**JAVA**

**1 자바 언어의 개요**

1-1 자바 언어의 특징

1-1-1 자바의 역사

1991년부터 제임스 고슬링을 구축으로 가전제품을 제어하는 아주 단순하고 버그 없는 프로그래밍 언어 자바를 개발하기 시작.

(제임스 고슬링은 앞으로 하나의 거대한 네트워크로 연동될 것이라고 예측하였고, 사용 중이던 가전제품들의 기능이 향상되어 내장 프로그램을 업그레이드할 때마다 네트워크를 통해 서버로부터 새로운 프로그램을 다운받아 가전제품의 기능을 향상시키고자 하였다.)

1-1-2 자바 언어의 특징

(1) 단순하다. : C++/C언어와 문법적으로 유사, C++의 복잡성을 단순화.

(2) 객체지향이다. : 객체 개념을 적용하는 완벽한 객체지향 언어.

(3) 플랫폼 독립적이다. : 플랫폼은 프로그램이 실행될 수 있는 하드웨어나 소프트웨어 (운영체제기반 환경으로 자바 프로그램은 자바 가상 머신(vm)에 의해 실행되므로 플랫폼에 독립적.

(4) 컴파일러와 인터프리터 언어의 두 가지 특징을 모두 갖는다.

(5) 분산 처리에 용이하다. (네트워크에 강함, 프로토콜이 사용 가능한 라이브러리 제공)

(6) 견고하다.

-포인터 사용X

-자동으로 가비지 컬렉션 기능을 수행

-엄격한 데이터형 검사를 통해 에러를 조기에 발견

-실행 시간에 발생하는 에러를 처리

(7) 안전하다. (보안에 강함)

(8) 멀티스레드 지원. (속도가 빠르다)

(9) 동적이다. (라이브러리의 확장이 용이하다)

1-2 자바 개발 환경

=>자바에는 다른 프로그래밍 언어와 달리 하드웨어나 운영체제에 관계없이 실행 가능한 자체 플랫폼이 있다, 이를 자바 개발 환경(JDK : Java Development Kit)라고 한다.

|  |
| --- |
| **자바 프로그램** |
| **Java API**  자바 개발 환경 |
| **JVM** |
| **하드웨어 의존적 플랫폼**  **(윈도우, 유닉스, 매킨토시)** |

1-2-1 자바 개발 환경 구조

* JVM (Java Virtual Machine : 자바 가상 머신)

컴파일 후 생성되는 바이트 코드를 다양한 컴퓨터 플랫폼에 맞는 기계어로 해석하여 실행 ->자바 프로그램은 플랫폼에 관계없이 실행 가능

* Java API (Java Application Programming Interface)

자바가 지원하는 클래스 라이브러리로, 프로그래밍에 필요한 기능들을 클래스로 구현하고 서로 관련 있는 클래스를 패키지 단위로 묶어서 제공한다.

1-2-2 자바 개발 환경 구조

=>JDK1.2 (JAVA2) 부터 개발되는 자바 프로그램의 성격에 따라 J2SE(Java2 Standard Edition), J2EE(Java2 Enterprise Edition), J2ME(Java2 Micro Edition) 3가지 버전의 자바 개발 환경이 있다.

|  |  |
| --- | --- |
| 종류 | 특징 |
| J2SE | * 일반 자바 애플리케이션을 개발하기 위한 기본 자바 개발 환경 |
| J2EE | * 서버 관련 프로그램을 개발하기 위한 자바 개발 환경 * 웹 서버 환경에서 많이 활용됨(JSP, Servlet, EJB 등) |
| J2ME | * 다양한 제품과 임베디드(Embedded) 기기 프로그램을 개발하기 위한 자바 개발 환경 * 모바일 환경에서 많이 활용됨(PDA, 핸드폰 등) |

+자바 설치

자바개발환경 다운로드 받기-설치-환경 변수 설정하기- JAVA API설치하기

1-3 자바 프로그램의 종류

1. 일반 프로그램

자바 애플리케이션

1. 웹 클라이언트 프로그램 : 애플릿

웹 브라우저로 접속한 클라이언트 컴퓨터에 자바 프로그램(애플릿)을 전송하여 클라이언트 쪽에서 프로그램이 실행되는 기술.

