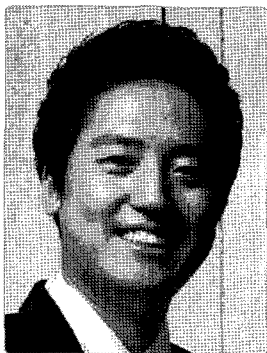
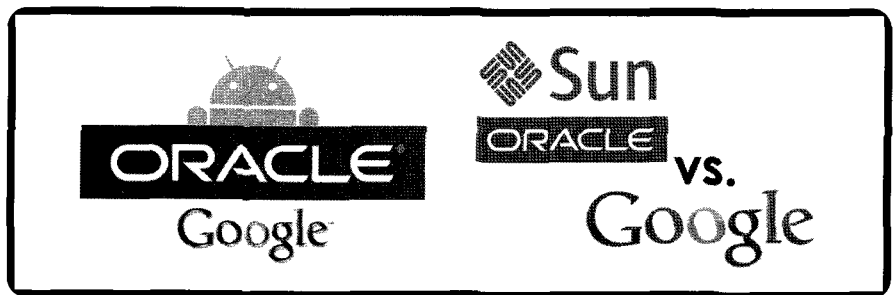


Oracle Vs. Google 특허분쟁

오라클은 지난해 자바기술과 특허를 보유한 썬마이크로시스템즈를 인수하면서, 구글의 안드로이드가 자바 특허기술을 침해했다며 구글을 상대로 고소장을 제출하였다. 오라클은 이번 특허소송에서 자바표준기술에 대한 특허 7건을 구글이 침해했다고 주장하고 있다.



김 병 년 선임연구원
한국특허정보원 표준특허센터



최 근 삼성전자와 애플의 특허전쟁에 대한 관심이 매우 뜨겁다. 하지만, 그에 못지않게 전 세계 많은 사람들이 큰 관심을 보이는 특허전쟁이 또 하나 있다. 바로 오라클과 구글의 특허전쟁이다.

오라클은 지난해 자바기술과 특허를 보유한 썬마이크로시스템즈를 인수하면서, 구글의 안드로이드가 자바 특허기술을 침해했다며 미국 샌프란시스코 연방법원에 구글을 상대로 고소장을 제출하였다.

당초 두 업체의 소송전은 비공개로 진행돼 왔으나, 최근 오라클이 해당 소송내용을 공개하는 것을 법원에 요청했고, 이에 해당 내용이 공개되었다. (2011년 6월)

오라클은 최초 61억 달러의 피해배상액을 구글에 요구했고, 구글은 터무니 없는 금액이라며 오라클의 주장을 일축했다.

지난 2011년 9월 20일, 21일 양사의 대표들이 만나서 협상을 진행했으나, 이견

을 좁히지 못한 것으로 알려졌다.

외신들은 오라클이 당초 61억 달러 피해배상액에서 한 발 물러나 11억 6천만 달러를 요구했다고 보도했다. 그러나 구글은 이를 거부하고 1억 달러 이상은 낼 수 없다고 말했다고 한다. 구글이 말하는 '1억 달러'는 법원에서 오라클과 합의를 하라고 권고했기 때문에 나온 금액이며 자신들은 여전히 자바 특허를 침해하지 않았다는 입장을 표명하고 있다.

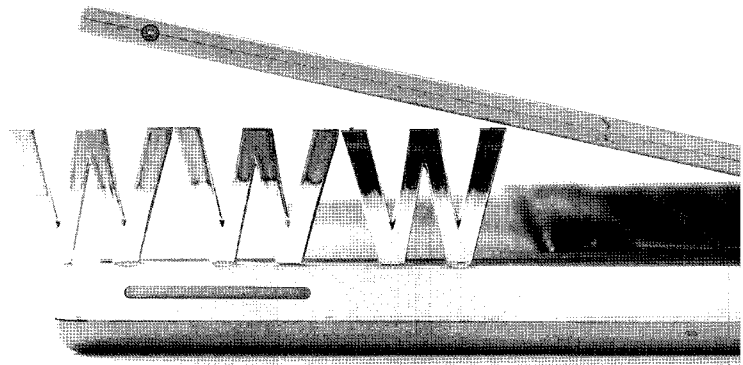
향후 어떠한 결과가 나올지는 아직 모르겠지만, 중요한 것은 오라클이 61억 달러라는 엄청난 피해보상액을 요구했다는 것이다. 물론, 최초에 피해보상액을 최대한 부풀려서 요구했을 가능성이 있다. 하지만, 최근 협상에서 요구한 11억 6천만 달러라는 금액 역시 엄청난 금액임에는 틀림없다. 이렇게 엄청난 액수의 피해보상금을 요구할 수 있었던 것은 바로 "자바표준특허"를 보유하고 있기 때문이다.

