JAVA についての一般的な情報

1-1　Java について（特長と利点）

Java はどのようなメリットがあり、なぜこれほど注目されているか？に注目して解説します。

Java とは何か

Java という単語が示している本来の意味はプログラミング言語です。つまり Java は C や C++の流れをくむプログラミング言語なのですが、昨今のニュースなどで扱われる Java という言葉は、もっと広い意味で使われています。Java というプログラミング言語をベースにして作成されるソフトウエアの利用形態全般を指して、現在では「Java」と呼んでいると言って良いでしょう。Java は単にアプリケーションを作る１つの言語という枠組みを超え、異なる OS の上でも同一のソフトウエアが機能するなど、従来にはない機能をもったソフトウエア環境なのです。その意味では、OS であるとも言えます。既存の OS の上に構築された独自の OS でもありますし、JavaOS として独立した OS も開発されています。

最初はインターネットで注目された

Java が本格的に世の中に登場したのは、1996 年の初頭です。Java 開発の素材や実行環境のおおもととなる JDK 1.0 のリリースが、96 年の１月なのです。その時期、Java は Web ブラウザで動くソフトウエアであるアプレットにフォーカスしていました。Web というシステムは、世界中の文書をリンクしてつなぐなど、従来のペーパーによる文書とは異なる独自の特徴を持ち、こうした文書をブラウザさえあれば、OS やパソコンの機種に関係なく参照できました。また、文書自体は HTML として、テキストで記述できる手軽さもあります。しかし、Web の本来の機能だけを使った文書は、文字やグラフィックスをレイアウトする機能しか利用できません。当然、HTML 文書としての作成機能にはあきたらず、たとえば動きを付けたり、ユーザーの操作を受け付けるなど、もっといろいろなことを Web という基盤をもとにやりたくなってきます。そのためにいろいろな仕掛けが考えられたのですが、その中でも Java は汎用的なプログラミングが可能なことで、大きな可能性に期待が集まりました。一言で言えば、Web ブラウザで表示した文書内で、ソフトウエアが機能するわけです。

インターネットでの“動くページ”からは脱皮している

ただし、97 年頃からは、整備されていないライブラリから来る Java のシステムとしての完成度の低さがネックとなり、また Shockwave、Flash といった、インタラクティブで動きのある Web ページをより作りやすくするシステムが次々と登場したことによって、このような用途での注目は現在では浴びていません。Flash や Shockwave は Web ブラウザではプラグインが必要ですが、OS あるいはブラウザにバンドルされる傾向にあり、Web 上でのマルチメディアプレゼンテーションの標準プラットフォームになりつつあります。

このような状況を背景に Java は当初注目されるきっかけとなったインターネットの分野にとどまらず、もっと広い範囲で利用できるのではないかと期待されてきました。たとえば、業務システム開発や、あるいは組み込みシステムなどでの利用に注目が集まっています。また、分散オブジェクト開発のツールとしても Java は有力視されています。

つまり Java はプログラミング言語なのですが、見方を変えれば Java は OS そのものです。既存の OS に組み込まれた独自の OS なのです。既存の特定の OS を超えて Java という独自の OS で機能するソフトウエアを作るということが可能なのです。もはや Java はインターネットという枠組みを超えて大きく広り新たな発展をし始めていると言えるでしょう。

