**Report**

**For java Programming**

**학과 :**

**학번 :**

**이름 :**

컴퓨터공학과

20184071

김도현

**문제 1번**

1. 프로그램 소스코드 및 설명과 결과

**public** **class** Dog { //도그 클래스 생성

String Name; //String 형 Name Field 생성

**int** Age; //int 형 Age Field 생성

String Color; //String 형 Color Field 생성

**static** **int** *count* = 0; //int 형 count Field 생성, 정적으로 선언한다.

//정적으로 선언하면 변수 한개를 공유한다.

**public** Dog(String Name,**int** Age,String Color) { //오류가 나지 않게 생성자로 값을 받는다.

**this**.Name = Name; //인자로 받은 Name 을 Name 필드에 저장

**this**.Age = Age; //인자로 받은 Age 을 Age 필드에 저장

**this**.Color = Color; //인자로 받은 Color 을 Color 필드에 저장

*count* ++; //정적변수 count 를 1 증가시킨다.

}

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

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

Dog puppy1 = **new** Dog("Molly",10,"Brown"); //Molly 10 Brown 을 가진 Dog 객체 puppy1 을 생성한다.

Dog puppy2 = **new** Dog("Daisy",6,"Black"); //"Daisy",6,"Black" 을 가진 Dog 객체 puppy2 를 생성한다.

Dog puppy3 = **new** Dog("Bella",7,"White"); //"Bella",7,"White" 을 가진 DOg 객체 puppy3 을 생성한다.

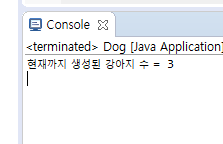
System.***out***.println("현재까지 생성된 강아지 수 = " + Dog.*count*);

//생성자가 호출 되면 즉, 객체가 생성되면 Dog 객체가 가진 Count 변수는 정적 변수라 공유하게 되므로 3이 출력된다.

}

}

실행 결과 ))



**문제 2번**

1. 프로그램 소스코드 및 설명과 결과

**import** java.util.ArrayList;

**class** Witch{

**public** **static** **void** younger(Dog2 obj) { //static 메소드로 사용하여 new 연산없이

//witch.younger 를 사용할수 있게 만든다.

obj.Age -= 5; //obj 는 객체이므로 객체를 함수로 받았을때는 call by reference 이므로

//원본 Age 가 수정된다.

}

}

**public** **class** Dog2 {

String Name; //String 형 Name Field 생성

**int** Age; //int 형 Age Field 생성

String Color; //String 형 Color Field 생성

**static** **int** *count* = 0; //int 형 count Field 생성, 정적으로 선언한다.

//정적으로 선언하면 변수 한개를 공유한다.

**public** Dog2(String Name,**int** Age,String Color) {//오류가 나지 않게 생성자로 값을 받는다.

**this**.Name = Name; //인자로 받은 Name 을 Name 필드에 저장

**this**.Age = Age; //인자로 받은 Age 을 Age 필드에 저장

**this**.Color = Color; //인자로 받은 Color 을 Color 필드에 저장

*count* ++; //정적변수 count 를 1 증가시킨다.

}

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

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

Dog2 puppy1 = **new** Dog2("Molly",10,"Brown"); //Molly 10 Brown 을 가진 Dog2 객체 puppy1 을 생성한다.

Dog2 puppy2 = **new** Dog2("Daisy",6,"Black"); //"Daisy",6,"Black" 을 가진 Dog2 객체 puppy2 를 생성한다.

Dog2 puppy3 = **new** Dog2("Bella",7,"White"); //"Bella",7,"White" 을 가진 DOg2 객체 puppy3 을 생성한다.

ArrayList<Dog2> DogLists = **new** ArrayList<Dog2>(); //3개 반복하여 출력/witch 사용을 위한 Dog2 를 담는 배열 생성

DogLists.add(puppy1); //DogLists 에 puppy 1 2 3 를 추가한다.

DogLists.add(puppy2);

DogLists.add(puppy3);

**for** (Dog2 Dogs:DogLists) { //for each 문을 사용하여 DogLists 요소 하나하나를 Dogs 에 할당한다.

Witch.*younger*(Dogs); //가져온 Dogs 를 witch 클래스의 younger 메소드를 사용하여

//원본 Age 를 조절한다.

}

**for** (Dog2 Dogs:DogLists) { //for each 문을 사용하여 Doglists 요소 하나하나를 Dogs 에 할당한다.

//할당한 Dogs 가 가진 Name Age Color 필드를 출력한다.

