비주얼 자바 클래스 파일 브라우저의 설계 및 구현

옥 재호, 정 민수, 김 도우, 류 동항, 이 수진
경남대학교 컴퓨터공학과

Design and Implementation of Visual Java Classfile Browser

Jea-Ho Ock, Min-Soo Jung, Do-Woo Kim, Dong-Hang Ryu, Soo-Jin Lee
Kyungnam University Computerscience

요 약

자바는 객체 지향 프로그래밍언어로써 클래스 파일내에 자바 소스코드에 나타나지 않는 여러 가지 정보를 내포하고 있다. 비주얼 자바 클래스 파일 브라우저는 가상 머신에 의해 실행되는 자바 프로그램내의 클래스들 사이의 호출관계와 상속관계, 각 클래스 파일의 내용을 브라우저를 통해 시각적으로 표시함으로써 하부 구조인 자바 가상 머신에서 동작하는 자바 클래스 파일의 다양한 분석을 수행한다.

1. 서론

특정 프로세스 혹은 하드웨어에 관계없이 동작할 수 있는 높은 이식성을 가진 자바는 자바 컴파일러에 의해 생성된 목적 코드가 특정 프로세스 혹은 하드웨어에 맞춰져 있지 않고 자바 가상 머신이라는 가상의 명령의 집합과 실행 환경에 맞춰져 있기 때문이다. 자바 가상 머신의 명령어 집합을 바이트 코드 또는 클래스 파일이라 한다.

하지만 자바 언어로 프로그램을 작성하거나 작성된 자바 프로그램을 분석하는 것은 쉽지 않다. 이것은 자바 언어가 기존의 절차 중심 언어(procedural language)가 아닌 객체 지향 언어로서 프로그램의 수행 순서가 명시적으로 잘 드러나지 않기 때문이다. 또한 객체 지향 언어의 특징인 시스템에서 제공되는 방대한 클래스와 메쏘드에 대한 사전 지식의 요구 또한 프로그램의 작성과 분석을 어렵게 하는 이유이다.[1, 8]

이러한 문제를 해결하기 위한 여러 가지 자바 개발/분석 도구가 개발되고 있지만, 이런 분석 도구들은 원시 소스 프로그램에 나타난 명시적인 정도의 분석만을 제공한다. 프로그램의 단순성이라는 측면에서는 상부 구조만을 사용자에게 보임으로써 복잡성을 줄여준다는 이점은 있으나, 그 하부 구조인 자바 가상머신에 대해서 감추어 버림으로써 컴파일러의 개발이나 운영체제 연구와 같은 시스템 프로그래밍 연구 분야에서는 그 유용성이 떨어지는 한계점이 있다.

본 논문에서 개발한 비주얼 자바 클래스 파일 브라우저는 가상 머신에 의해 실행되는 자바 프로그램내의 클래스들 사이의 호출관계와 상속관계, 각 클래스 파일의 내용을 브라우저를 통해 시각적으로 표시함으로써 하부 구조인 자바 가상 머신에서 동작하는 자바 클래스 파일의 상세한 분석과 자바 가상 머신에 대한 연구와 개발, 관련 도구들의 연구, 개발에도 많은 도움이 될 것이다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구로서 자바 가상 머신과 클래스 파일에 대해 기

술하며, 3 장에서는 비주얼 자바 클래스 파일 브라우저의 설계와 구현에 대해 기술한다. 그리고 4 장에서는 결론과 향후 연구 방향에 대해 기술한다

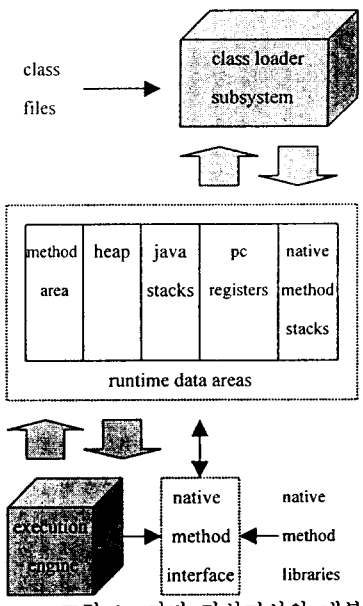
2. 관련 연구

본 장에서는 비주얼 자바 클래스 파일 분석기의 설계 및 구현에 사용된 자바 가상 머신과 클래스 파일에 대해 설명한다.

2.1 자바 가상 머신

자바 가상머신은 자바 프로그램이 실행되는 기반 환경으로 여러 플랫폼에서 운영되는 자바의 실행환경을 제공해 주는 일종의 소프트웨어 CPU 이다.