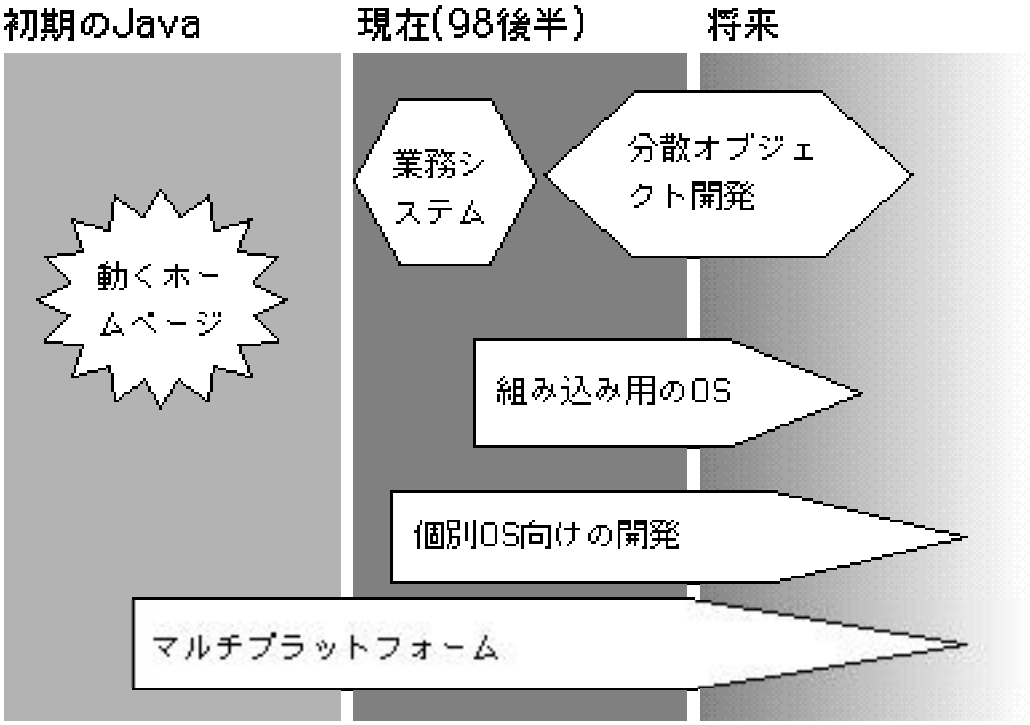

図 1-1　Java の役割の拡大

言語としてみた Java の利点

まず、Java をプログラミング言語として見てみましょう。Java をプログラミング開発言語として見た場合、従来の開発言語とどこが違い、どんな点にメリットがあるかを検討しましょう。

Java は C++をベースに開発されたもので、基本的な記述方法は、C あるいは C++と共通な部分があります。特に変数や繰り返し、条件分岐などの基本的な部分はほとんど同じと言ってよいでしょう。そのため、C や C++の経験者にとって、Java はそれほど違和感がありません。

Java プログラミングの特徴を一言で言えば、オブジェクト指向プログラミングを全面的に取り入れているということでしょう。

C はオブジェクト指向プログラミングの機能はほとんどないと言えるのですが、C++はオブジェクト指向プログラミングの中心的な開発言語となっており、開発ではもっともよく利用されています。しかしながら、C++は拡張を繰り返したために、高機能ではあるものの、非常に複雑で難しい言語になってしまいました。Java は C++よりもずっとシンプルでわかりやすい構成になるように改良されています。その反面 C++よろは機能がないと言うこともできますが、C++にあって Java にはない機能を補いう補う包括的な機能もあり、Java は C++と概ね同等のことができると言えます。また、シンプルな言語体系の方が理解しやすく、また、プログラム自体も筋の通ったものになりやすい傾向があります。こうした言語としての分かりやすさ、そこから派生して生産性の高い言語であるというのが Java の特徴です。

何でもオブジェクトとして扱う

Java の特徴は、データを一般的に「オブジェクト」として扱うことです。オブジェクトは言い換えれば構造を自由に定義できる柔軟なデータです。逆に言えば、Java ではオブジェクトとして定義しないとプログラム中でのデータの利用は非常にやりにくくなります。オブジェクトとして一元的にいろいろな形式のデータを保管したり処理をするというのがともかく基本になります。

Java ではC言語での「ポインタ」というデータはないのですが、「オブジェクトへの参照」というデータを変数に代入しておくことはできます。

Cでは、「ポインタ」というデータのアドレスを利用して、データにアクセスするような手法がよく利用されていました。しかし、ポインタはプログラムを分かりにくくすると同時に、誤って関係のない部分をアクセスして書き直し、暴走する原因にもなっていました。しかしながら、Java はそうした心配はありません。

Java のオブジェクト参照は、物理的にはオブジェクトを保存したメモリへのアドレスということになるのですが、Java ではオブジェクトへの参照は演算対象になりません。オブジェクトへの参照は他の変数に代入することしかできないので、ポインタ演算のようなことはできません。オブジェクトへの参照が保存されているということを概念的に理解していればいいのです。

Java にも配列はありますが、C や C++と使い方は異なり、オブジェクト的に扱えるようになっています。

こうした特徴をメリットと見るか、デメリットと見るかはもちろん意見が分かれるでしょうが、やはりメリットと見るべきでしょう。分かりやすく安全なソフトウエアにつながるからです。しかしながら、オブジェクト指向を全面的に採用しているために、うまくオブジェクトを定義しないと、プログラム自体が混乱するということもあります。

オブジェクトとして定義されたライブラリ

Java でソフトウエアを組む場合、言語本来の機能だけを使って組むわけではありません。Java に限らずOS をベースにしたプログラム開発では、OS の機能やライブラリなどを利用したプログラミングが必要です。したがって、プログラミングにあたっては Java の文法うんぬんよりも、ライブラリの使い方の知識の方が膨大になり、必要となります。