오라클은 이번 특허소송에서 자바표준기술에 대한 특허 7건을 구글이 침해했다고 주장하고 있다.

History

오라클은 2010년 자바기술과 특허를 보유한 썬마이크로시스템즈를 74억 달러에 인수하였고, 그해 8월 미국 샌프란시스코 연방법원에 구글을 상대로, "구글이 만든 모바일 운영체제(OS) 안드로이드가 우리의 자회사인 썬마이크로시스템즈 자바 특허기술을 침해했다", "구글은 오라클의 139개의 특허 청구항을 침해했으므로 61억 달러의 피해배상액을 지불하라"는 내용의 고소장을 제출하였다.

오라클이 고소장을 제출하자마자 구글은 반박에 나서며, "안드로이드는 썬의 자바 특허를 침해하지 않았고, 우리가 사용한 자바기술들은 이미 오픈소스에 가까운 것들이다", "안드로이드는 아파치소프트웨어재단(Apache Software Foundation, ASF)의 아파치 하모니 자바 임플리멘테이션(Apache Harmony Java Implementation, AHJI)의 서버셋을 사용했다"고 주장하였다. 구글 스코트



인개트너 고문변호사는 "오라클이 우리에게 요구한 61억 달러의 피해보상금은 터무니없는 금액", "그 이전에 61억 달러가 오라클에게도, 우리에게도 의미가 없다. 아직 끝나지 않은 것"이라는 성명서를 발표하였으며, 이후, 구글은 연방법원에 "오라클이 14~61억 달러의 피해배상액을 책정한 절차를 이해할 수 없으며, 그 정도의 가치가 있는 것도 아니다"라며, 오라클이 요구한 금액에 대한 반박서신을 보내기도 하였다.

지난 2011년 9월 20일, 21일 양사 대표가 미국 캘리포니아 산호세 연방법원에서 만나 협상을 진행하였는데, 협상은 비공개로 진행됐으며 결과가 공개되지는 않았으나 양사의 이견을 좁히지 못한 것으로 알려졌다.

자바표준 vs AHJI

AHJI(Apache Harmony Java Implementation)는 아파치 하모니 프로젝트에서 나왔고 ASF(Apache Software Foundation)가 개발하였다. ASF는 자바기술표준협회(JCP)에 소속돼 있었는데, 오라클이 썬을 인수하면서 오픈소스인 자바를 상업적인 용도로 사용하려고 하고 있어 더 이상 함께 활동하지 못하겠다고 밝히며, 2010년 11월 JCP에서 탈퇴하였다.

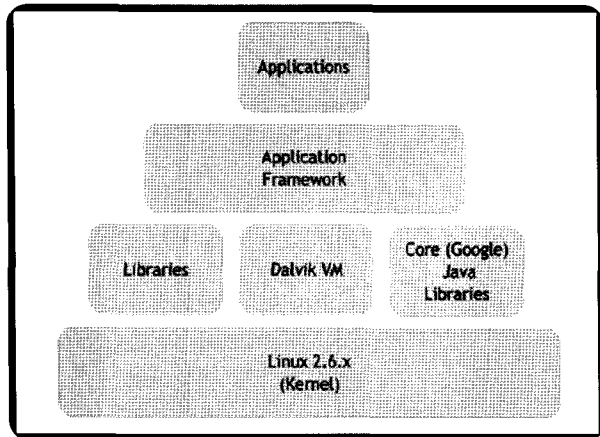
썬은 2007년 자바 스탠다드 에디션(Java SE)를 GPL(자유소프트웨어라이선스)로 공개했고, 그 이후 오픈소스라고 불리게 된다.(ASF는 썬이 자바를 공개하기 전부터 자바를 오픈소스로 만들려는 노력을 해 왔음)

그러나, 썬은 자바 모바일 에디션(Java ME)와 자바 엔

터프라이즈 에디션(Java EE)는 라이선스를 취득해야 사용할 수 있도록 했으면서, 자바 스탠다드 에디션을 모바일에 사용하지 못하도록 규제한다는 내용의 단서를 달았다.

