**JAVA LAB**

20234014 컴퓨터 공학과 이은정

1. 피자 상속

**package** pizza;

**class** Circle {

**protected** **int** radius;

**public** Circle(**int** r) {

radius = r;

}

}

**class** Pizza **extends** Circle {

**public** Pizza(String name, **int** r) {

**super**(r);

System.***out***.println("피자의 종류: " + name + " , 피자의 크기 : " + r);

// **TODO** Auto-generated constructor stub

}

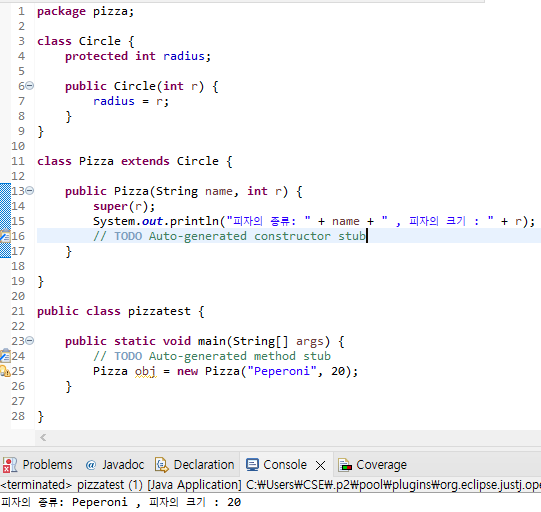
}

**public** **class** pizzatest {

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

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

Pizza obj = **new** Pizza("Peperoni", 20);

 }

}

Pizza 클래스는 Circle 클래스의 radius 값을 상속 받아 사용한다. Extends 를 통해 Circle을 상속 받은 이후에 super을 통해 먼저 초기화를 해준다. 이 때, 부모 클래스의 생성자가 r을 매개변수로 받으므로, 자식 생성자도 r을 매개변수로 받는다. 이후 피자의 종류 크기를 객체 생성시 받으면 이를 출력한다.

main에서 Pizza 객체를 만들 때에도 String 변수인 이름과, int 변수인 크기를 넣어서 자동으로 지정된 형식에 맞추어 출력할 수 있도록 코드를 작성했다.

2. 스포츠 출력하기

**class** Sports {

String getName() {

**return** "아직 결정되지 않음";

}

**int** getPlayers() {

**return** 0;

}

}

**class** Soccer **extends** Sports {

String name;

**int** getPlayers;

**public** Soccer(String name, **int** getPlayers) {

**super**();

**this**.name = name;

**this**.getPlayers = getPlayers;

}

@Override

String getName() {

**return** name;

}

@Override

**int** getPlayers() {

**return** getPlayers;

}

}

**public** **class** SportsTest {

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

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

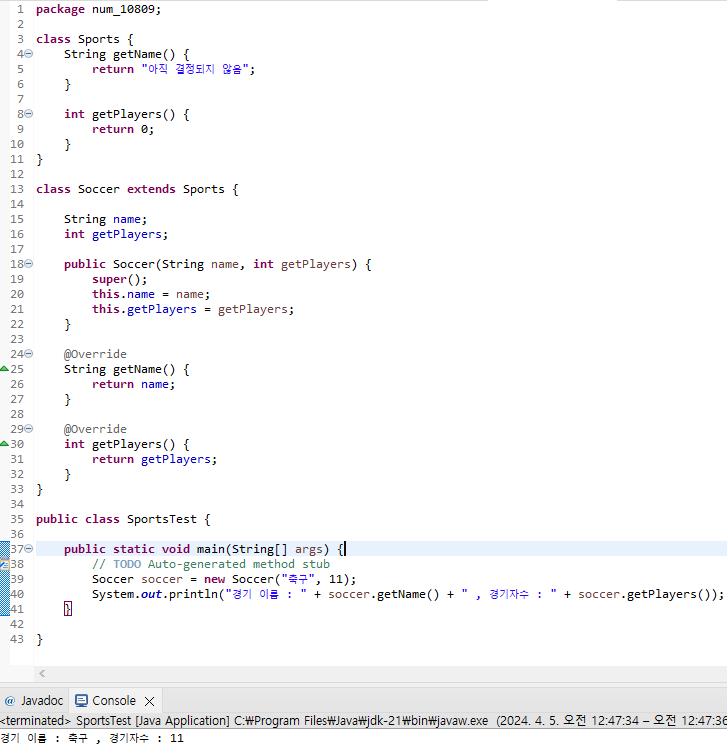
Soccer soccer = **new** Soccer("축구", 11);

