<Review>

program 언어

1. 인식/ 실행을 위한 개발환경 구축

2. 개발(현재는 cs 기반보다는 Web 기반, mobile 기반으로 많이 다뤄지며, 그 중추가 되는 프로그램 언어가 JAVA다.

  cf) 혼자서 독자적으로 개발하는 프로그램 : SE 프로그램

- Web에서 개발환경을 자바 DB 과정이후 배울것이다. But, SE 문법이 기반이 됨.

3. 문법 : 변수 선언 / 변수 초기화 / data type(기본 8개, 그외 참조 Data)

명령문 / 제어문, 반복문

1. if문, if/else문, if/elseif/else문 // else 구문만 따로 독자적인 사용은 불가능 하다.

2. 반복문

1) for(변수선언 및 초기화; 조건식; 증감식){

명령문

}

 2) 변수 선언 및 초기화

while(조건식){명령문, 증감식}

 3) 변수 선언 및 초기화

do{

명령문, 증감식

}while(조건식);

3. switch문

4. continue / break 사용처

#. 배열 : 무조건적으로 동일한 타입이 쓰여야 하며, 배열의 크기가 정해지면, 변경할 수 없다.

- 문법 : Web P 에서 더욱 민감! why? >> 실행속도와 메모리에 직접적으로 영향을 끼치기 때문이다.

- JAVA에서 만들어진 메모리는 삭제 불가, but 삭제할 수 있는 환경은 구축 가능.

- Garbage : 메모리는 할당되어 있지만, 사용하지 못하는 객체

cf) java Test // Test.class 파일을 실행하세요.

>> JVM(JAVA Vutural Machine)에서 실행

① classLoader // 메모리 로딩

② 유효성 검사(소스코드가 구동 가능하게 짜여져 있는지 검사)

③ ByteCode >>>Compile>>> MachineCode

But, 메모리 부족시, GarbageCollector 실행됨 // Garbage(참조가 끊긴 객체들)을 삭제시켜줌.

- 1차원 배열 ex) new type [숫자];

- 2차원 배열 ex) new type[행][열];

cf) 과거 은행에서 COBOL 프로그램 사용 >> 차세대 프로젝트 >> 현재 JAVA 사용

객체지향 언어 : 유지보수성이 뛰어남

프로그램 구동 방법의 차이

- C : 절차적 프로그램 언어

- JAVA : 객체지향 프로그램 언어

C와 JAVA의 차이
- 객체 모델링을 진행하는 동안 특성과 관련된 것들(이름, 나이, 키, 무게. 색상 등)은 변수로 표현하고, 동작과 관련된 것들(이/착륙, 레이저빔, 미사일 등)은 함수로 표현하면 된다.

객체 모델링 & 클래스 다이그램
접근제한자(class or 변수 선언시 사용가능) : 메소드, 변수, class를 가져다 쓸수 있는 범위를 지정하는 값(접근 범위 설정)

1) private : 동일 클래스 내의 범위

2) (default) : 동일 패키지 내의 범위

3) protected : 동일 클래스 혹은 상속 내의 범위

4) public : 접근 제한 없음.

시스템의 분야 - 도메인으로 나뉨.

제어자 리턴값타입(매개 변수 선언(arg 값이 저장됨)){

명령문

}

class를 가져다 쓸 때

1. 메모리에 가져다 쓸 class 파일의 메모리 생성

2. new "class명"

cf JAVA 실행 시 사용되는 메모리 영역

Code Heap Stack
Method 영역 멤버 변수
new class() 입력 시 생성
참조가 끊긴 후(Garbage), Garbage Collector에 의해 삭제 지역 변수
메소드 실행 시 생성
메소드 종료시 삭제
>> Stack 메모리 타입만 프로그램 개발자가 접근 가능하다.

- 변수 선언

변수는 멤버 변수와 지역 변수로 나누어 볼 수 있는데, 메모리의 저장 위치, 생성 시점과 삭제 시점이 다르다. 데이터의 성격 또한 다르다.

- 지역변수 : 메소드 안에서 선언

- 멤버변수 : 클레스의 특성을 나타내는 값, 클레스의 멤버로 선언

① 클래스 선언

② 클래스 생성

③ 사용

- 클레스를 정리해두면 몇 번이고 명령문을 만들어 사용 가능하다.

- 인스턴스 - 실제로 메모리에 만들어지는 객체

cf) 인스턴스

생성(새로운 객체를 가져다 사용하는 경우) : new 클래스명()