Day14(8/20)

\*내부 클래스

**package** com.ict.nested;

**public** **class** MyClass {

**public** MyClass() {

System.***out***.println("MyClass의 생성자");

}

// 인스턴스 멤버 Class

**class** B {

**int** field1;

// static int field2;

// 인스턴스 멤버 클래스 안에서는 static멤버 생성 불가능

**public** B() {

System.***out***.println("B의 생성자");

}

**void** method1() {

}

// 인스턴스 멤버 클래스 안에서는 static메소드 생성 불가능

// static void method2{ }

}

// static(정적) 멤버 class

**static** **class** C {

// static멤버 class내부에서는 static, 인스턴스 멤버 필드와 메소드를 모두 생성 가능

**int** field1;

**static** **int** *field2*;

**public** C() {

System.***out***.println("C의 생성자");

}

**void** method1() {

}

**static** **void** method2() {

}

}

// MyClass의 멤버 메소드

**void** method() {

// 메소드의 로컬 클래스 선언- 메소드에서 클래스 선언가능

**class** D {

**int** field1;

// 로컬 클래스에서는 정적멤버 생성 불가능

// static int field2;

**public** D() {

System.***out***.println("D의 생성자");

}

**void** method() {

System.***out***.println("field1=>" + field1);

}

// 로컬 클래스에서는 정적 메소드 생성 불가능

// static void method() {}

}// end of D 클래스 선언

//멤버메소드내의 로컬클래스 객체화 가능

D d = **new** D();

d.field1 = 3;

d.method();

}

}

**package** com.ict.nested;

**public** **class** MyClassClient {

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

MyClass a = **new** MyClass();

//인스턴스 멤버 클래스 객체 생성

MyClass.B b = a.**new** B();

b.field1 = 30;

b.method1();

//정적(static) 멤버 클래스 객체 생성

MyClass.C c = **new** MyClass.C();

c.field1 = 500;

c.method1();

//class C는 static 멤버이기 때문에 객체화 하지 않고 접근가능

MyClass.C.*field2* = 1000;

MyClass.C.*method2*();

//로컬 클래스 객체 생성을 위한 메소드 호출

a.method();

}

}

//java.lang package

//-java 프로그래밍에 사용되는 기본 클래스를 담고 있다.

//-java.lang package는 import하지 않아도 사용 가능하다.

//\*Object 클래스: 자바 클래스의 최상위 클래스로 사용

//\*System 클래스:

// -표준입력장치(키보드)로부터 데이터를 입력받을 때 사용 System.int

// -표준출력장치(모니터)로 출력하기 위해 사용 System.out

// -자바 가상머신을 종료시킬 때 사용 System.exit()

// -가비지컬렉션 호출 System.gc()

//\*Class 클래스: -클래스 저보 확인 또는 메모리 로딩 시 사용

//\*String 클래스: -문자열을 저장하고 여러가지 정보를 얻을 때 사용

//\*StringBuffer: -String의 단점을 보완한 문자열 처리 클래스

//\*StringBuilder 클래스 :-String의 단점을 보완한 문자열 처리 클래스

//\*Math 클래스 : - 수학 함수를 이용할 때 사용

//\*Wrapper : 기본 타입을 클래스로 만든 클래스

// -Byte

// -Short

// -Character

// -Integer

// -Float

// -Double

// -Boolean

// -Long

//java.util package

//\*Arrays 클래스: -배열을 조작(비교, 복사, 정렬, 찾기)할 때 사용

//\*Calendar 클래스: -운영체제의 날짜와 시간을 얻을 때 사용

//\*Date 클래스: -날짜와 시간정보를 저장하는 클래스

//\*Objects 클래스: -객체를 비교하고 null 정보등을 조사할때 사용

//\*StringTokenizer: 문자열에서 특정 구분자로 문자열을 분리 시킨다.

//\*Random 클래스 : -난수를 얻을때 사용한다.

\*equals() Override

**package** com.ict.am;

//public class Member extends Object

//컴파일시 자동으로 extends Object가 붙는다.