System.***out***.println("경기 이름 : " + soccer.getName() + " , 경기자수 : " + soccer. getPlayers());

}

}

Sports는 마더 클래스로 getNames(), getPlayers() 메서드를 가지고 있다. 이 클래스는 각각 글자와 숫자를 반환하는 역할을 한다. 우리는 축구 경기에 관한 정보를 출력하려고 한다. 따라서 Soccer 클래스를 만들되 Sports 마더 클래스를 상속 받아 사용한다. 축구 경기에 대한 정보는 이름과 경기자 수이다. 따라서 이 정보를 저장할 변수인 String name, int players 변수를 만든다. 생성자를 통해 값을 입력 받는 즉시 필드에 값을 저장할 수 있도록 한다. 또한 Soccer는 Sports를 상속 받아 사용하기 때문에 super을 통해 Sports를 가져온다. 이제 축구 경기 정보에 대해 출력을 하기 위해서는 Sports의 메서드를 수정해야 한다. 메서드 수정은 @Override를 사용해 수정한다. 이를 메서드 오버라이딩이라고 한다. 따라서 정보를 출력하기 위해서는 아까 생성자를 통해 저장한 값을 출력한다. 이 때 메서드 오버라이딩을 하기 위해서는 변수 명, 반환 값이 같아야 한다. 따라서 getNames(), getPlayers() 는 각각의 형식에 맞는 변수를 return해주면 된다. 이런 과정을 통해 Sports에서 가지고 있던 메서드를 다시 수정해 재사용할 수 있게 된다. 이제 main에서 객체를 생성할 때 이름과 경기자 수를 전달해주면, 이 변수들을 getName, getPlayers를 통해 출력한다면 축구에 대한 정보를 출력할 수 있다.

3. 페이지 생성

**import** javax.swing.JFrame;

**class** FrameTest {

**public** **void** motherclass() {

JFrame f = **new** JFrame("Frame Test");

f.setSize(300, 200);

f.setVisible(**true**);

f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}

**class** FrameTest2 **extends** FrameTest {

String name;

**public** FrameTest2(String name) {

**super**();

**this**.name = name;

motherclass();

}

@Override

**public** **void** motherclass() {

JFrame f = **new** JFrame(name);

f.setSize(300, 200);

f.setLocation(300, 300);

f.setVisible(**true**);

f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}

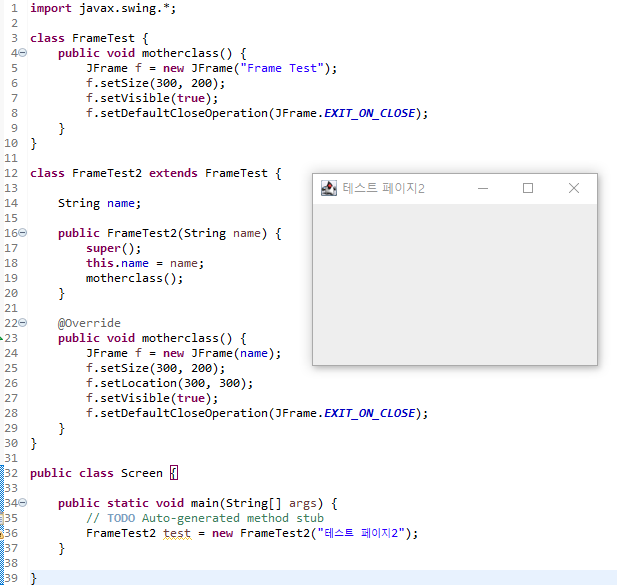
**public** **class** Screen {

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

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

FrameTest2 test = **new** FrameTest2("테스트 페이지2");