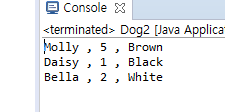
System.***out***.println(Dogs.Name +" , " + Dogs.Age +" , "+ Dogs.Color);

}

}

}

실행결과 ))



원래있던 정보

Molly , 10 , Brown

Daisy , 6 , Black

Bella , 7 , White 가 witch.younger 메소드를 통해 원본이 수정됬으므로 Age 에서 5 빠진

Molly , 5 , Brown

Daisy , 1 , Black

Bella , 2 , White 가 나온다.

**문제 3번**

1. 프로그램 소스코드 및 설명과 결과

**public** **class** Complex {//Complex 클래스를 생성

**double** real,imag; //double 타입의 정수부 real, 허수부 imag 필드를 생성한다.

**public** Complex(**double** real,**double** imag) { //double 형의 real 과 imag 를 인자로 받는다,

//생성자이므로 호출 즉시 인자로 넣어야 된다.

**this**.real = real; //인자로 받은 real 을 complex 클래스의 real 필드에 할당한다.

**this**.imag = imag; //인자로 받은 imag 를 complex 클래스의 imag 필드에 할당한다.

}

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

Complex C1 = **new** Complex(10,20); //complex 클래스를 가진 C1 객체를 생성한다.

//이 객체는 real 은 10 imag 는 20 이라는 값 (필드)를 갖는다.

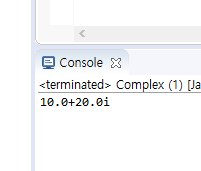
System.***out***.println(C1.real + "+" + C1.imag + "i");

//C1 이 가진 real 과 imag 를 출력한다.

}

}

실행결과 ))



인자로 넣어준 10 , 20 이 각각 10 +20 i 가 되어 출력이된다.

**문제 4번**

1. 프로그램 소스코드 및 설명과 결과

**public** **class** Complex2 {//Complex 클래스를 생성

**double** real,imag;//double 타입의 정수부 real, 허수부 imag 필드를 생성한다.

**public** Complex2(**double** real,**double** imag) {//double 형의 real 과 imag 를 인자로 받는다,

//생성자이므로 호출 즉시 인자로 넣어야 된다.

**this**.real = real; //인자로 받은 real 을 complex 클래스의 real 필드에 할당한다.

**this**.imag = imag; //인자로 받은 imag 를 complex 클래스의 imag 필드에 할당한다.

}

**public** Complex2 add(Complex2 o1, Complex2 o2) { //Complex2 객체를 반환하는 add 메소드 생성

//인자로는 Complex2 형의 o1, Complex2 형의 o2 를 받는다.

Complex2 temp = **new** Complex2(o1.real+o2.real,o1.imag+o2.imag);

//반환할 COmplex2 객체 temp 생성, 이 객체는 각각 o1 o2 가 가진 real, o1 o2가 가진 imag 를 더해서

//1번쨰 2번째 인자로 넣어준다,

**return** temp; //temp 객체가 생성되었으므로 반환을 해준다.

}

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

Complex2 C1 = **new** Complex2(10,20); //complex 클래스를 가진 C1 객체를 생성한다.

//이 객체는 real 은 10 imag 는 20 이라는 값 (필드)를 갖는다

Complex2 C2 = **new** Complex2(30,40);//complex 클래스를 가진 C2 객체를 생성한다.

//이 객체는 real 은 30 imag 는 40 이라는 값 (필드)를 갖는다

Complex2 C3 = C1.add(C1, C2);//static 메소드가 아니기 때문에 이미 생성된

//C1 객체를 사용하여 add 메소드를 호출한다.

//그 후 출력을 하면 된다.

System.***out***.print("("+C1.real + "+" + C1.imag + "i) 와"); //print 를 사용하여 개행을 하지 않는다

System.***out***.print("("+C2.real + "+" + C2.imag + "i)를 더하면");

System.***out***.println("(" + C3.real + "+" + C3.imag + "i) 가 됨");

}

}

실행결과 ))



C1 정수부인 10 과 C2정수부 30 을 더한 값 40 이 C3의 정수부가 되고 C1 허수부 20과 C2 허수부 40을 더한 60이 C3 객체의 허수부가 되서 출력이 된다.

**문제 5번**

1. 프로그램 소스코드 및 설명과 결과

**import** java.util.ArrayList;

**import** java.util.Scanner;

**public** **class** User {

String userid,password; //User 클래스는 각각 userid 와 password 라는 필드를 갖는다.

**static** **int** *count*; //모든 user 클래스가 공유하는 static 변수 count 를 생성한다.

**public** User(String userid,String password) { //String 형의 userid, String 형의 password 를

//생성자의 인자로 받는다.

