**[ 13 ] API-II**

**목표 : 수학관련 API, Object**

1. 수학적 계산에 사용되는 클래스 Math

※ Math 클래스가 제공하는 정적 메소드

※ Math 클래스의 상수

**public** **class** MathEx1 {

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

//Math의 static method 실습

System.***out***.println("2의 3승 : "+Math.*pow*(2, 3));

System.***out***.println("-9.9의 절대값:"+Math.*abs*(-9.9));

System.***out***.println("16의 제곱근 : "+Math.*sqrt*(16));

System.***out***.println("4와 7중에 작은값:"+Math.*min*(4, 7));

System.***out***.println("7과 9중 최대값:"+Math.*max*(7, 9));

//Math의 static final 변수(상수)

}

}

※ Math 수학 계산에 유용한 메서드들

**public** **class** MathEx2 {

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

System.***out***.println(" 소숫점에서 반올림, 올림, 버림");

System.***out***.println("9.12의 올림 : "+Math.*ceil*(9.12));

System.***out***.println("9.69의 반올림 : "+Math.*round*(9.69));

System.***out***.println("9.69 버림 : "+Math.*floor*(9.69));

System.***out***.println("소숫점 한자리에서 반올림, 올림, 버림");

System.***out***.println("9.12의 올림 : "+Math.*ceil*(9.12\*10)/10);

System.***out***.println("9.69의 반올림 : "+Math.*round*(9.69\*10)/10.0);

System.***out***.println("9.69 버림 : "+Math.*floor*(9.69\*10)/10);

System.***out***.println("십의 자리에서 반올림, 올림, 버림");

System.***out***.println("11의 올림 : "+Math.*ceil*(11/10.0)\*10);

System.***out***.println("19의 반올림 : "+Math.*round*(19/10.0)\*10);

System.***out***.println("19 버림 : "+Math.*floor*(19/10.0)\*10);

}

}

2. 난수 발생이 쉬운 Random 클래스

댜양한 랜덤 숫자를 구하는 작업을 제공합니다.

※Math.random()

Random() 메소드는 double타입의 임의의 수를 발생합니다.

API documents(<http://docs.oracle.com/javase/8/docs/api>)을 보면 static이고, 반환 타입이 double임을 확인 할 수 있습니다. Math클래스의 random()메소드가 static이기 때문에 객체 생성하지 않고 Math.random() 형으로 바로 실행할 수 있다

public class RandomEx {

public static void main(String[] args) {

double d = Math.random(); //random메소드는 static

System.out.println(d); //0이상 1미만의 double난수

int di = (int)(d\*10);

System.out.println(di);

Random random = new Random();

int i = random.nextInt();

System.out.println(i)

System.out.println((int)(d\*10)+1);//1~10까지의 난수

}

}

※Random 클래스

사용법이 아주 쉬워 요즘 많이 쓰이고 있는 클래스. 일반메소드는 객체를 생성 후 실행가능

public static void main(String[] args) {

Random random = new Random();

int iran = random.nextInt(10); //0~9까지의 난수

double doubleran = random.nextDouble(); //Math.random()과 같음

}

<예제>

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

Random random = **new** Random();

System.***out***.println("0부터 100까지의 난수 : " + random.nextInt(101));

System.***out***.println("0부터 50까지의 난수 발생 :" + random.nextInt(51));

System.***out***.println("0부터 20까지의 난수 발생 : " + random.nextInt(21));

System.***out***.println("int형 전체 범위의 난수 발생 : " + random.nextInt());

System.***out***.println("float 타입의 난수 발생 : " + random.nextFloat());

System.***out***.println("double 타입의 난수 발생 : " + random.nextDouble());

System.***out***.println("long 타입의 난수 발생 : " + random.nextLong());

System.***out***.println("boolean 타입의 난수 발생 : " + random.nextBoolean());

}

}

3. 자바 클래스의 상속 계층

여러 클래스의 공통된 특성(공통데이터와 메소드)은 슈퍼클래스로 선언하는 것이 좋다는 사실을 상속에서 접했습니다. JDK 라이브러리 개발자들도 그런 식으로 상속관계를 갖도록 클래스를 설계했기 때문에 비슷한 기능의 클래스들이 같은 슈퍼 클래스를 갖게 되는 경우를 종종 볼 수 있습니다.