상황이 이렇게 되자 구글은 썬의 자바 코드를 사용하지 않고 AHJI를 사용하게 되었는데, 오라클은 이를 인정하지 않고 구글이 자신들의 자바 코드를 사용해 부당이익을 취하고 있다고 주장하고 있다.

구글 “우리는 달빅가상머신 쓴다” Vs 오라클 “그것도 우리 기술 침해”¹⁾



구글이 자바의 특허를 침해하지 않았다고 주장하는 것에는 AHJI도 있지만 기저에는 자체적으로 개발한 달빅가상머신이 깔려있는 것도 그 이유 중 하나이다.

달빅가상머신(DVM)은 구글에서 자바가상머신(JVM)을 기초로 만든 새로운 가상머신인데, 레지스터 머신형태를 띄고 있으며 안드로이드에 탑재되어 있다. 달빅가상머신은 자바가상머신과 달리 낮은 메모리에서도 구동될 수 있도록 개발되었으며, 모바일 디바이스에 최적화된 가상머신으로 알려져 있다. 구글은 달빅가상머신에서 구동되는 코드(.dex)는 자바가상머신에서 돌아가는 코드와 전혀 다르기 때문에 ‘자바가상머신과 달빅가상머신은 다른 것으로 봐야한다’고 주장하고 있다.

그러나 오라클은 달빅가상머신이 자바 스탠다드 에디션을 기초로 JNI(Java Native Interface), OEM(자바 API)를 섞어서 만들었다고 주장하고 있으며, 2011년 4월 청문회에서 오라클 담당 변호사는 안드로이드 개발자들이 자바 내에서 코딩을 하고, 컴파일러를 통해 코드를 실행하는데, 비록 코드는 자바 가상머신이 아니라 구글의 달빅가상머신에서 구동되는 .dex 파일로 변환되지만 기본적인 원칙은 똑같다고 주장하고 있다.

분쟁특허 7건

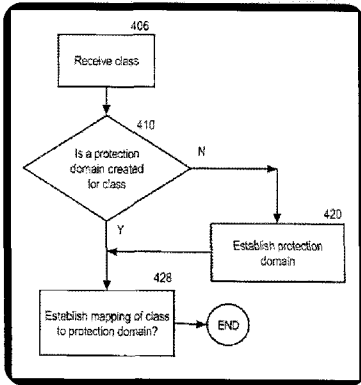
특허번호	발명의 명칭	출원년도
1 6,125,447	Protection domains to provide security in a computer system	1997
2 6,192,476	Controlling access to a resource	1997
3 5,966,702	Method and apparatus for pre-processing and packaging class files	1997
4 7,426,720	System and method for dynamic preloading of classes through memory space cloning of a master runtime system process	2003
5 RE38,104	Method and apparatus for resolving data references in generated code	1999
6 6,910,205	Interpreting functions utilizing a hybrid of virtual and native machine instructions	2002
7 6,061,520	Method and system for performing static initialization	1998

■ 분쟁특허 요지 리스트

NO. 1		US6125447	
발명의 명칭	Protection domains to provide security in a computer system		
출원번호	988439		
출원일	1997.12.11	출원인	Sun Microsystems
우선일	우선권주장번호		
IPC	H04L9/00	공개번호	