**public** **class** Member {

**public** String id;

**public** Member(String id) {

**this**.id = id;

}

//source->Override/Implement method

//기본 equals()는 객체가 같은 것인지에 대해서 만 결과 값을 알려준다.

//Object 클래스에 정의되어 있는 equals를 Override해서 비교한다.

//\*깊은 비교: 실제 필드를 비교해주는 것

@Override

**public** **boolean** equals(Object obj) {

Member member = **null**; //지역변수 초기화

**if**(obj **instanceof** Member) {

//instanceof : 타입이 같은 것인 지 묻는 것

member = (Member)obj;

//String에는 equals가 재정의 되어 있다.

**if**(member.id.equals(**this**.id)) {

**return** **true**;

}

}

**return** **false**;

}

}

**package** com.ict.am;

**public** **class** Ex02\_Equals {

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

//객체비교 : equals()

Member mem01 = **new** Member("Blue");

Member mem02 = **new** Member("Blue");

Member mem03 = **new** Member("Orange");

**if**(mem01.equals(mem02)) {

System.***out***.println("같은 멤버입니다");

}**else** {

System.***out***.println("다른 멤버입니다");

}

//같은 멤버입니다.

**if**(mem01.equals(mem03)) {

System.***out***.println("같은 멤버입니다");

}**else** {

System.***out***.println("다른 멤버입니다");

}

//다른 멤버입니다.

}

}

\*toString Override

**public** **class** SmartPhone {

**private** String company;

**private** String os;

**public** SmartPhone(String company, String os) {

**this**.company = company;

**this**.os = os;

}

//toString을 오버라이드하지 않으면 hashcode가 출력된다.

@Override

**public** String toString() {

**return** company + ", " + os;

}

}

**public** **class** SmartPhoneExample {

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

SmartPhone myPhone = **new** SmartPhone("Google", "Android");

//System.out.println(myPhone.toString());

//.toString()은 생략 가능

System.***out***.println(myPhone);

}

}

\*clone() Override

복사

1)primitive type의 복사

Object a, Object b

a = b;

reference값만을 복사하게 되어 같은 참조를 가진 두 개의 변수가 생긴다.

2)shallow copy(얕은 복사)

변수 a, b는 x100이라는 같은 reference 값을 가지게 되므로 한쪽에서 데이터를 변경하면 다른 한쪽도 변경되는 데이터를 가지게 되는 것

3)deep copy(깊은 복사)

사용자가 정의한 클래스의 객체가 모든 값을 복사하는 깊은 복사(deep copy)를 하기 위해서는 사용자 정의 클래스에 Cloneable interface를 implement하여 Object clone()이라는 메소드를 override하면 된다

-CloneNotSupportException을 예외처리하고 method 내부에 리턴할 객체를 생성한다.

//implements Cloneable을 해야 clone 기능 사용 가능

**public** **class** Member **implements** Cloneable{

**public** String id;

**public** String name;

**public** String password;

**public** **int** age;

**public** **boolean** adult;

//생성자

**public** Member() {}

**public** Member(String id, String name, String password, **int** age, **boolean** adult) {

**this**.id = id;

**this**.name = name;

**this**.password = password;

**this**.age = age;

**this**.adult = adult;

}

//clone : Object

//얕은 복사

**public** Member cloneMember() {

Member cloned = **null**;

**try** {

cloned = (Member)**this**.clone();

}**catch**(CloneNotSupportedException e){

}

**return** cloned;

}

@Override

**public** String toString() {

**return** "Member [id=" + id + ", name=" + name + ", password=" + password + ", age=" + age + ", adult=" + adult

+ "]";

}

}

**public** **class** MemberSample {

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

Member original = **new** Member("hong", "gildong", "1234", 24, **true**);

Member cloned = original.cloneMember();

cloned.name = "길동";

System.***out***.println(original);

System.***out***.println(cloned);

}

}

**public** **class** Car {

**public** String model;

**public** Car(String model) {

**this**.model = model;

}

@Override

**public** String toString() {

**return** "Car [model=" + model + "]";

}

}

**import** java.util.Arrays;

**public** **class** Member **implements** Cloneable {

**public** String name;

**public** **int** age;

**public** **int**[] score;

**public** Car car;

//생성자

**public** Member(String name, **int** age, **int**[] score, Car car) {

**this**.name = name;

**this**.age = age;

**this**.score = score;

**this**.car = car;

}

@Override

**protected** Object clone() **throws** CloneNotSupportedException {

Member cloned = (Member)**super**.clone();

//Object의 clone()을 이용해야 하기 때문에 super를 사용한다.

