教育用JVMエミュレータの設計と試作

4L−3

梅津 亮　井田 昌之

青山学院大学 国際政治学研究科 / 青山学院大学 国際政治経済学部

1 はじめに

Java言語教育用のソフトウェアの一環として、Java Virtual Machine[1]のエミュレータを試作したので報告する。Javaプログラム開発の際に、生成されるクラスファイルの構造や、そのターゲットマシンであるJVMの実行時の動態を知る一助となることを目標としている。設計方針として実行過程の可視化、自身のアプレット化による高可搬性、動作原理理解のため各種情報表示に重点をおいた。また、複数ウィンドウによる表示法に特徴がある。

以下では、先ず背景に触れた上で、本研究の設計・実装の特徴を詳述し、今後の予定を述べる。

2 背景

Javaのクラスファイルは、Java言語ソースの目的プログラムであると同時に、コンパクトにオブジェクト情報を詰め込んだものである[2, 3]。Constant Pool、属性情報、メソッドのバイトコードやソースファイル名が入っており、本試作ではこうした情報を抽出して利用している。

本試作に関連した他の開発には、Jasmin(Javaアセンブラ)、JavaClass(クラスファイル解析・書き換え)、D-Java(クラスファイル解析)、mocha(デコンパイラ)、Zelix KlassMaster(クラスファイル解析・書き換え)、ClassViewer(クラスファイル解析)といったもの、JVM動作理解を意図したアプレット[4]などが存在するが、これからJVMを学ぼうという者にとってはあまり適していない。GUIベースのものもあるが、商品であり、JVMの理解というよりはクラスファイルの構造分析ツールとして取り扱われている。

このように、本稿の目的のためにそのまま流用できるツールは存在しないのが現状である。

3 設計

3.1 portability

エミュレータ自身をJavaで書くことにより、アプレットとして、多くのプラットフォーム上で実行できるようにする。

3.2 デバッガ機能

逐次実行時に、プログラムカウンタ(PC)の指す番地のメモリ内容を変更して実行できるようにする。但し、この変更は、ソースファイルウィンドウなどには反映されないものとする。

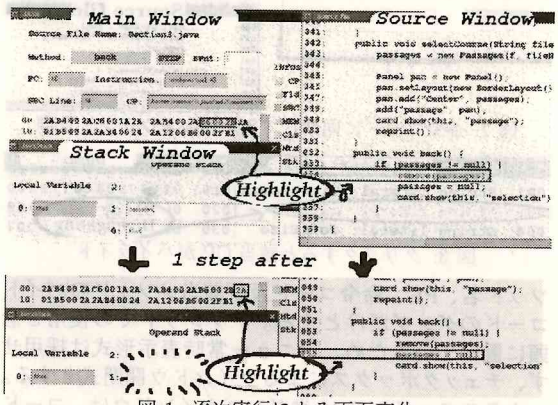

図1: 逐次実行による画面変化

3.3 逐次実行時の変化の表示

本試作では、逐次実行に伴って、該当命令コード表示、Constant Pool引数があればその表示、ソースファイル該当行のハイライト、そしてスタック状態遷移の表示をしている。例えば、図1上部はPCが「0C」の時の様子を表わしている。左上メインウインドウでは命令コードの「0C」番地部分がハイライトされ、当該命令は「invokevirtual 43」であることを示している。左下のスタックには「this」と「passage」が積まれ、右側のソースファイル表示は54行目をハイライトしている。ここで1ステップ進めると、図1下部のように、PCが「0F」に進み、ソースファイルでは55行目にハイライトが移っている。

又、例外処理については、強制的に例外を発生させたい場合、PCがtry節内にある時に、メインウインドウのブレイクポイント指定欄に「ex1」などと入力すれば1番目の例外処理部分に飛ぶようにする。

3.4 別ウインドウによる各種情報の表示

メインウインドウでは、限られた画面スペースから不要な情報を出来る限り排除するため、Constant Poolの情報等は全てポップアップウインドウによる表示とする。図2に示すような16進ダンプウインドウやConstant Poolを抽出したウインドウの他にも、メインウインドウで常に表示しておく必要のないクラスファイル属性やメソッド属性についても別ウインドウ表示としている。これによって、メインウインドウはファイル名、メ

Design and Implementation of JVM Emulator for Education.
Ryo UMETSU, Masayuki IDA
SIPEB, Aoyama Gakuin Univ.
u-ryo@sipeb.aoyama.ac.jp, ida@sipeb.aoyama.ac.jp