}

****}

위 코드는 FrameTest를 상속하여 FrameTest2를 통해 수정된 창을 창작하는 것이다. 먼저 FrameTest는 Jframe 클래스를 사용해 300,200 크기의 “Frame Test”이라는 이름의 창을 하나 생성하도록 하는 코드를 가지고 있다. 이 클래스를 FrameTest2라는 이름의 클래스가 상속하게 된다. 이 클래스는 객체를 생성하기 위해 이름을 매개변수로 전달하면, 이 매개변수를 창의 이름으로 하여 창을 생성하는 일을 한다. 일단 FrameTest2는 FrameTest를 상속받은 자식 클래스이기 때문에 생성자에서 super을 사용해 FrameTest를 불러온다. 그리고, 객체 생성 시 받아온 이름을 필드에 저장한다. 그리고 메서드 오버라이딩을 통해, 창의 이름을 필드 값으로 바꾼다. 마더 클래스는 Motherclass() 메서드를 사용해 창을 생성한다. 이를 수정해 사용하기 위해 자식 클래스도 MotherClass()를 메서드 오버라이딩하여 사용한다. 창의 이름을 정하는 메서드는 new JFrame()이다. 이 때 매개변수로 필드 값을 넣어줌으로써 내가 원하는 대로 창의 이름을 바꿀 수 있다.

4. 랜덤으로 그림 그리기

**package** Paint;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** java.util.Scanner;

**import** javax.swing.JFrame;

**class** MyFrame **extends** JFrame {

**public** MyFrame() {

**super**("My Frame");

setSize(300, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(**true**);

}

@Override

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

**int** num;

**super**.paint(g);

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

num = (**int**) (Math.*random*() \* 3);

g.setColor(

**new** Color((**int**) (Math.*random*() \* 255), (**int**) (Math.*random*() \* 255), (**int**) (Math.*random*() \* 255)));

**if** (num == 0)

g.drawOval((**int**) (Math.*random*() \* 200), (**int**) (Math.*random*() \* 200), (**int**) (Math.*random*() \* 50),

(**int**) (Math.*random*() \* 50));

**else** **if** (num == 1)

g.drawRect((**int**) (Math.*random*() \* 200), (**int**) (Math.*random*() \* 200), (**int**) (Math.*random*() \* 50),

(**int**) (Math.*random*() \* 50));

**else**

g.fillRect((**int**) (Math.*random*() \* 200), (**int**) (Math.*random*() \* 200), (**int**) (Math.*random*() \* 50),

(**int**) (Math.*random*() \* 50));

