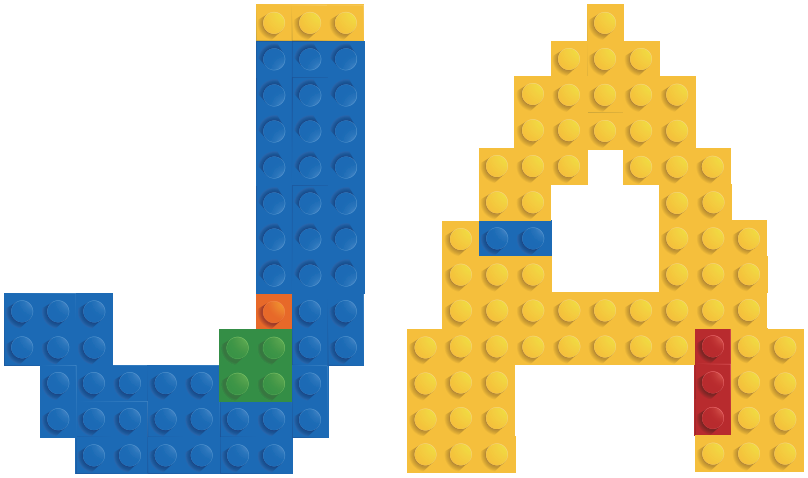
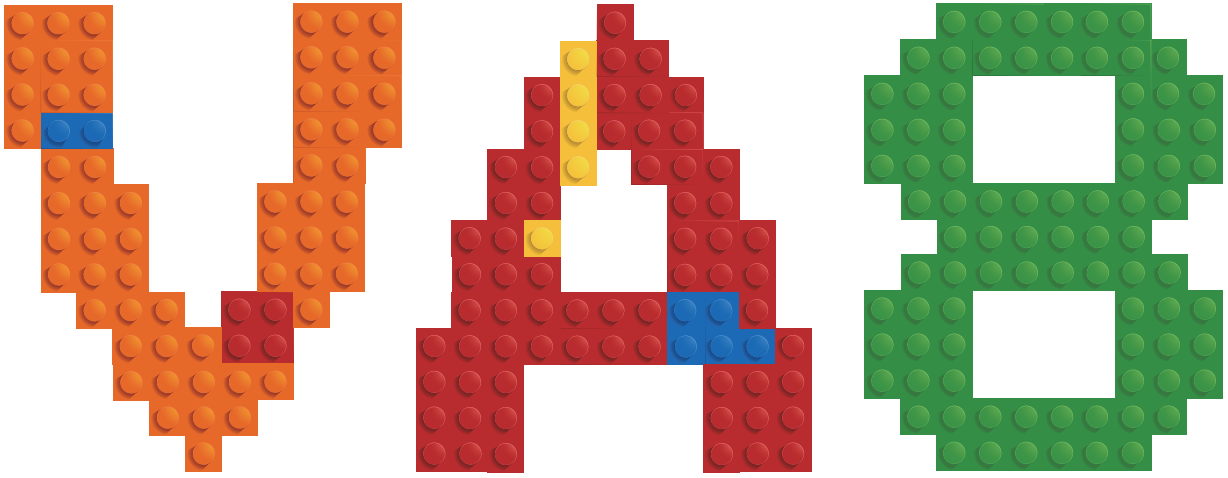
わかりすぎる

Java8の教科書



サンプルゲームを
動かしながら
Javaプログラムの
基礎が学べる！



中島省吾 著

SCC

・本書に記載されたURL等は執筆時点でのものであり、予告なく変更される場合があります。
・本書の使用（本書のとおりに操作を行う場合を含む）により、万一損害が発生しても、出版社、著者（著作権者）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。

・Oracle と Java は、米国 Oracle Corporation およびその子会社、関連会社の米国およびその他の国における登録商標です。
・Microsoft、Windows は米国 Microsoft Corporation の米国およびその他の国における登録商標です。
・その他、本書に記載されている会社名、製品名などは、一般に各社の商号、登録商標または商標です。
・本書では™および®の記載は省略しました。

目次

はじめに

著者が「Java」というプログラミング言語を知ったのは、もう 20 年近くも前の話です。当時のプログラミング言語といえば「FORTRAN」や「COBOL」が有名で、パソコン用には「Basic」や「C 言語」があるといった状態でした。

そんな 1996 年 1 月、「Java」と呼ばれるプログラミング言語が正式にリリースされました。Java は今後主流となるであろう「オブジェクト指向」と呼ばれる言語仕様をサポートしているにもかかわらず、同じオブジェクト指向言語である「C++」に比べてとてもシンプルな仕様であり、標準でネットワークにも対応するなど、その発表内容はかなり衝撃的でした。

その後、当然のように Java 言語の勢力は拡大を続け、現在では多くの SE 向け新人研修で、初めて学ぶプログラミング言語に Java が採用されるようにまでなりました。