Java でもやはりライブラリが用意されています。特にコアになる基本的なライブラリは、Java の供給元である Sun Microsystems より供給されており、標準化されています。これら Java で使えるライブラリは、すべてがクラスとして定義されたオブジェクト指向のライブラリになっています。つまり、ライブラリを利用する上では、オブジェクト指向プログラミングをしなければならないのです。ここでも、全面的にオブジェクト指向が取り入れられていることになります。

メンテナンスが不要なメモリ利用

プログラムでは何らかの形で必ずメモリを必要とします。特に C や C++ではメモリを確保し、そして解放するということを行わないと正しくメモリが利用できません。メモリの取得はともかく、確実に解放するというのは、プログラムする上ではかなり難しい作業です。処理途中のエラーなどで、確保されたメモリが使われずに放置されることは実際によくあることです。

Java ではメモリの解放という作業は不要で、全面的に自動的に行われます。取得は明示的に行いますが、不要になったかどうかはそのオブジェクトを利用しているかどうかがチェックされ、いらなくなったら自動的に解放するということを行っています。そのため、プログラマはともかく必要なときにメモリを確保しさえすれば良いのです。結果的に、プログラムは非常に作りやすくなります。

C あるいは C++でのプログラミングと、Visual Basic など BASIC 言語あるいはマクロ系のプログラミングとの１つの決定的な違いに、メモリ利用をプログラミングしないといけないかどうかがあります。BASIC 言語はそれが不要なために、気軽にプログラミングができるという面もあります。Java ではそうした BASIC 言語的な気軽さと、C や C++に匹敵するオブジェクト指向あるいは高性能なソフトウエアをつくり出すという両方のいいところを併せた特徴を持っていると言えるでしょう。

なお、不要になったメモリ領域を解放することは「ガベージコレクション」と呼ばれます。この作業自体は自動的に行われますが、そのために時間を使ってしまい、ソフトウエア全体のパフォーマンスが低下するというデメリットもあります。

スレッドによるマルチタスク

Java の基本的なライブラリには、スレッドを利用したマルチタスクの機能が組み込まれており、気軽にマルチタスク処理を組み込むことができます。これを利用して非同期処理を組み込むことができますし、アニメーションのような画像処理も比較的手軽にできるようになっています。

実行環境としてみた Java の利点

Java をソフトウエアの実行環境としてみてみます。やはりいちばんの特徴は「マルチプラットフォーム」でしょう。Java のデビューのときにもっとも注目されたのがこの点で、同じソフトウエアが Windows でも Mac OS でも、UNIX でも実行できるということです。

クロスプラットフォームでのバイナリ互換

C 言語で標準ライブラリだけを使ったプログラムなら、若干修正の必要はありますが、どの OS でも機能します。しかしながら、Windows でコンパイルして作った EXE ファイルは、Mac OS や UNIX では実行できません。また逆も同様です。

Java のソフトウエアは、Windows で作った実行可能なファイルを、Windows ではもちろん、Mac OS でも UNIX でもそのまま実行することができます。「プログラムのソースに手を加えずに、各 OS ごとにコンパイルして実行プログラムを作成できる」というレベルではないのです。Mac OS で作った実行プログラムでも、原理的には Windows でも UNIX でも機能するというのが Java で言う「クロスプラットフォーム」という特徴なのです。このような特徴を「バイナリ互換」とも呼んでいます。

