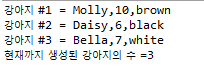
< JAVA PROGRAMMING LAB -4장: 객체 활용하기>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과



**1. 강아지를 나타내는 클래스 Dog를 작성하고 정적변수 count를 이용하고 객체를 이용해서 강아지들을 출력하라.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

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

**class** Dog{

String name;

**int** age;

String color;

**static** **int** *num*;

**public** Dog(String n,**int** a,String c) {

name=n; age=a; color=c;

*num*++;

}

}

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

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

Dog dog1=**new** Dog("Molly",10,"brown");

Dog dog2=**new** Dog("Daisy",6,"black");

Dog dog3=**new** Dog("Bella",7,"white");

System.***out***.println("강아지 #1 = "+dog1.name+","+dog1.age+","+dog1.color);

System.***out***.println("강아지 #2 = "+dog2.name+","+dog2.age+","+dog2.color);

System.***out***.println("강아지 #3 = "+dog3.name+","+dog3.age+","+dog3.color);

System.***out***.println("현재까지 생성된 강아지의 수 ="+Dog.*num*);

}

}

◇ 제목

◆ DESCRIPTION

◇ 제목

◆ 프로그램 설명

◇ 프로그램 소스

◆ DESCRIPTION

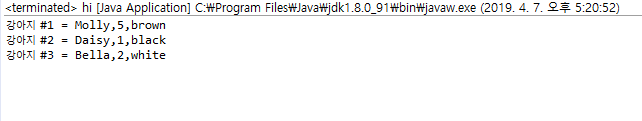
먼저 강아지 정보가 담긴 클래스를 만들었다. 이름,나이,색,번호를 저장할 수 있는 필드를 먼저 생성하고 생성자를 만들어서 필드에 적용이 되록 하였다.

그리고 main메소드에서는 dog1,dog2,dog3객체를 만들었다. 객체를 만들 때는 필드에 값을 수정하는 것부터 시작된다. 즉 객체를 만드는 도중 num필드의 값을 수정하지는 않았으니 dog1의 num값은 ++된 1, dog2의 num값은 ++이 2번 되서 2가 저장된다. 클래스 필드에 값을 넣는 것부터 시작되니 객체를 만들면서 강아지에 대한 정보는 다 다르게 주었다. 이후 dog1객체의 이름,나이,색을 가져와서 출력하고 마찬가지로 다른 객체들의 정보도 출력하였다.

< JAVA PROGRAMMING LAB -4장: 객체 활용하기>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과



**2. younger메소드를 이용하면 나이가 5살이 어려진다고 한다. 메소드를 이용해서 어려진 강아지들을 출력하라.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

앞의 소스에서 younger이라는 메소드가 추가되었다. 이 메소드는 static으로 정의되어있어서 Dog클래스의 필드에 접근할 수 없다. 그러나 main함수에서 참조변수 dog1을 넘겨주면 말이 달라진다. dog1은 객체라고 생각해서 문제를 풀지만 사실 객체를 가리키는 주소이다. dog1.color을 굳이 원래대로 해석하면, ‘dog1참조변수(주소)가 가리키는 객체에 있는 color필드’라고 하는 것이다. 응용해서 객체를 가리키는 참조 변수를 넘기면 객체의 주소를 넘긴거고 이 주소를 가지고 younger메소드에서 필드를 자유롭게 접근할 수 있다. 이 소스에서는 강아지 객체의 나이를 5살 줄인 것을 알 수 있다.

최종적으로는 나이가 준 강아지들의 정보를 출력한다.

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

**class** Dog{

String name;

**int** age;

String color;

**public** **static** Dog younger(Dog obj) {

obj.age-=5;

**return** obj;

}

**public** Dog(String n,**int** a,String c) {

name=n; age=a; color=c;

}

}

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

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

Dog dog1=**new** Dog("Molly",10,"brown");

Dog.*younger*(dog1);

Dog dog2=**new** Dog("Daisy",6,"black");

Dog.*younger*(dog2);

Dog dog3=**new** Dog("Bella",7,"white");

Dog.*younger*(dog3);

System.***out***.println("강아지 #1 = "+dog1.name+","+dog1.age+","+dog1.color);

System.***out***.println("강아지 #2 = "+dog2.name+","+dog2.age+","+dog2.color);

System.***out***.println("강아지 #3 = "+dog3.name+","+dog3.age+","+dog3.color);

}

}

◇ 제목

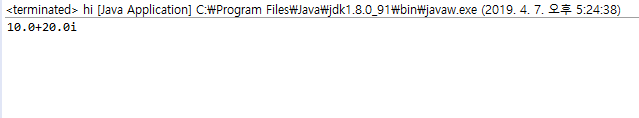
◇ 제목

◇ 프로그램 소스 및 설명

< JAVA PROGRAMMING LAB -4장: 객체 활용하기>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과



