**JAVA LAB**

20234014 컴퓨터 공학과 이은정

1. COUNTER 코드

**import** java.util.Scanner;

**class** Counter {

**int** counter = 0;

String getcounter() {

**return** "Counter Value: " + counter;

}

**void** up() {

counter++;

System.***out***.print("up 메소드 호출 후 " + getcounter() + "\n");

}

**void** down() {

counter--;

System.***out***.print("down 메소드 호출 후 " + getcounter() + "\n");

}

**void** reset() {

counter = 0;

System.***out***.print("reset 메소드 호출 후 " + getcounter() + "\n");

}

}

**public** **class** Countertest {

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

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

Counter c;

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

c = **new** Counter();

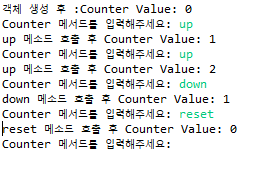
System.***out***.print("객체 생성 후 :" + c.getcounter() + "\n");

**do** {

String input;

System.***out***.print("Counter 메서드를 입력해주세요: ");

input = sc.next();



**if** (input.equals("up")) {

c.up();

} **else** **if** (input.equals("down")) {

c.down();

} **else** **if** (input.equals("reset")) {

c.reset();

} **else** **if** (input.equals("q")) {

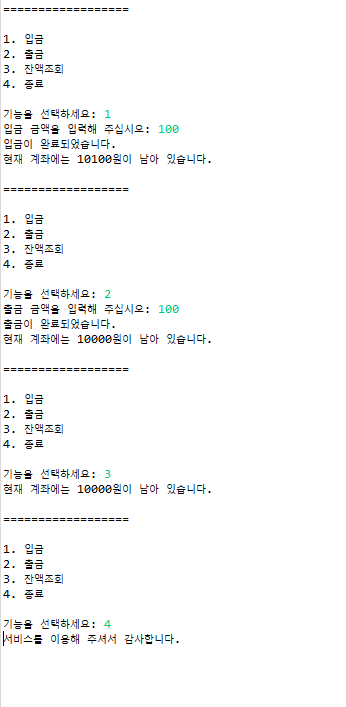
**break**;

}

} **while** (**true**);

}

}

COUNTER 클래스는 메인에서 값을 입력 받아 외부 데이터를 변화시킨다. 이 코드에서는 MAIN에서 scanner을 통해 값을 입력 받은 이후에 이를 class 메서드에 전달을 해준다. Counter이라는 class에서 up, down, reset 메서드를 통해 counter 변수를 변화시킬 수 있다. Main class에서 new scanner(System.in) 메서드를 통해 값을 입력 받는다. 이후에는 Counter c를 통해 c에 Counter의 주소를 저장한다. 이후에 new Counter을 통해 c라는 클래스를 생성한다. 이후부터는 c의 메서드를 사용할 수 있다. Counter 변수 값 변화는 한 번만 일어나는 것이 아니므로 do-while문을 사용한다. 만약 input을 입력 받는다면 c.up()을, down을 입력 받으면 c.down()을 reset을 입력받으면 c.reset을 q를 누르면 아예 do-while문을 벗어난다. 각각의 메서드를 누름으로써 Counter++, counter—을 통해 counter 변수 값을 변화시키고, getcounter 메서드를 통해 그 값을 불러온다. 마지막으로 reset은 counter변수를 0으로 할당한다. 이 코드를 작성할 때 주의해야 할 점은 문자열 비교는 ==가 아니라 c언어처림 c.equals같은 특정 메서드를 사용해야 한다는 것이다.

2. ATM

**class**  Account{

**int** money;

**int** getAcc{

return money;

}

**void** setAcc(**int** n) {

money += n;

}

**void** withdraw(**int** n) {

n = -n;

setAcc(n);

}

}

**public** **class** ATM {

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

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

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

**int** input;

Account a;

a = **new** Account();

**do** {

System.***out***.println("==================\n");

System.***out***.println("1. 입금");

System.***out***.println("2. 출금");

System.***out***.println("3. 잔액조회");

System.***out***.println("4. 종료\n");

System.***out***.print("기능을 선택하세요: ");

input = sc.nextInt();

**if** (input == 1) {

**int** deposit;

System.***out***.print("입금 금액을 입력해 주십시오: ");

deposit = sc.nextInt();

a.setAcc(deposit);

System.***out***.println("입금이 완료되었습니다.");

System.***out***.println("현재 계좌에는 " + a.getAcc() + "원이 남아 있습니다.\n");

}

**else** **if** (input == 3) {

System.***out***.println("현재 계좌에는 " + a.getAcc() + "원이 남아 있습니다.\n");

}

**else** **if** (input == 2) {

**int** withdraw;

System.***out***.print("출금 금액을 입력해 주십시오: ");

withdraw = sc.nextInt();

a.withdraw(withdraw);

