rect객체를 만들었다. 객체의 초기 위치는 10,100을 주었다. 이후 윈도우를 보이게 하였다. MyPanel클래스가 보이는데 MouseMotionListener를 구현한다고 한다. MouseAdapter를 해도 되지만상속은 한 번 밖에 받지 못하니 JFrame이 있는 이상 불가능하다. 일단 1번과 같은 형태로 paintcomponent를 재정의하였다. 그 전에 addMouseMotionListener가 보이는데, 마우스 이벤트를 이 객체에서 하겠다고 등록하는 것이다. 그리고 MyFrame에서 add가 됐으니, MyFrame내에서만 이벤트가 적용된다. 마우스 이벤트는 drag이벤트인데 드래그한 마우스의 위치를 바탕으로 설정자를 호출하고 repaint하면 사각형이 움직이는 구조이다. 이 이벤트는 조건문에 해당할 때 진행된다. 일반적으로 도형의 x,y위치는 x,y 좌측 상단 꼭지점이다. 그래서 일단 마우스의 x좌표가 r.x보다 크거나 같고 (r.x+사각형 width)보다 작거나 같아야 한다. 마우스의 y좌표도 r.y보다 크거나 같고 (r.y+사각형 height)보다 작거나 같아야 한다. 이 조건이 성립하고 설정자에 마우스 x,y위치를 그냥 보내면 도형의 좌측 상단 꼭지점이 내려가므로 -20을 보내서 올린 것이다. 그림을 그려보면 더욱 이해가 쉽다. 최후로 이 MyFrame을 윈도우에 추가했다.

◇ 제목

◇ 제목

◇ 프로그램 설명

< JAVA PROGRAMMING LAB -10장: 그래픽 이벤트 처리>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과

->

**3. 여러 몬스터를 만들고 사용자가 몬스터를 클릭하면 사라지게 만들어라**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.MouseAdapter;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** java.awt.event.MouseMotionListener;

**import** java.util.ArrayList;

**import** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

**import** javax.swing.Timer;

**class** Monster **extends** JPanel{

 **int** x,y,hp;

 **public** Monster(**int** x,**int** y,**int** hp) {

 **this**.x=x;

 **this**.y=y;

 **this**.hp=hp;

 }

 **public** **void** setMonster(**int** hp) {

 **this**.hp=hp;

 }

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

 **if**(hp==0) {}

 **else** {

 g.drawImage(window.*mon*,x,y,100,100,**this**);}

 }

}

**class** store{

 **int** monx,mony;

 **public** store(**int** x,**int** y){

 monx=x;

 mony=y;

 }

}

**public** **class** window **extends** JFrame{

 ImageIcon img;

 **static** Image *mon*;

 Monster r;

 **int** ran1,ran2;

 **public** window() {

 setSize(500,500);

 setTitle("Animation");

 img=**new** ImageIcon("e://monster.png");

 *mon*=img.getImage();

 ArrayList<Monster> ms=**new** ArrayList<>();

 ArrayList<store> ss=**new** ArrayList<>();

 **class** Mypanel **extends** JPanel{

 **public** Mypanel() {

 setBackground(Color.***WHITE***);

 **for**(**int** i=0;i<10;i++) {

 ran1=(**int**)(Math.*random*()\*300);

 ran2=(**int**)(Math.*random*()\*300);

 ms.add(**new** Monster(ran1,ran2,90));

 ss.add(**new** store(ran1,ran2));

 }

 addMouseListener(**new** MouseAdapter() {

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

 **for**(store store1:ss) {

 **if**(e.getX()<=store1.monx+100 && e.getX()>=store1.monx && e.getY()<=store1.mony+100 && e.getY()>=store1.mony) {{

**int** idx=ss.indexOf(store1);

 ms.get(idx).setMonster(0);

repaint();

 **break**;

 }

 }

 }

 });

 }

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

 **super**.paintComponent(g);

 **for**(Monster mon:ms)

 mon.paint(g);

 }

 }

 add(**new** Mypanel());

 setDefaultCloseOperation(***EXIT\_ON\_CLOSE***);

 setVisible(**true**);

 }

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

 // **TODO** Auto-generated method stub

 window f=**new** window();