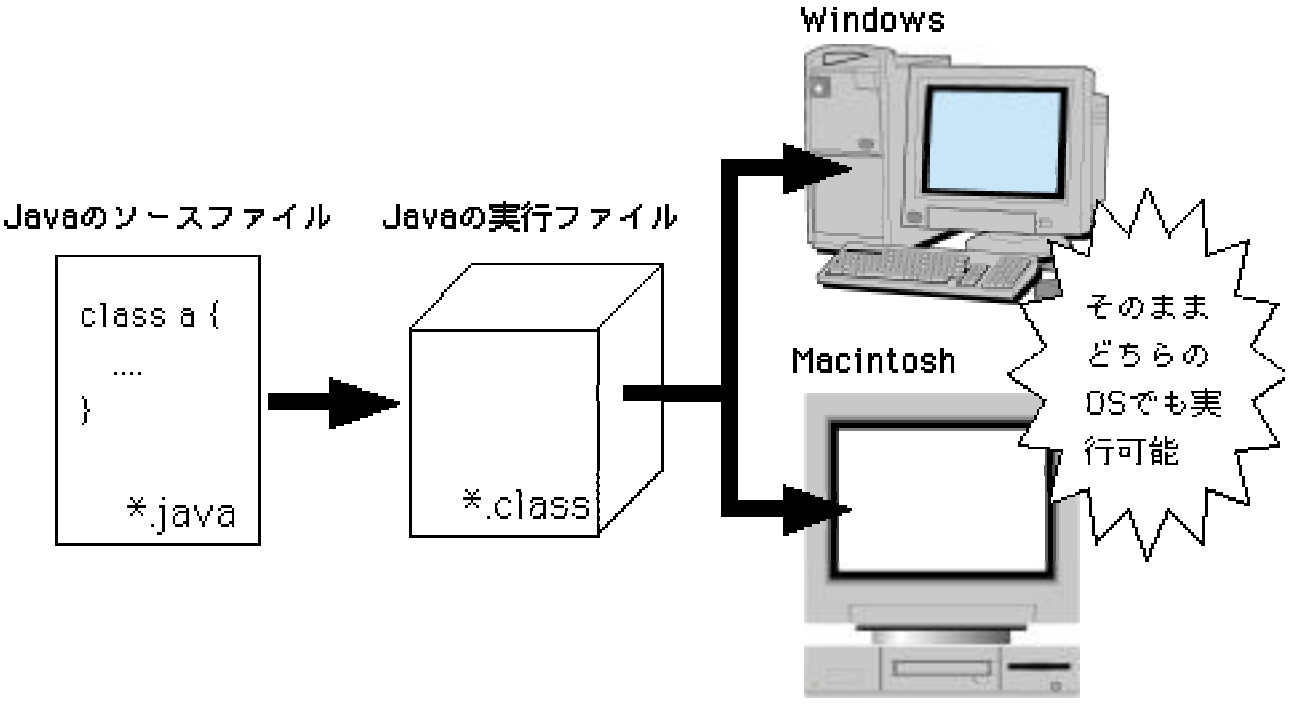

図 1-2　Java におけるバイナリ互換

通常、CPU や OS が違うと、実行可能なバイナリの形式も異なり、共通には使えません。では Java ならなぜできるかと言えば、簡単に言えば、OS や CPU とは別に共通の「Java 実行環境」を構築しているからです。つまり、Windows や Mac OS には、Java を実行するためのコンピュータが内蔵されています。そのコンピュータはソフトウエアで実現されており、バーチャルマシン（VM：Virtual Machine）あるいは VM などと呼ばれています。

この VM が OS に内蔵されていれば、同一のバイナリファイルが、どの OS ででも実行できるのです。もちろん、各 OS に VM が用意されていないと Java は実行できませんが、現実には Windows も Mac OS も、VM をブラウザメーカーあるいは OS メーカーによって積極的に構築されており、結果的に世の中に存在するほとんどのパソコンで、Java の VM が機能しているという状況になっています。

また、VM そのものに加えて、各 VM の実行環境で、クラスライブラリが提供されています。VM とセットにして提供されるべき基本的なクラスライブラリがしっかり定義されており、それが十分に OS 機能を果たしているため、Java は OS に依存しない共通の基盤を持った OS として機能することになるのです。

なお、VM は完全に 1 から構築されたものではなく、部分的に各 OS の機能を利用します。たとえば、Java のソフトウエアがウィンドウを表示したりキー入力を受け付けるときには、実際には背後で稼動している OS の機能を流用します。つまりクラスライブラリの見えない部分で OS の機能を利用しているのです。このように Java には稼動している OS の機能を利用するメカニズムも用意されています。ただし、一般のプログラマはその部分を扱う必要はなく、クラスライブラリの機能だけを見ていればいいでしょう。

ちなみに先程触れたように、Java はデビュー時には Web ブラウザでの利用にフォーカスされていたため、Web ブラウザ自体に VM が組み込まれました。Netscape Communicator/Navigator では、ブラウザ自身が用意した VM しか利用できません。Internet Explorer はブラウザ向けに用意したものと OS に組み込んだものを切り替えることができます。Netscape の VM のうち、Macintosh 版は Ver.4.5 でも最新版 Java のライブラリ機能にすべては対応していません。Netscape もいずれは OS の VM を利用するような形態になる予定です。

VM の問題点

Java は VM 上で機能することから、OS に依存しないソフトウエアを、しかもバイナリ互換で作成できるのが大きなメリットです。しかし、逆にこのことがデメリットにもなっています。VM を実現するには、Java プログラムのバイナリコードを実行する CPU をエミュレーションすることになり、どうしても実行速度は遅くなります。つまり、バイナリーコードを実際に CPU が実行するのではなく、搭載している CPU 上のプログラムを使ってあたかも Java 対応 CPU が機能しているかのように見せ掛けているわけです。