}

}

**public** **class** Paint {

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

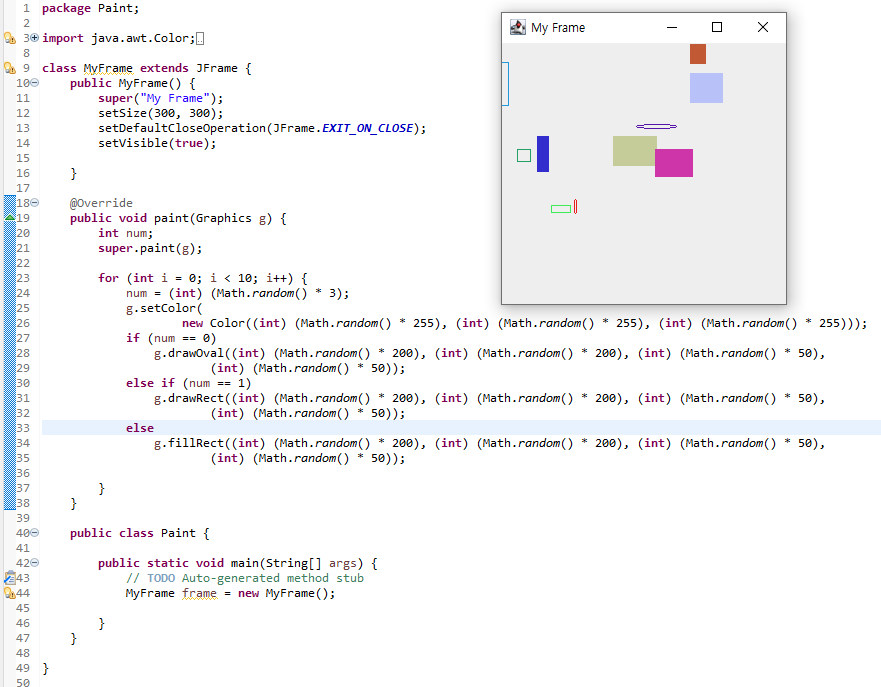
// **TODO** Auto-generated method stub

MyFrame frame = **new** MyFrame();

}

}

}

그림을 그리는 코드이다. 이 코드는 기존에 존재하던 클래스 JFrame을 상속 받아 MyFrame 클래스를 생성한다. 이 MyFrame은 랜덤하게 10개의 도형을 그리는 코드이다. 이 때 그림을 그린ㄴ 메서드는 paint인데, 사실 paint는 이미 JFrame에 존재하는 메서드이다. 따라서 메서드 오버라이딩을 통해 내가 원하는 형식으로 메서드를 수정한다. 먼저 랜덤하게 어떤 도형을 그릴지 정하기 위해 num 변수를 사용한다. num에 (int)(Math.random()\*3) 를 통해 0~2까지 랜덤한 값을 하나 저장한다. 이 때 num이 0일 경우 빈 동그라미를 그린다. Num이 1일 경우는 사각형을 그리고, 3일 경우는 색이 채워진 사각형을 그린다. 이 때 도형 색을 정하기 위해서 New Color을 사용해 색상을 정한다. 색상은 RGB 값을 사용해 지정할 수 있다. 따라서 값을 랜덤하게 모두 설정하고, 그림을 그릴 위치, 도형의 크기 모두 랜덤하게 설정함으로써 그림을 그린다. main에서는 클래스를 이용해 객체를 생성하기만 한다면 바로 그림을 그릴 수 있도록 메서드 오버라이딩을 한다.

5. 그림 그리기 2

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** javax.swing.JFrame;

**class** Shape {

**int** x, y;

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

}

}

**class** CircleInfo {

**int** x;

**int** y;

**int** size;

}

**class** Circle **extends** Shape {

**int** x;

**int** y;

**int** size;

**public** Circle(**int** x, **int** y, **int** size) {

**super**();

**this**.x = x;

**this**.y = y;

**this**.size = size;

}

@Override

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

g.setColor(Color.blue);

**int** num;

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

num = (**int**) (Math.*random*() \* 2);

g.setColor(

**new** Color((**int**) (Math.*random*() \* 255), (**int**) (Math.*random*() \* 255), (**int**) (Math.*random*() \* 255)));

**if** (num == 0)

g.drawOval(MyFrame.*CircleSize*[i].x, MyFrame.*CircleSize*[i].y, MyFrame.*CircleSize*[i].size,

MyFrame.*CircleSize*[i].size);

**else**

g.fillOval(MyFrame.*CircleSize*[i].x, MyFrame.*CircleSize*[i].y, MyFrame.*CircleSize*[i].size,

MyFrame.*CircleSize*[i].size);

