< JAVA PROGRAMMING LAB -5장: 클래스 심화 개념 >

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과

**1. 윈도우 setLocation메소드를 이용하고 Graphics객체를 이용하여 어떤차가 가장 빠른지 나타내는 프로그램을 만들어라.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목



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

**import** java.awt.Graphics;

**import** java.util.\*;

**class** Vehicle **extends** JFrame {

 **public** Vehicle (String s) {

 **super**(s);

 setSize(100,100);

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

 setVisible(**true**);

 }

 @Override

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

 **super**.paint(g);

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

 g.drawArc(30,70,20,20,0,360);

 g.drawArc(90,70,20,20,0,360);

 }

}

**class** Car **extends** Vehicle{

 **public** Car(String s) {

 **super**(s);

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

 }

 @Override

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

 **super**.paint(g);

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

 g.drawRect(40,35,50,15);

 g.drawRect(20,50,100,30);

 }

}

**class** Bus **extends** Vehicle{

 **public** Bus(String s) {

 **super**(s);

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

 }

 @Override

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

 **super**.paint(g);

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

 g.drawRect(20,40,100,40);

 g.drawRect(20,50,100,15);

 }

}

**class** Truck **extends** Vehicle{

 **public** Truck(String s) {

 **super**(s);

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

 }

 @Override

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

 **super**.paint(g);

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

 g.drawRect(20,40,70,40);

 g.drawRect(90,60,30,20);

 }

}

**public** **class** hi{

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

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

 Vehicle[] varray = **new** Vehicle[10];

 **int** x=100,yc=0;

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

 **if**(i<5) {

 varray[i]= **new** Car("Car");

 varray[i].setLocation(x,100+yc);}

 **else** **if**(i<8) {

 varray[i]= **new** Bus("Bus");

 varray[i].setLocation(x,100+yc);}

 **else** {

 varray[i]= **new** Truck("Truck");

 varray[i].setLocation(x,100+yc);}

 yc+=100;

 }

 **int**[] xarray=**new** **int**[10];

 **int**[] yarray= {100,200,300,400,500,600,700,800,900,1000};

 Random r=**new** Random();

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

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

 **int** ran=r.nextInt(10);

 x+=ran;

 varray[j].setLocation(x,yarray[j]);

 xarray[j]=x;

 }

 }

 **int** max= xarray[0];

 **for**(**int** i=1;i<10;i++) {

 **if**(max<xarray[i])

 max=i;

 }

 System.***out***.println("가장 빠른 차는"+max+"입니다.");