Java がこれほどまでに人気なのは、「オープンな仕様で商用利用も可能なこと。歴史があるにもかかわらず、現在も進化し続ける現役の言語であること。サーバーからスマートフォンまで、広範囲のデバイスをサポートしている」などの点が理由にあげられるでしょう。

このような現状の中、自分も Java でプログラミングを学習してみたいと思う人が増えているようです。ただ、Java を独学で習得するのは、決して簡単なことではありません。

Java の習得を妨げる要因はいくつか考えられますが、最も大きな壁はやはり「オブジェクト指向」でしょう。特に「なぜ、オブジェクトなのか？」、「なぜ、ポリモーフィズムで考えるのか？」といった基本的な概念は、プログラミングの歴史的な背景も絡むことから、初学者には「難しさ」ばかりが先行してしまい、結局挫折してしまう人も少なくありません。

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. Java 言語の、基本文法を学習します。
2. 簡単なゲームプログラムを紹介し、ソースコードの解説を行います。
3. 最後に、復習と確認のため、簡単な練習問題を解きます。

このように、文法を解説した後、その文法が登場するゲームプログラムで利用方法を確認することで、「楽しく、飽きずに」Java の学習を進められ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

著 者

第1章

Java 8の概要

この章では、Java 8 の概要と開発環境のインストール方法を紹介し、開発環境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。

1.1 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。

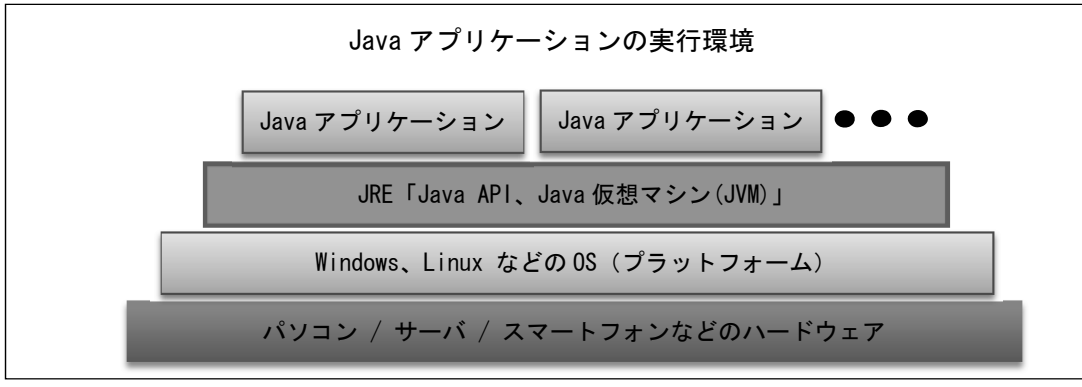
通常「Java」といった場合、プログラミング言語である「Java言語」を指します。本書も同様ですが、Java言語で開発したアプリケーションの実行環境（プラットフォーム）やJava技術の総称を、単に「Java」と呼ぶ場合もあります。

Javaの特徴は、直接コンピュータ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や「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の機能を呼び出す「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。

注意 ただし、Java アプリケーションが OS 独自の機能を利用している場合は、その OS 専用の Java アプリケーションになります。



1.2 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。

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Java SE のダウンロードページ

このダウンロードサイトには、Windows、Mac、Linux 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。

2014 年 8 月時点で最新の JDK は、バージョン 1.8 です。このバージョンは、通称「Java 8」と呼ばれ、それ以前の Java に対して、いくつか文法的な拡張が施されています。そのため本書では、Java 8 で拡張された重要な文法について、コラムや付録で解説を追加しています。

JDK に含まれるツールの中で、重要なのが「Java コンパイラ」と、実行環境である「JVM」です。

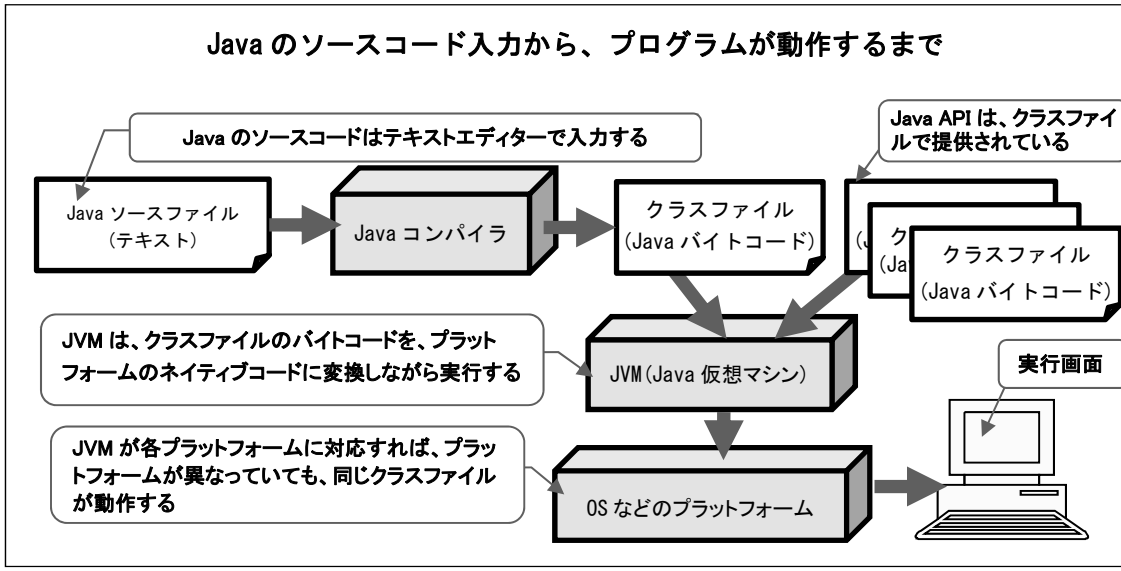
Java コンパイラとは、Java 言語で記述されたプログラムコード（これを「ソースコード」と呼びます）を、JVM 上で実行できるコードに変換するソフトウェアのことです。