1. 웹 서버 프로그램 : 서블릿과 JSP

웹 브라우저로 클라이언트가 요청하면 웹 서버 쪽에서 자바 프로그램(서블릿, JSP)을 실행하고 그 실행 결과만 HTML로 작성하여 클라이언트 컴퓨터에 전송하는 기술

1. 웹컴포넌트 : EJB

EJB는 네트워크 상에 필요한 객체를 분산시켜 두고 이 객체를 자바 애플리케이션에서 불러 쓸 수 있도록 하는 분산 객체 기술. J2EE는 분산 객체 환경을 제공하여 기업 등의 웹 개발 시에 필요한 자바 플랫폼. (EJB는 J2EE에 속함)

1. 모바일과 임베디드

휴대폰으로 영화를 보거나 무선 인터넷 서버로부터 게임을 다운로드하여 무선기기에서 사용하는 기술로 무선 환경뿐만 아니라 TV, 자동차 내비게이션 등 임베디드 분야(특정 기능의 프로그램이 내장된 하드웨어)로 개발 분야가 확대되고 있다.

1-4 자바 프로그램

1-4-1 자바 프로그램의 기본 구조

Public class 클래스 이름{

Public static void main(String args[]){

System.out.println(“ ”);

//한 줄 주석 (주석이란 프로그램 실행에는 영향 없는 메모기능)

/\*여러 줄 주석\*/

}

}

1-4-2 cmd 단축기

- dir : 보여주기

- javac 파일이름.java : 번역(컴파일)

- java 파일이름 : 실행

1-4-3

* + 자바 프로그램은 클래스들의 집합이다.
  + 클래스 중 하나는 반드시 main( ) 메서드를 포함한다.
  + VM은 실행 시 main( ) 메서드를 가장 먼저 호출한다.
  + 클래스명 앞에 public 키워드를 추가하면 그 클래스의 이름을 파일명과 동일해야

하는 규칙이 있다.

* + 클래스 이름은 항상 대문자로 시작한다.

1-4-4 오류

* 번역오류 : 번역 과정에서 발생하는 문법적 오류
* 실행오류 : 잘못된 실행을 하는 경우 프로그램의 동작이 멈추는 경우
* 논리오류 : 원하는 결과 출력X, 틀린 결과O. 프로그램의 설계상의 문제가 있는 경우가

대부분=>디버깅을 수시로 하는 것이 중요.

­­­1-4-5 식별자 프로그램 구성요소를 나타내기 위해 사용하는 단어

>식별자의 3가지 유형

1. 다른 프로그래머가 선택한 단어
2. 프로그래머가 선택한 단어
3. 예약어 (제임스 고슬링)

**2 자바 언어의 기본 문법**

2-1 변수

2-1-1 변수

- 기억 장소의 이름을 ‘변수’라고 함. => 데이터는 필요에 따라 변할 수 있기 때문에 변수.

>작성규칙

- 알파벳 대/소문자, 숫자, \_ , $만 사용가능 (일반적으로 소문자로 시작)

- 예약어는 사용 못함. - 대소문자 구분

- 길이 제한X - 첫 문자로 숫자는 X

2-1-2 예약어

- 모두 소문자이며 대/소문자 구분. – C/C++의 sizeof는 사용되지 않는다.

- const, goto는 현재 사용되지 않으나, 변수명으로는 사용할 수 없다.

2-2 데이터형 프로그램에서 사용되는 여러 종류의 데이터를 분류하여 형으로 규정한 것.

