

Java 研修 WisdomsSoft

<http://www.wisdomsoft.jp/1.html>

石井裕太

2020 年 6 月 16 日

目次

第 1 章	Java と Java 言語	v
1.1	Java 仮想マシン	v
1.2	オブジェクト指向	vi
1.3	プログラミング言語 Java	vii
1.4	開発環境とコンパイラ	viii
第 2 章	初めての Java 言語	ix
2.1	初めての Java 言語	ix
2.2	コメント	x
2.3	字句変換と Unicode	x
2.4	トークン	xi
2.5	文	xi
2.6	変数と型	xii
2.7	リテラル	xiv
参考文献		xvii
参考文献		xvii

第 1 章

Java と Java 言語

1.1 Java 仮想マシン

1.1.1 なぜ、今 Java なのか

Java...Sun Microsystems 社が開発

- Java... システムとして
- Java 言語... 言語として

Java 言語で開発されたソフトウェアは、Java がサポートされている全ての環境で実行可能

- C,C++ など コンパイラ言語
- BASIC,Perl... インタプリタ言語

1.1.2 どこでも動く Java の秘密

コンパイルの流れ：ソースファイル→コンパイラ→オブジェクトファイル→リンカ→実行可能ファイル

インタプリタの流れ：ソースファイル→インタプリタ→実行

Java：ソースファイル→コンパイラ→Java バイトコード→仮想マシン→実行

1.2 オブジェクト指向

1.2.1 オブジェクトの役割

高水準言語は、手続き型と非手続き型に分類される

- 手続き型

命令手順を記述して問題を解決するアプローチ C,Pascal,FORTRAN など

- 非手続き型

手順ではなく必要な情報を与えることで問題を解決するアプローチ LISP,Prolog など

Java は、非手続き型に属するオブジェクト指向型に分類される。これは、プログラムの流れではなく、オブジェクトの役割に注目している

継承と抽象化

これまでのプログラムは、プログラムを組むときに流れを意識

- 順番に命令が実行される逐次
- 条件によって命令が選択される分岐
- 条件が満たされるまで繰り返す反復

CPU はこれらの計算を延々と繰り返している。一方、オブジェクト指向はこのような流れよりもプログラムを部品化して独立させることに注目する。

正しく部品化されたプログラムは他のプログラムコードの影響を受けない。このような状態をプログラムの独立性として評価される。

オブジェクト指向プログラミングでは、プログラムの部品をオブジェクト（物）として考える。実行手順ではなく、役割を与えられたオブジェクトを部品として開発し、最終的にオブジェクトを結びつけ、それぞれのオブジェクトが関係し合うことによって目的が達成される。

- 継承 既存プログラムが提供する提供する機能を引き継ぎ、拡張することができる機能。抽象→具体的な実装という流れでプログラムを設計できる。
- 多様性 目的を達成する達成するための方法にバラエティを持たせる手段。
- カプセル化 プログラムのオペレーションを帰省すること。外部に公開する必要のない内部事情のコードを隠蔽し、プログラムを利用する人間に必要なコードだけを公開できる。

1.3 プログラミング言語 Java

1.3.1 Java の誕生と歴史

家電製品市場のアプリケーション開発のため、プラットフォームに依存しない開発システムが求められた。まず、Oak が開発される。しかし、成功の目を見ることなく、製品名を Java に変更。

Java の開発環境 JDK(Java Development Kit) は Sun Microsystems の Web サイトから無償でダウンロードでき、誰もが自由に使える。

JDK1.0

- アプレットなどを中心に、クライアント側のネットワーク関連ソフトウェアや、小規模な GUI プログラミング、コマンドラインで利用する変換プログラムや圧縮、暗号などの研究プログラムに利用可能。
- 当時は完成度が不十分で、標準ライブラリが提供する昨日も貧弱

その後、サーブレットや JDBC と呼ばれる API の普及により、サーバー側のネットワーク関連ソフトウェアやデータベース関連ソフトウェアが注目を集める。セキュリティシステムも堅牢で、国際プログラムを得意とし、ビジネスの世界で支持されている。