ソースコードは、入力するためにテキストエディター（Windows の「メモ帳」のような、テキストを編集するアプリケーション）を用います。そして、テキストファイルとして保存します。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、Java のソースファイルの拡張子は、通常「.java」です。

次に、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。

また、Java のプログラムは、プログラマーがすべて入力するのではなく、既にあるプログラム部品を利用します。このように、さまざまな機能を持った部品も JDK 内にクラスファイルとして用意されています。そして、これらの部品の具体的な使い方は「Java アプリケーションインターフェース（以降 Java API）」と呼ばれる仕様で決まっています。

※Java API については、第6章で詳しく解説します。



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の種類や用途によって使い分けます。次に主なエディションを紹介します。

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。

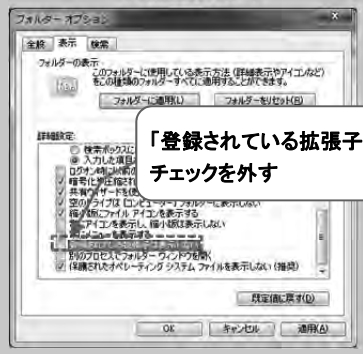
Windows で拡張子を表示する

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、Windows 7の場合、エクスプローラーの[整理]メニューにある「フォルダーと検索のオプション」をクリックし「フォルダーオプション」ダイアログを表示します。[表示]タブをクリックして、「登録されている拡張子は表示しない」のチェックをクリックして外し、[適用]ボタンをクリックしてダイアログを閉じます。

Windows 8 以降では、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 7



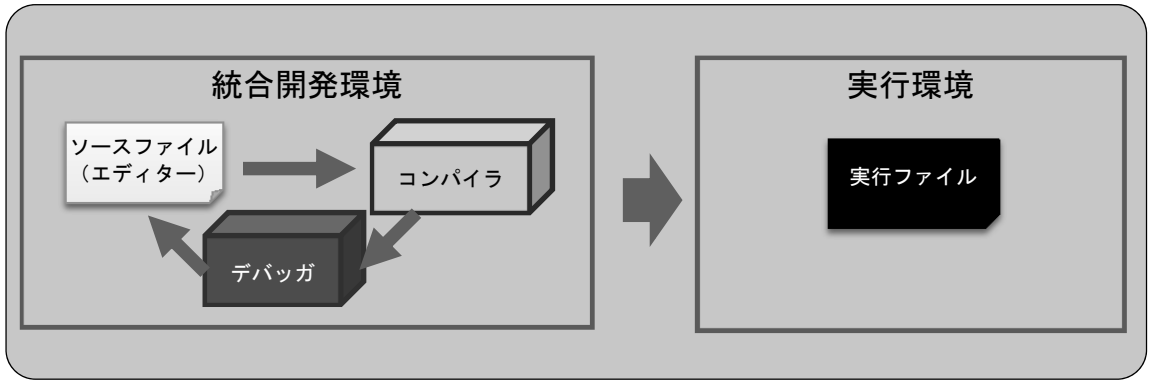
Windows 8 以降

1.3 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します。

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であるデバッガなど、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。