<기본형과 참조형으로 나뉜다. >

1. 기본형

변수에 실제 사용되는 데이터가 저장된다.

|  |  |  |  |
| --- | --- | --- | --- |
| 종류 | 데이터형 | 바이트 수 | 범위 |
| 논리형 | boolean | 1바이트 | True 또는 false |
| 단일 문자형 | char | 2바이트 | ‘\u0000’ ~ ’\uffff’ |
| 정수형 | byte | 1바이트 | -2­7­ ~ 2­­­7-1 |
| short | 2바이트 | -215~215-1 |
| int | 4바이트 | -231~231-1 |
| long | 8바이트 | -263~263-1 |
| 실수형 | float | 4바이트 | 1.4e-45 ~ 3.4028235e38 |
| double | 8바이트 | 4.9e-324 ~1.7876931348623157e308 |

논리형 (Boolean) : C/C++에서는 정수값을 이용하여 0일 때는 거짓을 나타내고 0 이외의 값일 때는 참을 나타내도록 하지만, 자바는 true와 false만 제공. 따라서 다른 데이터형으로 변환X, 대/소문자 구분하므로 TRUE와 FALSE는 논리형이 아니다.

문자형 (char) : 비영어권 국가를 위해서 2byte의 유니코드를 기반으로 단일문자를 표현. 반드시 따옴표로 감싸줘야 한다.

정수형 (byte, short, int, long) : 소수점이 없는 숫자를 표현하는 자료형. C/C++과 달리 자바는 시스템에 관계없이 int형이 4byte의 크기를 갖는다.

실수형 (float, double) : 소수점이 있는 숫자를 표현하는 자료형. 실수형에 데이터는 기본으로 8byte인 double로 표현. float으로 표현하려면 숫자 뒤에 ‘f’또는 ‘F’를 붙여야 한다.

1. 데이터형 변환 : 이미 선언된 데이터를 다른 데이터형으로 변환하는 것.

자동 형 변환 : 데이터 손실 없이 안정적으로 형 변환이 가능할 경우에만 내부적으로 자동 형 변환을 수행. Ex) byte -> int

강제 형 변환 : 더 작은 범위를 나타내는 데이터형으로 변환하는 것이다. 축소 형 변환이라 데이터의 손실 가능성이 있으므로 반드시 형 변환 연산자를 사용해서 명시.

1. 데이터 타입 (참조형)

데이터가 저장되어있는 메모리 주소를 기억, 이 주소를 통해 해당 데이터를 참조.

(object -> 대표적인 참조타입, 객체)

|  |  |  |
| --- | --- | --- |
|  | 기본 자료형 변수 | 참조 자료형 변수 |
| 정의 | 기본 자료형 값을 가진다. | 값에 대한 참조, 즉 주소를 가진다. |
| 종류 | 정수, 실수, 문자, 논리값 등 | 배열, 클래스 참조, 변수, 인터페이스 등 |
| 예 | Int a=100  - a라는 변수에 실제 100이라는 값이들어감. | Int [ ] a = new int [3];  a[0]=10; , a[1]=20;  - a라는 변수에는 a[0] 값이 저장된 위치의 주소값이 저장. |

2-3 상수

> 프로그램의 실행 중 변화하지 않는 값.

>정수형, 실수형, 문자형 (‘ ‘), 논리형 상수가 있다.

>final 예약어를 변수타입 앞에 쓰며 상수명은 대문자로 작성한다.

- 상수 사용에 유용한 점

1. 불분명한 값에 의미를 부여

2. 프로그램 수정 용이

3. 프로그래머에 의해 잘못 입력되는 것을 방지

2-4 연산자와 우선순위

|  |  |  |
| --- | --- | --- |
| + | a + b | 덧셈 |
| - | a - b | 뺄셈 |
| \* | a \* b | 곱셈 |
| / | a / b | 나눗셈 |
| % | a % b | 나머지 |

1. 산술 연산자

가감승제를 표현할 수 있도록 산술 연산자를 제공.

% 연산은 피연산자가 정수형이어야만 하는

C/C++과 달리 실수형도 가능.

|  |  |  |
| --- | --- | --- |
| > | a > b | 크다 |
| >= | a >= b | 크거나 같다 |
| < | a < b | 작다 |
| <= | a <= b | 작거나 같다 |
| == | a == b | 같다 |
| != | a != b | 같지 않다 |

1. 관계(비교)연산자

피연산자의 크기나 객체의 타입 비교 등에 사용.