JDK1.2

- Swing と呼ばれる GUI(Graphical User Interface) コンポーネントが登場し、クライアント側のソフトウェア開発で注目される。
- Java の動作は遅かったが、マシンの高速化と Java 技術の工場により十分採用できる範囲になった。
- 携帯コンピュータや携帯電話も発達し、これらに実装するアプリケーションとしても Java が流行った。

JDK1.3,JDK1.4

- マルチメディアの分野でも Java が実用可能な範囲になってきた。
- XML, サウンド、イメージなどの高度な入出力をサポートする API が新たに実装。
- 今後はゲーム開発、音楽編集、動画関連ソフトウェアにも期待

現在も Java は安定しているとは言えず、改良と拡張を繰り返して Java は進化し続けている。

1.3.2 Java を学ぶ

オブジェクト指向型言語である Java はプログラムの流れでなく、部品としての役割を重視して目的のソフトウェアを構築していく。しかしオブジェクト指向プログラミングには高度な設計力が必要となる。

Java 言語のようなオブジェクト指向型の開発は、C 言語のような手続き型とは異なり、巨大なソリューションの小さな部品を開発するという性質をもつ。このためオブジェクト指向は大規模な開発に向いている。大規模な開発には、設計図が必要

1.4 開発環境とコンパイラ

1.4.1 Java SDK

Java プラットフォーム専用のアプリケーションを開発するためには、Java バイトコードを生成しなければならない。Java にとっては Java バイトコードこそが機械語であり実行データ。

Java バイトコードを生成するためには、C/C++ 言語などによる開発と同様、高水準言語を用いて開発する方法が用いられる。Java バイトコードを生成するための専用の高水準言語が Java 言語。Java 言語のコンパイラは JDK に付属している。

第 2 章

初めての Java 言語

2.1 初めての Java 言語

2.1.1 最初のプログラム

Java 言語は C 言語のような手続き型言語とは異なり、ルーチンや関数といったプログラムコードの集積ではなく、何らかの役割を持った小さな部品単位で開発し、それらを組み立ててシステムを構築する。

このときの部品をオブジェクト指向ではクラスと呼んでいる。Java のあらゆるコードはクラスの中に属し、様々なクラスの関係によって 1 つのシステムが実現する。プログラムの流れは、クラスの動作として記述し、Java 言語ではクラスの動作のことをメソッドと呼ぶ。

オブジェクト指向プログラミングは手順ではなく役割単位で設計を行うため、実世界のような複雑な構造をもつオブジェクトを表現できる。

2.1.2 文字を表示する

```
System.out.print("表示する文字列");  
System.out.println("表示する文字列");
```

Java の標準クラスライブラリが様々な機能を提供することで、システムを隠蔽する。文字表示は System クラスの機能を使って文字を表示する

2.2 コメント

2.2.1 注釈を残す

プログラムを改変する必要があるとき、オープンソースなど共同作業の場合など、開発者以外の人ソースコードを読む際の理解の手助けにする

```
/* 注釈テキスト*/  
// 注釈テキスト
```

2.3 字句変換と Unicode

2.3.1 Unicode エスケープ

コンピュータの世界では文字も二進数として表現されている。国際的にどのコンピュータでも文字データは統一して扱うことができるように規格が定められた。これがいわゆる文字コードである。

1 バイトコード

- ASCII コード (American Standard Code for Information Interchange)
ANSI(American National Standard Institute) 米国規格協会が制定

しかし、日本語、中国語など文字の種類が多い言語には対応できない。そこで各国独自の文字コードが誕生。

- JIS
- Shift JIS
- 日本語 EUC

など

文字コードの乱立が問題となり、Microsoft や IBM などの米国企業が中心となって提唱したのが Unicode。ISO(International Organization for Standardization 国際標準化機構) で国際規格 ISO/IEC 10646 の一部として標準化されている。ASCII と異なり、全ての文字を 2 バイトで表現。

Java はどこでも動くのは、マシン環境だけでなく、世界の様々な国の言語環境も Unicode で記述することによって解決している。Java 言語は ASCII だけでプログラムできるように設計されている。

