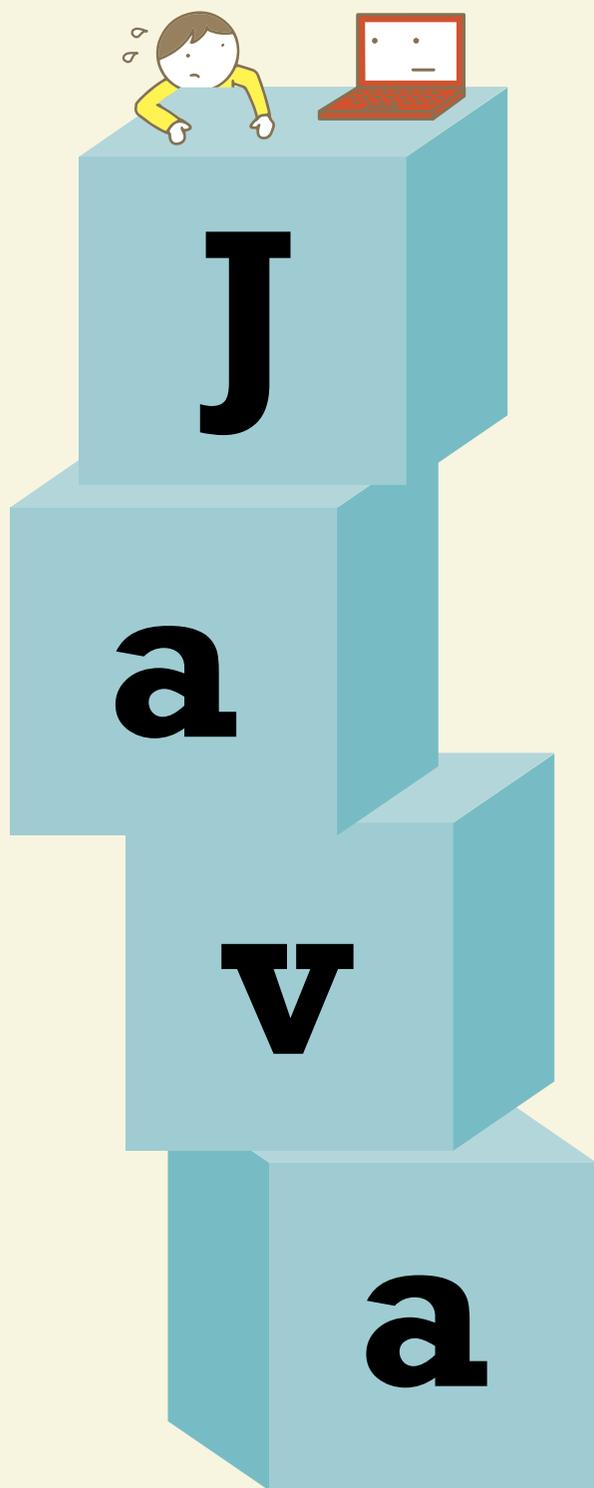
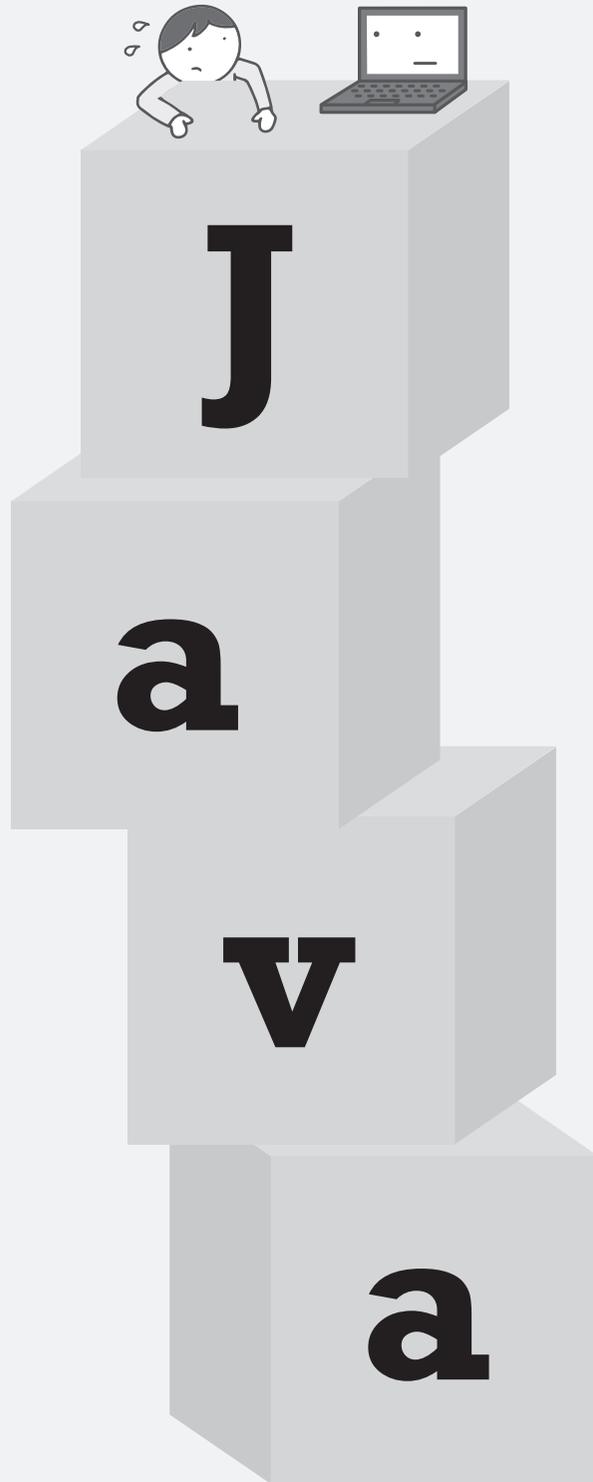