//this.clone()은 Member의 clone()을 재귀 호출하는 것이기 때문이다.

//깊은 복사 구현

//Arrays.copyOf(배열 명, 배열 길이) - 배열을 복사 한다.

cloned.score = Arrays.*copyOf*(score, score.length);

car = **new** Car(car.model);

**return** cloned;

}

**public** Member cloneMember() {

Member cloned = **null**;

**try** {

cloned = (Member)**this**.clone();

}**catch**(CloneNotSupportedException e){

}

**return** cloned;

}

@Override

**public** String toString() {

**return** "Member [name=" + name + ", age=" + age + ", score=" + Arrays.*toString*(score) + ", car=" + car + "]";

}

}

**public** **class** MemberSample {

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

Member original = **new** Member("홍길동", 25, **new** **int**[] {5,10,15}, **new** Car("제네시스")) ;

Member cloned = original.cloneMember();

cloned.score[0] = 20; //얕은 복사 original, cloned의 값이 모두 바뀜

//깊은 복사 cloned의 배열 값을 바꿔도 original의 값이 바뀌지 않는다.

//클래스가 기본데이터형으로만 구성되어 있지 않을 때는 깊은 복사를 이용해서

//값을 변경해야 한다.(ex)배열, 클래스 등)

System.***out***.println(original);

System.***out***.println(cloned);

}

}

\*finalize() Override

**public** **class** Counter {

**private** **int** no;

**public** Counter(**int** no) {

**this**.no= no;

}

@Override

**protected** **void** finalize() **throws** Throwable {

System.***out***.println(no+"번의 Counter 객체 소멸");

}

}

**public** **class** FinalizeExample {

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

Counter counter = **null**;

**for**(**int** i=0; i<30 ; i++) {

counter = **new** Counter(i);

counter = **null**;

System.*gc*(); //가비지 컬렉션 강제 실행 요청

//참조값이 null값을 가르키면 가비지 컬렉션에 의해서 소멸된다.

//가비지 컬렉션이 자동으로 호출되서 소멸된다.

}

}

}

\*StringBuffer

**public** **class** StringBufferExam {

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

String str1 = "Korea-japan";

//StringBuffer strbf = "korea-japan" (x)잘못된 선언

StringBuffer strbf = **new** StringBuffer("Korea-japan");

System.***out***.println(strbf);

//strbf = strbf+ "China";(x)잘못된 사용

//문자열을 추가하기 위해 append()사용

strbf.append("-china");

System.***out***.println(strbf);

//subString(시작 점, 끝점)

//subString(시작 점)

//끝나는 점 명시안하며 문자열 끝까지 출력됨

String str2 = strbf.substring(6);

System.***out***.println(str2);

String str3 = strbf.substring(6,11);

System.***out***.println(str3);

//setCharAt(바꾸고자 하는 자리 index, '바꿀 문자')

strbf.setCharAt(6, 'H');

System.***out***.println(strbf);

//charAt(index): Char -index에 해당하는 문자를 반환

System.***out***.println("strbf.charAt(6) =>" +strbf.charAt(6));

//indexOf("H"); 해당 문자의 index를 반환

System.***out***.println("strbf.indexOf(\"H\")=>" +strbf.indexOf("H"));

}

}

\*StringTokenizer

**public** **class** StringTokenizerExam {

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

String info = "HONG|서울시 은평구|010-1111-111|34";

//StringTokenizer(문자열, 구분자);

StringTokenizer strtk = **new** StringTokenizer(info, "|");

String[] str1 = **new** String[strtk.countTokens()];

**int** i = 0;

**while**(strtk.hasMoreTokens()) {

str1[i++] = strtk.nextToken();

}

**for**(i=0; i<str1.length; i++) {

System.***out***.println(str1[i]);

}

}

}

\*HangMan GAME 구상도

Main

1.회원가입 2.게임하기 3.랭킹보기 4.종료하기

선택>>>1

--------------------

ID>> fbwodud

password>> 1234

회원가입이 완료되었습니다!