그런데 클래스의 어떤 특성은 특정 부류의 클래스 뿐 아니라 자바의 모든 클래스들이 공통으로 갖습니다. 그래서 자바에서는 그런 공통 특성을 추출하여 Object라는 클래스로 만들어 두었습니다. 그리고 다른 모든 클래스들은 Object 클래스를 상속 받도록 만들어 두었습니다.

extends Object라는 단어가 없어도 컴파일 과정에서 자동적으로 Object 의 서브 클래스가 됩니다.

※ Object클래스의 메소드들

* public String toString() 객체의 문자열을 반환한다.
* public boolean equals(Object obj)
* public int hashCode() 자바에서 객체를 식별하는 정수 값인 해시 코드를 반환(다량의 데이터를 저장&검색하는 해싱 기법에 사용된다.)
* public Class getClass() 객체의 클래스 이름을 Class형으로 반환한다

1. toString() ; 객체의 정보를 문자열로 제공할 목적으로 정의된 메서드.

자바는 사용자가 모르게 내부적으로 동작하는 메소드 들이 많은데 그 중 하나가 toString().

<예제>

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**public** **class** Sawon {

**private** String num; // 사번

**private** String name; // 이름

**private** String part; // 부서

**private** Date enterDate; // 입사일

**public** Sawoon(){}

**public** Sawoon(String num, String name, String part){

**this**.num = num;

**this**.name = name;

**this**.part = part;

enterDate = **new** Date();

}

/\*public void print(){

System.out.print("[사번]"+num+"\t[이름]"+name);

System.out.print("\t[부서]"+part+"\t[입사일]");

SimpleDateFormat sdf =

new SimpleDateFormat("yy년MM월dd일(E)");

System.out.println(sdf.format(enterDate));

}\*/

@Override

**public** String toString() {

String temp = "[사번]"+num+"\t[이름]"+name;

temp += "\t[부서]"+part+"\t[입사일]";

SimpleDateFormat sdf =

**new** SimpleDateFormat("yy-MM-dd");

temp += sdf.format(enterDate);

**return** temp;

}

}

**package** com.ch.ex3tostring;

**public** **class** SawoonMain {

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

Sawoon kang = **new** Sawoon("23A01","강동원","전산실");

Sawoon yu = **new** Sawoon("22A10","유아인","비서실");

/\*kang.print();

yu.print();\*/

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

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

}

}

1. equals()를 오버라이딩 할 수도 이다.

**package** com.ch.ex4equals;

**public** **class** Person {

**private** **long** juminNo;

**public** Person(){}

**public** Person(**long** juminNo){

**this**.juminNo = juminNo;

}

@Override //p1.equals(p2)

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

// juminNo == obj.juminNo 이면 true리턴

// 아니면 false 리턴

**if**(obj!=**null** && obj **instanceof** Person){

**return** juminNo==((Person)obj).juminNo;

}**else**{

**return** **false**;

}

}

}

**package** com.ch.ex4equals;

**public** **class** PersonMain {

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

Person p1 = **new** Person(9101012085215L);

Person p2 = **new** Person(9101012085215L);

Person p3 = **new** Person(9805241052009L);

**if**(p1.equals(p2))

System.***out***.println("두 객체는 같은 사람이구만");

**else**

System.***out***.println("두 객체는 다른 사람이구만");

**if**(p1 == p2)

System.***out***.println("두 객체는 같은 객체네");

**else**

System.***out***.println("두 객체는 따로 네");

}

}

<예제> Card 데이터 :카드종류(♥ ◆ ♠ ♣), 번호(1~13)

생성자 new Card('♥',2);

toString() 오버라이드

equals() 오버라이드

**public** **class** Card {

**private** **char** kind; // ♥ ◆ ♠ ♣

**private** **int** num; // A(1), 2~10, 11, 12, 13

// Card c1 = new Card('♥',2);