 }

}

◇ 제목

◇ 제목

◇ 프로그램 소스

◇ 제목

◇ 제목

◇ 프로그램 소스

◇ 제목

◇ 제목

◇ 프로그램 소스

이 프로그램은 두더지 잡기와 유사하다. 다만 개체가 몬스터이니, Monster객체를 만들었다. 생성자는 1,2번 문제와 같은데 hp라는 필드가 추가되었다. 이 객체는 hp를 0을 가지면 그림을 그리지 않는다. 예를 들어 몬스터 30마리가 있다. 이는 몬스터 객체를 30개 만들었다는 것이다. 그런데 특정 객체에 hp를 0주면 그림을 그리지 않는다. 그리고 repaint를 해서 그림을 다시 세팅하면 29마리가 되는 것이다. 이런 방식으로 몬스터를 없앤다. 그 내용이 paint메소드의 if문에서 나온다. 그 위의 설정자가 hp필드만 받는 것도 이를 위해서이다. 이번엔 store클래스를 살펴보자. 이 클래스에서는 몬스터 객체의 x,y필드 값만 monx,mony로 따로 저장하기 위해 만들어졌다. 물론 몬스터 클래스로 문제를 해결할 수도 있지만 어레이 리스트를 다채롭게 쓰기 위해 이렇게 해보았다. window클래스를 살펴보자. 몬스터로 쓸 이미지를 ImageIcon객체가 갖게 하고 이 객체가 가진 이미지를 getImage메소드를 통해 가져와 이미지 객체가 갖게 하였다. 이미지 객체의 생성자에서 파일 가져오는 처리가 불가능하기에, 이게 프로그램에서 이미지를 가져올 가장 빠른 방법이다. 다음은 어레이 리스트가 2개 만들어진다. 이 리스트는 각각 Monster와 store객체를 원소로 담을 것이다. 필자는 Monster가 인스턴스화되면 store도 그 다음에 인스턴스화해서 어레이 리스트의 원소 인덱스를 같게 할 것이다. 그래야 매치가 되서 문제를 푸는데 지장이 없다. 이번엔 MyPanel클래스를 살펴 보자. 이 클래스에서는 for문이 쓰이는데 앞서 만든 리스트에 add를 한다. 즉 10번 반복한다는 것은 add를 10번하는 것이니, 몬스터 객체를 10개 만들고 원소로 추가하는 것이다. 몬스터 객체 각각의 x,y필드는 랜덤 값을 갖게 된다. 다음은 마우스 이벤트를 등록하는 장면이 나온다. 여기선 무명 클래스와 어댑터 클래스가 쓰였다. 무명 클래스는 클래스의 이름을 갖지 않는 클래스로 구성은다음과 같다. -> (상속이나 구현할 클래스 인스턴스화{클래스 정의}) 여기서는 마우스 어댑터 클래스를 상속하였는데, 이 클래스는마우스 리스너의 추상 메소드를 모두 구현해놓은 메소드이다. 즉 이 메소드를 상속받아서 필요한 마우스 이벤트 메소드만 재정의하면되겠다. 문제에서 클릭을 누르면 몬스터가 사라지게 만들랬으니, MouseClicked를 재정의하였다. for-each문을 통해 ss어레이 리스트의 원소를 하나씩 꺼내서 store1이라는 store객체를 가리킬 수 있는 참조변수에 가리키게 하였다. 이렇게 store1은 반복할 때마다 다른 store객체를 가리키게 된다. if문에서 사용자가 클릭한 x좌표가 monx<=x<=monx+100(이미지 width)에 속하고 y좌표도 mony<=y<=mony+100(이미지 height)에 속하면 몬스터 중 한 마리를 클릭하고 있다는 것이다. (몬스터 객체의 x,y필드가 좌측 상단이므로, 이렇게 문제를 풀었으나, 실제로 풀렸다.)만약 몬스터를 가리킨다면, 어레이 리스트의 indexof라는 메소드를 이용해서 현재 store1에 해당하는 객체가 인덱스 몇에 있는지를 가져온다. 이 인덱스를 idx라는 정수형 변수에 담고 몬스터 어레이 리스트인 ms.get(idx)를 해서 위 인덱스에 해당하는 몬스터 객체를 가져온다. 이후 이 객체의 설정자를 이용해서 hp를 0으로 만든다. 그리고 repaint를 하면 한 마리가 사라지는 것으로 보이는 것이다. 이렇게 이벤트 처리를 다 했으면 break문으로 for-each문을 빠져 나온다. 다음은 paintComponent를 재정의했는데, MyPanel이 인스턴스화되고 어레이 리스트에 원소를 추가한 다음 바로 호출되게 된다. 호출되면 for-each문을 통해 몬스터 객체의 paint를 호출해서 패널에 몬스터가 보이게 한다. 앞에서 추가를 10번했으니 몬스터 10마리가 보이게 되겠다. 마지막으로 이 패널을 윈도우에 추가했고 윈도우를 보이게 하였다.

◇ 제목

◇ 제목

◇ 프로그램 설명