보호 도메인을 사용하여 보안 룰을 유지하고 강화시키기 위한 방법 및 장치

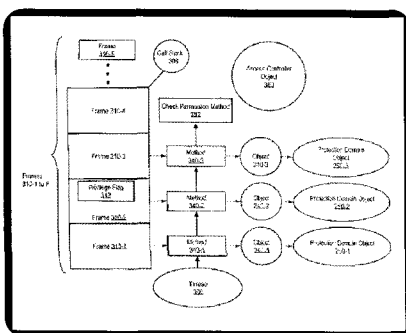
A method and apparatus are provided for maintaining and enforcing security rules using protection domains. As new code arrives at a computer, a



determination is assigned to a protection domain based on the source from which the code is received. The protection domain establishes the permissions that apply to the code. In embodiments where the code to be executed by the computer belongs to object classes, an association is established between the protection domains and the classes of objects. When an object requests an action, a determination is made as to whether the action is permitted based on the class to which the object belongs and the association between classes and protection domains.

determination takes into account the sources of the code on the call stack of the principal at the time the access is desired. Because the source of the code on the call stack will vary over time, so will the access rights of the principal. Thus, when a request for an action is made by a thread, a determination is made of whether the action is authorized based on permissions associated with routines in a calling hierarchy associated with the thread. The determination of whether a request is authorized is based on a determination of whether at least one permission associated with each routine encompasses the permission required to perform the requested action. Support for "privileged" routines is also provided. When a routine in the calling hierarchy is privileged, the determination of whether an action is authorized is made by determining whether at least one permission associated with each routine between and including the privileged routine and a second routine in the calling hierarchy encompasses the permission required to perform the requested action.

NO. 2		US6192476	
발명의 명칭	Controlling access to a resource		
출원번호	988431		
출원일	1997.12.11	출원인	Sun Microsystems
우선일		우선권주장번호	
IPC	H04L9/00	공개번호	



principal(e.g. thread)이 특별한 리소스에 접근할 수 있는지를 결정하기 위한 방법 및 시스템

A method and system are provided for

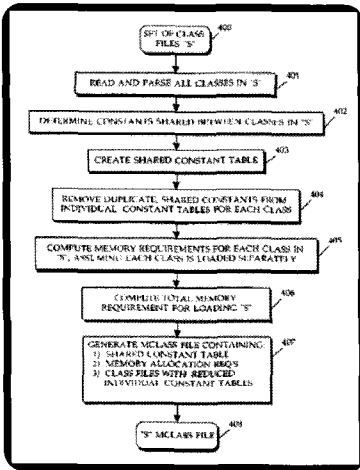
determining whether a principal (e.g. a thread) may access a particular resource. According to one aspect of the invention, the access authorization

NO. 3		US5966702	
발명의 명칭	Method and apparatus for pre-processing and packaging class files		
출원번호	961874		
출원일	1997.10.31	출원인	Sun Microsystems
우선일		우선권주장번호	
IPC	G06F17/30	공개번호	

전처리 및 클래스 파일 패키징을 위한 방법 및 장치.

A method and apparatus for pre-processing and packaging class files. Embodiments remove duplicate information elements from a set of class files to reduce the size of individual class files and to

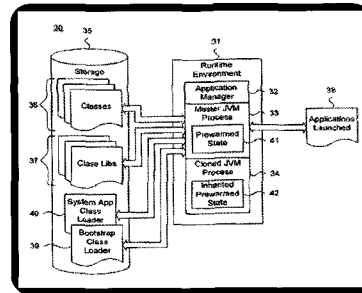
1) 출처 : 디지털데일리, [구글vs.오라클]총성없는 전쟁이 시작되다, 이민형 기자, 2011.09.26



prevent redundant resolution of the information elements. Memory allocation requirements are determined in advance for the set of classes as a whole to reduce the complexity of memory allocation when the set of classes are loaded. The class files are stored in a single package for efficient storage, transfer and processing as a unit. In an embodiment, a pre-processor examines each class file in a set of class files to locate duplicate information in the form of redundant constants contained in a constant pool. The duplicate constant is placed in a separate shared table, and all occurrences of the constant are removed from the respective constant pools of the individual class files. During pre-processing, memory allocation requirements are determined for each class file, and used to determine a total allocation requirement for the set of class files. The shared table, the memory allocation requirements and the reduced class files are packaged as a unit in a multi-class file.