System.***out***.println("출금이 완료되었습니다.");

System.***out***.println("현재 계좌에는 " + a.getAcc() + "원이 남아 있습니다.\n");

}

**else** **if** (input == 4) {

System.***out***.println("서비스를 이용해 주셔서 감사합니다.");

**break**;

}

} **while** (**true**);

}

}

ATM 코드는 1. 내 잔금 확인, 2. 출금, 3. 입금의 기능이 있다. 먼저 main 함수에서 new Scanner(System.input)을 통해 내가 어떤 동작을 할 것인지 정한다. 1을 누르면 입금, 2를 누르면 출금, 3은 잔액조회, 4는 종료이다. 이 행동은 계석 반복하므로 do-while문을 사용한다. 4를 누르면 break;를 사용해 빠져나간다. 각각의 숫자를 입력하면 입력한 값에 따라 Account 클래스의 메서드를 수행한다. 이 클래스에는 크게 setAcc, getAcc, withdraw 3개의 기능이 있다. setAcc는 내 잔액을 저장하는 money 변수의 값을 변화시키는 기능을, getAcc는 잔액을 return하는 기능을, withdraw는 main에서 받아온 값을 -값으로 변환하여 setAcc로 값을 보내는 역할을 한다. main에서 new Counter라는 새 클래스를 생성한다. 이후에 내가 무슨 행동을 할 지 input값으로 입력을 받는다. 이 때 1,2번을 누를 시에는 지역 변수를 사용해 얼마나 값을 입금하고 출금할지 다시 입력을 받는다. 이후에 입금은 그냥 setAcc로, 출금은 withdraw로 그 값을 보낸다. 입금은 ‘잔액+입금액’으로 출금은 ‘잔액+(-입금액)’이라고 볼 수 있으므로 이렇게 코드를 만들었다. 마지막으로 1,2,3 모두 마지막에는 getAcc로 잔액을 return해 그 값을 main으로 보낸다.

3. Circle

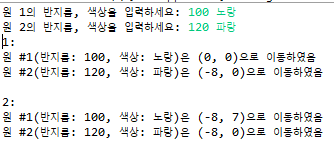
**import** java.util.Scanner;

**class** Circleset{

String name;

**int** x;

**int** y;

 **int** r;

**int** num;

String color;

**void** move(**int** n, **int** m){

**int** dx, dy;

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

**if** (dx %2==0) x = -n;

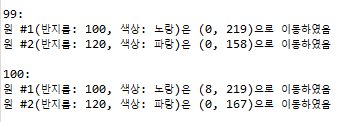
**else** x+=n;

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

**if** (dy %2==0) m = -m;

**else** y+=m;

System.***out***.print("원 #"+num+"(반지름: "+r+", 색상: "+color+")은 ("+x+", "+y+")으로 이동하였음\n");

 }

**void** ran() {

**int** dx, dy;

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

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

move(dx, dy);

}

}

**public** **class** circle {

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

**int** x,y;

String color;

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

Circleset c1;

Circleset c2;

c1 = **new** Circleset();

c2 = **new** Circleset();

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

System.***out***.print("원 1의 반지름, 색상을 입력하세요: ");

c1.x = (**int**)Math.*random*()\*100;

c1.y = (**int**)Math.*random*()\*100;

c1.r = sc.nextInt();

c1.color = sc.next();

c1.num = 1;

System.***out***.print("원 2의 반지름, 색상을 입력하세요: ");

c2.x = (**int**)Math.*random*()\*100;

c2.y = (**int**)Math.*random*()\*100;

c2.r = sc.nextInt();

**while**(c2.r==c1.r)

{

System.***out***.print("반지름의 값이 c1과 동일합니다.\n다른 값을 입력해주십시오: ");

c2.r = sc.nextInt();

}

c2.color = sc.next();

c2.num = 2;

**for** (**int** i = 0; i<100 ; i++) {

System.***out***.print(i+1+": \n");

c1.ran();

c2.ran();

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

}

}

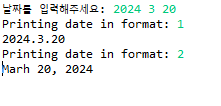
}

위 코드는 원의 색상, 반지름을 입력 받고 랜덤으로 위치를 이동시키는 코드이다. 먼저 메인 코드에서 랜덤으로 각각의 원의 x, y좌표를 정하고, Scanner을 통해 색상, 반지름을 입력 받는다. 이를 c1, c2라는 Circleset 클래스에 의해 만들어진 인스턴스의 각 값에 저장한다. 이후에 100번 for문을 통해 랜덤으로 각 원을 이동시킨다.