}

}

◇ 제목

◇ 제목

◇ 프로그램 소스

◇ 제목

◇ 제목

◇ 프로그램 소스

◇ 제목

◇ 제목

◇ 프로그램 소스

그림 그리는 클래스가 담긴 패키지인 java.awt를 가져오고, 윈도우를 만드는 클래스가 담긴 패키지인 swing을 가져왔다. Random함수를 쓰기 위해 util패키지도 가져왔다. 일단 Vehicle이란 클래스는 awt패키지의 Jframe클래스를 상속받는다. Super메소드를 이용해서 전달받은 문자열을 생성자로 보내서 윈도우의 제목을 정하고, setSize메소드를 이용하여 윈도우의 사이즈를 설정한다. setDefaultCloseOperation메소드를 이용하여 x표를 누르면 jvm에서도 완벽히 사라지게 만들었다. 마지막으로 윈도우를 우리 눈에 보이게 했다. 다음은 paint메소드이다. 이 메소드는 override된다. 그래픽 객체 g를 전달받고 슈퍼 클래스의 paint메소드에 g를 전달하였다. 이 과정은 슈퍼 클래스의 paint메소드에서 설정하는 부분으로, 그림을 생성한 윈도우에 그리겠다는 소스로 보인다. 이후 g의 setColor메소드를 이용하여 그림 그릴 색을 정하고 drawArc메소드를 이용하여 호를 그린다. 인수의 0,1인덱스 자리엔 x,y좌표가, 2,3인덱스 자리엔 반지름이, 4,5인덱스 자리엔 호의 각도가 들어간다. 여기까지가 Vehicle클래스이다. 다른 car,bus,truck클래스는 이 Vehicle클래스를 상속받는다. 이렇듯 관계가 비슷한 클래스들에서 공통적으로 사용하는 필드나 메소드는 하나의 공통 클래스에서 묶고 상속받아 사용하는 것이 편하다. Car클래스에서는 문자열을 전달받아 Vehicle로 넘기고 다시 paint메소드를 오버라이딩한다. Vehicle의 paint메소드에 g를 넘기고 그림 그릴 색을 정하고 이번엔 drawRect를 이용하여 사각형을 그린다. 이 사각형은 자동차의 몸통 부분이 된다. 사실 Vehicle클래스에서 drawArc를 이용하여 바퀴는 미리 그려진 상태이니, 또 다시 바퀴를 만들 필요가 없는 것이다. DrawRect 인수의 0,1 인덱스 자리에는 x,y좌표가, 2,3 인덱스 자리에는 가로,세로 길이가 들어간다. Truck과 Bus클래스도 Car클래스와 비슷한 내용이 들어가겠다. 이제 main메소드를 살펴보자. 먼저 Vehicle객체를 담을 수 있는 객체 배열 varray를 크기가 10으로 만들었다. 이후 for문을 이용하여 varray의 0~4인덱스에는 Car객체가, 5~7인덱스에는 Bus객체가, 8~9인덱스에는 Truck객체가 저장되게 하였다. 이 서브 클래스 객체들은 슈퍼 클래스를 포함하고 있기 때문에 varray객체 배열에 들어갈 수 잇는 것이다. For문 이전에 만든 x,yc 변수는 객체를 저장 후 setLocation할 때 좌표로 쓰이게 된다. X좌표는 100으로 동일하지만 yc는 반복할 때마다 100씩 증가해서 각 윈도우들은 y좌표만 다른 것을 알 수 있다. 이후 xarray정수 배열을 만든다. Yarray정수 배열도 만드는데 앞에 만든 윈도우들(객체를 만들면서 차가 그려진 윈도우가 자동으로 만들어진다)의 y좌표를 다 포함하고 있는 배열이다. 다음에는 for문 중첩과 랜덤 함수를 이용하여 윈도우를 움직이게 할 것이다. 안쪽 for문을 통해 윈도우0부터 9까지 위치를 이동시키고 바깥쪽 for문을 통해 이를 10번 반복시킨다. 위치를 이동시킬 때마다 xarray[j]에 값이 들어가게 된다. For문이 다 끝난 최종적인 상태로 보면, xarray[0]에는 윈도우0의 x좌표가,... xarray[9]에는 윈도우9의 x좌표가 들어간다. 이제 xarray배열 요소에서 최댓값을 골라내 출력만 하면 된다. Max를 xarray[0]으로 두고 xarray[1]부터 [9]까지 비교하여 max보다 비교한 것이 더 크면 max를 바꾸는 작업을 반복하면 되겠다. 이로써 윈도우0부터 9까지 각각 차 그림이 담기면서 이동하는 모습을 사용자가 볼 수 있고 가장 빨리 도착한 윈도우(차)를 출력할 수 있다.

◇ 제목

◇ 제목

◇ 프로그램 설명

< JAVA PROGRAMMING LAB -5장: 클래스 심화 개념 >

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과

**2. @가 사용자고, G가 골드이고 M이 몬스터인 게임을 만들어라.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목



\*

**import** java.util.\*;

**abstract** **class** Sprite{

 **int** x=3, y=3;

 **abstract** **void** move(String c);

 **abstract** **void** newset();

}

**class** Main **extends** Sprite{

 **void** move(String c) {

 **if**(c.contentEquals("w")) --x; // 위쪽

 **else** **if** (c.contentEquals("s")) ++x; // 아래 쪽

 **else** **if**(c.contentEquals("d")) ++y; // 오른 쪽

 **else** **if**(c.contentEquals("a")) --y; // 왼 쪽

 **if**(x==-1) ++x;

 **else** **if**(x==10) --x;

 **if**(y==-1) ++y;

 **else** **if**(y==20) --y;

 }

 **void** newset() { // 새롭게 사람 위치 정함

 x=3; y=3;}

}

**class** Monster **extends** Sprite {

 **public** Monster() {

 **do** {

 x=(**int**)(Math.*random*()\*10);

 y=(**int**)(Math.*random*()\*(10+10));

 }**while**(x==3); // 몬스터가 사람이랑 위치 다르게!

