**- 조건문**

**- 반복문**

**- 이동문**

**===> main 첫문장 – 마지막문장 순서대로 1번씩 실행 종료**

**조건문**

**if-else**

**if(boolean){ t }**

**else{ f }**

**switch-case**

**switch(int(byte, short, char) |String){**

**case 숫자1 또는 ""값:**

**같으면 수행 문장;**

**break;**

**case 숫자2 또는 ""값:**

**같으면 수행 문장;**

**break;**

**}  
}**

**- 반복문**

**for – 0번 이상 유한횟수 반복**

**while – 0번 이상 무한횟수 반복**

**do-while– 1번 이상 무한횟수 반복**

**for(시작문장 ; 반복종료조건식( t/f) ; 변화식){**

**반복 수행 필요 문장 정의**

**}**

**for(1 ; 2 ; 3){**

**4;**

**}**

**1->**

**2->4->3->**

**2->4->3->2->4->3**

**while(true ){**

**반복 수행 필요 문장 정의**

**if(반복종료조건식){ break; }**

**}**

**int i = 10;**

**do{**

**s.o.p(i\*i);**

**} while(i <10);**

**- 이동문**

**break**

**continue**

**if(조건식) { break; }**

**if(조건식) { continue; }**

**- 배열**

**기본 – 자바 내장 변수**

**논리값, 단일문자, 정수, 실수**

**stack 실제값 저장**

**참조형 - 문자열, 배열**

**stack 주소값 저장**

**(주소 참조 heap영역 실제저장값)**

**String s =null; ==> 주소값 없다**

**int [] ar = null; ==> 주소값 없다**

**if( s == null)**

**if(s != null) s.메소드명()**

**int i = null;===>error**

**String**

**== 연산자 – 두 String 변수 참조 주소값 동일 비교**

**equals() 메소드 - 두 String 변수 참조 문자열 내용 비교**

**- 배열**

**동일 타입 데이터를 여러개 모아놓은 참조형 변수**

**{true, false, true, false}**

**배열 선언**

**boolean b [];===> null**

**배열 생성**

**b = new boolean[4];===> null 아니다**

**b.length**

|  |  |
| --- | --- |
| **b:100** | **100:[true][false][true][false]** |

**배열 초기화**

**b[0] = true;**

**b[2] = true;**

**{1, 2, 3, 4}--->int ar[];**

**{1.0, 2.2, 3.456, 4.4}--->double ar[];**

**{1, 2.2, 3.456, 4}--->double ar[];**

**(자동형변환(기본형끼리만) – int -=-> double )**

**{"aaa", "가나다" } ==> String ar[]**

**{"b", "aaa", "가나다" } ==> String ar[]**

**- "동일" 타입 여러개 모음**

**- boolean int double String char**

**- 배열의 배열로 생성- 2차원 배열**

**4개 데이터 저장 1차원 배열을 3개 모은 2차원 배열 개념**

|  |  |  |  |
| --- | --- | --- | --- |
| **[0][0] 1** | **[0][1] 2** | **[0][2]3** | **[0][3] 4** |
| **5** | **6** | **7** | **8** |
| **9** | **10** | **11** | **12** |

