デジタル・ドキュメント 1－2
（1996. 5. 27）

Javaの技術動向と今後の展望

井田 昌之
ida@sipeb.aoyama.ac.jp
青山学院大学 国際政治経済学部

Java言語は，ワールドワイドウェブの爆発的な普及とともに注目を浴びている．このため，Javaにはそうしたアプリケーションからの期待が大きく，さまざまな議論がされている．本報告では，まず，Javaの論点を1）言語仕様，2）API／クラスライブラリ，3）アプリケーションの三層にわけて考えることから始めて，その本質を追求するには，Javaの言語としての側面について議論する必要があるとしている．さらに現在はJavaの言語仕様書が出版されていない前史段階であることに触れる．

言語仕様のレビューとリファインの作業に協力している立場から，Javaに関連する技術の枠組について論じ，次に言語仕様の策定過程ならびに現在の仕様原案，そこでの論点について述べている．

JDKという参照処理系はかならずしも現在作成中の仕様書とは同期していないこと，UTF8というUnicode表現形式について，また，Corbaや他言語，開発環境などの関連技術との関連，動向などについても最後に触れている．

Java: the Status Quo and the Perspective

Masayuki Ida

School of International Politics, Economics and Business
Aoyama Gakuin University

Java is getting huge population with the explosion of World Wide Web. Application side seems to be hot and expect greater advancement with it. Many dreams and issues have been discussed. In this paper, the author is trying to split the issues into three layers. And describes the discussion on the language aspect of Java is critical to touch it's essensials. The author also points out the current situation is pre-historic age for Java, since the langauge definition and specification are not published yet.

In this paper, with the author's stand point as a volunteer collaborator through reviewing and refining the draft Java Language Specification, the framework of Java related technologies and their movements are argued, then, the process to finalize the language specification, current draft specification, and the issues are described.

Some points to notes are the following: JDK is not always synchronising the draft specification document. We need to have knowledge on UTF8, which is a Unicode representation format. Lastly, the trend and technology related to Java, such as Corba, other languages and development environment, are described.

1 はじめに

1.1 論点の三層への切り分け

Java 言語 [1] は，Sun Microsystems によって開発され，そのホワイトペーパ [2] が出された頃から，ワールドワイドウェブ（WWW）の爆発的な普及とともに急速に注目を浴びて来た．このため，Java にはそうしたアプリケーションからの期待が大きい．また，さまざまな応用への可能性が取り沙汰され，それに付随して議論百出の観がある．

ここでは，以下の三層に分けて問題を議論する．

- **第一層：** コアとなる言語仕様
- **第二層：** API，クラスライブラリ
- **第三層：** アプリケーション（Hotjava，Netscape などのブラウザへの組み込みに関連した技術および動向の問題．WWW とは独立した応用および関連技術の問題）

この三層にわけられると考えた場合，一層と二層の間では，どのようなライブラリ機能を言語仕様に含めるかと言う課題がある．二層と三層の間では，アプリケーションで共通的に使われるような機能をライブラリするか／できるかという課題がある．また，一層については，それが土台とするマシン仕様をどう扱うかという課題がある．

1.2 現状での問題

このような分類をした場合，96 年 4 月の段階での問題は次の三点に起因すると思われる．

第一に，第一層，第二層の確定作業に比べて，アプリケーションへの期待が先行している．

第二に，米国 Sun Microsystems が開発したものであるが，必ずしも，そのもともとの開発者がリーダシップに固執していない．

第三に，Java 言語仕様書（JLS）ならびに関連する言語環境の仕様書が未公刊で，前史的状態にある．

1.3 本稿での立場

Java の本質を追求するには，Java の言語としての側面について議論する必要がある．アプリケーション層については特に触れない．

ここでは，その立場から，Java に関連する技術の枠組について論じ，次に言語仕様の策定過程ならびに現在の仕様原案，そこでの論点および関連する動向について述べている．JVM 問題は一応別とする．