基礎から学ぶ

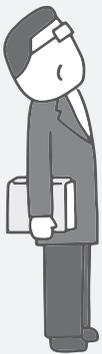


〜基本文法からオブジェクト指向まで〜

基礎から学ぶ



〜基本文法からオブジェクト指向まで〜



サポート情報／開発環境の準備手順／ソースコード／練習問題の解答例

本書に記載された項目に関する最新情報や、開発環境の準備手順、ソースコード、練習問題の解答例などを、必要に応じて公開しています。

本書専用サポートページ

<https://www.scc-kk.co.jp/scc-books/b-416.html>

なお、サポートページの内容は随時更新されますので、ご注意下さい。

本書に掲載した会社名、製品名などは、一般に各社の登録商標または商標です。
本書の使用（本書のとおり操作を行う場合を含む）により、万一、直接的・間接的に損害等が発生しても、出版社および編著者は一切の責任を負いかねますので、あらかじめご了承下さい。

はじめに

「Java」というプログラミング言語が登場してから、20年以上の年月が経ちました。巷にあふれるプログラミング言語の中では、もはや「枯れた言語」という印象もありますが、現在でもプログラマー人気のトップ3に入る最先端の言語でもあります。

Java言語の特徴は、「オブジェクト指向」と呼ばれる言語仕様を「本格的」にサポートしている点です。現在のプログラミング言語では当たり前となっていますが、Javaが登場した時代は「C++」など一部の言語がサポートするのみでした。しかも、標準ライブラリがマルチスレッドやネットワークに対応するなど、発表時にはかなりの衝撃を受けたことを覚えています。