자바 가상기계의 구조는 <그림 1>과 같이 클래스 로더 서브시스템(class loader subsystem), 실행 엔진(execution engine)과 런타임 데이터 영역(runtime data areas)으로 구성되어 있다.[2]



<그림 1> 자바 가상머신의 내부 구조

2.2 자바 클래스 파일 구조

클래스 파일(class file)은 자바 클래스 또는 자바 인터페이스가 컴파일된 바이트 코드로서 자바 소스 프로그램에서 정의된 클래스 수만큼 생성된다. 클래스 파일은 매직번호, 버전정보, 상수풀, 접근자, 클래스명, 부모 클래스명, 인터페이스, 필드, 메쏘드, 속성등과 같은 자바 소스 프로그램이 보여주지 못하는 많은 정보를 가지고 있다.[3]

클래스 파일의 전체적인 구조와 각 구조에 대한 설명은 다음과 같다.

클래스 파일(Class File)
매직 번호(magic number)
버전 정보(minor-major version)
상수풀(constant pool) 정보
접근자(access flag) 정보
클래스명
부모 클래스명(super class)
인터페이스(interface) 정보
필드(field) 정보
메쏘드(method) 정보
속성(attribute) 정보

<그림 2> 자바 클래스 파일의 구조

3. 비주얼 자바 클래스 파일 브라우저의 설계 및 구현

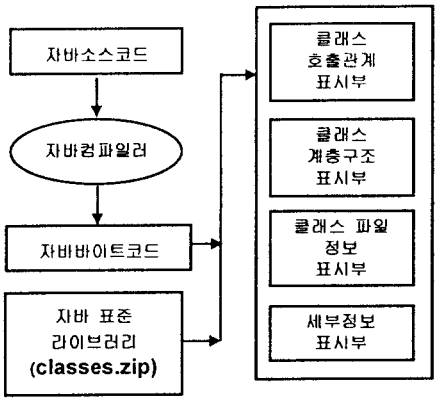
3.1 비주얼 자바 클래스 파일 브라우저

비주얼 자바 클래스 파일 브라우저는 현재 실행되고 있는 자바 프로그램의 클래스 호출관계(Caller/Callee)와 클래스 계층관계(Hierarchy), 각 클래스가 가지는 정보를 브라우저를 통해 시각적으로 표현한다. 비주얼 자바 클래스 파일 브라우저는 세 개의 트리 뷰와 1 개의 멀티리스트로 구현되어 있으며 각각 클래스 호출관계 표시부, 클래스 계층구조 표시부, 클래스 파일 정보 표시부, 세부 정보 표시부의 4 가지 영역과 디컴파일(Decompil)된 소스를 보여주는 프레임(Frame)으로 나눌 수 있다.[4, 7]

다음 장에서는 각 영역에 대해 간략히 설명한다.

3. 2 비주얼 자바 클래스 파일 브라우저 구성

비주얼 자바 클래스 파일 브라우저의 전체 구성과 각 영역에 대한 설명은 다음과 같다.



<그림 3> 비주얼 자바 클래스 파일 브라우저 전체 구성도

3.3 클래스 호출관계 표시부

클래스 호출관계 표시부는 입력 클래스 파일을 기점으로 하여 클래스내에 정의된 메쏘드에 의해 호출된 클래스들의 호출관계(Caller/Callee)를 분석하기 위한 영역이다. 즉 입력된 클래스 파일을 기점으로 하여 클래스내에 정의된 메쏘드에 의해 호출된 클래스들을 탐색하여 각 클래스들의 호출관계를 트리 구조로 분석하여 출력한다. 또한 클래스 호출관계 표시부에 선택된 클래스에 대해 클래스에 대한 정보를 클래스파일 정보 표시부와 세부정보 표시부에 표현한다.[5, 6]

3.3.1 클래스 호출관계 분석

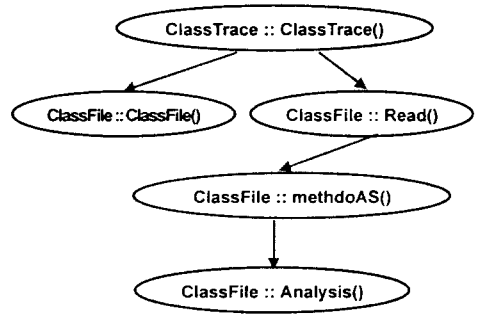
각 클래스 파일내부에는 현 클래스 파일에서 사용한 필드, 메쏘드, 클래스들에 대한 타입과 이름등을 콘스탄트 풀로서 정의 하게 된다. 따라서 현 클래스에 정의된 메쏘드 영역을 분석하여 각 메쏘드에 대한 호출관계(Caller/Callee)를 구한다음 Callee 의 클래스가 현 클래스와 다른 경우 차례로 각 클래스를 같은 방법으로 검색하여 클래스에 대한 호출관계를 분석하게 된다.