JLS の著者に対し作業協力をしている関係から，現在作成中の仕様書の内容に言及するが，本稿の内容は，著者の研究に基づく独自のものであって，それ以外の意味はない．将来の公的機関による言語標準の維持を期待するが，著者らの Common Lisp での 10 年余りの経験では，早期の公開討論は設計者のアートとしての一貫性に困難さを生じ，仕様の肥大化をもたらす傾向があることを明記したい．同時に，実際のユーザが求める仕様が生きた仕様となることはいうまでもない．その意味で，第二層のライブラリについての共通仕様作成作業は重要であり，大いになされるべきであろう．

2 Where is my Bible ?

2.1 参照処理系

Java の開発キットとして，その開発者である Sun Microsystems からは JDK（Java Development Kit）というものが出ている．JDK のプラットフォームは，Solaris 2.3 以降の SPARC と Windows NT, 95 である．現在のバージョンは JDK-1.0.1 である．また，JDK-1.0 beta1 が Mac 用に出ている．この JDK は，仕様の参照処理系としても役立ってきている．

JDK の開発の歴史としては，以下の順を認めることが出来る；Java Alpha3 (95 年 3 月)，その後，JDK 1.0 Pre-Beta，JDK 1.0 beta，JDK 1.0 Beta2，JDK 1.0，JDK 1.0.1 の順．JDK 1.0 は 96 年 1 月，JDK 1.0.1 は同年 3 月である．

JDK のさまざまな移植としては，JDK-1.0 for Linux が ftp://java.blackdown.org/pub/Java/linux にある．IBM は OS/2 用ならびに AIX 用の JDK beta1 を 1 月 26 日にアナウンスした．(MVS，OS/400，Win16 への移植を継続しているとその中で述べている) HTTP://ncc.hursley.ibm.com:80/javainfo/に説明がある．OSF では HPUX 10.0 用の JDK 1.0.1 を公開している．http://www.osf.org/ に説明がある．

2.2 言語仕様書

Javaの言語仕様としては，

- 95年10月のベータ版仕様書に基づく仕様[3]：これは，インターネットで公開された仕様書としては96年5月現在でも最新のものだといえる．その名のとおり，第一層の定義のみのベータ版で，さらにすべての項目に渡って完全に記述されてはいない．
- 96年1月のJDK-1.0（および1.0.1版）のソフトウェアの動き方から類推する仕様：96年1月23日にv1.0が正式にSun Microsystems社から出された．しかし，それに伴う仕様書の改訂はその時点では公開されなかった．
- いくつか出ている書籍などで述べられている仕様：(Javaを題材とした英語の本の多くは，ベータ版よりさらに前のアルファ版に基づいているものが多かった．また，それぞれの本での解釈が含まれている．日本語のものも含めて徐々にJDKベースのものが出てきている)

  また，SunのWWWや，JDKについている文書などから仕様を類推できるものもある．
- 96年4月現在で，かなり固まってきている『Java Language Specification マニュアル』の原稿に記された仕様（96年夏までにAddison-Wesley社より出版される予定とのこと）．また，それに加えてシリーズでマニュアルが出され，これによって，各層での定義の初版がそろうことになる．

の4種類のソースがあることになる．早晩，ベータ版に書かれた仕様は無意味になる．4番目の仕様書がもっとも最新かつ正確な仕様書ということになる．しかし，この仕様書は現段階ではまだ出版されていない．(The Java Seriesは，5冊よりなり，以下のようなものである．The Java Language Specification, The Java Virtual Machine Specification, The Java Application Programming Interface Vol.1: Core Package, Vol.2: Window Toolkit and Applet, The Java Programming Language)

この夏に出るJLS仕様書とベータ版の違いについて少し触れておく．

第1に，ベータ版ではクラスライブラリについて，そのアウトライン，その内容，ともにまったく触れられていなかった．言語の骨格は記されているが，実際のプログラミングに必要なシステムの持っているライブラリ機能がどうなっているかについては全く触れられていなかった．システムの標準となるライブラリ機能が仕様書中に規定されるようになった．