}

}

}

**class** MyFrame **extends** JFrame {

**static** Circle[] *CircleSize* = **new** Circle[10];

**int** x;

**int** y;

**int** size;

**public** MyFrame() {

**super**("My Frame");

setSize(300, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(**true**);

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

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

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

size = ((**int**) (Math.*random*() \* 100));

*CircleSize*[i] = **new** Circle(x, y, size);

}

}

@Override

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

**super**.paint(g);

**for** (Circle obj : *CircleSize*)

obj.draw(g);

}

}

**public** **class** ShapeDraw {

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

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

JFrame MyScreen = **new** JFrame();

**new** MyFrame();

}

}

이 코드는 리스트에 저장된 원의 정보를 불러와 원을 그리는 코드이다. 먼저 이 코드는 크게 2가지 클래스가 상호작용 하면서 작동한다. 먼저 Shape 마더 클래스이다. 이 클래스는 x, y좌표를 저장하는 int 형 변수를 가지고 있고, 그림을 그리는 draw 메서드를 가지고 있었다. 우리는 원을 그릴 것이므로, 원의 x, y 좌표를 저장할 필요가 있다. 따라서 Shape 클래스를 상속 받아 Circle 클래스를 정의한다. 따라서 생성자를 만들 때 super을 통해 Shape 클래스를 불러옴과 동시에 매개변수로 x(x좌표), y(y좌표), size(반지름) 등을 받아와 해당 클래스의 필드에 넣어준다. 이렇게 하면 객체를 생성하였을 때 받아온 정보가 그대로 필드에 저장될 수 있게 만들 수 있다. 그 다음 우리는 그림을 그려야 한다. 그림을 그리는 것은 아까 Shape가 가지고 있던 메서드를 통해 할 수 있다. 다만 Circle는 확실히 원을 그릴 것이라고 정해져 있기 때문에 g.drawOval을 통해 원을 그릴 수 있게 된다. 이 때 랜덤한 원을 그리기 위해서 (**int**) (Math.*random*() \* 2)를 사용해 내가 무슨 원을 그릴 것인지를 정한다. 만약 num 값이 0이라면 빈 원을 그리고, 1이라면 속이 채워진 원을 그린다. 속이 비든 채워졌든 원의 색은 변경 가능하므로, new Color(R,G,B)에서 R,G,B를 마찬가지로 랜덤한 값으로 설정하여 다채로운 원을 그릴 수 있도록 코드를 작성한다. 원을 실제로 그릴 때에도 원의 크기는 저장한 값을 불러오는 방식을 사용해 그림을 그린다.

그렇다면 이 원의 정보는 어디에 저장되어 있는가? 우리는 캔버스 위에 그림을 그린다. 그림을 그리기 위해서 우리는 JFrame을 통해 새로운 창을 하나 만들 필요가 있다. 그래서 우리는 JFrame을 상속해 만든 MyFrame 클래스를 사용한다. MyFrame은 원의 정보를 같이 저장하고 있어야 하기 때문에 원에 대한 정보를 저장할 CircleSize, x, y, size 필드를 가지고 있어야 한다. 원의 정보 또한 마찬가지로 x,y,size이므로 Circle 클래스를 배열에 저장해야 한다. 생성자를 통해 JFrame의 기능을 사용한 창을 제작한다. 그리고, 랜덤한 원에 대한 값을 저장한다. 원은 0~299까지의 x,y 좌표, 0~99까지의 크기 정보를 가지고 있다. 이 값을 Circle 클래스의 매개변수로 넣어 새로운 객체를 생성하고, 이를 CircleSize에 추가한다. 이렇게 만들어진 CircleSize 배열은 위의 Circle 클래스에서 그림을 그릴 때 사용된다. 그리고 for each 문을 사용해 각각 저장된 값을 draw를 통해 그려준다면 main에서 Jframe을 새로 생성한 것 만으로 원을 랜덤하게 그릴 수 있게 된다.