다음은 메쏘드 호출관계에 대한 정보를 저장하고 있는 findCallees 클래스이다.

```
Public class findCallees {
    public String  ClassName;   // 클래스 이름
    public int     classCount;  // 클래스의 개수
    public int     calleeCount; // 호출된 메쏘드의 개수
    public String  MethodName; // 호출된 메쏘드의 이름
}
```

<그림 4> findCallees 클래스의 구조

다음은 클래스 호출관계 분석에 사용된 클래스와 메쏘드를 보여주는 전체 흐름도와 기능이다.



<그림 5> 클래스 호출관계에 대한 전체 흐름도

- ClassTrace :: ClassTrace() : 클래스 호출관계를 분석하기 위한 구동 메쏘드이다. 또한 분석된 클래스 호출관계를 트리 뷰에 반영한다.
- ClassFile :: ClassFile() : 클래스 파일 분석에 필요한 필드를 초기화한다.
- ClassFile :: Read() : 입력된 클래스 파일을 분석하여 분석된 정보를 저장한다.
- ClassFile :: methodAS() : 메쏘드의 호출관계를 분석하여 Callee 중 다른 클래스에 정의된 Callee 를 검색하여 저장한다.
- ClassFile :: Analysis() : Caller 에 대한 Callee 를 검색하여 저장한다

3.4 클래스 계층구조 표시부

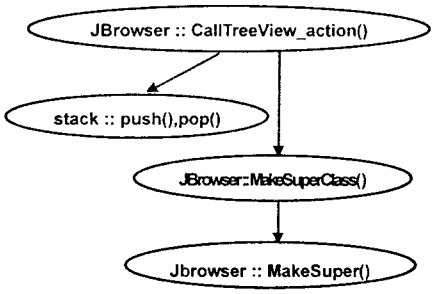
클래스 계층 구조 표시부는 클래스 호출 관계 표시부에 정의된 클래스들의 계층 관계를 표시하는 부분이다. 즉 선택된 클래스를 기점으로 하여 자신의 슈퍼 클래스(super class)를 탐색하여 최상위 클래스인 java.lang.Object 클래스까지의 클래스 계층관계를 표시하게 된다.

또한 클래스 계층관계에 나타난 클래스들의 정보를 쉽게 분석할 수 있도록 선택된 클래스에 대한 정보를 클래스파일 정보 표시부와 세부정보 표시부에 상세하게 표시한다.

3.4.1 클래스 계층관계 분석

클래스 계층관계에 대한 분석은 클래스파일의 정보를 이용한다. <그림 2>와 같이 클래스파일은 내부적으로 자신의 슈퍼클래스에 대한 정보를 저장하고 있다. 따라서 최상위 클래스인 java.lang.Object 까지 각 클래스의 슈퍼클래스를 역으로 추적하면 계층관계를 분석할 수 있다.

다음은 클래스 계층관계 분석에 사용된 클래스와 메쏘드를 보여주는 전체 흐름도와 기능이다.



<그림 6> 클래스 계층관계 분석에 대한 전체 흐름도

- JBrowser :: CallTreeView_action() : 클래스 호출관계 표시부에 대한 이벤트 처리 메쏘드로 선택된 클래스에 대해 클래스의 계층관계를 표시하게 된다. 이때 클래스의 슈퍼클래스를 검색하기 위해 스택을 이용하였다.
- JBrowser :: MakeSuperClass() : MakeSuper 메쏘드를 재귀적으로 호출하여 슈퍼 클래스를 검색하여 검색 완료후 스택에 저장된 클래스들을 pop 하여 트리 뷰에 반영한다.
- JBrowser :: MakeSuper : 클래스에 대한 슈퍼클래스를 차례로 검색하고 검색된 클래스를 스택에 push 한다.

3.5 클래스파일 정보 표시부

클래스파일 정보 표시부은 세부정보 표시부와 함께 클래스파일의 정보를 시각적으로 표현하며, 입력된 클래스 파일을 분석하여 클래스 파일의 정보를 트리 형식으로 출력한다.

3.5.1 클래스 파일 정보에 대한 분석

클래스 파일에 대한 정보를 트리 구조로 표현하기 위해서는 각 정보를 저장하기 위한 트리 구조의 연결리스트가 필요하다.