ただ、Javaのプログラムは「Java仮想マシン」上で動作するということもあり、高速動作を要求されるプログラムには不向きでした。そんな状況を大きく変えたのが、コンピュータの高速化と「Webアプリケーション」の存在です。Javaは、そもそもマルチスレッドやネットワークのプログラミングが標準で可能でしたが、そこへ「Servlet」や「JSP」といったWebアプリケーションを簡単に作るための各種ライブラリやツールが用意され、一気に「Javaブーム」が起きました。

つまり、Java言語は「インターネットと共に成長した言語」だといってよいでしょう。そのような現状の中、JavaプログラマーはWeb業界で引く手あまたとなり、Java言語を学習してみたいと思う人が増え、現在のIT研修では多く企業がプログラミング言語にJavaを採用しています。

そうはいつても、Java言語の習得はいうほど簡単ではありません。その最大の要因は「なぜ、オブジェクトなのか?」、「なぜ、ポリモフィズムで考えるのか?」といった概念の理解です。特に最近は「関数型」と呼ばれる「関数をオブジェクトとして、関数単位でプログラミングする」という新しい概念も加わり、ますます文法が複雑になりつつあります。

そこで、本書は「これからプログラミングを学習する人」のために、各章を次のような構成にしています。

1. この章を学ぶと何ができるようになるのかを知り、そのための概念と文法を学習します。
2. 章のまとめで、この章の内容を復習します。
3. 学習した文法を利用する演習課題を、サンプルプログラムを添えて紹介します。
4. 最後にテストを行い、本当に理解しているのか確認します。

このように、概念や文法を解説した後、サンプルプログラムの作成に挑戦し、さらにテストで確認することで、Java言語を本当に理解できるように工夫しました。ぜひ、本書でJavaプログラミングの楽しさと、奥の深さを体験していただければ幸いです

目次

はじめに

第1章 Javaの概要	1
01 Javaとは	3
02 Java言語でプログラムを作る方法	4
03 Javaのエディション	6
04 統合開発環境	7
05 開発環境をインストールしよう	8
06 動作確認	12
07 ソースコードの基本	23
08 第1章のまとめと練習問題	28
1. まとめ	28
2. 練習問題	29
3. 1章テスト	31
第2章 基本文法	33
01 変数	35
1. 変数とは	35
2. 変数の正体	36
3. 変数の名前付け規約	40
4. 変数の初期化	41
02 データ型	43
1. 基本データ型の種類	43
2. boolean型	44
03 リテラル	46
1. 整数リテラル	46
2. 実数リテラル	48
3. 文字リテラル	49
4. 文字列リテラル	50
5. Java 7以降で使用できるリテラル	52

04 算術演算子	53
1. 算術演算子	53
2. 式の値	54
3. オーバーフロー	55
4. 0による割り算	57
5. 計算による丸め	59
6. 文字列連結演算子	60
05 ビット演算子	62
1. ビット演算子	62
06 代入演算子	65
1. 代入演算子	65
2. 複合代入演算子	66
3. インクリメント演算子、デクリメント演算子	67
07 キャスト	69
1. 自動型変換	69
2. キャスト演算子	71
08 定数	72
1. 定数とは	72
2. 定数の宣言	73
09 配列	75
1. 配列とは	75
2. 配列変数の宣言	76
3. 配列の生成	77
4. 配列の値へアクセスする	78
5. 配列の初期化	80
6. 配列要素数を取得する	83
10 多次元配列	84
1. 多次元配列とは	84
2. 多次元配列の宣言と生成	85
3. 多次元配列の初期化	88
4. 要素数の異なる多次元配列	89
11 演算子の種類と優先順位	92
12 2章のまとめと練習問題	93
1. 2章のまとめ	93