게임을 시작하겠습니다(y/n)

>>n 🡪게임종료

>>y 🡪게임시작

게임시작

총 시도 횟수는 10회입니다.

>>\_ \_ \_ \_ \_

>>입력>>l

>>’입력한 값’은 존재합니다.

>>\_ \_ l l \_

>>시도 횟수가 9번 남았습니다.

>>입력>>k

>>’입력한 값’이 존재하지 않습니다.

>>시도횟수 8회입니다.

…10회내에 맞추면 1탄 완료!

🡪다음 탄으로 넘어가겠습니까?(y/n)

🡪y🡪다음 탄 시작

🡪n🡪게임 종료

…10회내에 못맞추면 1탄 실패!

🡪재도전 하시겠습니까?(y/n)

🡪y🡪현재 탄 다시 시도

🡪n🡪게임 종료

단, 게임 종료할 때 모든 현재 탄 기록

1.회원가입 2.게임하기 3.랭킹보기 4.종료하기

선택>>>2

로그인

아이디>> fbwodud

비번>>1234

아이디존재, 비번 일치 >> 로그인 성공!

아이디존재, 비번 불일치>> 로그인 실패

아이디존재x 🡪 아이디가 존재하지 않습니다. 1.회원가입

>>게임하기

---------------------------------------------------------------

1.회원가입 2.게임하기 3.랭킹보기 4.종료하기

선택>>>3

회원 ID | 현재 탄

>>높은 탄을 깨고 있는 순으로 보여주기

-------------------------------------------------------------------------------------------

**package** com.ict.hanaman;

**import** java.util.Scanner;

**public** **class** HangMan {

**static** Scanner *scan* = **new** Scanner(System.***in***);

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

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

HangManMethod.*makequiz01*();

System.***out***.println("1.회원가입 2.게임시작 3.랭킹보기 4.종료하기");

String answer = *scan*.next();

**switch** (answer) {

**case** "1":

HangManMethod.*register*();

**break**;

**case** "2":

HangManMethod.*gameStart*();

**break**;

**case** "3":

HangManMethod.*showRank*();

**break**;

**case** "4":

HangManMethod.*close*();

**break**;

**default**:

System.***out***.println("잘못 입력하셨습니다.");

}

}

}

}

**package** com.ict.hanaman;

**import** java.util.Scanner;

**public** **class** HangManMethod {

**static** String[] *str* = **new** String[9];

**static** Scanner *scan* = **new** Scanner(System.***in***);

**static** Player[] *player* = **new** Player[100]; // player를 저장할 Player타입의 객체 저장 배열

**static** **int** *top* = 0; // 현재 저장공간

**static** **void** makequiz01() {

*str*[0]="hello";

*str*[1]="disco";

*str*[2]="taste";

*str*[3]="visit";

*str*[4]="waste";

*str*[5]="basic";

*str*[6]="cisco";

*str*[7]="money";

*str*[8]="pushy";

//배열 생성하고 시작

}

**static** String makequiz() {

**return** *str*[*player*[*top*-1].getIstan()];

}

**private** **static** **void** assign(String id, String password) {

*player*[*top*] = **new** Player(id, password, 0); // 아이디가 생성될때는 무조건 0탄으로 생성

System.***out***.println("ID생성 완료!");

*top*++;

System.***out***.println("게임을 시작하겠습니까?(Y/N)");

String answer = "";

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

answer = *scan*.next();

**if** (answer.equalsIgnoreCase("y")) {

*gameStart*();