다음은 트리 구조의 연결리스트에 대한 자료 구조를 정의한 클래스 파일이다.

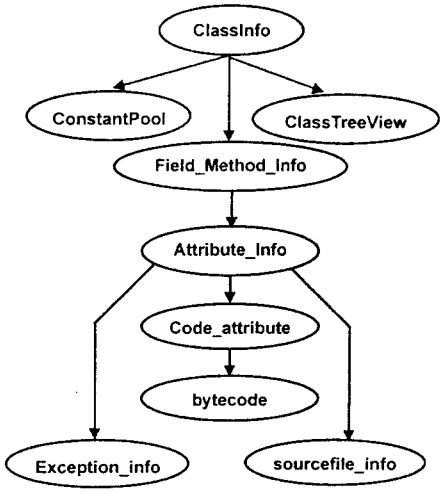
```
Public class TreeNode {
  // 클래스 정보를 나타내는 객체를 저장
   Object    data;
  // 현 노드와 관련 있는 노드 표현
   TreeNode sibling;      // 형제노드
   TreeNode child;        // 자식노드
   TreeNode parent;       // 부모노드
  // 현 노드의 이름을 저장
   String    text;
  // 트리뷰에 표현될 이미지 정보를 저장
   Image     collapsedImage;
   Image     expandedImage;
  // 노드의 깊이 서브노드 수를 정의
   int       depth = -1;
  // Expand 유무 저장
   boolean   isExpanded = false;
  // 자식 노드의 수
   int       numberOfChildren;
   }
```

<그림 7> 클래스의 각 정보를 저장하는 TreeNode 클래스

<그림 7>의 트리 노드로 구성된 클래스파일의 각 정보들은 트리 구조로 표현되며 중복된 노드를 피하기 위해 각 트리에 매달린 각 노드를 검색하여 새로운 노드가 연결될 위치를 정확하게 결정한다. 또한 현재 자신의 상위노드가 하위노드를 가지는 경우 자

신의 형제노드를 검색하여 새로운 노드를 연결하게 된다.

트리 구조로 표현된 클래스 파일 정보 분석에 사용된 클래스와 기능은 다음과 같다.



<그림 8> 클래스 파일 정보에 대한 분석의 전체 흐름도

- ClassInfo : 클래스 파일 분석의 주 클래스로써 클래스 파일의 각 정보를 순차적으로 분석하여 저장한다. 저장되는 정보로는 클래스 파일의 버전, 부모와 자식 클래스의 이름, constant pool, 필드 그리고 메쏘드 등이다.
- ConstantPool : 입력된 클래스 파일로부터 constant pool 의 내용을 추출하여 ClassInfo 클래스의 constant_pool[] 필드에 저장한다.
- Attributes : 입력된 클래스 파일로부터 필드, 메쏘드와 관련된 어트리뷰트를 추출한다. 메쏘드 어트리뷰트인 경우에는 실제 메쏘드의 코드와 예외처리에 대한 정보 그리고 소스파일 정보등을 저장한다.
- Fileds_Method_Info : 입력된 클래스 파일로부터 필드와 메쏘드의 정보를 추출하여 ClassInfo 클래스에 저장한다. 필드인 경우 필드의 이름과 수정자 그리고 필드의 형이고 메쏘드인 경우 메쏘드의 개수와 관련된 메쏘드의 이름, 수정자와 리턴 형 정보 그리고 어트리뷰트에 대한 정보를 저장한다.
- Bytecode : 메쏘드에 대한 실제 바이트 코드를 이셈블리 형식으로 변환하여 저장한다.
- ClassTreeView : ClassInfo 에 저장된 클래스 정보를 트리 구조로 표현한다.

3..6 세부정보 표시부

세부정보 표시부는 클래스파일 정보 표시부에 표현된 트리구조의 각 노드에 대한 정보를 자세하게 나타내는 부분으로 여러 개의 아이템으로 나누어 나타낼 수 있는 멀티리스트로 구현되어 있다.

TreeView 로 구현된 클래스파일 정보 표시부의 각각의 클래스 파일의 정보를 가리키는 노드 객체는 인터페이스 Displayable 내의 display()메쏘드를 호출함으로써 객체의 정보가 세부정보 표시부의 멀티리스트에 출력된다.