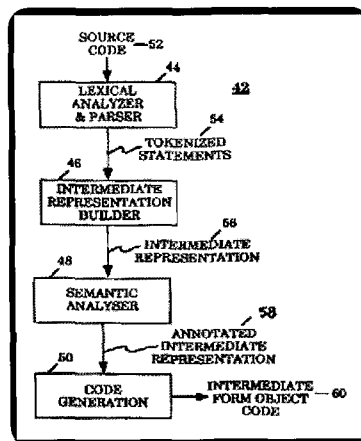
NO. 4		US7426720	
발명의 명칭	System and method for dynamic preloading of classes through memory space cloning of a master runtime system process		
출원번호	745023		
출원일	2003.12.22	출원인	Sun Microsystems
우선일		우선권주장번호	
IPC	G06F9/45	공개번호	

마스터 실시간 시스템 과정의 메모리 영역 클로닝을 통한 클래스의 동적 preloading을 위한 시스템 및 방법.



A system and method for dynamic preloading of classes through memory space cloning of a master runtime system process is presented. A master runtime system process is executed. A representation of at least one class is obtained from a source definition provided as object-oriented program code. The representation is interpreted and instantiated as a class definition in a memory space of the master runtime system process. The memory space is cloned as a child runtime system process responsive to a process request and the child runtime system process is executed, inheriting the memory state of the parent, which reflects the data structures and state corresponding to the preloaded classes.

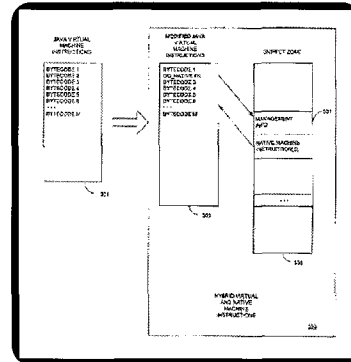
NO. 5		US RE38104	
발명의 명칭	Method and apparatus for resolving data references in generated code		
출원번호	261970		
출원일	1999.03.03	출원인	Sun Microsystems
우선일		우선권주장번호	
IPC	G06F9/45	공개번호	



소스 프로그램 코드를 컴파일링하는 컴파일러와 컴파일된 코드를 인터프리팅하는 인터프리터로 구성된 하이브리드 컴파일러-인터프리터

A hybrid compiler-interpreter comprising a compiler for "compiling"

source program code, and an interpreter for interpreting the "compiled" code, is provided to a computer system. The compiler comprises a code generator that generates code in intermediate form with data references made on a symbolic basis. The interpreter comprises a main interpretation routine, and two data reference handling routines, a dynamic field reference routine for handling symbolic references, and a static field reference routine for handling numeric references. The dynamic field reference routine, when invoked, resolves a symbolic reference and rewrites the symbolic reference into a numeric reference. After re-writing, the dynamic field reference routine returns to the main interpretation routine without advancing program execution to the next instruction, thereby allowing the rewritten instruction with numeric reference to be reexecuted. The static field reference routine, when invoked, obtain data for the program from a data object based on the numeric reference. After obtaining data, the static field reference routine advances program execution to the next instruction before returning to the interpretation routine. The main interpretation routine selectively invokes the two data reference handling routines depending on whether the data reference in an interaction in a symbolic or a numeric reference.



function are provided. A portion of the virtual machine instructions of the function are compiled into native machine instructions so that the function includes both virtual and native machine instructions.

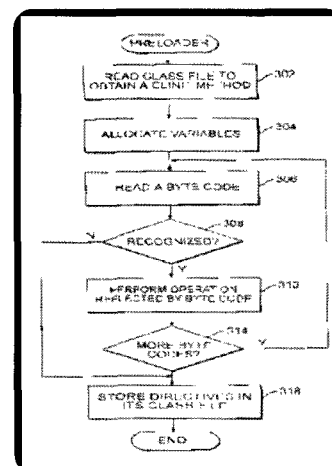
Execution of the native machine instructions may be accomplished by overwriting a virtual machine instruction of the function with a virtual machine instruction that specifies execution of the native machine instructions. Additionally, the original virtual machine instruction may be stored so that the original virtual machine instructions can be regenerated.