第2に，ベータ版で存在していたあいまいな記述が明確になってきた．たとえば，文字型の文字は数として扱えるのか，あるいは独立した型であるのかベータ版ではあいまいだった．それが，数として扱うことが明確になっている．

3 言語仕様上の特徴

Javaの特徴は次のようにまとめることができる．

1. **オブジェクト指向言語**である．
   基本機能は言語仕様と，いわゆるクラスライブラリから成り立っていて，Javaのライブラリはパッケージという概念でまとめられている．

   java.lang，**java.util**，**java.io**が標準で言語仕様に含まれるパッケージとなる．Javaのソフトウェア開発キットであるJDK（Java Development Kit）にはこの他にも多数のパッケージが含まれている．実際の問題としてはそれらも標準のように扱うことができる．そして，これらのライブラリ機能はオブジェクト指向の概念で利用することができる．

   マルチメディア時代に重要な，アニメーションやオーディオ機能などもライブラリとして入ってくる．いわゆるGUIのツールキットもあり，それらは，オブジェクトとして扱える．

   また，プログラムの開発に際しては，そのオブジェクト指向言語としての機能を発揮して，モジュラーで保守性が良く，ダイナミックなソフトウェアを開発することができる．

2. **機種独立**（machine independent and platform independent）
   簡単に言えば，Javaで書かれたソフトウェアは，どんなコンピュータにも持っていくことができ，どこへ行っても同じ動き方をするように意図されている．

   また，実際にもSparcをはじめ，インテルのPentiumなど複数のプラットフォームで動作できるような開発キットが用意されている．

   言語仕様には「処理系依存」「未定義」などというものはない．

3. シンプルな言語として設計された.
   C 言語や Lisp のように，言語の構文則としては単純にし，そのまわりにライブラリ機能として，さまざまなものを用意して開発を助けている.

   また，Java には構造体という概念がない．単純なデータか，オブジェクトしかない.

4. インターネット時代を前提とした言語として設計された.
   たとえば,「新しいパッケージの追加というのは単に TCP 接続にすぎない」などという記述がベータ版の文書にある．これは，インターネットを環境として考えると，新しい機能の追加は自分のマシンに置かなくとも，どこかで公開してくれていれば，それを引用するだけでいい，というようなセンスである.

   また，たとえば，WWW 用に URL を簡単に操作できるようになっている．URL を指定して，あるホームページからその内容をひっぱってくるというようなことが簡単に書ける.

   ソケットの処理などもライブラリ機能に入っている.

5. Java プログラムは，**Unicode** 文字セットで作られる.
   Java は，ASCII アルファベットの世界だけでなく，国際対応した文字を扱えるように考えられている．このために Unicode 1.1 への準拠をうたっている.

4 ９６年４月時点の JLS ドラフトをめぐって

4.1 目次案

JLS（Java Language Specification）のドラフトは，現在，以下のような構成になっている.

```
Chapter 1. Introduction : (6 pages)
Chapter 2. Grammars : (4 pages)
Chapter 3. Lexical Structure : (16 pages)
Chapter 4. Types, Values, and Variables : (20 pages)
Chapter 5. Conversions and Promotions :  (22 pages)
Chapter 6. Names :  (30 pages)
Chapter 7. Packages :  (14 pages)
Chapter 8. Class Declarations :  (46 pages)
Chapter 9. Interface Declarations :  (6 pages)
Chapter 10. Arrays :  (8 pages)
Chapter 11. Exceptions :  (12 pages)
Chapter 12. Execution :  (40 pages)
Chapter 13. Blocks and Statements :  (32 pages)
Chapter 14. Expressions :  (42 pages)
Chapter 15. Definite Assignment : (16 pages)
Chapter 16. Threads and Locks :  (18 pages)
Chapter 17. Documentation Comments :  (10 pages)
Chapter 18. LALR(1) Java Grammar :  (14 pages)
Chapter 19. The java.lang Package :  (152 pages)
Chapter 20. The java.util Package :  (46 pages)
Chapter 21. The java.io Package :  (98 pages)
```

このうち，本稿執筆時点のドラフトでは，第 1 章から第 18 章はおよそ 360 ページ，第 19 章以降は標準ライブラリ機能の定義で，それら 3 章をあわせると約 300 ページである.