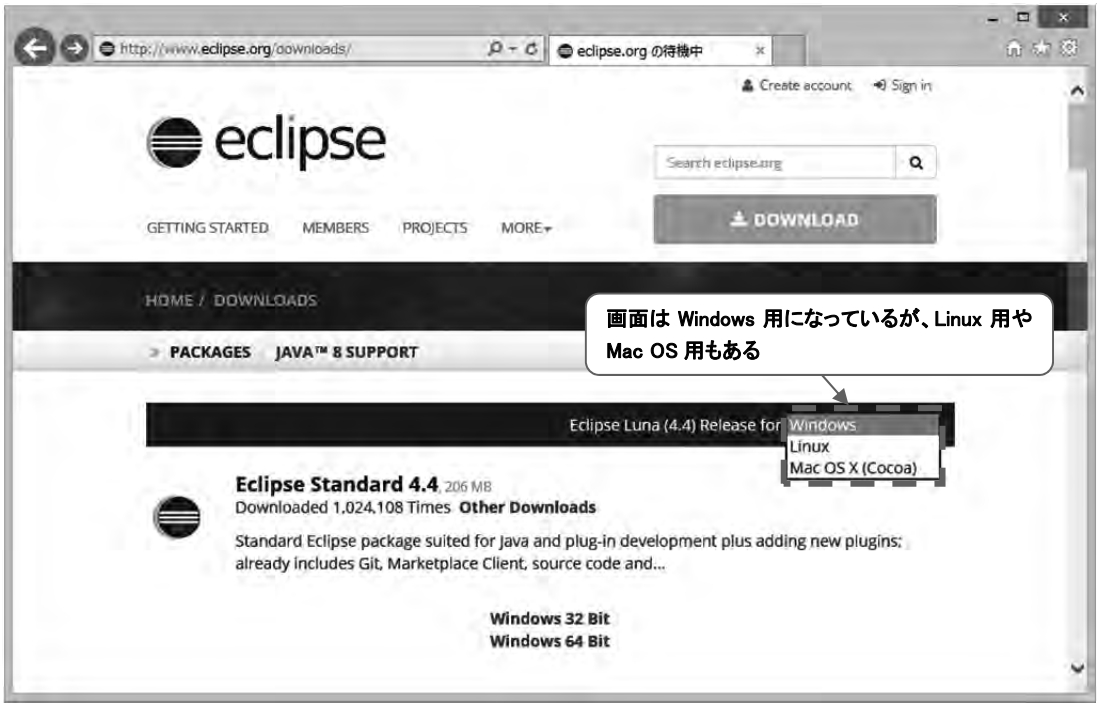
このように、Eclipse がさまざまなプログラミング言語に対応するのは、「プラグイン」と呼ばれる部品を組み合わせることで、機能を拡張できるためです。

プラグインの中には、メニューなどの表示を変更するプラグインもあり、さまざまな国の言語で画面表示をすることができます。もちろん、日本語表示用のプラグインもあるので、表示を日本語に変更することも可能です。

Eclipse は、以下のサイトからダウンロードすることができます。

http://www.eclipse.org/downloads/

Eclipse のダウンロードページ



1.4 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、本書では Windows 環境に限定されますが、「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 Mac OS などで 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、以下のサイトにブラウザでアクセスしましょう。

http://mergedoc.sourceforge.jp/

Pleiades All in One のダウンロードページ

最新の Java 8 に対応しているのは、「Eclipse 4.4 Luna Pleiades All in One」です。リンクになっているのでクリックしましょう。

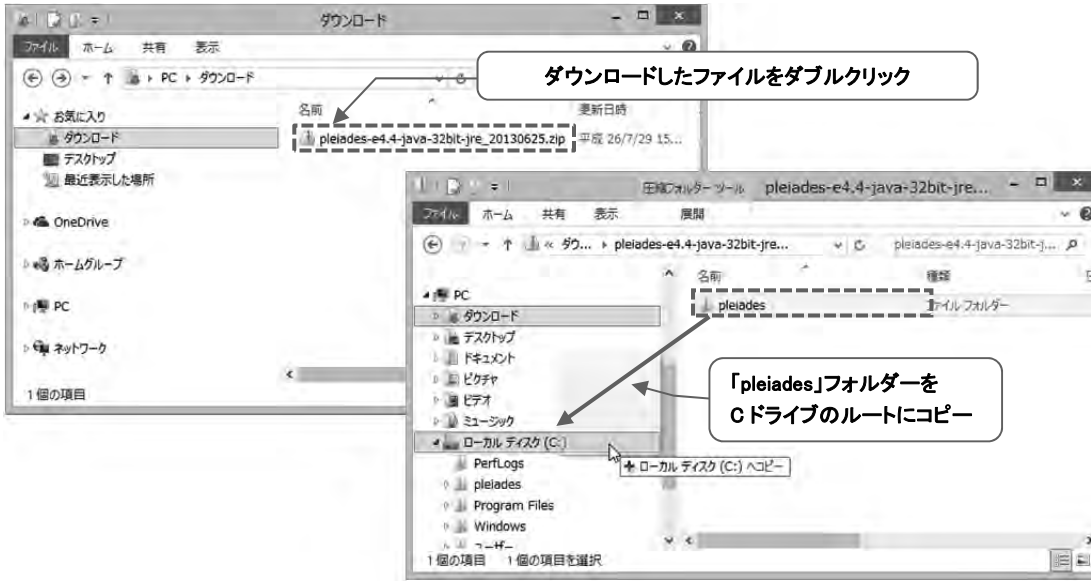
各言語に対応したパッケージの一覧画面になります。

本書の画面紹介は、Windows 8.1 Update 32bit 版で行っているため、ここでは 32bit 版の Full Edtion をクリックしています。

「Download」ボタンをクリックすると、ブラウザのダウンロード確認になるので、32bit の Full Edtion の場合「pleiades-e4.4-java-32bit-jre_20130625.zip」がダウンロードされます。