 }

 **void** move(String c) {

 x +=(Math.*random*()-0.5)>0? 1:-1;

 y +=(Math.*random*()-0.5)>0? 1:-1;

 **if**(x==-1) ++x;

 **else** **if**(x==10) --x;

 **if**(y==-1) ++y;

 **else** **if**(y==20) --y;

 }

 **void** newset() { // 새롭게 몬스터 위치 정함

 **do** {

 x=(**int**)(Math.*random*()\*10);

 y=(**int**)(Math.*random*()\*(10+10));

 }**while**(x==3); // 몬스터가 사람이랑 위치 다르게!

 }

}

**class** Gold **extends** Sprite {

 **public** Gold(Sprite mon) {

 **do** {

 x=(**int**)(Math.*random*()\*10);

 y=(**int**)(Math.*random*()\*(10+10));

 }**while**( ((x==3)&&(y==3)) || ((x==mon.x)&&(y==mon.y)) ); // 골드가 사람,몬스터랑 위치 다르게!

 }

 **void** move(String c) {

 // 아무 것 도 안 함, 호출도 안 됨

 }

 **void** newset() {

 **do** {

 x=(**int**)(Math.*random*()\*10);

 y=(**int**)(Math.*random*()\*(10+10));

 }**while**((x==3)&&(y==3)); // 골드가 사람이랑 위치 다르게!

 }

}

**class** hi{

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

 Scanner input=**new** Scanner(System.***in***);

 String[][] board=**new** String[10][20]; // 사람,몬스터,골드 위치시킬 배열 생성

 Sprite main,mon,gold; // 사람,몬스터,골드 객체 생성

 main=**new** Main();

 mon= **new** Monster();

 gold= **new** Gold(mon);

 **int** x=main.x, y=main.y; // 사람,몬스터,골드 좌표 저장

 **int** mx=mon.x, my=mon.y;

 **int** gx=gold.x, gy=gold.y;

 board[x][y]="@"; // 좌표를 이용해서 배열에 저장

 board[mx][my]="M";

 board[gx][gy]="G";

 **int** count=0; // 골드 먹은 횟수 변수

 **while**(**true**) {

 System.***out***.print("######################");

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

 **if**(i==0)

 System.***out***.print("\n");

 **for**(**int** j=-1;j<21;j++) {

 **if**(j==-1)

 System.***out***.print("#");

 **else** **if**(j==20)

 System.***out***.print("#");

 **else** **if**(j>=0 && j<20) {

 **if**(board[i][j]!=**null**)

 System.***out***.print(board[i][j]);

 **else**

 System.***out***.print(" ");

 }

 }

 System.***out***.print("\n");

 }

 System.***out***.print("######################");

 System.***out***.print("\n 골드 개수: "+count+"\n");

 **if**((x==mx)&&(y==my)) { // 사람이랑 몬스터랑 위치 같으면 게임 오버

 System.***out***.print("\n게임 오버!!");

 **break**;

 }

 **else** **if**((x==gx)&&(y==gy)) { // 사람이랑 골드랑 위치 같으면 카운트 증가

 count++;

 **if**(count==5) {

 System.***out***.print("->->->->->");

 System.***out***.print("\n 골드를 "+count+"개 획득하였습니다!! "+"("+count+"/5)\n");

 System.***out***.print("\n게임 승리!!");

 **break**;}

 **else** {

 System.***out***.print("->->->->->");

 System.***out***.print("\n 골드를 "+count+"개 획득하였습니다!! "+"("+count+"/5)\n");

 System.***out***.print("\n (다음 골드를 획득해보세요!!)\n");

 main.newset();

 mon.newset();

 **do** {

 gold.newset();

 }**while**( (gold.x==mon.x)&&(gold.y==mon.y) ); // 골드 위치가 몬스터 위치랑 같으면 다시 세팅

 gx=gold.x; gy=gold.y;

 board[gx][gy]="G";