**3. 복소수 2개를 더하는 프로그램을 만들어라.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

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

**class** Complex{

**double** real;

**double** imag;

Complex(**double** r, **double** i){

real=r; imag=i;

System.***out***.println(real+"+"+imag+"i");

}

}

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

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

Complex com=**new** Complex(10,20);

}

}

◇ 제목

◆ DESCRIPTION

◇ 제목

◆ 프로그램 설명

◇ 프로그램 소스

◆ DESCRIPTION

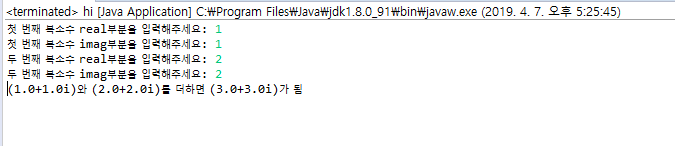
복소수는 real부분과 imag부분이 있다. 복소수를 여러 개 만들 수 있으니, 객체로 만드는게 용이할 것이다. 그래서 real과 imag를 클래스 필드로 일단 만들었다. 그리고 생성자를 이용해서 필드를 수정하는 작업을 하였다. 생성자는 인스턴스화할 때 인수로 넘겨주는 값을 받아서 필드를 수정하는 것이다.

그래서 Complex클래스를 객체로 만들 때 10,20을 넘겨주면 real이 10, imag이 20이 되고 20+10i처럼 출력이 가능한 것이다.

< JAVA PROGRAMMING LAB -4장: 객체 활용하기>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과



**4. 복소수 2개를 받아서 덧셈을 수행한 후에 반환하는 메소드를 만들고 반환 값을 출력하는 프로그램을 만들어라.**

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

◇ 제목

두 개의 복소수를 받으려면 두 개의 real과 imag가 필요하다. 그래서 사용자로부터 입력을 총 4번 받게 된다. 첫 번째 real,imag는 a,b변수에 저장하였고 두 번째 real,imag는 c,d 변수에 저장하였다. 그리고 Complex클래스를 o1객체로 인스턴스화하면서 a,b를 넘기었다. 이렇게 하면 생성자에서 필드를 수정해줘서 o1객체의 필드 real은 a값, imag는 b값을 갖게 되겠다. 마찬가지로 o2도 real은 c값, imag는 d값을 갖게 인스턴스화를 진행하였다. 이후 complex\_add라는 정적 메소드를 이용하여 o1복소수 객체와 o2복소수 객체를 더 하는 작업을 하겠다. 정적 메소드이니 앞 문제처럼 o1,o2객체의 주소를 넘겨준다. 객체를 넘겨 받으면 o1의 real필드와 o2의 real필드를 더 해서 e변수에 저장하고, o1의 imag필드와 o2의 imag필드를 더해서 f에 저장한다. 이것을 가지고 o3객체를 만든다. 최종적으로 만든 객체를 리턴하고 main에서 받은 o3객체를 출력하면 끝이 난다.

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

**class** Complex{

**double** real;

**double** imag;

Complex(**double** r, **double** i){

real=r; imag=i;

}

**static** Complex Complex\_add(Complex o1,Complex o2){

**double** e=o1.real+o2.real;

**double** f=o1.imag+o2.imag;

Complex o3=**new** Complex(e,f);

**return** o3;

}}

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

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

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

System.***out***.print("첫 번째 복소수 real부분을 입력해주세요: ");

**double** a= input.nextDouble();

System.***out***.print("첫 번째 복소수 imag부분을 입력해주세요: ");

**double** b= input.nextDouble();

System.***out***.print("두 번째 복소수 real부분을 입력해주세요: ");

**double** c= input.nextDouble();

System.***out***.print("두 번째 복소수 imag부분을 입력해주세요: ");

**double** d= input.nextDouble();

Complex o1=**new** Complex(a,b);

Complex o2=**new** Complex(c,d);

Complex o3=Complex.*Complex\_add*(o1,o2);

System.***out***.println("("+o1.real+"+"+o1.imag+"i)와 "+"("+o2.real+"+"+o2.imag+"i)를 더하면 "+"("+o3.real+"+"+o3.imag+"i)가 됨"); }}

}

}

◇ 제목

◇ 제목

◇ 프로그램 소스 및 설명

< JAVA PROGRAMMING LAB -4장: 객체 활용하기>

◇ 문제 및 요구하는 것

◆ 프로그램 실행결과

**5. 객체 배열을 이용해서 (회원가입, 로그인, 모든 유저 확인, 종료)가 담긴 프로그램을 만들어라.**

◇ 제목

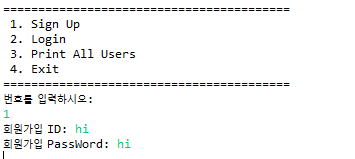
◇ 제목

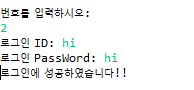
◇ 제목

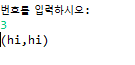
◇ 제목

◇ 제목

◇ 제목









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

**class** User{

String userid, password;