注意　ダウンロードするファイル名は、バージョンによって多少異なります。図のファイル名は、2014 年 8 月にダウンロードしたものです。

ダウンロードが完了したら、エクスプローラーでダブルクリックして開きます。



開くと、「pleiades」フォルダーがあるので、C ドライブのルートにドラッグアンドドロップしてコピーします。解凍しながらコピーするので、5 分～20 分ほど時間がかかる場合があります。完了までしばらく待ちましょう。

※コピーが終了したら、ダウンロードした「pleiades-e4.4-java-32bit-jre_20130625.zip」は削除してかまいません。

1.5 動作確認

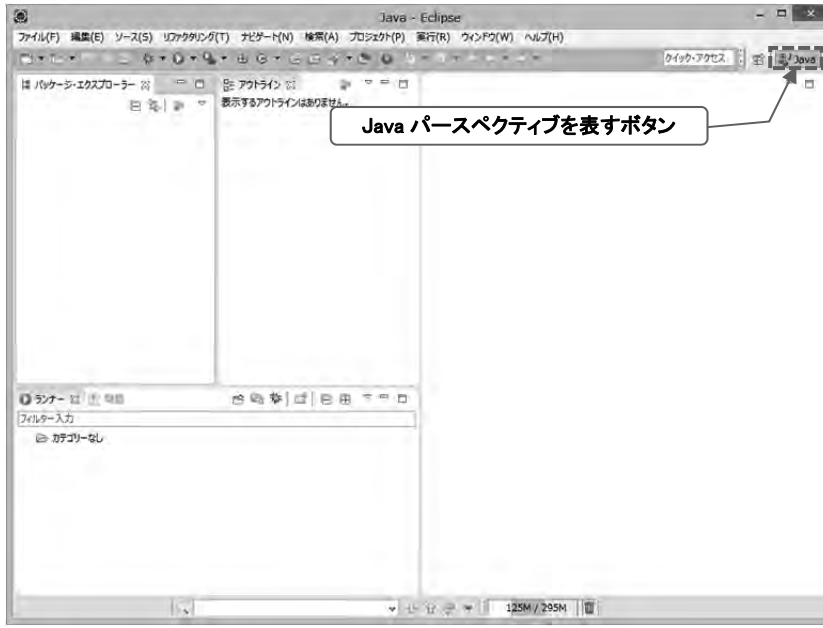
開発環境が正しくインストールされたか確認します。

エクスプローラーで、C:¥pleiades¥eclipse フォルダを開き、eclipse.exe をダブルクリックします。この eclipse.exe が Eclipse の実行ファイルです。

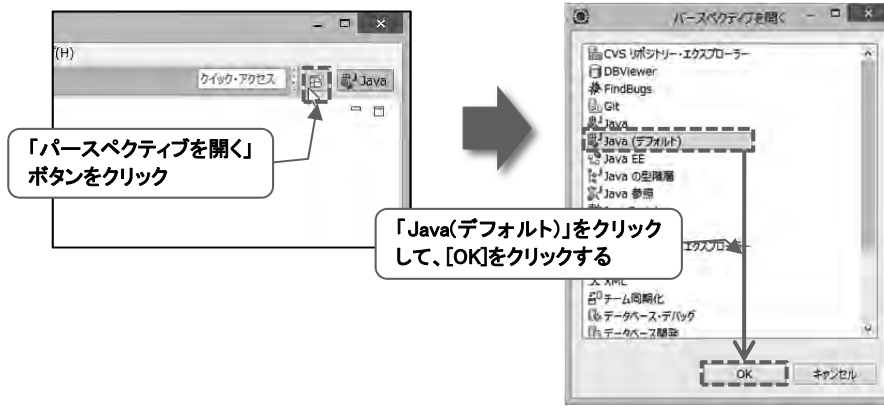
「ワークスペース・ランチャー」というウィンドウが開きます。「ワークスペース」とは、Eclipse が作業するフォルダーのことです。

本書では、デフォルトのフォルダー「C:¥pleiades¥workspace」で作業するので「この選択をデフォルトとして使用し、今後この質問を表示しない」チェックボックスをクリックしてチェックを入れ、[OK]ボタンをクリックします。

Eclipse が起動すると、「Java パースペクティブ」が開きます。

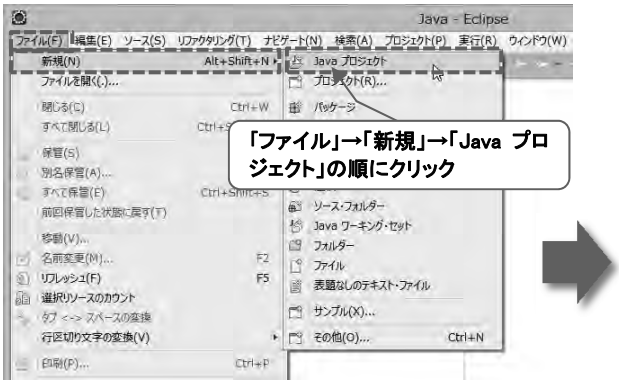


Eclipseは、言語や環境に合わせて、画面構成を変更できます。この画面構成を「パースペクティブ」と呼びます。デフォルトでは「Javaパースペクティブ」になりますが、他のパースペクティブに変わってしまったときは、右上端にある「パースペクティブを開く」ボタンをクリックします。「パースペクティブを開く」ダイアログが表示されるので、「Java(デフォルト)」を選択し、[OK]ボタンをクリックすると「Java パースペクティブ」に戻すことができます。