 }

 }

 **else** { // 다 아니면, 사람과 몬스터 이동

 System.***out***.print("\n 위쪽(w), 왼쪽(a), 아래쪽(s), 오른쪽(d): ");

 String c=input.next();

 main.move(c); // 사람,몬스터 이동하기

 **do** {

 mon.move(c);

 }**while**( (mon.x==gold.x)&&(mon.y==gold.y) ); // 몬스터가 이동 중 골드랑 위치 같으면 다시 이동

 }

 board[x][y]=**null**; // 기존 사람,몬스터 지우기

 board[mx][my]=**null**;

 x=main.x; // 사람,몬스터 좌표 다시 저장

 y=main.y;

 mx=mon.x;

 my=mon.y;

 board[x][y]="@"; // 좌표를 이용해서 배열에 다시 저장

 board[mx][my]="M";

 }

 }

}

◇ 제목

◇ 제목

◇ 프로그램 소스

◇ 제목

◇ 제목

◇ 프로그램 소스

◇ 제목

◇ 제목

◇ 프로그램 소스

◇ 제목

◇ 제목

◇ 프로그램 소스

Sprite는 추상 클래스이다. 추상 클래스는 추상 메소드를 포함하고 있는 클래스이고 추상 메소드는 몸체가 없는 메소드를 말한다. 이러한 클래스와 메소드를 쓰려면 abstract라는 키워드를 사용해야만 한다. 이러한 추상 클래스는 관계가 비슷한 클래스들이 공통의 이름으로 쓰는(공통적으로 쓴다고 보기는 애매함) 필드,메소드를 포함한다. 일반 클래스에서 추상 클래스를 상속받아 move,newset같은 추상 메소드를 구현(메소드 내용 정의)하는 식으로 사용된다. 필자는 main클래스를 사용자, monster클래스를 몬스터, gold클래스를 골드라고 지정해서 문제를 풀었는데, 이러한 개체들이 move메소드를 통해서 움직이기는 하겠지만 같은 방식으로 움직이지는 않을 것이다. 즉 move라는 같은 메시지를 주었지만 객체의 종류(클래스)가 달라 기능을 다르게 하는 ‘객체의 다형성’을 잘 볼 수 있는 예시가 되겠다. 추상 클래스를 상속받아 사용할 때는 추상 메소드를 빠짐없이 구현해야 된다는 점을 잊어서는 안된다.

Sprite 추상 클래스부터 살펴보겠다. X,y변수는 개체의 위치라고 생각하고 move는 개체가 움직이게 하는 추상 메소드, newset은 개체의 위치를 다시 세팅하는 추상 메소드로 보면 되겠다.

 Main클래스에서는 Sprite클래스를 상속받았다. Move메소드를 아래와 같이 구현한다. 만약 문자열 w를 전달받았으면 x를 1내리고, s를 전달 받았으면 1올린다. d를 전달받았으면 y를 1올리고 a를 전달받았으면 y를 1내린다. 만약 w가 전달받아서 x가 1내려가고 x=-1이 되었다면 다시 x를 1올린다.(w를 전달받았다는 것을 굳이 조건문에 안 써도 x가 0에서 -1이 되는 경우가 w가 전달되는 경우밖에 없음) 그렇지 않고 s를 전달 받아서 x가 10이 되면 x를 1내린다. 그렇지 않고 a를 전달받아서 y가 -1이 된다면 y를 1올리고, d를 전달받아서 y가 20이 된다면 y를 1내린다. 이런식으로 x가 -1,10, y가 -1,20이 되는 경우를 아예 없앤다. Newset메소드는 x,y를 다시 3으로 저장하였다.

이제 Monster클래스를 살펴보겠다. 이또한 Sprite클래스를 상속받는다. 이번엔 생성자가 쓰인다. Math.random()의 반환 값은 0.0부터 0.99까지 랜덤이다. 즉 Math.random에 10을 곱하고 정수형으로 형 변환을 하면 0부터 9까지 랜덤 수가 나오게 된다. 비슷한 이치로 Math.random에 10을 곱하는 것을 2번하고, 곱한 2개를 더한다면 부터 18안의 랜덤 수가 나오게 된다. 위의 방식으로 랜덤 값들을 x,y에 각각 넣는 것을 x=3이 될 때 반복한다. 즉 x가 3이 되지 않을 때(사용자의 x위치랑 다를 때) x,y값이 구해지게 될 것이다. Move메소드는 main클래스와 다르게 구현되는데 이번에는 전달받은 문자열을 쓰지 않는다. Math.random을 써서 50퍼 확률로 x을 1올리거나 1내린다. Y도 마찬가지이다. 여기서도 x가 -1,10, y가 -1,20이 되는 경우를 아예 없앤다. Newset메소드도 main과 다르게 구현되는데, monster생성자와 사실 같은 기능을 한다.