Unicode エスケープ

```
\u****  
”\u732B\u304C\u597D\u304D” → 猫 が 好 き
```

2.4 トークン

2.4.1 Java 言語の基本構造

コンパイラは、文字の並びを頭から順に解析して機械語 (Java の場合はバイトコード) へと変換しているにすぎない。

1. まず Unicode エスケープを検索して Unicode 文字に変換する。
2. 次に行末記号の分析が開始される。
3. Unicode エスケープと行末文字を処理した最終的な結果を入力要素と呼ぶ。
4. ソースプログラムは入力要素へと還元され、さらに入力要素はトークンへと分解される。
5. トークン：空白とコメントを除いた入力要素、ソースプログラムの最小単位

2.5 文

2.5.1 文を完成させる

トークンとは、英文に置ける英単語だが、英単語だけでは意味は伝わらない。複数のトークンを正しく組み合わせて何をやりたいのかを伝えなければならない。

- ブロック
- 空文
- 式文
- ローカル変数宣言文
- if 文
- switch 文
- while 文
- do 文
- for 文
- break 文
- continue 文

- return 文
- throw 文
- synchronized 文
- try 文
- 到達不能文

空文

;

何もせずに常に正常終了させることが保証されている。

2.6 変数と型

2.6.1 情報を保存する

プログラミングでは、与えられた情報を決められた手順で分析、加工して出力することが基本原理。その過程で必要な情報を保存することは極めて重要。保存といっても、いちいち記録ディスクに保存しているとプログラムは遅くて使い物にならない。

通常、プログラムが稼働中に必要な情報を保存するときは、主記憶装置に対して入出力が行われる。Java はどこでも動くという思想から、プログラマが実装システムの記録装置を意識する必要が内容に設計されているので、主記憶装置の扱いを意識する必要はない。

プログラム言語ではデータの保存には**変数**を用いる。変数は主記憶装置の記録場所の代名詞。プログラマはこの変数に情報を保存したり、保存した情報を引き出したりすることができる。

情報の保存のもう一つの大きな問題として、保存されている情報が計算用の数値なのか、表示する文字なのかの識別方法はどうなるか。

機械語レベルでは開発者の自己責任。どちらとして扱ってもいいが、人間のミスによりバグが発生しうる。

Java は保存された情報の種類の管理を行ってくれる。この管理を行うための情報を**型**と呼ぶ。

Java 言語の設計者たちは仕様書の中で「Java は強く型づけされた言語である」と表現している。

2.6.2 ローカル変数宣言文

変数の利用には、メソッド内でローカル変数宣言文を使う。ローカル変数はメソッド内でしか宣言することができず、メソッドが終了すると破棄される。

2.6.3 識別子

ローカル変数宣言文では、肩を表すキーワードと変数の名前を設定しなければならない。Java 言語ではこのような、何らかの実態を認識するための名前を識別子と呼ぶ。識別子は自由に設定できるが、Java 言語が定める命名規則に従わなければならない。Java 言語では、識別子は Java 文字と Java 数字の並びであると定義している。

Java 文字

ASCII 文字のアルファベット A-Z(`\u0041-\u005A`), a-z(`\u0061-\u007A`)、ASCII のアンダーライン_ (`\u005F`) とドル記号\$ (`\u0024`) のことを表す。

Java 数字

ASCII の 0-9(`\u0030-\u0039`) までの数字を含む Unicode 文字。

ただし、識別子の先頭 1 文字は Java 文字でなくてはならない。

2.6.4 確実な代入

Java 言語ではローカル変数宣言文で宣言された変数は、その値がアクセスされるまでに必ず値が保持されていなければならない。これを確実な代入と呼ぶ。コンパイラは変数のアクセスを認識すると、厳密な手順の解析を行って確実な代入状態であるか調べる。値が保持されていることが保証できない場合、コンパイル・エラーを発生させる。

2.6.5 変数宣言子

変数宣言

型 変数宣言子, 変数宣言子...