2. 練習問題	94
3. 2章テスト	96

第3章 制御構文..... 99

01 分岐処理	101
1. プログラムの流れ	101
2. if文	102
3. else文	104
4. else if文	106
5. switch文	109
02 関係演算子	111
1. 関係演算子の種類	111
2. 関係演算子の演習	112
03 論理演算子	114
1. 論理演算子の種類	114
2. 論理演算子の演習	115
3. ショートサーキット演算子	117
04 繰り返し処理	119
1. 繰り返し処理とは	119
2. while文	120
3. do文	122
4. for文	123
5. break文	125
6. continue文	126
7. 拡張for文	127
8. 多重ループ	129
05 メソッド	131
1. メソッドとは	131
2. メソッドの宣言	133
3. return文	136
4. 戻り値のあるreturn文	137
5. メソッドの引数	138
6. 変数のスコープ	140

06 3章のまとめと練習問題	144
1. 3章のまとめ	144
2. 練習問題	145
3. 3章テスト	147
第4章 オブジェクト指向	151
01 クラス	153
1. オブジェクトとは	153
2. クラスとは	154
3. クラスとオブジェクト	155
4. Javaでクラスを定義する	156
02 インスタンス	158
1. インスタンスとは	158
2. new演算子	159
3. クラスのインスタンス化	161
4. なぜオブジェクト指向で考えるのか	167
5. メソッドのオーバーロード	168
03 コンストラクタ	170
1. コンストラクタとは	170
2. コンストラクタに引数を渡す	172
3. なぜコンストラクタが必要か	173
4. デフォルトコンストラクタ	176
5. this	177
6. 初期化ブロック	180
04 インスタンスメンバとクラスメンバ	182
1. インスタンスメンバ	182
2. クラスメンバ	186
3. インスタンスに共有されるオブジェクト	188
4. static初期化ブロック	191
05 パッケージ	192
1. パッケージとは	192
2. パッケージ宣言	193
3. import文	196
4. static インポート	198

06 カプセル化	200
1. カプセル化とは	200
2. アクセス修飾子	201
07 アクセサメソッド	205
1. フィールドへのアクセス問題	205
2. アクセサメソッド	208
3. アクセサメソッドを作るときの注意点	211
08 4章のまとめと練習問題	212
1. 4章のまとめ	212
2. 練習問題	214
3. 4章テスト	217
第5章 高度なオブジェクト指向	221
01 継承	223
1. 継承とは	223
2. 継承の宣言	225
3. アップキャスト	229
4. 暗黙に継承されるObjectクラス	231
5. ダウンキャスト	233
02 スーパークラスのコンストラクタと参照値	235
1. コンストラクタは継承されない	235
2. 引数付きコンストラクタの呼び出し	239
3. super	242
4. クラスのfinal修飾子	244
5. サブクラスのオーバーロード	245
03 オーバーライド	248
1. オーバーライドとは	248
04 ポリモフィズム	250
1. ポリモフィズムとは	250
2. なぜポリモフィズムが必要か	251
05 抽象クラス	252
1. 抽象クラスとは	252
2. 抽象クラスの定義	253
3. 抽象メソッド	255

4. 抽象メソッドによるポリモフィズムの保証	257
06 インターフェース	262
1. 多重継承	262
2. インターフェースとは	264
3. インターフェースの定義	265
4. USBの機能にインターフェースを使う理由	268
5. インターフェースを継承する	272
07 匿名クラス	273
08 アノテーション	275
09 ラムダ式	278
10 メソッド参照のサポート	280
11 仮想拡張メソッドのサポート	281
12 5章のまとめと練習問題	283
1. 5章のまとめ	283
2. 練習問題	285
3. 5章テスト	290
第6章 さまざまな言語仕様	295
01 モジュール	297
02 Java SE コアAPI	298
1. Java APIの主なパッケージ	299
2. EclipseでJava APIリファレンスを参照する	300
03 例外処理	302
1. 実行時エラー	302
2. 例外とは	304
3. 例外処理	307
4. finallyブロック	311
5. 例外を自分でスローする	313
6. Exception で例外をキャッチする	315
7. チェック例外と非チェック例外	317
8. 例外の委譲	318
9. 複数の例外キャッチ	322
04 基本データ型とラッパークラス型の自動変換	323
1. ラッパークラス	323

