

제 1강 참고 자료 - 자바의 특성

류대우

davidryu@newtc.co.kr

작성일 : 2007년 03월 01일

자바의 특성

자바를 개발한 선 마이크로는 자바를 아래와 같이 정의 하고 있는데 여기에 사용한 단어들이 자바의 특징을 잘 나타내 주고 있다.

Java: A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language

자바는 매우 간단하다. 자바는 C++에 가깝지만 훨씬 간단하다. 자바에는 고급 언어들에 들어 있는 여러 가지 요소들 중에서 반드시 필요하지 않다고 생각된 부분들은 모두 빼 버렸다.

객체 지향적이다. 자바는 숫자(int, float, long 등)나 논리값(true, false)을 제외한 거의 모든 것이 객체로 구성되어 있다. 실제로 자바는 Object 클래스에서 모든 클래스를 파생한다.

C++과 비교할 때 자바는 단일 클래스 상속만 지원하여 다중 클래스 상속 시 발생할 수 있는 개념의 혼란을 막았다. 또, 자바는 연산자 오버로딩과 함축적 유형 변환을 지원하지 않는다.

분산 환경을 지원한다. 자바에는 TCP/IP 라이브러리가 기본적으로 포함되어 있다. 또한 http와 ftp 프로토콜을 기본적으로 지원한다.

인터프리터 언어이다. 자바는 정확하게 말하면 컴파일 언어인 동시에 인터프리터 언어이다. 자바는 먼저 텍스트 소스를 컴파일 하여 2진 파일(클래스 파일)로 만든 다음 자바 런 타임이 클래스 파일을 인터프리트 하면서 실행한다. 먼저 시스템에 무관한 2진 파일을 만듦으로써 자바는 컴파일 언어에 가까운 속도와 시스템 독립성을 동시에 얻을 수 있었다.

강고하다. 이것은 자바가 C++과 마찬가지로 컴파일 때에 유형 검사를 하여 에러를 막을 뿐만 아니라 도구들까지 명확한 유형 선언을 하도록 한다. 자바는 포인터 개념에 있어 포인터 연산을 지원하지 않는다. 이는 잘못된 주소를 가리킬 가능성을 사전에 없앤 것이다. 자바는 포인터를 표현하기 위해 실제 배열을 사용하여 메모리 경계를 범할 가능성을 없애고 메모리 검사를 가능하게 하였다. 자바는 모든 메모리 접근을 자바 시스템이 관리하고 제한하며 또한 예외 핸들링을 하여 시스템 붕괴의 우려가 없다. 예를 들어 자바는 리소스 관리(garbage collection)를 하는데 사용이 끝난 리소스를 시스템이 메모리에서 삭제하는 방식을 채택하고 있어 메모리 누출에 대한 고민을 프로그래머가 할 필요가 없다.

자바는 안전하다. 자바는 포인터 개념이 없고 유형 정의가 강고하여서 실행 전에 클래스 파일을 이용한 프로그램의 검사가 가능하다. 실제로 웹 브라우저가 애플릿을 실행하기 전에 보안 검사를 한다.

기계 종립적이다. 웹의 서버-클라이언트 환경에서 이 기종 서버-클라이언트의 지원은 매우 중요한 문제이다. 자바는 웹을 기본 환경으로 하고 있기 때문에 처음부터 이 문제를 염두에 두고 만들어졌다. 자바의 실행 파일은 기계 종립적인 이진 코드(클래스) 파일이다. 이 클래스 파일을 기계 종속적인 자바 런타임이 인터프리트 하여 실행시키는 것이다. 따라서 자바 런타임이 설치된 시스템에서는 어디서나 자바 프로그램을 실행할 수 있다.

이식성이 높다. 자바는 유형 정의를 시스템에 무관하게 정의하고 있다. 따라서 C++ 에서 처럼 시스템에 따라 동일한 유형(예를 들어 int)이 다르게 사용되는 경우가 없다. 또한 기계 종립적인 자바의 특성은 곧바로 자바의 높은 이식성을 말해 준다.

높은 수행능력을 제공한다. 자바는 기계 종립적인 이진 코드로 미리 컴파일된 것을 실행 때 기계어 코드로 번역하여 수행하는 방식을 채택하고 있고 또 이진 코드는 기계어로 번역될 것을 감안하여 설계되었으므로 그 수행능력이 C나 C++ 컴파일러에 의한 기계어 코드에 비견할 만하다. 현재 자바 이진코드의 수행속도는 C 컴파일러의 약 20분의 1이라고 한다.

멀티 쓰레딩을 지원한다. 윈도우 NT, 윈도우95나 솔라리스는 쓰레드를 지원하는 운영 체제들이다. 윈도우 3.1은 멀티 쓰레드를 지원하지 않으므로 자바 런타임을 구현할 수 없다. 쓰레드는 하나의 프로세스 인스턴스(인스턴스는

객체의 실제 변수로 생각하면 된다.)가 실행되는 경로로 정의되는데 멀티 쓰레드를 지원할 경우 하나의 프로그램 단위가 동일한 쓰레드를 동시에 수행할 수가 있다. 보통 사용자 인터페이스 쓰레드와 장시간 계산이 필요한 쓰레드가 동시에 필요할 때 멀티 쓰레드가 효과적이다. 특히 자바는 멀티 프로세서 하드웨어를 지원하도록 설계되었으므로 멀티 CPU 시스템에서 높은 효율을 낼 수 있다.

동적이다. 자바 인터페이스를 이용하면 하나의 모듈을 갱신할 때 다른 모듈을 모두 갱신할 필요가 없다. 이것은 인터페이스가 모든 인스턴스 변수와 도구의 실행문을 배제한 채 객체 간의 상호 작용을 정의하기 때문이다.