**static** **int** *count*=0;

**public** User(String userid, String password) {

**this**.userid=userid;

**this**.password=password;

}

**public** String idget() {

**return** userid;

}

**public** String pwget() {

**return** password;

}

}

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

**static** **int** *num*;

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

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

User[] array=**new** User[10];

**while**(1>0) {

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

System.***out***.print(" 1. Sign Up \n 2. Login \n 3. Print All Users \n 4. Exit");

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

System.***out***.println("번호를 입력하시오: ");

*num*=input.nextInt();

input.nextLine();

**if**(*num*==1) {

System.***out***.print("회원가입 ID: ");

String id=input.next();

input.nextLine();

System.***out***.print("회원가입 PassWord: ");

String pw=input.next();

input.nextLine();

array[User.*count*]=**new** User(id,pw);

User.*count*++;

}

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

System.***out***.print("로그인 ID: ");

String comid=input.next();

input.nextLine();

System.***out***.print("로그인 PassWord: ");

String compw=input.next();

input.nextLine();

**for**(**int** i=0;i<User.*count*;i++) {

String realid=array[i].idget();

String realpw=array[i].pwget();

**if**( (realid.equals(comid)) && (realpw.equals(compw)) ) {

System.***out***.println("로그인에 성공하였습니다!!");

**break**;

}

**else** {

**if**(i==User.*count*-1)

System.***out***.println("로그인에 실패하였습니다!!");

**else**

**continue**;

}

}

}

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

**for**(**int** i=0;i<User.*count*;i++)

System.***out***.println("("+array[i].idget()+","+array[i].pwget()+") \n");

}

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

System.***out***.println("이용해주셔서 감사합니다!!");

**break**;

}

}

}

}

◇ 제목

◇ 제목

◇ 프로그램 설명

◇ 제목

◇ 제목

◇ 프로그램 소스

계정 저장을 위해서는 아이디와 비밀번호가 필요하다. 그리고 모든 계정을 출력하려면계정 번호도 필요하다. 일단 아이디와 비밀번호는 클래스의 필드로, 계정 번호는 계정이 저장될 때마다 1씩 올라가는 식으로 쓰기 위해 정적 변수로 만들었다. 아이디와 비밀번호 필드는 많이 오고 가니 접근자를 따로 만들었다. 그리고 아이디,비밀번호 필드의 수정은 생성자가 담당한다. Main함수에서 num은 사용자가 로그인을 할 것인지, 회원가입을 할 것인지, 계정 정보를 볼 것인지 선택하기 위해 만들어진 정적 변수이다. 사용자가 숫자를 입력하면 1이 입력되면 회원가입을 진행한다. 2가 입력되면 로그인을 하고, 3이 입력되면 모든 계정 정보를 출력한다. 4가 입력되면 프로그램을 종료시킨다. 먼저 회원가입이다. Scanner클래스를 인스턴스한 input객체를 통해서 사용자로부터 사용할 아이디와 비밀번호를 입력받고 각각 문자열 변수에 저장한다. 회원가입 소스 이전에 계정 객체를 저장할 수 있는 배열이 이미 만들어져 있다. 그 배열의 User.count인덱스에 객체를 만든다. 이 객체는 앞에서 저장된 문자열 변수를 필드로 받게 된다. 이렇게 객체 배열 첫 번째 요소에 계정에 대한 객체 하나가 만들어진다.

이후 User.count를 1올리면 다음에 회원가입하면 다음 요소에 계정 객체가 만들어지게 될 것이다. 다음은 로그인에 관련된 내용이다. 마찬가지로 사용자로부터 아이디와 비밀번호를 받고 문자열 변수에 각각 저장한다. 그리고 for문을 통해서 User.count까지 i를 1씩 올리며 저장된 객체 필드와 문자열 변수를 비교하는 과정에 들어간다. 여기서 i는 객체 배열의 인덱스로 작용하게 된다. 만약 반복 도중 문자열 변수와 필드들이 하나라도 같으면 로그인에 성공하였다고 알려준다. 그렇지 않으면 로그인에 실패하였다고 알려준다.

다음은 ‘모든 계정 보기’이다. 이것도 i를 인덱스로 사용하고, for문을 통해서 User.count까지 i를 1씩 올리며 저장된 객체 필드를 출력하는 과정을 거친다.

마지막으로 ‘프로그램 종료’ 이다. 이 프로그램은 숫자 입력(num)을 계속해서 수행한다. 즉 반복문으로 작성되었으니, break를 하면 끝이 날 문제이다.

◇ 제목

◇ 제목

◇ 프로그램 설명