2. オートボクシングとアンボクシング	324
05 Stringクラス	325
1. Stringクラスの主なメンバ	325
2. Stringクラスのメンバを利用する際の注意点	326
3. 文字列リテラルの正体	328
4. 文字列リテラルの注意点	329
06 ジェネリクス	331
1. ArrayListクラス	331
2. ジェネリクス	333
3. インスタンス生成時の型推論	335
07 Stream API	337
08 入出力	340
1. 文字の出力	340
2. コンソールからの文字入力	341
3. try-with-resources文でclose文を省略する	344
4. java.util.Scannerクラスでコンソール入力	345
5. ファイルから行単位でテキストを読み込む	346
6. ファイルへ行単位でテキストを書き込む	349
09 日時API	352
10 列挙型	354
1. 連続した定数	354
2. 列挙型の宣言と定義	357
3. 列挙型のメソッド	359
11 6章のまとめと練習問題	360
1. 6章のまとめ	360
2. 練習問題	362
3. 6章テスト	366
 章テストの解答	 369
 索引	 383

第 1 章

Javaの概要

この章では、Javaの概要と統合開発環境「Eclipse」のインストール方法を紹介し、Eclipseを利用してプログラムを作成するための基本操作を学びます。

本章で学ぶこと

▶ 達成目標：

1. Java の概要を説明できる
2. JDK、JVM、Java API とは何か説明できる
3. Java 言語でプログラミングする手順を説明できる
4. Java のソースファイル、クラスファイルの拡張子をいえる
5. Eclipse とは何か説明できる
6. オブジェクト指向におけるクラスの役割を説明できる
7. main メソッドの役割を説明できる
8. ソースコード内に、コメントが記述できる
9. コンソール画面に、文字列を表示するプログラムが書ける

01

Java とは

Java は、サン・マイクロシステムズ社が開発した、オブジェクト指向プログラミング言語です。サン・マイクロシステムズ社は、2010 年にオラクル社に買収されたため社名は消滅しましたが、Java の開発はオラクル社が引き継いでいます。

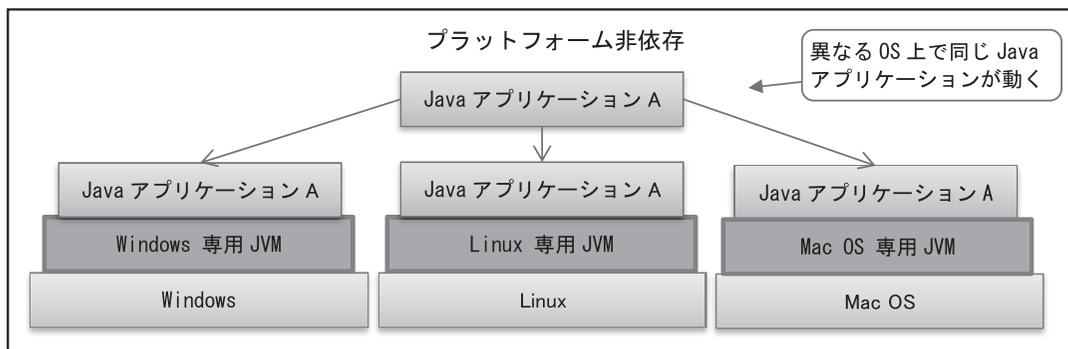
通常「Java」といった場合、プログラミング言語である「Java 言語」を指しますが、Java 言語で開発したアプリケーションの実行環境（プラットフォーム）や Java 技術の総称を「Java」と呼ぶ場合もあります。

Java 言語の特徴は、直接コンピューターのメモリーへアクセスできないようにした安全性と、「オブジェクト指向」と呼ばれる言語仕様、「例外処理」の採用、ネットワーク機能の標準装備などです。