2.6.6 名前つけ規約

Java 言語ではソースの可読性を高めるために変数の識別子に対して名前つけ規約を定めている。

型	1 文字識別子
byte	b
byte 以外の整数型	i,j,k
char	c
float	f
double	d
long	l
String	s

変数は、意味のある名前を省略した、単語とならない短い小文字の並びであるべきとされる。

- temp:ごく一時的な利用に限られた変数
- val:値を保持する

原則として 1 文字の変数は推奨されない。しかし、一時てきな利用や汎用的な利用をされる変数であれば、変数型を連想させる文字を採用するような規約が定められている。

2.7 リテラル

2.7.1 整数リテラル

リテラル：そのままの値のトークン,10,"Kitty" などリテラルの値はコンパイル時に確定し、実行中に変化することはない。

とはいうものの、実は変数と同時にリテラルにも型が存在する。意味は変数リテラルかだけの違いであり、型の考え方は全く同じ。リテラルを変数に保存する場合、型が一致しなければならない。

小数点を含まない数字は**整数リテラル**と呼ばれ、整数型の変数に代入することができる。iValue = 10 という文では、整数リテラルは int 型として認識されている。

2^{31} 以上の数値を表現したい時は int 型では不十分。long 型の整数リテラルを記述するためには、数字の末尾に l または L を付加する。(小文字の l は 1 と誤認しやすいので L を使うべきと考えられている。) long 型では 2^{63} までの値を表現することが可能。

2.7.2 浮動小数点数リテラル

float や double 型の定数は浮動小数点数リテラルと呼ばれ、整数部、小数点、小数部、指数部、型接尾辞で構成される。

整数部 . 小数部 E 指数部 型接尾辞

浮動小数点数リテラルは、整数リテラルとは異なり、10 進数でのみ表記できる。

浮動小数点数リテラルもまた、型接尾辞を指定することで float 型なのか double 型なのかを区別することができる。float 型は f または F、double 型は d または D を末尾につける。接尾辞がないときは double 型であると解釈される。

2.7.3 文字と文字列リテラル

Java では文字が Unicode として扱われる。C 言語の char 型変数は ASCII 文字を表すための 1 バイトだったが、Java 言語の char 型は 2 バイトであることに注意。

char 型で表現できる 1 つの Unicode 文字を文字リテラルと呼ぶ。文字と文字列は異なることに注意。

- 文字... 一つの Unicode
- 文字列... 複数の文字の並び

文字リテラルは ' で囲む。

char 型に整数リテラルを代入できる理由は、char 型が整数型に分類されることから理由づけされる。print() および println() メソッドは char 型のデータを受け取るとそれを文字として表示する性質がある。逆も可能で、

```
int iValue = '猫';
```

iValue には Unicode 文字 '猫' を示すコード 0x732B が格納される。

2.7.4 エスケープ・シーケンス

文字や文字列リテラルは、引用符の間に改行することはできない。改行を表現したい場合、文字列リテラルの中に二重引用符や水平タブを使いたい場合も同様。

このように文字としては表現が難しい「'」「”」のような記号、表示できない文字などを表現したい場合はエスケープ・シーケンスを使う。エスケープ・シーケンスはバックスラッシュ \ (\u005C) とそれに続く ASCII 文字で構成される。エスケープ・シーケンスはこれで 1 つの文字と解釈されるため、文字リテラルとして指定することも可能。Unicode

表 2.1 エスケープ・シーケンス

エスケープ・シーケンス	Unicode	意味
\b	0008	バックスペース
\t	0009	水平タブ
\n	000A	改行
\f	000C	フォーム・フィード
\r	000D	復帰
\”	0022	二重引用符
\’	0027	引用符
\\	005C	バックスラッシュ
\XXX	0000 ~ 00FF	00~FF までの 8 進数で表現されたコード

エスケープとは異なり、コンパイルの軸変換サイクルで処理されるわけではない。コンパイラがエスケープ・シーケンスを処理する頃にはすでに \u は変換済みであることに注意。

2.7.5 審議リテラル

boolean 型には true と false の二つの値が入る。この true と false を審議リテラルと呼ぶ。

2.7.6 null リテラル

変数が保持する情報が存在しないことを示す null 型が存在。null 型は唯一、null という値を保持し、ASCII 文字を用いた **null** リテラルは常に null 型となる。null は数値型の変数に代入することができず、代入できるのは何らかのオブジェクトの位置を表す参照型と呼ばれるタイプの変数だけ。

参考文献

- [1] 赤坂玲音, 読本 Java 2011/11/14 毎日コミュニケーションズ
<http://www.wisdomsoft.jp/1.html>