**public** Card(**char** kind, **int** num) {

**this**.kind = kind;

**this**.num = num;

}

@Override

**public** String toString() {**return** "카드 모양은 "+kind+" "+num;}

@Override

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

// c1.equals(c2) this.kind와 c2.kind가 같고 this.num과 c2.num이 같으면 true

}//equals

}

**public** **class** Ex04\_CardMain {

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

Card[] cards = {**new** Card('♥',2),

**new** Card('◆', 7),

**new** Card('♣', 8)};

Card yours = **new** Card('♣', 8);

**char**[] kinds = {'♥','◆','♠','♣'};

Random ran = **new** Random();

Card yours = **new** Card(kinds[ran.nextInt(4)], (**int**)(Math.*random*()\*13)+1);

System.***out***.println("당신카드는 "+yours);

**for**(**int** idx=0 ; idx<cards.length ; idx++) {

System.***out***.print(cards[idx]);

**if**(yours.equals(cards[idx])) {

System.***out***.println(" - 당신 카드와 일치합니다");

}**else** {

System.***out***.println(" - 당신 카드와 일치하지 않습니다");

}//if

}//for

}//main

}

실행결과 : 당신 카드 : 카드 모양은 ♣ 8

카드 모양은 ♥ 2 - 당신 카드와 일치하지 않습니다

카드 모양은 ◆ 7 - 당신 카드와 일치하지 않습니다

카드 모양은 ♣ 8 - 당신 카드와 일치합니다

1. hashCode() 해쉬코드를 반환

public static void main(String[] args) {

String str1 = new String("ABC");

String str2 = new String("ABC");

System.out.println(str1.hashCode());

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

System.out.println(str1.equals(str2));

}

1. getClass메소드 : 객체가 속하는 클래스의 정보(이름)를 리턴

<quiz> Point3D.java / QuizPoint3Dmain.java 를 첨부한 메일 부탁드립니다(yisy0703@naver.com)

3D 공간의 좌표값(x,y,z)을 갖는 Point3D 클래스를 다음의 조건에 맞게 구현하시오

1. 매개변수를 갖지 않는 생성자 호출시 (x=0.0, y=0.0, z=0.0)

매개변수 3개를 갖는 생성자 호출시 (x,y,z) 값으로

1. x,y,z 실수의 좌표값이 같으면 true 값을 리턴하고

그렇지 않으면 false를 리턴하는 equals() 재정의

1. 좌표값 [x값, y값, z값]을 리턴하는 toString() 재정의
2. 위의 세 조건을 모두 보여주는 main()함수를 구현하세요

<총괄예제>

|  |  |
| --- | --- |
| **Friend 클래스형 배열에 저장된 친구들의 생일을 검색하여 해당 생일의 친구들의 모든 정보를 출력하고, 존해하지 않을 경우 “해당 생일의 친구는 없습니다”라고 출력하는 프로그램을 구현하시오.**   |  | | --- | | **<요구사항>**  1) 친구 정보를 저장할 class Friend는 다음과 같은 구조로 구현한다  데이터 – name(String;이름), phone(String;핸드폰), address(String;주소), birthday(Date;생년월일)  기능 – toString을 override 한다.  생성자, getter/setter  2) main 메서드에서 친구정보를 배열로 저장한다.  main 메서드에서 x를 입력하기 전까지 반복되는 반복문을 이용해서 프로그램의 흐름이 유지되도록 한다.  3) 저장된 친구는 같은 생일인 친구가 있을 수 있다  4) 실행화면은 다음과 같다. | |
| |  | | --- | | 검색할 생일을 00-00로 입력 (단, 종료를 원하시면 x를 입력하시오) : 05-22  이 름 : 홍길동  주소 : 서울시 용산구  핸드폰 : 010-9999-1234  생 일 : 05-22  이름 : 신길동  주소 : 서울시 마포구  핸드폰 : 010-9999-9999  생일 : 05-22  검색할 생일을 00-00로 입력 (단, 종료를 원하시면 x를 입력하시오) : 05-23  해당 지역의 친구는 없습니다  검색할 생일을 00-00로 입력 (단, 종료를 원하시면 x를 입력하시오) : X | |