다음은 Displayable 인터페이스를 나타내고 있다.

```
Public interface Displayable {
    Public void display(MultiList multi);
}
```

<그림 9> Displayable 인터페이스

다음은 클래스 파일 정보 표시부에 나타난 트리 구조의 노드를 선택하는 경우 노드에 대한 개체정보가 출력되는 과정을 처리한 부분이다.

```
// TreeView 에 대한 이벤트 처리 함수
void CtreeView_Action(Event event) {
// 특정 노드가 선택되는 경우
   if (((CTreeView.getSelectedNode()).data) != null){
    // 해당 객체에 대한 display()메쏘드를 호출한다.
        ((Displayable)(CtreeView.getSelectedNode()).data
)).display(multiList);
}
```

<그림 10> TreeView 에 대한 이벤트 처리 함수

3.7 Mocha 를 이용한 자바 소스코드 뷰(View)

3.7.1 모카에 대한 소개 및 사용

모카(mocha)는 자바 클래스 파일(*.class)로부터 자

바 소스 코드(*.java)를 역으로 생성시켜주는 자바 디컴파일러이다. Hanpeter van Vliet 에 의해 1995,1996 년에 걸쳐 개발된 모카는 현재 베타(beta)1 이 공개용으로 사용할 수 있다

모카는 "mocha.zip"으로 제공되며 압축을 풀 필요없이 배포된 zip 파일 자체로 사용한다. 적당한 디렉토리에 복사한 후 기존의 CLASSPATH 환경변수에 추가하여 사용한다. 또한 mocha 는 100%자바로 구현되어 있으며 자바를 지원하는 모든 플랫폼에서 동작한다. 모카는 win95 의 경우 도스창에서 커맨트라인(commandline)에서 사용하며 사용법은 다음과 같다.

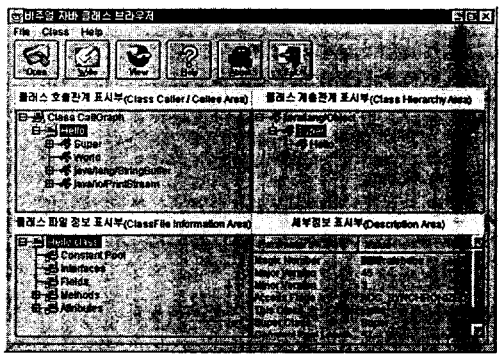
javamocha.Decompiler [-v][-o]Class1.class Class2.class..

3.7.2 Mocha 를 이용한 자바 소스코드 뷰(View)

비주얼 자바 클래스 파일 브라우저는 클래스 정보 표시기에 표시된 클래스 파일에 대해 디컴파일된 자바 소스코드를 생성하여 보여준다. 내부적으로 모카를 이용하여 디컴파일된 자바 소스코드를 생성하여 새로운 프레임(frame)에 디컴파일된 소스코드를 출력한다.

3.8 비주얼 자바 클래스 파일 브라우저의 실행

다음 그림은 앞서 설명한 기능을 수행하는 비주얼 자바 클래스 파일 브라우저의 실행 화면이다.



<그림 11>비주얼 자바 클래스 파일 브라우저의 실행

4. 결론 및 연구결과의 활용

본 논문에서 구현한 비주얼 자바 클래스 파일 브라우저는 원시 소스 프로그램에 나타난 명시적인 정도의 분석만이 아닌 그 하부 구조인 자바 가상머신에서 동작하는 자바 클래스 파일에 대한 상세한 분석을 수행함으로써 자바 가상머신에 의해 실행되는 자바 프로그램내의 클래스들 사이의 호출관계와 상속관계, 각 클래스 파일의 내용을 브라우저를 통해 시각적으로 알 수 있다.

또한 자바 가상 머신에서 동작 자바 클래스 파일에 대한 상세한 분석을 통해 자바 가상 머신에 대한 연구와 개발, 관련 도구들의 연구, 개발에도 많은 도움이 될 것이다.

참고 문헌

[1] J. Gosling, "Java™ Language Specification", Addison-Wesley, 1997.

[2] T. Lindholm and F. Yellin, "The Java™ Virtual Machine Specification" ADDISON-WESLEY, 1997.

[3] A. Taivalsaari, "Implementation a Java Virtual Machine in the java programming Language ",SUN Lab, 1997.

[4] B. Venners, "Inside Java Virtual Machine", McGraw-Hill, 1997.

[5] D. Grove, G. DeFouw, J. Dean, and C. Cahmbers, "Call Graph Construction in Object-Oriented Languages", OOPSLA, 1997

[6] 류동항, 정민수 "자바 바이트 코드 분석기의 설계 및 구현", 한국정보과학회 98 봄학술발표논문집, 제 25 권 1 호, pp 77-79, 1997.

[7] http://www.artima.com/, ARTIMA SOFTWARE COMPANY

[8] http://java.sun.com/, Sun Microsystem, Java Home Page