動作確認のため、「ようこそ、Java の世界へ」と表示するだけのプログラムを作りましょう。

Eclipse の「ファイル」メニューをクリックして表示される項目から「新規」→「Java プロジェクト」を選択します。

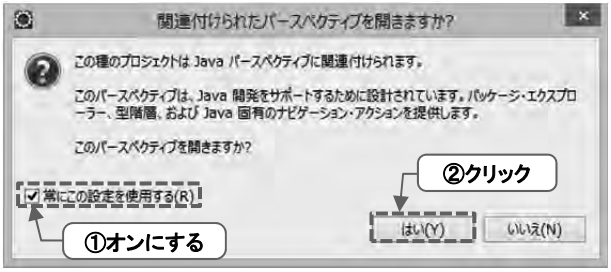


「Java プロジェクトの作成」ダイアログが開くので、「プロジェクト名」の欄に「Welcome」と入力し、[完了]ボタンをクリックします。

「プロジェクト」とは、作成したプログラムが利用するさまざまなファイルをまとめたものです。Eclipse で Java プログラムを作成するときは、この「プロジェクト」単位で作成します。

プロジェクト名は、識別しやすければどのような名前でもかまいません。今回は「Welcome」というプロジェクト名にしました。

「Java プロジェクトの作成」ダイアログで[完了]ボタンをクリックすると、最初のプロジェクトの場合、「この種のプロジェクトは Java パースペクティブに関連付けられます。」ダイアログが表示される場合があります。



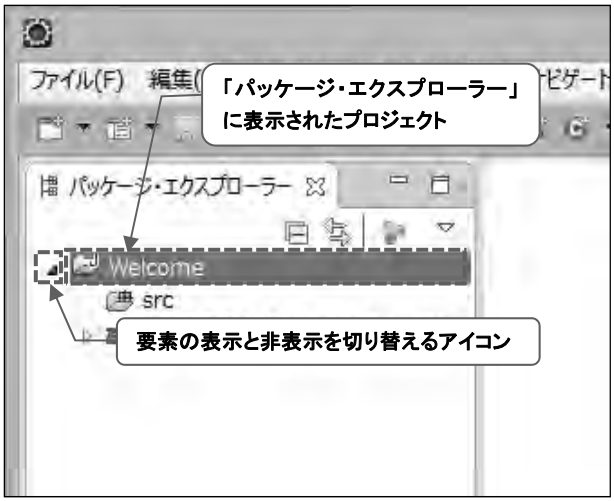
その場合は、「常にこの設定を使用する」チェックボックスにチェックを入れて[はい]をクリックします。

すると、新たに作成したプロジェクト専用の Java パースペクティブが開きます。

プロジェクトができると、Eclipse の左側にある「パッケージ・エクスプローラー」という領域にプロジェクト名が表示されます。

「パッケージ・エクスプローラー」のプロジェクト名の左側にある小さな「三角アイコン」をクリックすると、現在このプロジェクトに含まれる要素を表示します。

また、この「三角アイコン」は、クリックするたびに要素の表示と非表示が切り替わります。



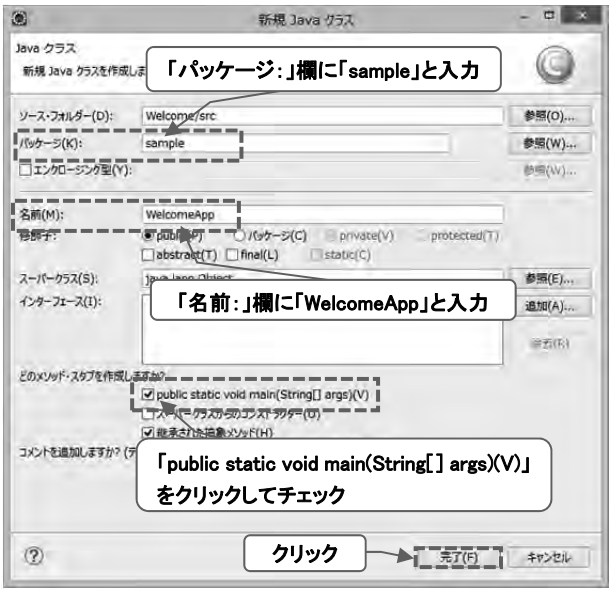
「パッケージ・エクスプローラー」のウィンドウは、表示と非表示を切り替えることができます。誤ってタイトル横の[×]ボタンをクリックして消してしまった場合や、最初から表示されていない場合は、Eclipse の「ウィンドウ」メニューをクリックして表示される項目から「ビューの表示」→「パッケージ・エクスプローラー」を選択すると再表示することができます。