Circleset에 사용하는 변수는 크게 5가지이다. Num, 반지름, x, y , 색상이다. 이 때 num 값은 내가 입력 받지 않고 인스턴스 생성시에 내가 만들어준다. 이를 통해 하나의 함수를 사용해도 추후에 원 #num값을 제대로 출력할 수 있게 했다. 반지름 값은 nextInt를 통해 입력 받고, x, y값은 Math.random()를 통해 랜덤으로 지정한다. 이 값은 항상 정수형만 사용하기 때문에 앞에 (int)를 붙여준다. 이 값을 모두 입력 받은 후에는 ran 함수를 호출한다. ran함수는 x, y가 얼만큼 변화할지 그 값을 정하는 함수이다. 마찬가지로 Math.random()함수를 통해 값을 정해 이를 move함수로 이동시킨다. move에서는 그 값을 받아 값을 이동시킨다. 이 때 x, y 좌표는 증가할 수도 감소할 수도 있다. 이는 (int)Math.random()\*2를 통해 0, 1 둘 중 랜덤한 값을 받아 감소 증가를 결정한다. 만약 2로 나눈 값이 0이면 음수로 1이면 양수로 처리하여 각각의 값에 더한다. 이후에 print 메서드를 통해 반지름, 색상, x, y좌표를 출력하면 해당 코드는 잘 작동하게 된다.

4. date 출력

**class** DatePrint {

**private** **int** year;

**private** **int** month;

**private** **int** day;

**void** setDate(**int** a, **int** b, **int** c) {

year = a;

month = b;

day = c;

}

**void** p1() {

System.***out***.print(year + "." + month + "." + day + "\n");

}

**void** p2() {

String[] Month = { "January", "Febrary", "Marh", "April", "May", "June", "July", "Agust", "September",

"October", "November", "December" };

System.***out***.print(Month[month - 1] + " " + day + ", " + year + "\n");

}

}

**public** **class** Date {

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

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

**int** year, month, day;

**int** input;

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

DatePrint d;

d = **new** DatePrint();

System.***out***.print("날짜를 입력해주세요: ");

year = sc.nextInt();

month = sc.nextInt();

day = sc.nextInt();

d.setDate(year, month, day);

**for** (**int** i = 0; i < 2; i++) {

System.***out***.print("Printing date in format: ");

input = sc.nextInt();

**if** (input == 1) {

d.p1();

} **else** **if** (input == 2) {

d.p2();

}

}

}

}

날짜를 입력하면 이를 변형 출력하는 코드를 작성한다. Main 클래스에서 연, 월, 일을 입력 받고, 이를 어떻게 출력할지를 고른다. 만약 1을 입력 받으면 p1 메서드를, 2를 입력 받으면 p2 메서드를 실행한다. 이는 DatePrint클래스에서 실행할 수 있다. 이 클래스를 실행하기 위해서는 setDate 메서드를 통해 연, 월, 일을 전달한다. 만약 p1 메서드를 실행할 시, 각 변수의 순서를 바꿔 출력해준다. P2 메서드를 실행 시 리스트에서 내가 입력한 month에 해당하는 값을 출력할 수 있도록 한다. 입력 받고, 전달 받은 값이 int형이기 때문에 해당 숫자-1 번째의 리스트 값을 출력하면 리스트에서 월을 출력할 수 있다.

5. 화면 이동시키기

**package** race;

**import** javax.swing.JFrame;

**class** move {

**int** setloc(**int** n) {

**int** dx;

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

**return** n + dx;

}

}

**public** **class** race {

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

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

JFrame f1 = **new** JFrame("Frame Test1");

f1.setSize(300, 200);

f1.setLocation(30, 20);

f1.setVisible(**true**);

f1.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

JFrame f2 = **new** JFrame("Frame Test2");

f2.setSize(300, 200);

f2.setLocation(30, 300);

f2.setVisible(**true**);

f2.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

move mv;

mv = **new** move();

**int** x1, x2;

**try** {

**do** {

x1 = mv.setloc(f1.getX());

x2 = mv.setloc(f2.getX());

**if** (f1.getX() >= 1400) {

System.***out***.print("Test1의 승리입니다.");

**break**;

}

f1.setLocation(x1, 20);

Thread.*sleep*(20);

**if** (f2.getX() >= 1400) {

System.***out***.print("Test2의 승리입니다.");

**break**;

}

f2.setLocation(x2, 300);

Thread.*sleep*(20);

} **while** (f1.getX() <= 1400 || f2.getX() <= 1400);