Gold클래스를 살펴보겠다. 마찬가지로 Sprite클래스를 상속받는다. 생성자에서는 몬스터와 같은 방식으로 랜덤으로 x,y위치가 정해진다. 그런데 이번엔 위치를 사람과 몬스터 위치와 모두 다르게 세팅시켰다. 골드는 위치가 지정되면 고정되야 하므로 move를 구현하지만 아무 것도 작성되지 않는다. Gold클래스의 move는 호출도 되지 않을 것이다. Newset메소드도 구현되는데 생성자와 같은 소스가 담긴다.

드디어 main메소드를 살펴보겠다. 스캐너 객체를 만들고 input객체로 쓰기로 하였다. 문자열을 담을 수 있는 2차원 배열을 만드는데 크기를 행 기준 10, 열 기준 20으로 만들었다. Sprite객체를 참조할 수 있는 참조변수 main,mon,gold를 만들고 각 참조변수에 Main,Monster,Gold객체를 가리키게 하였다. 이 객체들은 Sprite를 상속받는 서브 클래스들이니 Sprite클래스를 포함한다. 고로 Sprite객체를 참조할 수 있는 참조변수를 써도 된다. 이후 x,y는 main의 x,y로 정의하고 mx,my는 mon의 x,y로 정의하고 gx,gy는 gold의 x,y로 정의한다. 여기서 x,y가 들어간 변수들은 행 인덱스, 열 인덱스로 본다. 앞전 클래스들에서 x,y의 값 제한을 둔 이유도 행 인덱스 최대가 9이고 열 인덱스 최대가 19이기 때문이다. 이후 2차원 배열의 x인덱스(에 위치한) 값,y인덱스 값은 @로 저장하고 mx인덱스 값,my인덱스 값은 M으로 저장하고 gx,gy인덱스 값은 G로 저장한다. 즉 이 배열에 (사람,몬스터,골드)가 (@,M,G) 형태로 위치되게 된다..

다음은 게임이 끝날 때까지 반복문이 진행된다. 일단 틀을 #으로 만들고 i는 0부터 9까지 1씩 증가하고 중첩으로 j는 -1부터 20까지 증가한다. 만약 j가 -1이나 20이면 #을 출력해서 틀을 만들고, 0부터 19면 2차원 배열의 i인덱스(에 위치한) 값,j인덱스 값을 출력하게 되겠다. 만약 I,j인덱스 값이 null이면 출력하지 않고 띄어쓰기를 출력한다. 이 방식이 게임 형태를 만드는 부분이다. 다음에는 if문을 써서 만약 x랑 mx랑 같고 y랑 my랑 같으면 게임을 종료시킨다. 그렇지 않고 x는 gx랑 같고 y는 gy랑 같으면 count(골드 먹은 개수)를 1증가시킨다. Count가 5가 되면 게임 승리라고 출력하고 게임을 종료하고 그렇지 않으면 골드를 몇 개 획득했는지 알려주고 각 객체의 newset메소드를 이용하여 게임을 리셋팅한다. 만약 골드를 리셋팅햇는데 몬스터 위치랑 같으면 다시 세팅한다. 만약 x가 gx,mx랑 다르고 y도 gy,my도 다르다면 각 객체의 move메소드를 이용해서 개체의 위치를 이동시키고 x,mx,y,my도 다시 정의한다. 다시 정의한 것을 토대로 @와 M도 다시 저장한다. 물론 이전 @와 M은 지워야 될 것이다.

◇ 제목

◇ 제목

◇ 프로그램 설명

◇ 제목

◇ 제목

◇ 프로그램 설명