**int two\_ar [][];**

**int [] two\_ar [];**

**int [] [] two\_ar = new int[3][4];**

**two\_ar.length ===> 3**

**two\_ar 타입 --> 2차원 배열**

**two\_ar[0] 타입 --> 1차원 배열**

**two\_ar[0].length ===> 4**

**two\_ar[1].length ===> 4**

**two\_ar[2].length ===> 4**

**two\_ar[3].length ===> error**

**two\_ar[0][1] = 2;==> 초기화**

**int two\_ar[][] = new int[3][];**

**two\_ar[0] = new int[1];**

**two\_ar[1] = new int[2];**

**two\_ar[2] = new int[3];**

**for(int j = 0; j < 3; j++){**

**two\_ar[j] = new int[j+1];**

**}**

|  |
| --- |
| **1** |
| **5** | **6** |
| **9** | **10** | **11** |

**String-문자열**

**배열- 동일타입 데이터 여러개**

**열거타입- Enum 타입. 데이터 여러개- x**

**계절 = 봄 여름 가을 겨울**

**월 = 12개 ..**

**요일 = 7개 . 한정**

**-객체와 클래스**

**20분**

**1>1995 년 자바 언어 탄생**

**c-> c++ ->java -> c#,.....**

**2> 객체지향언어 / 1980 c cobol 절차지향언어**

|  |  |
| --- | --- |
| **절차지향언어**  **성적처리프로그램{**  **학생이름, 과정명, 점수-데이터**  **수강하다(){}-절차**  **시험보다(){}**  **평가하다(){ 수정구현 }**  **}//1파일**  **취업관리프로그램{**  **학생이름, 회사이름, 과정명**  **수강하다(){}**  **회사설명회하다(){}**  **}** | **객체지향언어**  **학생{**  **학생이름, 과정명**  **수강하다(){}-절차**  **시험보다(){}**  **}//1파일**  **평가자{**  **점수**  **평가하다(){ 수정구현 }**  **}**  **성적처리프로그램{**  **학생.수강하다(){}-절차**  **학생.시험보다(){}**  **평가자.평가하다()**  **}**  **취업관리{**  **학생.수강하다();**  **회사.설명회하다();**  **}**  **회사{**  **회사명, 급여, 위치, 직원수**  **설명회하다(){......}**  **}** |

**1> 수정 = 유지보수 쉽다**

**2> 1번 작성 코드 재사용 가능**

**= 반복적 정의하지 않고 간결성**

**- 객체**

**현실세계 컴퓨터내부 구현 자바 단계**

**- 현실세계 명사 표현-객체**

**컴퓨터, 회사원, 학생, 수강생, 딸, 아들,**

**1>객체 특성 파악**

**정적 특성=상태, 정보, 속성 표현 데이터들**

**동적 특성=행위, 동작, 기능 (변화)**

**허진호객체=교육생 구현 -**

|  |  |
| --- | --- |
| **이름=허진호**  **나이**  **수강명**  **위치**  **일시**  **수료여부**  **성적** | **수강하다**  **등하원하다**  **휴식하다**  **졸다**  **점심먹다** |

**허진호객체 회사원 구현**

|  |  |
| --- | --- |
| **이름=허진호**  **나이**  **급여**  **위치**  **직급**  **부서** | **일하다**  **출퇴근하다**  **휴식하다**  **졸다**  **점심먹다** |

**100000분 모두 교육생객체 표현**

**동일 정적 + 동적 특성 객체 무수히 많이 존재**

**===> 객체 표현 template 정의**

**붕어빵틀 1개 ====> 붕어빵 여러개**

**2>클래스 정의**

**==> 객체 표현 template**

**교육생 클래스 정의**

|  |  |
| --- | --- |
| **class 교육생 {**  **정적특성 = 변수 정의**  **String 이름;**  **int 나이;**  **String 수강명;**  **String 위치;**  **Calendar 일시;**  **boolean 수료여부;**  **double 성적;** | **동적특성=메소드 정의**  **수강하다(){.......}**  **등하원하다(){...}**  **휴식하다(){}**  **졸다(){}**  **점심먹다(){}**  **}** |