また、Java 言語で記述されたプログラムは、「Java バイトコード」と呼ばれる中間言語に変換されます。この Java バイトコードは、「Java Runtime Environment（以降 JRE）」と呼ばれるソフトウェアのツールセットを利用します。JRE には、さまざまな Java の機能呼び出す「Java API (Application Programming Interface)」や、OS などのプラットフォーム上で動作する「Java 仮想マシン (Java Virtual Machine、以降 JVM)」が用意されており、JVM は Java アプリケーションのバイトコードを読み込んで、プラットフォームが実行する「ネイティブコード」へ変換していきます。

このような仕組みにより、Java のアプリケーションは、異なるプラットフォーム（例えば、OS が Windows や Linux）でも、JVM が動作しさえすれば同じ Java アプリケーションが動作します。

※異なる OS 上で同じ Java アプリケーションが動作することを「プラットフォーム非依存」と呼びます。



02

Java 言語でプログラムを作る方法

Java でプログラムを作るには、Java SE Development Kit (以降 JDK) と呼ばれるツールセットを用意します。JDK の中には、JRE など Java プログラムの実行と作成に欠かせないさまざまなツールが含まれており、以下のオラクル社のサイトからダウンロードすることができます。

<https://www.oracle.com/technetwork/jp/java/javase/downloads/index.html>

このダウンロードサイトには、Windows、Linux、MacOS など、各種パソコン用 OS の JDK が 1 つの圧縮ファイルとしてダウンロードできます。

OpenJDK

現在「JDK」には、「Oracle JDK」と「Open JDK」の 2 種類があります。どちらも基本的な内容は同じですが、「Oracle JDK」にはオラクルの有償のサポートが含まれており、バグフィックスなども「Open JDK」よりも先に行われます。

対して「Open JDK」は、ソースコードが公開された「オープンソース」としてライセンスされており、無料で Java プログラムを開発することができます。

Open JDK のサイト

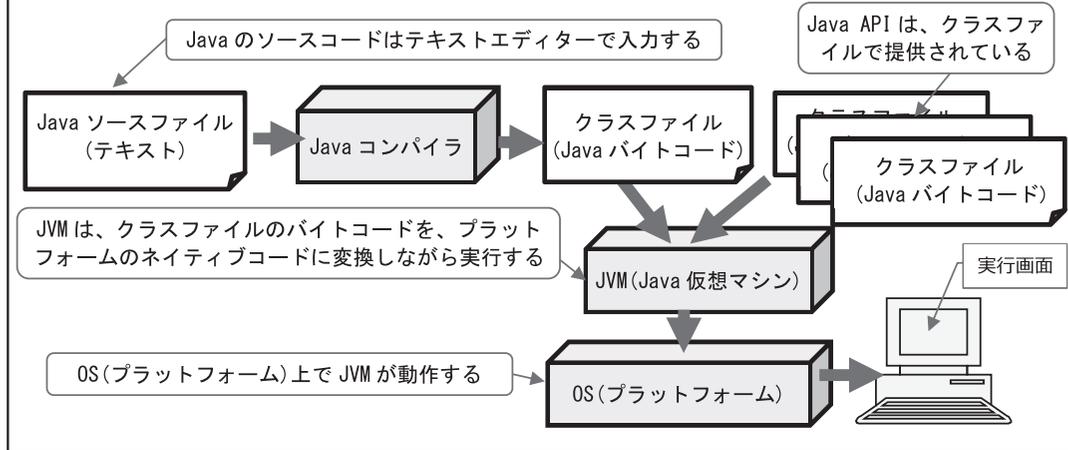
<https://openjdk.java.net/>

JDK には、ソースファイルをバイトコードに変換する「Java コンパイラ」と、実行環境である「JVM」が含まれます。Java コンパイラとは、Java 言語で記述されたプログラムコード (これを「ソースコード」と呼びます) を、JVM 上で実行できるコードに変換するソフトウェアのことです。ソースコードは、入力するためにテキストエディター (テキストを編集するアプリケーション) を用います。このソースコードを記述し保存したテキストファイルを「ソースファイル」と呼びます。また、ソースファイルの拡張子は、「.java」とします。