プログラムの世界では、記述したプログラムのことを「ソース」あるいは「ソースコード」と呼びます。また、ソースコードが記述されたファイルを、「ソースファイル」と呼びます。

今回の「Welcome」プロジェクトに、ソースファイルを追加しましょう。

「パッケージ・エクスプローラー」内のプロジェクト名で右クリックして、表示される項目から「新規」→「クラス」を選択します。

「新規 Java クラス」ダイアログが表示されるので、「パッケージ:」欄に「sample」、「名前:」欄に「WelcomeApp」と入力します。「どのメソッド・スタブを作成しますか？」にある「public static void main(String[] args)(V)」のチェックボックスをクリックしてチェックを入れたら、「完了」ボタンをクリックします。



「新規 Java クラス」ダイアログで[完了]ボタンをクリックすると、Eclipse の画面中央にエディターが表示され、プログラムの「ひな型」が表示されます。

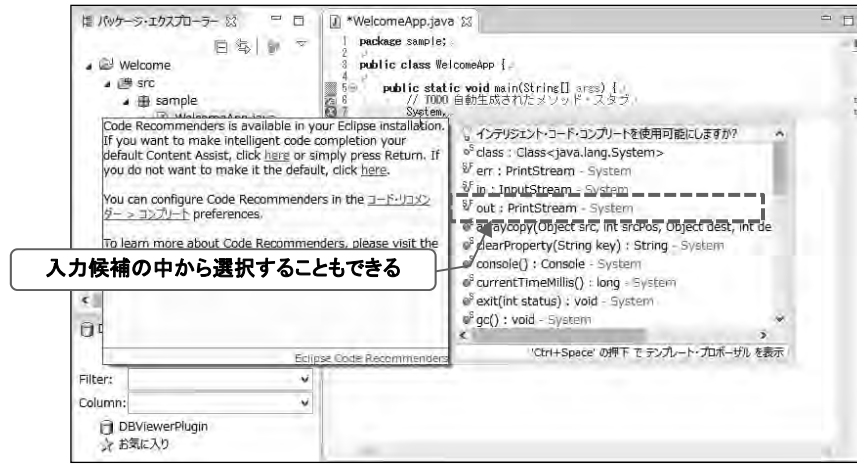
エディターに表示されているのは、Eclipse がプログラムの外枠だけ(ひな型)を自動的に生成したものです。ソースファイルのファイル名は「WelcomeApp.java」になり、プロジェクトに追加されます。

このソースファイルに、オリジナルの処理を追加します。エディター内にある「// TODO 自動生成されたメソッド・スタブ」の下に、以下のオリジナルプログラムを入力しましょう。

```
System.out.println("ようこそ、Java の世界へ");
```

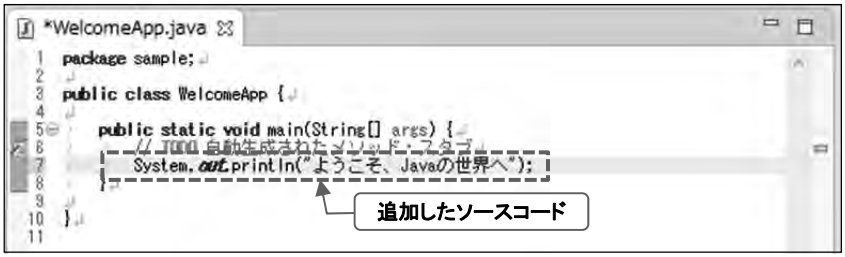
プログラムの入力方法は、「// TODO 自動生成されたメソッド・スタブ」の「ブ」の後ろをマウスでクリックして、カーソルを合わせ[Enter]キーを押します。すると、改行されて自動的に字下げ(インデント)を行った位置にカーソルが移動します。ここで System と半角で入力します。1 文字目の S だけを大文字にします。

System の後ろに「.(ドット)」を入力して、out 以降の入力を続けます。Eclipse のエディターは、入力可能な候補を表示する機能(コンテンツ・アシスト機能)があるので、表示された入力候補の中から選択することも可能です。ただし、似た別の候補と間違えることが多いため、ここは無視してキーボードからすべて入力してください。

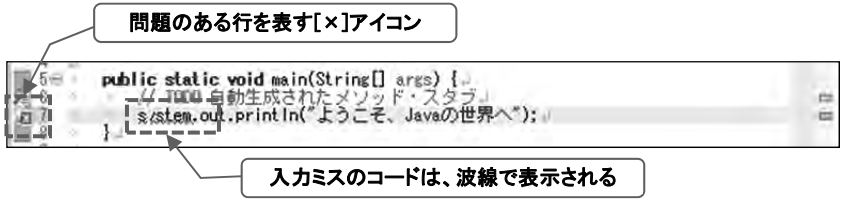


注意　もし、「.(ドット)」を入力した際、コンテンツ・アシストのエラーダイアログが表示される場合は、ダイアログの[デフォルトの設定]ボタンをクリックして、設定を戻してください。