ただ、VM であるために、通常のアプリケーションなどに比べて実行速度が低かった点については年々解決されてきています。1 つの技術は「JIT（Just In-Time）コンパイラ」と呼ばれるもので、Symantec 社の製品が有名です。これは、Java のバイナリーコードを実行直前に、稼動している CPU 向けに書き直して、そのパソコンで動いている CPU のコードに翻訳して実行するというものです。実際には CPU 本来のバイナリで実行するので、スピードは速くなります。たとえば、Mac OS だと、Java のバイナリコードが、PowerPC の実行可能なコードに翻訳されて実行するわけです。最近の VM には JIT も標準で組み込まれるようになり、実行速度は目に見えて改善されています。ただし、JIT の場合は、実行前にコードを翻訳する時間が必要になり、CPU があまり高いパフォーマンスでない場合には、ソフトウエアの起動に時間がかかるようになってしまいます。

さらに、99 年には、HotSpot と呼ばれる記述が実用化される予定です。これは言うならば動的コンパイラであり、実行時間がかかる処理を順次稼動している CPU のコードにコンパイルするような処理を実現します。また、メモリー利用や複数のタスクの実行機能などに改良が加えられ、とにかく効率の良いソフトウエアの処理が実現するはずです。

近い将来、このようにエミュレーションで動く VM であるために実行速度が遅いという Java の現状のデメリットは、解決されると期待して良さそうです。

動的なリンクにより軽量ソフトウエアを実現

Java で作成したソフトウエアは、動的なリンク機能により、実行するときに必要なライブラリを探して結合することができます。C や C++でも動的ライブラリは不可能ではないのですが、一般にはそうした機能は OS の機能などとして提供されています。通常、コンパイラとしてはあらかじめライブラリを取り込んで、実行ファイルを作ります。

Java は、コアになるクラスライブラリについては、コンパイルしてバイナリを作成するときには一体化しません。実際に実行する段階で、その場にあるライブラリの本体を呼び出して結合し、実行します。従来のアプリケーションでは、ライブラリを実行ファイルに含める必要があったため、シンプルなアプリケーションでもそれなりに大きなサイズになっていました。しかしながら、Java ではライブラリをいっしょにしなくてもいいので、シンプルなものは非常に小さいサイズのファイルで事が足ります。

こうした機能はインターネットを経由して取り込まれたアプレットの実行において、ダウンロード時間の節約になるというメリットにもつながります。いずれにしても、重複したライブラリ利用は避けることができるというわけです。

安全で強固な環境

先に Java の言語としての特徴として、「ポインタ」がないことを説明しました。C 言語のようにポインタ演算でデータの位置を決める場合、バグやプログラマが想定していない状況が起きた結果、データ範囲外を読み書きしてしまう可能性が出てきます。そうすると、ソフトウエアの動作が不安定になることがあります。Java 言語ではデータの範囲外をアクセスするようなプログラミングはほとんど不可能なので、関係ないところを読み書きするようなプログラムは書くことができません。これだけでも、データやロードしたソフトウエアA 自体を保護する機能が働いていると言えるでしょう。

また、Java の実行環境では「セキュリティ」という考え方が導入されています。VM の動作として実行可能な範囲を制限できるのです。たとえば、ファイルの読み書きの処理は、Web ページに組み込まれるアプレットではできない処理です。仮にできるようになっていれば、Web ページを読むと、ハードディスクの中身を書き換えるというアプレットも作成できますが、悪意を持ったソフトであれば被害も大きくなります。また、悪意はなくても動作が不安定だと何をするか分かりません。そこで Web ブラウザで開いた Web ページにあるアプレットは、そのページ内、あるいは同一サーバーに対してのみ処理ができるという制約が付けられています。こうした制約を施すというセキュリティがあるために、Java で作成したアプレットは安全なソフトウエアとして機能することが保証できるのです。

1-2　Java で開発できるソフトウエア

Java ではいろいろな形態のソフトウエアが作成可能ですが、現状では、アプリケーションとアプレットというのが一般的な形です。最近はサーバーで機能するサーブレットも作成できるようになってきています。また、コンポーネントとしての JavaBeans も注目を集めています。これらについて説明をしましょう。

Web ページに埋め込むアプレット

アプレットは Java の登場時から存在するソフトウエアの形態で、Web ページに埋め込んで利用されるものです。以下に説明するように、従来にはない形態のソフトウエアであったために、Java に注目が集まったと言っても過言ではありません。

アプレットは、拡張子が.class というファイルに保存された Java の実行形式のバイナリファイルを、Web ページ内で機能させたものです。なお、アプレットは.class ファイルにあるのが一般的ですが、アーカイブファイルの.jar ファイルに存在することもあります。Web ページは HTML テキストでその内容を記述