次に、このソースファイルを Java コンパイラで、「Java バイトコード」に変換します。Java バイトコードとは、JVM 上で動作するプログラムコードのことです。このように、JVM 上で動作する Java バイトコードでできたファイルを「クラスファイル」と呼び、ファイル拡張子は「.class」になります。

さらに、Java のプログラムは「Java アプリケーションインターフェース (以降 Java API)」と呼ばれるプログラム部品を利用します。このプログラムの部品も JDK 内にクラスファイルとして用意されています。

Java のソースコード入力から、プログラムが動作するまで



03

Javaのエディション

Java には、複数のエディション（種別）があり、使用されるコンピューターの種類や用途によって使い分けます。

Java Platform, Standard Edition (Java SE)	パソコンなどで動作する Java アプリケーションを作成するための基本セット。本書で扱う環境も、この Java SE です。
Java Platform, Enterprise Edition (Java EE)	メール、分散処理、Web などのライブラリやツールを追加した企業用セット。「Java EE」はブランド名で、中身は Java SE を含む複数の Java 技術の複合体です。
Java Platform, Micro Edition (Java Me)	リソースに制限のあるデバイス向けに、少ないリソースでも動作するように設計されたサブセット。サブセットとは「J2SE から機能を削除した」という意味です。

Windows で拡張子を表示する

Windows は、デフォルトでファイル拡張子が非表示になっています。「拡張子」とは、ファイル名の最後に「. (ドット)」と共に追加するファイルの種類を表す文字列のことです。

この「拡張子」を表示するには、エクスプローラーの「表示」リボンにある [ファイル名拡張子] のチェックボックスにチェックを入れます。

Windows 10 の場合

The screenshot shows the Windows 10 File Explorer interface. The 'View' ribbon is active, and the 'File name extensions' checkbox is checked. A callout box labeled '① 「ファイル名拡張子」にチェックを入れる' points to this checkbox. Below, the file list shows 'Sample.txt' with its extension visible. A callout box labeled '② 拡張子が表示されるようになる' points to the file name.

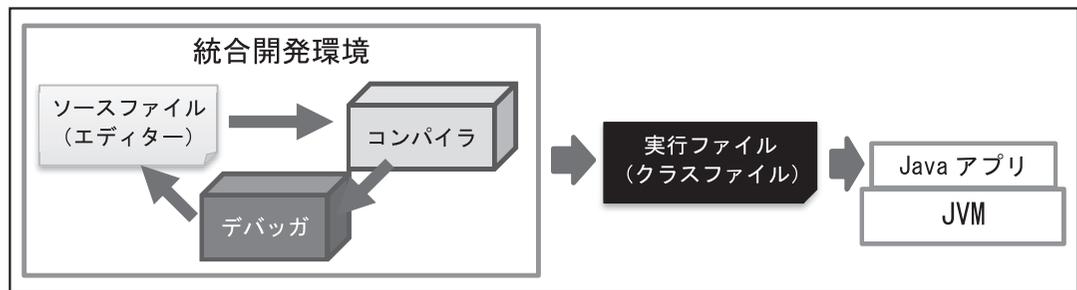
04

統合開発環境

Java プログラムは、JDK とテキストエディターがあれば作ることはできますが、本書では「統合開発環境」を利用して、Java プログラムを学習することになります。

統合開発環境とは、アプリケーション開発に必要な、ヘルプ、エディター、コンパイラ、バグの修正をするときに役立つツール「デバッガ」など、さまざまなツールがあたかも 1 つのアプリケーションであるかのように統合された環境のことで、「Integrated Development Environment (以降 IDE)」と呼ばれます。