} **else** **if** (answer.equalsIgnoreCase("n")) {

*close*();

} **else** {

System.***out***.println("y/n중 입력하세요.");

**continue**;

}

}

}

**static** **void** register() {

System.***out***.println("----회원가입----");

String id = "";

String password = "";

chk: **while** (**true**) {

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

id = *scan*.next();

// 중복되는 id가 있는지 검사

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

**if** (*player*[i].getID().equals(id)) {

System.***out***.print("중복된 ID입니다. 다시입력하세요");

**continue** chk;

}

}

**break**;

}

System.***out***.println("Password>>");

password = *scan*.next();

// 객체 생성 함수

*assign*(id, password);

}

**static** **void** gameStart() {

System.***out***.println("----게임 시작----");

**if**(*top*==0) {

System.***out***.println("회원가입하세요");

*register*();

}

**int** count = 10;

String quiz = *makequiz*();

StringBuffer sb = **new** StringBuffer("-----");

String answer = "";

**int** end = 0;

// 답을 맞추는 코드

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

**if** (count == 0) {

System.***out***.println("기회가 모두 끝났습니다.");

**break**;

}

System.***out***.println("총 기회는" + count + "입니다.");

System.***out***.println(sb);

System.***out***.println("입력>>>");

answer = *scan*.next();

**boolean** result = **false**;

// - --> 입력한 문자로 바꾸기

**char** a = answer.charAt(0);

**for** (**int** i = 0; i < 5; i++) {

**if** (quiz.charAt(i) == a) {

sb.setCharAt(i, a);

end++;

result = **true**;

}

}

**if** (result) {

System.***out***.println("입력한 문자" + a + "존재합니다.");

} **else** {

System.***out***.println("입력한 문자" + a + "존재하지않습니다.");

}

// 10회 이내에 답을 맞출경우

**if** (end == 5) {

System.***out***.println("정답입니다");

System.***out***.println("현재"+(*player*[*top*-1].getIstan()+1)+"탄 입니다");

**int** nextan = *player*[*top* - 1].getIstan() + 1;

*player*[*top* - 1].setIstan(nextan);

**break**;// while문 빠져나감

}

count--;

}

// 정답을 맞춤

System.***out***.println("계속하시겠습니까?");

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

String answer2 = *scan*.next();

**if** (answer2.equalsIgnoreCase("y")) {

**if**(*player*[*top*-1].getIstan()<=8) {

System.***out***.println(*player*[*top*-1].getIstan()+1+"탄 시작합니다.");

*gameStart*();

}**else** {

System.***out***.println("모두 깨셨습니다. 게임을 종료합니다");

*close*();

}

} **else** **if** (answer2.equalsIgnoreCase("n")) {

**return**;

} **else** {

System.***out***.println("y/n중 입력하세요");

**continue**;

}

}

}

// 지금까지 player 순서대로

**static** **void** showRank() {

System.***out***.println("show Rank");

System.***out***.println("아이디|\t현재 탄");

**for** (**int** i = 0; i < *top* - 1; i++) {

**for** (**int** j = i; j < *top*; j++) {

**if** (*player*[i].getIstan() < *player*[j].getIstan()) {

Player temp = **new** Player();

temp = *player*[i];

*player*[i] = *player*[j];

*player*[j] = temp;

}

}

}

}

**static** **void** close() {

System.***out***.println("수고하셨습니다");

System.*exit*(0);

}

}

**package** com.ict.hanaman;

**public** **class** Player {

**private** String ID;

**private** String password;

**private** **int** istan; //현재 진행중인 탄 표시

**public** Player() {

}

**public** Player(String ID, String password, **int** istan) {

**this**.ID = ID;

**this**.password = password;

**this**.istan = istan;

}

//getter, setter

**public** String getID() {

**return** ID;

}

**public** **void** setID(String ID) {

**this**.ID = ID;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** **int** getIstan() {

**return** istan;

}

**public** **void** setIstan(**int** istan) {

**this**.istan = istan;

}

}

ㅁ