できますが、APPLET タグを利用して、アプレットを Web ページに埋め込みます。見かけ上は JPEG などのグラフィックスと同じように長方形の範囲を確保し、その中でソフトウエアの実行が行われます。

アプレットの.class ファイルは、HTML ファイルと同じサーバーに置かれます。画像ファイルもサーバーに置かれますが、それと同じイメージで考えて下さい。HTML テキストの中に APPLET タグがあると、サーバーからクライアントのブラウザにアプレットがダウンロードされます。そして、Web ページを参照しているクライアントの中にある VM を利用して、アプレットが実行されます。

このとき、Web ブラウザが Mac OS で機能していても、Windows で機能していても、サーバーから同一の.class ファイルがクライアントにやってきます。VM という共通の基盤が各 OS あるいはブラウザに用意されているので、同じ.class ファイルが違う OS で機能するということになるのです。

サーバー上で公開されているアプレットは、場合によっては不特定多数のクライアントで実行されます。そこでセキュリティを確保するために、アプレットで利用できる VM の機能は制限されています。その制限はブラウザである程度はコントロールできますが、中にはファイル処理など絶対にできないこともあります。

また、アプレットは Java の場合、極めて簡単にプログラミングできる点も見逃せません。アプレットの原形にあたるクラスが用意されているので、そのクラスを継承して、望む機能を追加するだけでアプレットは作成できます。サーバーからのダウンロードするなどのややこしいことは全部 Web ブラウザと VM がやってくれます。

Java のアプリケーション

Java でも一般的なアプリケーションを作成できます。Mac OS では、ダブルクリックして実行可能なごく普通のアプリケーションを作成できます。Windows では、EXE ファイルを作成することが可能です。ただし、一般にはその OS で実行可能になっているファイルをそのまま別の OS に持って行っても実行することはできないでしょう。たとえ Java のバイナリ部分はどの OS で動いたとしても、アプリケーションとして実行させる部分はファイルに含まれることになり、それが OS ごとに違うからです。

だからといって、OS ごとにアプリケーションを 1 から作る必要があるわけではありません。Java VM は共通なので、Java のソースコードは、Mac OS、Windows 用で同一のものを利用できます。それをもとに、Mac OS 用のアプリケーションを作ったり、Windows 用のアプリケーションを作るということをすればいいわけです。ただし、1 つの開発ツールで Mac OS 用と Windows 用の実行可能ファイル作成ができるとは限らず、それぞれ別の開発環境を使わないといけないかもしれません。

なお、Java のソースコードをコンパイルした結果は.class ファイルに保存されますが、ファイルの拡張子についてはこのようにアプレットだけでなくアプリケーションも同様です。

Mac OS にも Windows でも、.class 形式のアプリケーションを実行するツールは存在します。それらのツールを利用すると、同一の.class ファイルのアプリケーションが、Mac OS でも Windows でも実行できます。

サーブレット

Web ページで、入力するテキストボックスなどがあって、閲覧者の入力を受け付けるものも見られます。そのようなページの多くは「ボタンをクリックすると入力データがサーバーに送付され、サーバー側でプログラムを実行して、入力データをデータベースに保存する」などの処理を行うようなメカニズムを CGI（Common Gateway Inteface）で実現しています。このような、ユーザーのリクエストによってサーバー上で実行するようなソフトウエアも、Java で作成できるようになっています。こうしたソフトウエアを「サーブレット」と呼んでいます。

サーブレットを実行するには、サーバーにVMが組み込まれていることに加えて、Webサーバーがサーブレット実行の機能を持って行なければなりません。一般には、Webサーバーがサーブレットを実行するという手順になります。もともとサーブレットを実行可能なWebサーバーもありますが、サーバーのプラグインとしてサーブレットの実行環境が提供されている場合もあります。現状ではむしろ後者の方が一般的なサーブレット実行の方法になるでしょう。

JavaBeansとコンポーネント

ソフトウエアを部品化することを「コンポーネント化」などと言いますが、Javaでは、JavaBeansという枠組みで、基本的なコンポーネント化の機能をサポートしています。JavaBeansに従ったソフトウエアの部品を「Bean」と呼んでいます。

通常Beanだけでは必要な機能は満たされません。Beanを組み合わせることによって、アプレットやアプリケーションを作ります。このとき、グラフィカルな開発環境などで部品として取り扱い可能な形式はJavaBeansとして仕様が定められています。開発環境で部品と部品を線でつないで、アクションに対する処理を選択するようなことを可能にするのがJavaBeansの１つの目的です。