これらのツールが同じ環境から利用できることによって、ソースコードの入力、実行、デバッグといった一連の作業を、ストレスなく繰り返すことができます。



Java の開発で利用される IDE には、有料のものからフリーのものまで多くの種類がありますが、今回は、オープンソースの代表的な統合開発環境である、「Eclipse (エクリプス)」を使用します。

Eclipse は、実際の開発現場でも利用されている高機能な IDE で、最初は IBM が Java の開発用に作ったものですが、後にオープンソースへ移管されました。Eclipse 自体も Java で記述されており、インストールするには JDK が必要です。ただし、Eclipse で開発できるプログラミング言語は、Java だけでなく「C」「C++」「PHP」など、多岐にわたります。

このように、Eclipse がさまざまなプログラミング言語に対応するのは、「プラグイン」と呼ばれる部品を組み合わせることで、機能を拡張できるためです。

05

開発環境をインストールしよう

それでは、さっそく Eclipse をインストール・・・といきたいところですが、Eclipse をインストールするには、Java が動作する環境が必要です。また、英語表示であるため日本語にするためのプラグインも必要です。そこで、Windows または MacOS に限定されますが、「Pleiades All in One」という Eclipse のパッケージを利用します。

Pleiades All in One は、プログラミング言語別にパッケージングした Eclipse のことです。本体と日本語プラグインである「Pleiades」がセットになっていて、ZIP ファイルをダウンロードして解凍するだけで、日本語化された Eclipse を使うことができます。

もし、Linux や他の OS で Eclipse を利用するときは、それぞれ専用の JDK と Eclipse をダウンロードしてください。日本語化には、Pleiades プラグインをダウンロードして利用できます。

それでは、「Pleiades All in One」をインストールするため、以下のサイトにブラウザでアクセスしましょう。

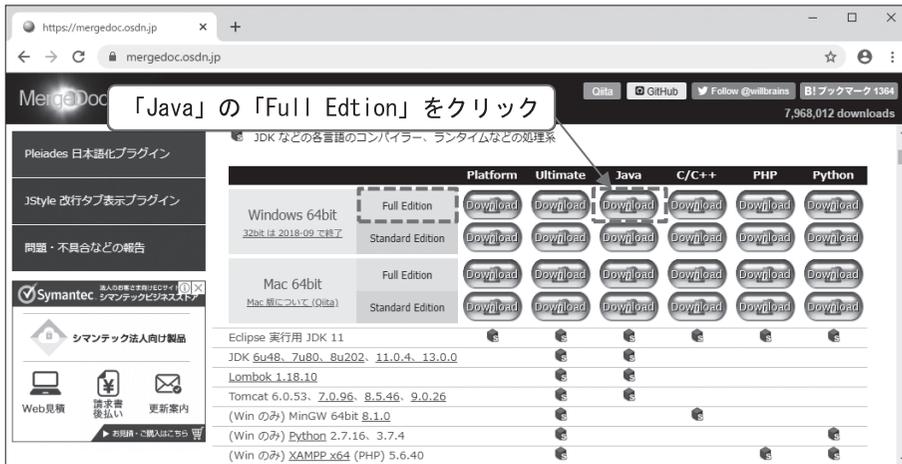
<https://mergedoc.osdn.jp/>

最新の Java に対応しているのは、「最新版」と書かれている Eclipse です。リンクになっているのでクリックしましょう。

Pleiades All in One のダウンロードページ



各言語に対応したパッケージの一覧画面になります。



本書は、64bitのWindowsを使って学習します。Windows 64bit Java Full Editionをクリックしましょう。ダウンロード画面になるので、インストーラー（※今回は pleiades-2019-09-java-win-64bit-jre_20191007.zip）をダウンロードします。

※ファイル名は、常に更新されます。

ダウンロードが完了したら、エクスプローラーで確認します。