4.2 Random Quotes from the Recent Correspondences

進行中の作業の過程から，基礎的ないくつかを抜きだし，それを文献 [4] に含めたが，それ以後，最近のやりとり（著者からの照会／リクエストとそれに対する応答）から主なものをいくつか拾っておく.

1）「Unicode 準拠」の定義は $\Longrightarrow$ Unicode 1.1
2）ラベルと名前空間との関連は $\Longrightarrow$ 独立
3）0（ゼロ）というリテラル表現の意味は $\Longrightarrow$ 10 進数のゼロ

4)"`public static void main (String args [])`" あるいは `argv` か $\Longrightarrow$「どちらでも問題ではない．JDK は argv だが，Bill は JLS では args としている」

5）無名パッケージの個数は $\Longrightarrow$ 一つだけ.（ただし，これはパッケージ名とファイルシステムのマッピングの問題で，複数のホームディレクトリでコンパイルする環境では別の話となりうる）

6）言語標準のパッケージには java.net は入らないのか $\Longrightarrow$ java.net は標準パッケージの一つだが，JLS 出版の時間的制約で検討が間に合わなかったので入っていない．

7）System.out.print の fully qualified name は java.lang.System.out.print か $\Longrightarrow$ Yes

8）abstract class/method と interface の役割の整理に関して $\Longrightarrow$ 検討すべき点はいろいろあるが，ひとまずこのままで行く

9）初期値を並べた配列の初期化ではサイズを陽に指定することはできない $\Longrightarrow$ Yes

10）JLS の対象読者に関連して $\Longrightarrow$ JLS はチュートリアルではないので，厳密な定義を助ける例はつけるが，導入的な例などを特に意識はしない

11）スタンドアロンの場合，public static void main メソッドを起動するというのは JDK の要求か JLS の要求か $\Longrightarrow$ JLS

12）単純式をメソッドの中に文として書けないのは意図か $\Longrightarrow$ Yes

13）if 文の conditional compilation に関連する議論から $\Longrightarrow$ `if(false){ foo();}`では，`foo();` は unreachable だとして文句をいわれることは無い．

14）「hashCode() で，true については 1231，false については 1237 を返す」というのはどういう拘束性を考えているのか $\Longrightarrow$ この値でなければならない．このためのハッシュアルゴリズムは文章的に明記する．

15）unicode でのハングルの割り付けに関連する部分 $\Longrightarrow$ Unicode 2.0 で変わるのであれば，JLS では Unicode 1.1 のハングル文字を識別子として使わないように警告を入れる．

16）getYear() は西暦 2001 年に対しては何を返すか $\Longrightarrow$ 101．西暦マイナス 1900 を値とする（なお，Java 1.1 ではこれ以外のインタフェースも検討している）

4.3 ライブラリ機能の標準化について

java.io については継続した検討の必要があると思われる．とくに，現在の仕様では文字ストリームの扱いに 8 ビットを意識した部分があり，これを真に Unicode 対応させるための API の定義には，さまざまな知恵を集めることが重要と思われる．

java.net についても継続した検討の必要があると思われる．

JLS では対象外である java.awt，java.applet については急速なマルチメディアインタフェースへの要求に対応するべく根本的な変更も検討されているようだが，これは実際の使用形態や academic feedback を見た，息の長いプロセスが必要だと思われる．VRML，スクリプト言語などとの関連，セキュリティ機能 [5] というテーマもある．

5 Java 実行モデルとその課題

一般に，Java で記述されたプログラムの処理は，JDK を使うと，図 1（[4] より）のような手順により実行される．スタンドアロンで実行される場合と，クライアント／サーバモデルにより（WWW）サーバ上にバイナリを置き，それがクライアントから呼び出される際にクライアントに渡されて実行される場合の大別して二つの流れがある．後者の場合，実行可能オブジェクトはアプレットと呼ばれる．