さらに、たとえばサーバーで機能するソフトウエアを作るためのEnterprise JavaBeansのような規格も作られています。サーバーで実行されるソフトウエアはクライアントからのリクエストに応じた形式で必要な処理を行うので、その意味ではシステム全体から見ればソフトウエアの部品のように見えるわけです。こうした部品を作る枠組みがEnterprise JavaBeansということになります。

1-3　フレームワークとしての Java

Javaを特徴付けるものとして、そのクラスライブラリがあります。クラスライブラリは誰もが自由に作ることができますが、基本になる部分はSun Microsystemsで開発され公開されています。基本部分は、いわばOSとしての機能と言っても良い部分です。

作成ソフトで使う機能を提供するライブラリ

Javaが初期の頃は、基本クラスライブラリの機能が、一般的なOSに比べて目に見えて貧弱でした。しかしながら、Javaのリリースが進むにつれて、一般のOSにかなり近付いてきています。また、ネットワーク機能のように、一般のOSよりもより使いやすく充実した機能もあります。

こうした基本クラスライブラリは、アプリケーションやアプレットなどJavaソフトウエアを作るフレームワーク（枠組み、あるいはひな形）となります。もちろん、こうしたクラスライブラリを無視して独自に構築してもいいのですが、普通はそうはしません。与えられたクラスライブラリをいかにうまく利用するかと言うのがプログラミングの基本になります。基本クラスライブラリの内容についての概略を説明しましょう。

なお、いくつかのクラスを集めたものを「パッケージ」と呼んでいます。パッケージは概念的なものと思ってもいいのですが、実際には特定のクラスを集めるものとしてプログラム上で定義されているものです。

充実したクラスライブラリ

クラスライブラリの中には、Webブラウザに組み込むアプレットの原形であるAppletクラスが定義されています。アプレットとして機能するために必要なさまざまなものがすでに用意されています。アプレットは、このクラスを継承して作成します。極端に言えば、アプレットの中身をどうするかという記述だけを加えればアプレットは作成できてしまいます。しかも、テキストボックスやドロップダウンリストなどのユーザーインタフェースを構築するオブジェクトは、次に説明するAWTで定義されており、アプレットには簡単に追加できます。また、再描画処理も簡単に記述できます。このようにアプレット作成にはかなり最適化されているのは、当たり前と言えば当たり前なのですが、Javaの特徴としては顕著な部分です。

Javaでは基本的なデータとして整数のint型などは使えるのですが、データはオブジェクトとしてある意味では一般化した使い方をします。クラスライブラリでは基本的なデータ型のオブジェクトに加えて、オブジェクトを複数まとめるような使い方をするためのクラス定義などもあり、オブジェクトを利用するための基本的な機能も提供しています。

グラフィックスとユーザーインタフェース

グラフィックス処理の中心になるのが、AWT（Abstruct Windowing Toolkit）と呼ばれるパッケージで、文字通りウィンドウ処理を行うものですが、現実にはもっとたくさんのことができると考えてよいでしょう。グラフィックス処理や基本的なGUIの一切を取り仕切るパッケージです。

まず、AWTにはウィンドウ表示のもとになる、一般的な画面表示領域を定義するクラスがあります。また、その中にオブジェクトを表示領域に簡単に加えることができるようなメカニズムを持ったクラスも定義されます。これらを利用して、テキストボックスやドロップダウンリストなどのクラスも定義されており、ユーザーインタフェースのオブジェクトを簡単に処理できるようになっています。

ただし、グラフィックス描画については、ごく基本的なことしか行なえません。複雑なグラフィック処理については、Java 2Dと呼ばれるSun Microsystemsより提供されるパッケージでサポートしています。

Javaのライブラリは「イベント」として、何か変化があったことを別のオブジェクトに伝達する手段を備えています。ユーザーインタフェースオブジェクトを使うときには、必ず利用することになるでしょう。たとえば、ユーザーがボタンを押したり、リストを選択したときなどにイベントが発生します。このイベントの扱いが、初期のJavaと、97年頃にリリースされたJDK 1.1とではやや異なっています。当然、後から出た方がより高機能なのですが、Macintosh版のNetscapeでは未だに新しいイベント処理がVMに組み込まれていません。ここは注意が必要なところです。

98年頃より、Swingという名称のパッケージ群がSun Microsystemsより提供され始めており、より高度なユーザーインタフェース構築のための機能が利用できるようになってきています。AWTでは限られていたオブジェクトの種類も、Swingでは豊富に利用できます。正しくは、SwingはJFC（Java Foundation Classes）というライブラリの一部分で、JFCにはさらにドラッグ&ドロップの扱いなども組み込まれます。