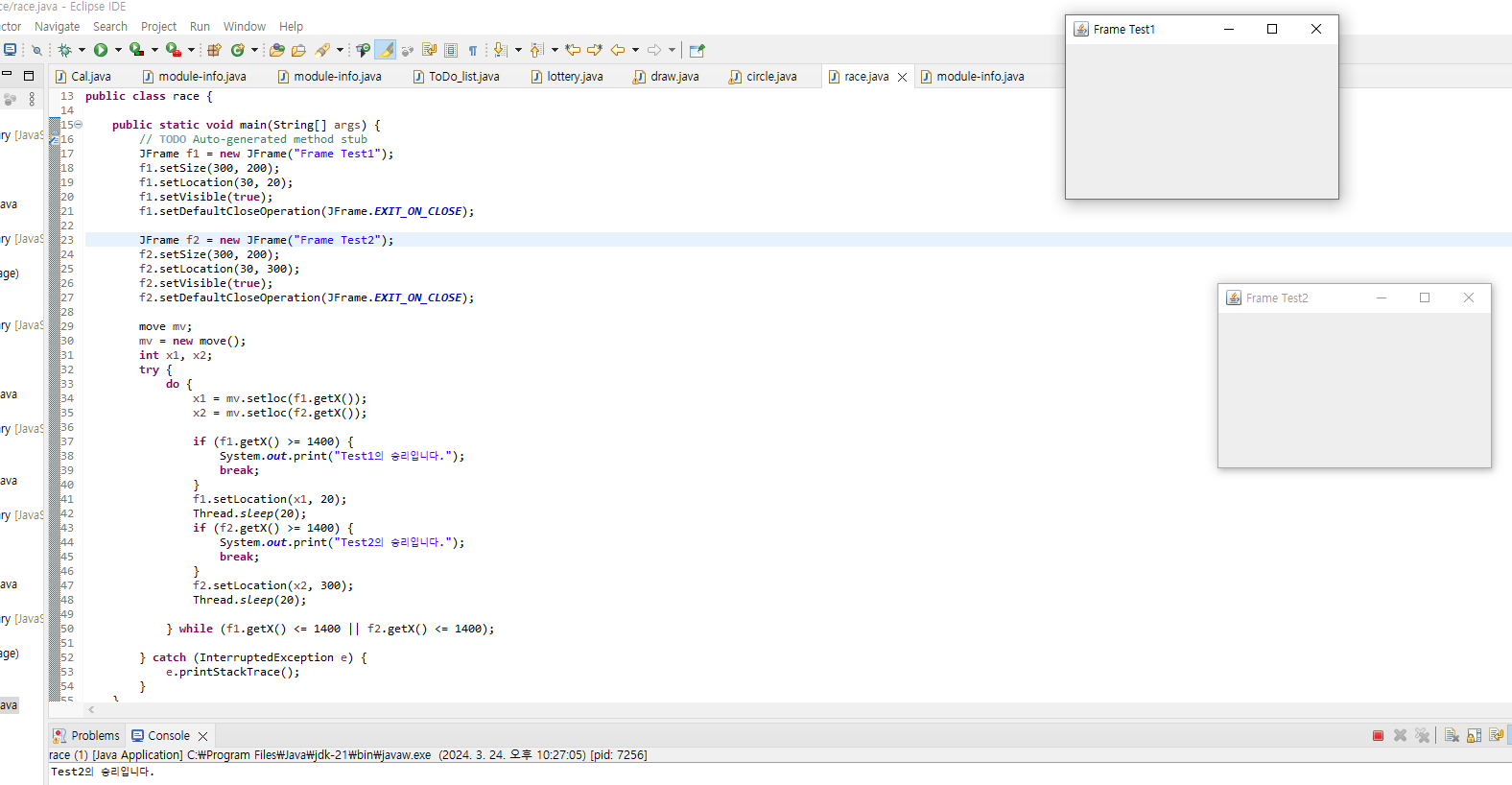
} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}



이 코드는 창 2개를 생성해 경주를 하는 게임이다. 우선 테스트 1, 테스트 2라는 이름의 창 2개를 만든다. eclipse에서는 지정된 몇 개의 라이브러리를 제외하고는 따로 module에서 임포트를 해주어야 한다. 따라서 module 파일에서

**requires** java.desktop;

**requires** java.sql;

코드를 실행해 창을 생성할 수 있게 한다. 화면에서 창이 움직이는 것을 볼 수 있게끔 하기 위해 Thread.sleep을 사용했다. for문을 사용해 f1 혹은 f2의 x좌표가 1400 미만일때만 반복할 수 있게끔 한다. F1, f2의 x좌표를 move 클래스의 setloc 메서드에 넘겨주면 기존의 x좌표+랜덤한 값(0~49; int 처리를 해줌)을 더한 값을 다시 리턴해준다. main에서는 setLocation을 통해 리턴 값을 x 좌표에 넣어주어 창의 위치를 수정할 수 있다. 만약 f1 혹은 f2의 x좌표가 1400이상이 된다면 각각의 창이 이겼다는 문구를 출력한 이후 break문을 통해 반복문을 종료한다.