JDK では，Java Virtual Machine (Java VM) に対する実行可能形式をバイナリオブジェクトとしている．このバイナリオブジェクトは，クラスライブラリをリンクして実行が可能になる．アプレットとよばれる形態の実行形式オブジェクトの場合，このクラスライブラリはクライアント側が供給するものである．このことから，次のような基本的な性質が生まれる．

> Java の実行環境は仮想マシンの中に閉じている．また，アプレットの実行機能の実体はクライアントが供給するライブラリを使う．仮想マシンのインタプリタはクライアント側に居る．したがって，クライアント側で，渡されたアプレットの実行の制御が可能である．

また，用意されたデータ型は，大別するとプリミティブ型とリファレンスオブジェクト型になる．ポインタは数値ではなく，その実体はシステムにより管理される．

アプレットで使用するスタティックなオブジェクトはコンスタントプールと呼ばれる形式に圧縮されその中に含められて渡される．また，JDK では，Java VM のエミュレータをソフトウェアで実現しているので，実行速度としては必ずしも速くない．

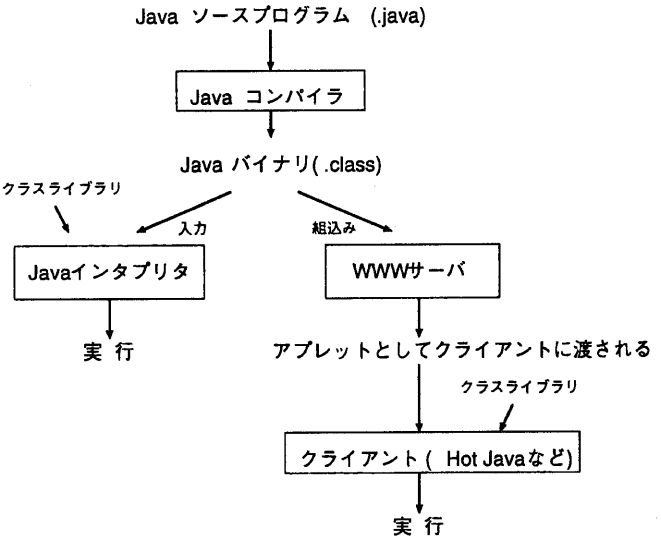

Figure 1: Javaプログラムの処理の流れ

これに関連する研究テーマとして，直接，Java VM を実現するハードウェアを用意し，それによって実行すること [7]，あるいは，使用するプラットフォームの機械語に再変換し実行する JIT コンパイラ（Just In Time Compiler），などがある．

また，アプレットの永続性に関する議論もある．アプレットは上述の性質から基本的にはクライアントに対してクリーンな「一時的な滞在者」である．このアプレットの永続性の議論も重要であり，分散コンピューティング環境の研究に本質的な場を与えてくれるものとも考えられる．

6 UTF8 形式による互換性の導入

Java の文字列は Unicode であり，名前などは UTF8 形式の Unicode で表現されなければならない．そして，ネットワーク環境で，相互に接続されたコンピュータ上の文字コードをどうするのか，という問題意識が必要となる．閉じたマシン環境では，それぞれ固有の文字コードとその表現形式で十分だった．それが，通信環境が整ってきて，特に異機種の間でのデータあるいはプログラムのやりとりができるようになってきた．そうすると，新しい次元のコードの共通性が問題になってくる．国際環境の分散コンピューティングでは，各国語のテキストのリアルタイムのやりとりということがでてくる．日本における Java の普及，あるいはこういったネットワーク環境における文字列の国際化対応は，これからもっと議論されなければいけない．もちろんこれは Java だけの問題ではなく，歴史的に今その問題が来ていると言うことができる．