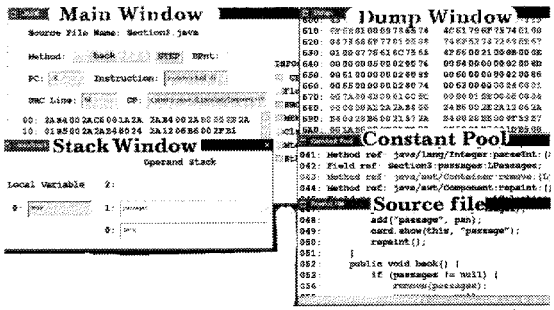

図 2: 命令コードに同期して表示されるウインドウ群

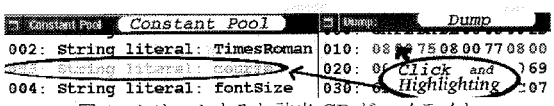

図 3: クリックすると該当 CP がハイライト

ソッド名、PC、命令コード及び該当メソッドのバイトコードのダンプだけとする。アプレットでの使用を念頭に置いているため、メニュー常時表示形式は採用せず、チェックボックスによるウインドウ開閉式とする。

クラスファイルの 16 進ダンプウインドウは、コードの単なる羅列とせず、クラスファイルの大半を占める Constant Pool に関する情報部分を色分けして表示する。更にダンプ列の Constant Pool 表現部分をクリックすると Constant Pool ウインドウで該当部分が呼応してハイライトして示すようにする (図 3)。

ローカル変数の遷移やオペランドスタックが 1 ステップずつ積まれる様子を可視的に表わすことで、JVM スタックマシンの理解を促進させることを意図した。メソッドによってスタック数が可変であるので、大きさを自由に調整できる別ウインドウ表示にする。

又、当該クラスファイルの構造や Constant Pool の内容をわかりやすくするために、クラスファイルの 16 進ダンプや Constant Pool、フィールドデータの情報や、クラスファイルの属性、メソッドの属性も別ウインドウにして表示する方針をとった。

3.5 エミュレーションの方針

本試作では JVM の動作理解に重点を置く。メソッドの実行順序も、ユーザがメインウインドウ中の選択肢から選んだ任意のメソッドから開始できるようにする。フィールド値は putfield 命令や getfield 命令でメソッドを越えて値が受け渡されるように配慮する。

4 実装

4.1 API

Java2 では、以下の API が定義されている。java.lang、java.io、java.applet、java.awt、java.math、java.beans、java.rmi、java.security、java.util、javax.accessibility、javax.swing、org.omg.CORBA、org.omg.CosNaming。これらの API を全てカバーすることにより、完全な開発環境へと進化させることが出来る。インスタンシエーションは reflection の Constructor クラスの newInstance を用い、メソッドの実行は Method クラスの invoke を利用して実現している。

4.2 逐次実行

1 命令ずつの逐次実行とブレイクポイントまでの一括実行、ソースファイルの行指定による実行を用意した。この場合でも、呼び出した先のメソッドの実行は逐次とはせず、返り値が積まれるまで一括して実行される。

4.3 幾つかの実装上の工夫

バイト列表示の際、当初は描画の度に各バイト列の位置を算出し表示していたが、オーバーヘッドが激しいので、ウインドウ生成時にオフスクリーンイメージに全て描画した。ために後の際描画を滑らかなものにすることが出来た。

ソースファイル名は、クラスファイル名を単純に流用するのではなく、クラスファイル内に埋め込まれている情報を抽出するようにした。

invokevirtual 命令 [1] で、System.out.print[ln] の場合は標準出力に書き出し、System.in の場合はウインドウをポップアップさせて入力を促し、入力値をスタックに積むようにして、簡単ながら入出力を実現した。

4.4 URL の公開

本試作は以下の URL にて公開している。

`http://noa.sipeb.aoyama.ac.jp/jvm-emu/`

5 今後の可能性

本ツールの今後の展開であるが、初学者用途及び専門家養成用途の 2 つの方向性がある。前者に対しては、16 進ダンプ画面にてバイト列をクリックした際の情報表示を Constant Pool 以外のメソッドの部分やフィールドデータの部分にも拡充したり、命令コードの簡単な解説も出るようにして、マニュアル無しでも JVM を理解出来るようなツールにしていく。

後者に関しては、本ツールをオブジェクトインスペクタ、オブジェクトエグザミナへ発展させることで、オブジェクト指向言語開発への一助となる。又、SUN が 1999 年 6 月に発表した小型機器用 JVM の KVM[5] のバイトコードにも対応すれば、プログラムサイズが大きな問題になる J2ME[6] 対応アプリケーション開発にも使用できる。

参考文献

[1] Tim Lindholm, Frank Yellin, "The Java Virtual Machine Specification," `http://java.sun.com/docs/books/vmspec/html/VMSpecTOC.doc.html`

[2] 井田昌之, 『New はやわかり Java』, 共立出版, 1997

[3] Jon Meyer, 鷲見豊訳, 『JAVA バーチャルマシン』, オライリー・ジャパン, 1997

[4] Bill Venners, "Under the Hood — JavaWorld June 1996", `http://www.javaworld.com/javaworld/jw-06-1996/jw-06-vm.html`

[5] K Virtual Machine(KVM), `http://java.sun.com/products/kvm/`

[6] Java 2 Platform, Micro Edition, `http://java.sun.com/j2me/`