NO. 6		US6910205		
발명의 명칭	Interpreting functions utilizing a hybrid of virtual and native machine instructions			
출원번호	194040			
출원일	2002.07.12	출원인	Sun Microsystems	
우선일		우선권주장번호		
IPC	G06F9/45	공개번호		

기능을 위한 가상 머신 명령어의 실행 속도를 증가시키기 위한 시스템 및 방법

Systems and methods for increasing the execution speed of virtual machine instructions for a

NO. 7		US6061520		
발명의 명칭	Method and system for performing static initialization			
출원번호	942292			
출원일	1998.04.07	출원인	Sun Microsystems	
우선일		우선권주장번호		
IPC	G06F3/00	공개번호		



어레이를 정적으로 초기화하기 위한 가상 머신에 의해 실행 코드의 양을 감소시킴으로써 정적 배열을 초기화하기 위한 종래의 시스템을 개선함

The disclosed system represents an improvement over conventional systems for initializing static arrays by reducing the amount

of code executed by the virtual machine to statically initialize an array. To realize this reduction, when

consolidating class files, the preloader identifies all <clinit> methods and play executes these methods to determine the static initialization performed by them. The preloader then creates an expression indicating the static initialization performed by the <clinit> method and stores this expression in the .mclass file, replacing the <clinit> method. As such, the code of the <clinit> method, containing many instructions, is replaced by a single expression instructing the virtual machine to perform static initialization, thus saving a significant amount of memory. The virtual machine is modified to recognize this expression and perform the appropriate static initialization of an array.

맺음말

지금까지 오라클과 구글의 특허분쟁에서 제기되었던 썬 마이크로시스템의 자바기술 특허 7건과 특허분쟁 관련 내용에 대해 알아보았다.

오라클은 자기의 자회사인 썬마이크로시스템의 자바표준특허 7건에 대해서 139개 청구항을 구글이 침해했다고 주장하고 있다.

오라클이 침해를 주장하는 139개의 특허는 미국특허청에서 재심 과정을 통하여 청구항 수가 줄어들 여지가 있다.

이미, 미국특허청은 “오라클이 구글 안드로이드가 침해했다는 7개의 특허 중 1개의 특허를 심사한 결과 21개의 청구항 중 17개의 청구항이 부적합하다고 판단해 기각했다” 고 밝혔다. 구글의 입장에서는 참으로 다행스러운 일처럼 보였을 것이다.

그렇다 하더라도 나머지 4개의 청구항은 여전히 남아있는 것이고, 남은 6개의 특허들 내에 118개의 청구항들도 재심 과정이 진행되고 있어, 그 결과를 지켜봐야 하는 구글로서는 애간장을 태울 수밖에 없을 것이다.

오라클 입장에서도 자신들의 특허 권리범위가 줄어드는

것을 그저 바라보고만 있지는 않을 것이다. 어떻게 해서라도 청구항이 삭제되는 것을 막을 것이며, 설령 7건 중 2~3건의 특허가 무효가 된다 하더라도, 나머지 4~5건의 핵심특허를 갖고도 충분히 구글을 압박할 수 있다.

그 핵심특허가 바로 표준특허이기 때문이다.

오라클은 구글을 상대로 소송을 진행하는 것 이외에도, 앞으로 안드로이드 OS를 탑재한 기기에 대해서 대당 15~20 달러를 로열티로 받을 것이라는 이야기가 흘러나오고 있다. 만일, 이러한 사안이 현실화로 이어진다면 안드로이드를 사용하는 국내 단말기 제조업체인 삼성전자와 LG전자는 큰 부담을 갖게 될 것이다.

이미 현실화되어버린 MS의 특허공세에 더불어 오라클마저 특허로 제조업체들을 압박하려는 일련의 사태들이 참으로 안타깝다.

하드웨어를 잘 만드는 것만큼, 아니 그 이상으로 소프트웨어도 잘 만들기 위해 노력해야 한다는 것을 이제는 많은 이들이 공감하고 있다.

지피지기면 백전백승이라고 했다. 본 보고서에서 소개되고 있는 특허뿐만 아니라 그와 관련 있는 많은 특허들을 잘 살펴보고, 특히 R&D와 관련 있는 특허들은 면밀히 분석하여 R&D에 적용한다면, 그 이상의 부가가치를 창출할 수 있는 질 좋은 특허, 특허전쟁에서 무조건 승리를 불러올 수 있는 가장 강력한 무기인 표준특허를 창출하는 것은 시간문제일 것이다. 2011. 12]