また、マルチメディアでは、ビデオ画像などを扱うJava Mediaや、3Dオブジェクトを扱うJava 3Dなども利用されはじめています。

入出力処理とネットワーク

Javaにはファイルの処理を行うパッケージも用意されています。このパッケージには単純なバイト単位の入出力だけでなく、行単位の入出力を行う機能なども用意されていますが、どちらかと言えば基本的な機能が中心です。ドライブの情報などもアプリケーションなどでは知りたいところですが、そうしたOSに依存するような情報については標準のライブラリでは得ることができません。従ってパッケージで行なえるのは一般的なファイル処理に限られます。また、アプレットではファイル入出力がまったく利用できない点も注意が必要です。ただし、将来的には適切なセキュリティをかけた上で、アプレットでもファイル処理ができるようになるとされています。

Javaのライブラリを、一般のOSから見たときに大きく特徴付けているのがネットワーク関連のパッケージでしょう。Javaが注目され始めた当初はアプレットによってWebページで動く画面が作成できる点が強調されていましたが、TCP/IPをベースに通信するソフトウエアを簡単に作成できるという隠れた面も見逃せません。そのことがJavaがネットワークに強いと評価される大きな要因です。TCP/IPのソケットを簡単に用意でき、しかも、行単位の入出力などもできるので、電子メールを送付するようなプログラムくらいなら簡単に作成できます。

また、電子メールで言えば、JavaMail というパッケージにより、電子メールのやりとりも比較的簡単にプログラミングができます。ネットワーク、特にインターネットを使うプログラムとなると、Java はだんぜん強味を発揮するようになります

データベース環境

データベースにアクセスするためのパッケージとして、JDBC（Java DataBase Connectivity）が用意されています。JDBC 用のドライバを利用すれば、Java のソフトウエアからデータベースエンジンにアクセスできます。SQL 言語レベルの処理はもちろん、テーブルのデータをオブジェクトとして扱って処理するようなことも可能です。

分散オブジェクト

Java は分散オブジェクト環境での開発も視野に入っており、特に大規模な業務システムでの利用も始められているようです。オブジェクト間の通信では、CORBA をベースにすることもできるようになってます。また、RMI（Remote Method Invocation）として、別のマシンにあるオブジェクトのメソッドを呼び出すようなメカニズムも、分散オブジェクト環境を実現するのに使われています。そして、JavaIDL というパッケージで、CORBA などのオブジェクトリクエストブローカー（オブジェクト間通信を行う基本システム）を利用した分散オブジェクト構築までが可能になっています。

JDK について

JDK は Java Development Kit の略で、Sun Microsystems によって開発された Java 開発のための基本となる素材を集めたものです。JDK が Java 環境の正式なリリースで、その中には、開発ツール、基本クラスライブラリ、ドキュメントなどが含まれています。JDK はそのバージョンとともに記述されるのが一般的で、バージョンによりサポートされている機能が異なります。いずれにしても、JDK というのは、Java 開発の中ではよく出てくる言葉です。

JDK には基本クラスライブラリが含まれており、JDK を入手することによって、クラスライブラリが手に入ります。従って、そのクラスライブラリを使ったプログラムの作成もできます。あるいは、配付されたクラスライブラリを利用するソフトウエアの実行もできるようになるというわけです。

Java 開発ツールのメーカーは、JDK をライセンスされており、自社の開発ソフトに JDK としてリリースされたクラスライブラリを含めることができます。しかしながら、一般には JDK に対応した開発ソフトが発売されるのは、その JDK がリリースされてから時間が経過した後です。Sun Microsystems 以外のメーカーより供給される開発ツールを使いながらなおかつ JDK も入手する理由として、新しくリリースされるクラスライブラリをなるべく早く手に入れたいということが挙げられます。

また、JDK には Java のコンパイラなど、開発に必要なツールが含まれています。市販の開発ツールを利用するまでもなく、JDK だけでプログラムを実行可能な形式にすることができます。Sun の Web ページなどからダウンロードできるので、事実上フリーの開発ツールとして JDK が存在すると言えば良いでしょうか。

コンパイラ以外のツールもある

JDK はコンパイラだけでなく、いろいろなツールを含んでいます。アーカイブファイルの処理をするものもありますし、分散オブジェクト開発用のツールなどもあります。さらに、作成したプログラムの解説ドキュメントを作成する JavaDoc と呼ばれるツールも JDK に含まれます。クラスライブラリの解説ドキュメントを見たことがあるかもしれませんが、プログラム中のクラス定義から、そのクラスにあるメソッドなどの使い方を解説した HTML 文書を作成する機能もあります。