비교한 결과에 따라 true, false의 논리형 결과를 반환.

|  |  |  |
| --- | --- | --- |
| & | a & b | a와 b가 모두 T인 경우에만 T. 나머지 경우 F. |
| && | a && b | a와 b가 모두 T인 경우에만 T. a가 F인 경우 F. |
| | | a | b | 둘 중 하나만 T여도 T. 둘 다 F면 F. |
| || | a || b | 둘 중 하나만 T여도, T. a가 T일때, b 수행X |
| ! | !a | a가 T면 F, a가 F면 T. |

1. 논리 연산자

주어진 조건식이 참인지

거짓인지를 판단하여

true or false의 논리형

결과를 변환.

|  |  |  |
| --- | --- | --- |
| ++ | ++a | a값 1 증가 후 수식 적용 |
| a++ | 수식 적용 후 a값 1 증가 |
| - - | --a | a값 1 감소 후 수식 적용 |
| a-- | 수식 적용 후 a값 1 감소 |

1. 증감 연산자

증감 연산자는 변수의 값에 1을 증가 또는

감소 시킨 후 그 값을 다시 변쉥 저장하는

연산자.

1. 비트 연산자

피연산자의 비트 별로 연산을 수행. 피연산자로는 실수형을 제외한 산술 데이터를 취함.

* & 연산은 특정 비트를 0으로 만들기 위해 사용.
* | 연산은 특정 비트를 1로 만들기 위해 사용.
* ^ 연산은 특정 비트를 비트 반전시키기 위해 사용.
* ~ 연산은 1의 보수를 만들기 위해 사용.
* >>, <<, >>> 연산은 비트를 이동시켜 곱셈 나눗셈에 응용.

|  |  |  |
| --- | --- | --- |
| = | a = b | b를 a에 대입 |
| += | a += b | a 값에 b를 더한 후, a에 대입 |
| -= | a -= b | a 값에 b를 뺀 후, a에 대입 |
| \*= | a \*= b | a 값에 b를 곱한 후, a에 대입 |
| /= | a /= b | a 값을 b를 나눈 후, a에 대입 |
| %= | a %= b | a 값을 b를 나눈 후 나머지를, a에 대입 |

1. 대입 연산자

대입 연산자는 우변식의

최종 결과를 좌변의

기억 장소에 대입하는 연산자.

1. 조건 연산자

조건 연산자는 조건문 if-else를 축약하여 사용할 수 있는 연산자. 세 개의 피연산자를 취하는 삼항 연산자. 첫 번째 피연산자는 true, false값을 갖는 논리식이어야 하며, 두 번째와 세 번째 피연산자는 같은 데이터형을 갖도록 해야한다.

|  |  |  |
| --- | --- | --- |
| ?: | 조건 ? 처리1 ; 처리2 | 조건이 참이면 처리1을 수행하고 거짓이면 처리2를 수행. |

1. 연산자 우선 순위

2-5 제어문

2-5-1 선택문

1. if 문

조건에 따라 선택적으로 문장이 수행.

if나 else절애 실행해야 하는 문장이 여러 개일 경우 블록 { }으로 감싸준다.

if (조건식){

~참일 때 문장

}else {

~거짓일 때 문장

}

1. switch ~ case문

switch문은 여러 갈래로 분기되는 if문과 유사한 구조를 갖는다.

Switch( 조건식) {

Case 상수1 : 실행문 ; [ break; ]

Case 상수2 : 실행문 ; [ break; ]

Case 상수3 : 실행문 ; [ break; ]

[ default : 실행문; ]

}

------------------------------------------\* 시험범위는 아님!! \*--------------------------------------------

제어문 (for 반복문)

1. 반복 변수 초기화
2. 조건체크 -> 참이면 실행, 거짓이면 종료
3. 실행 후 반복 변수 증가

Ex) for ( i=1; i<10; i++ ) -> 9번 실행.

>for 실행 순서

초기식 -> 조건 비교 -> 참이면 실행문 실행 -> 증감식 ->거짓이면 빠져나가기

>따라서 처음 조건식 비교에서 거짓이면 for문을 한 번도 실행하지 않을 수 있음.

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