**this**.userid = userid; //인자로 받은 userid 를 필드 userid 에 할당한다.

**this**.password = password; //인자로 받은 password 를 필드 password 에 할당한다.

*count* ++; //생성자가 호출될때 마다 static int count 는 1이 증가한다.

}

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

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

**int** input = 0; //do while 문 실행과 종료를 위한 input 변수 생성

Scanner sc = **new** Scanner(System.***in***); //while 문이 돌때마다 값을 받아야 하므로 while 문밖에 scanner 객체를 생성한다

//while 문안에 있으면 계속 객체되는데 while 문 밖에 있으면 1회만 생성된다.

ArrayList<User> Users = **new** ArrayList<User>(); //계정정보를 담아야 하므로

//User 객체를 담는 Users 이름의 ArrayList 객체 생성.

**do** {

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

System.***out***.println("1.Sing UP");

System.***out***.println("2.Login");

System.***out***.println("3.Print All Users");

System.***out***.println("4.Exit");

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

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

input = sc.nextInt(); //사용자에게서부터 int 형 값을 입력받는다.

**if** (input == 1) { //입력 값이 1이라면

String id,password; //id 와 password 변수를 생성하고

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

id = sc.next(); //id 값을 입력받고

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

password = sc.next();//password 값을 입력반는다.

Users.add(**new** User(id,password)); //그후 새로운 User 객체를 생성하여 Users ArrayLists 에 넣는다.

//Users 안에 user 객체가 들어있으므로 인스턴스를 (User us = new ..)를 할 필요가 없다.

//ArrayLists 에 넣음으로써 id 조회를 할 수 있다.

}

**else** **if**(input == 2) { //입력값이 2 로그인이라면

String id,password; //id 와 password 변수를 생성하고

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

id = sc.next(); //id 값을 입력받고

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

password = sc.next(); //password 값을 입력받는다.

**boolean** isLogin = **false**; //로그인 유무를 하기 위해 3항연산자 사용을 위해 Bollean 타입의 isLogin을 생성한다

//로그인이 되었으면 true 고 아니면 false 이다.

**for** (User user : Users) { //for each 문을 사용하여 Users 배열의 각각 값을 for 문이 돌때마다

//user 에 할당한다.

**if**((user.userid).equals(id) && (user.password).equals(password)) {

//로그인의 조건 id 와 비밀번호가 일치할 경우

isLogin = **true**; //로그인은 성공으로 바꾸고

**break**; //해당 루프는 더 할 필요 없이 종료시킨다.

}

}

System.***out***.println((isLogin) ? "로그인 되었습니다." : "로그인에 실패했습니다 .");

//isLogin 이 true 면 로그인이 된것이므로 로그인이 되었습니다 를 출력한다.

//아니면 로그인에 실패하였습니다 를 출력한다.

}

**else** **if**(input == 3) { //입력값이 3이라면

**for** (User user : Users) { //for each 문을 통해 Users 배열의 각각 값을 for 문이 돌때마다 user 에 할당한다.

//user 객체가 가진 userid 와 password 를 출력한다.

System.***out***.println("{ " + user.userid + " , " + user.password + "}");

}

}

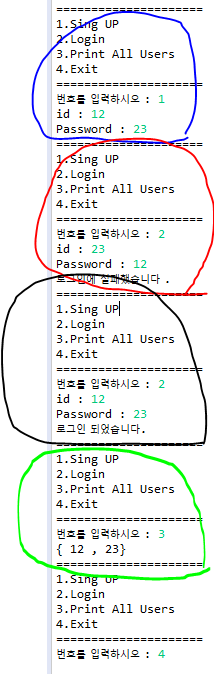
//do while 문 조건중에 4 가 있으므로 else if input==4 then break 는 만들 필요 없다.

}**while** (input != 4); //해당 do while 문은 input 이 4일때 종료된다.

}

}

실행결과 ))



입력 숫자 1 2 3 에 따라 출력이 다 다르다. 파란색과 같이 1을 선택을 했으면 아이디가 12, 비밀번호 23 으로 저장이 된다.

3을 선택해서 출력을 해보면 연두색 원과 같이 앞서 저장한 아이디 12 와 23 이 저장이 된다.

2를 선택해서 출력을 해보면 로그인에 실패할 경우 ( 아이디 비밀번호가 다를 경우) 빨간색 원처럼 나온다.

로그인의 성공할 경우 ( 아이디 비밀번호 가 일치할 경우) 검정 원 처럼 로그인에 성공이 나온다.

4를 선택할 경우 프로그램이 종료가 된다.