UTF8 形式というものは，大変たくみに考えられている．この形式について紹介する．

UTF8 は，UTF 8 ビットという意味であり，UTF は，Unicode characterset Transformation Format の頭文字をとったものである．これは，簡単に言えば，Unicode の文字を転送するためのコード化の形式である．ISO/IEC JTC1/SC2/WG2 N 1036 を見るとその定義があるようだが，著者はまだそれは確認していない．UTF は 2 バイトの ISO 10646 だけでなく，4 バイトのコードも同一の仕方でコード化できるアイデアになっている．

`http://www.stonehand.com/unicode/standard/utf8.html`

に UTF8 形式についての記述がある．

UTF8 の基本的なアイデアは次のようなものである．

- 16 進の 0000 から 007F までの値のコードは先頭 1 バイトの 00 を省いて，1 バイトで表す．つまり，Unicode のそれらの範囲は ASCII 文字と同一なので，それらは 1 バイトで表される．ようするに，

ふつうの英文字を使っている分には，Unicode であるといっても今までと同様に扱われているので，なにも今までとは違いがなく互換だということである．ここに UTF8 形式の鍵がある．

Java の構文要素は UTF8 形式の Unicode で書かれている，といっても ASCII コードを使っている国でパニックにならないのは，こうした点がある．

- それ以外のコードは複数バイトで表す．1 バイト目が C0 から FD まではマルチバイトエンコーディングの最初の文字を表す．そして，最初のバイト以外のマルチバイトのエンコーディングのためのバイトでは，80 から BF までの形が使われる，つまり，マルチバイト表現の 2 バイト目以降は頭が二進で 10，下に 6 ビット有効な文字が入るというふうになる．さらに具体的に言うと，
  - 11 ビットまでの大きさのコードは 2 バイトで表す．先頭の 1 バイトは，C0 から DF まで．
  - 16 ビットまでの大きさのコードは 3 バイトで表す．先頭の 1 バイトは，E0 から EF まで．
  - 21 ビットまでの大きさのコードは 4 バイトで表す．先頭の 1 バイトは，F0 から F7 まで．
  - 26 ビットまでの大きさのコードは 5 バイトで表す．先頭の 1 バイトは，F8 から FB まで．
  - 31 ビットまでの大きさのコードは 6 バイトで表す．先頭の 1 バイトは，FC から FD まで．
  - 32 ビットの大きさのコードは 7 バイトで表す．先頭の 1 バイトは，FE．

なお，Java の場合には，さらにちょっと変更が加えられていて，null 文字は単に 1 バイトの 0 ではなくて，C080 と表される約束になっている．
以下に図示する．

`0xxxxxxx`	Unicode 文字 `00000000xxxxxxx` を表す
`110xxxxx 10yyyyyy`	Unicode 文字 `00000xxxxxyyyyyy` を表す
`1110xxxx 10yyyyyy 10zzzzzz`	Unicode 文字 `xxxxyyyyyyzzzzzz` を表す
`11110xxx 10yyyyyy 10zzzzzz 10wwwwww`	21 ビット文字を表す
`111110xx plus four bytes 10------`	26 ビット文字を表す
`1111110x plus five bytes 10------`	31 ビット文字を表す

このようなエンコーディングの仕方は，ほとんど ASCII だけれども，多少 ASCII でない文字があるという場合の文字列には，都合が良い．しかし，ほとんど ASCII でないが，ときどき ASCII であるというような文字列に対しては，サイズが大きくなる．たとえば，16 ビットであらわすような文字に対しては 2 バイトではなく 3 バイトで表現される．
（本章の内容は文献 [4] の 3.2 節をまとめ直したものである．）

7 Java に関連する動向より

7.1 Java と Corba

- 分散オブジェクト指向 OS として研究試作された Spring OS では，その 1.1 版で，Java interface を持った．

- PostModern Computing という会社が BlackWidow という Java Object Request Broker を発表している．ベータ版を http://www.pomoco.com からダウンロードし，自由使用できるとしている．