**3> 객체로 생성**

**=클래스 변수와 메소드 복사하여 메모리 저장**

**교육생 s = new 교육생();**

**교육생 s2 = new 교육생();**

**교육생 s3 = new 교육생();**

**대문자 명사 , 숫자시작 x, 키워드 x, \_, $**

**modifier- 활용요소앞 선언 가능.**

**사용방법이나 접근제한 지정 역할 키워드들**

**- 클래스 문법**

**6 장 클래스**

**7장 extends**

**8장 implements**

**[modifier] class 이름 [extends ...][implements ...]**

**==>클래스 선언**

**{ 클래스 몸체**

**[modifier] 변수=> 변수타입 이름 = 값;**

**[modifier] 생성자**

**[modifier] 메소드**

**}**

**- 회사원 클래스 작성 / 객체들 생성**

**new – heap 영역 해당데이터값 저장 공간 생성**

**변수와 메소드 복사 저장 공간 생성**

**int i = 10;==>기본형변수**

**int [] ar = new int[5];**

**String s = new String("java");(String s = "java" 가능)**

**===> 참조형 변수로 취급**

**패키지 – package 키워드로 표현**

**1> class 정의 = 1개 파일 단위**

**2> 100개 클래스-폴더 단위(파일저장묶음)**

**3> 패키지 - 유사한 성격의 클래스 모아놓은 폴더**

**폴더 – win**

**디렉토리 - unix linux mac**

**board - 10개 게시판관련클래스들**

**package board;**

**class BoardWriting{}**

**member - 20개 회원관리관련클래스들**

**shop - 80개 쇼핑몰클래스들**

**private – 변수,메소드,생성자앞 선언.**

**사용 영역을 현재 클래스 제한.**

**no modifier – 클래스, 변수,메소드,생성자앞 선언.**

**사용 영역을 현재 패키지 제한**

**protected - 변수,메소드,생성자앞 선언.**

**사용 영역을 현재 패키지 + 다른 패키지의 상속받은 클래스로 제한.**

**public - 클래스, 변수,메소드,생성자앞 선언.**

**사용 영역을 무제한**

class 이름{

멤버변수

타입 변수명;

메소드

[modifier] 리턴타입 메소드명( [매개변수] ) {

수행 기능 문장 ; ;

}

}

- 리턴타입

[modifier] 리턴타입 이름 ([매개변수])

public > protected > no modifier > private

class A{

private int i ;

}

class B{

public static void main(String args[]){

A a1 = new A();

a1.i = 100;

s.o.p(a1.i);

}

}

class Login{

String id;

private String pw ;

//setter 메소드 / getter 메소드들

public void setPw(String newpw){

pw = "1234";

}

public String getPw(){

return pw(암호);

}

class LoginTest{

public static void main(String[] args){

Login l = new Login();

l.id = "multi";

l.setpw("campus");

......

String pw = l.getPw();

}

}

- 매개변수

1> 메소드 정의

결과물 메소드이름(입력값전달){

기능수행문장;

결과물 리턴;

int ma(int j){

int i = 10;

return j\*j;

}

2> 메소드 호출 실행 가능

ma(10);

ma(20);

ma(30);

Call c = **new** Call();

**int** j = 10;

c.add(j);

System.***out***.println

("c.add(j) 호출 후의 j = " + j);

|  |  |
| --- | --- |
| i:11  j:10  c:100 | 100:  void add(int i){  s.o.p(i++)//10  } |

기본형 매개변수로 전달시에는

메소드 i변수로 j변수값 전달한 이후에

i 변수값 변경되어도 j 변경되지 않는다.

Call c = **new** Call();

MyObject obj = **new** MyObject();

c.addObject(obj);

System.***out***.println

("c.addObject(obj) 호출 후의 obj.a = " + obj.a);//11

|  |  |
| --- | --- |
| mo:200  obj:200  c:100 | 100:  add(int i){.....}  addObject(MyObject mo){s.o.p(mo.a++);}  //10  200:  int a = 11 |

참조형변수 매개변수 전달시에는 주소값 복사 전달

obj 주소값을 mo 복사 전달

obj, mo 같은 객체 참조

obj.a mo.a -->동일

call by value 방식

-"실제값 |주소값 "복사 전달

Employee e1 = **new** Employee();

클래스명 객체참조변수명 = new 클래스명();

e1.id="100"; e1.name=" 최대리"

e1.title = "대리" e1.dept="인사부"

객체참조변수명.멤버변수

Employee e2 = **new** Employee();

e2.id="200";

e2.name = "최사원";

e2.title = "신입사원";

e2.dept = "it 개발부";

|  |  |
| --- | --- |
| e2=200  e1 =100 | 100:  id="100"  name="최대리"  title="대리"  dept="인사부"  200:  id="200";  name= "최사원";  title= "신입사원"  dept="it 개발부" |