  BlackWidow は完全な CORBA 2.0 準拠のプログラムで Java で書かれている．クライアント側サーバ側両方の機能を提供している．BlackWidow を使ったアプレットを，Netscape Navigator 2.0 などにダウンロードして，それが，インターネット上の任意の場所にある Corba 2.0 準拠のオブジェクトと会話できる．

  この会社は，SmallTalk Object Request Brokers などというものもやっているらしい．

- 大学などでの実験的なシステムとしては Scott Hassan という人がやっているものなどがある．この人は，ILU (Xerox Inter-Language Unification) の関係者である．http://www-db.stanford.edu/~hassan/ に説明がある．

7.2 日本での Java 関連の話題のいくつか

日本でのものは HORB（平野氏@電総研）あるいは j2c（Yukio Andoh 氏による）などがある．

j2c は java to c のトランスレータで，http://www.webcity.co.jp/info/andoh/java/j2c.html に説明がある．FreeBSD, Solaris 2, SunOS, HP/UX, Linux, Win32 などで動作するらしい．

ETL HORB は，分散オブジェクト指向の研究の中にあるもので，http://ring.etl.go.jp/openlab/horb/ に説明がある．

7.3 他言語と Java

Tcl に関しては，96 年 3 月 6 日に，Tcl を Java のスクリプト言語として使うインタフェースである TclJava のアナウンスがある．(ftp://ftp.smli.com/pub/tcl/tcljava0.1.tar.gz)

GUILE と呼ぶ GNU 拡張言語には，Java インタフェースが用意されようとしている．

7.4 Java コンパイラ，JVM インタプリタ

GPL 準拠の Java コンパイラとしては，David Engberg 氏による guavac がある．これは JVM のコードを出力する．その 0.2.1 版では javac との互換性，高速性をうたっている．http://http.cs.berkeley.edu/~engberg/guavac に情報があり，ソースは， ftp://summit.stanford.edu/pub/guavac にある．gcc 2.7.2 with libg++ 2.7.1 で書かれている．

JVM の JIT インタプリタに関しては，Kaffe というものがある．FreeBSD, Linux, Solaris 2.4 (i386), BSDI などで動作する．http://www.sarc.city.ac.uk/homes/tim/kaffe/kaffe-0.3.tgz にある．

7.5 その他

開発環境などに関連するツールの話題がある．例えば Symantec の Cafe など，いろいろな商品が開発中である．Sun からは Java Workshop というものもある．

また，Java 対応のできる WWW ブラウザとしては，NetScape Navigator 2.0 が先行している．HotJava は最新の仕様に対応する版が近々出される予定とのことである．最近の話題では NetScape では Atlas PR2 (Mac) もあるが筆者はまだテストしていない．なお，Java とデータベースとのインタフェースも重要な課題であり，たとえば，[6] などがある．

JavaScript などのスクリプト言語については直接の関連がないので触れなかった．VBScript あるいは Microsoft 社の対応製品などの話題もあるが，これらも多分に流動的であり，また，本稿の目的とする部分には関連がないので触れなかった．

JavaVM 機能を包含するチップの開発などについては，たとえば [7] がある．

また，Java 関連ツールの紹介は，たとえば http://www.cybercom.net/~frog/javaide.html にある．

Java に関連する FAQ としては，例えば [8] あるいは [9] などがある．

References

[1] http://java.sun.com （http://www.javasoft.com も同じ）

[2] J. Gosling, H. McGilton, The Java Language Environment - A White Paper, Sun Microsystems Computer Company, May 1995

[3] The Java Language Specification v1.0beta, Sun Microsystems, Oct. 1995

[4] 井田，はやわかり Java，共立出版， April, 1996

[5] Java Products Group, Java Security API, draft alpha 0.9, JavaSoft, Sun MicroSystems, Feb.1996

[6] http://splash.javasoft.com/jdbc （Java and Database）

[7] http://www.sun.com:80/sparc/java/ （Java chip）

[8] http://java.sun.com/java.sun.com/faqIndex.html

[9] http://www-net.com/java/faq