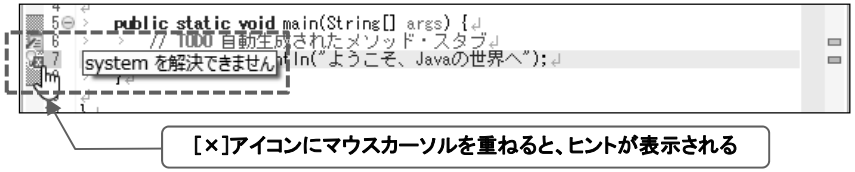
最終的には、次のように入力します。



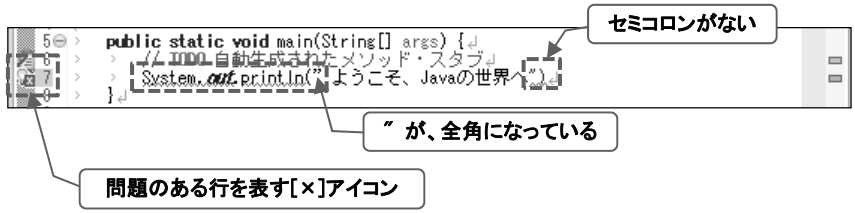
Eclipse のエディターには、文法的な間違いをリアルタイムで指摘する機能があります。例えば、最初の S を小文字で入力したとしても、入力し終わるとその行の先頭に[×]アイコンが現れます。さらにコード上に波線が表示され、間違い箇所を指摘してくれます。



この[×]アイコンや波線の部分にマウスカーソルを重ねると、エラーのヒントが表示されます。最初のうちは、Java のプログラムに慣れていないので、ヒントを表示しても意味が理解できないと思いますが、できるだけ確認するようにしておきましょう。



入力の際、「″(ダブルクォーテーション)」や最後の「;(セミコロン)」は半角で入力します。全角で入力したり、最後の「;」を忘れた場合も、エディターは何らかのエラーを表示します。

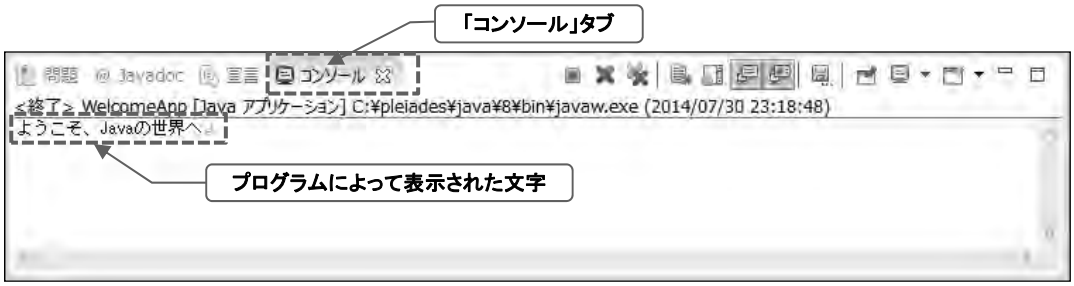


ソースコードの入力が完了したら、実行してみましょう。Eclipse 画面、上部中央にある「実行」ボタンをクリックします。

ソースファイルが保存されていない場合は、実行する前に「保存して起動」ダイアログが表示されるので、「常に起動前にリソースを保管する」をチェックして[OK]ボタンをクリックします。

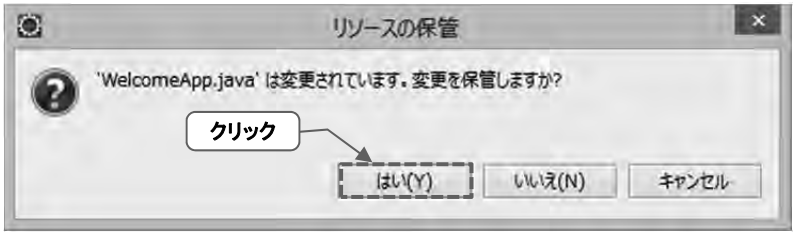
これでソースファイルは保存され、ソースコードのコンパイルと実行が行われます。

正しく実行されると、Eclipse の画面下部にある「コンソール」と書かれたタブ領域に、「ようこそ、Java の世界へ」と表示されます。



これで、動作の検証は終了です。

Eclipse を終了するときは、Eclipse のクローズボタンをクリックします。プロジェクトで利用したファイルが保存済みの場合、そのまま終了します。もし、保存していないファイルがある場合は、ファイルを保存するかどうかを確認するメッセージボックスが表示されます。[OK]もしくは[はい]ボタンをクリックして保存して終了してください。



1.6 ソースコードの基本

それでは、動作確認で使用したソースコードの概要を解説します。

```
package sample;

public class WelcomeApp {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        System.out.println("ようこそ、Java の世界へ");
    }

}
```

上記がソースコードの全体像ですが、まだコードの意味を完璧に理解する必要はありません。ただし、いくつか覚えておいて欲しい決まりや用語があるので解説します。

まず最初に覚えて欲しいことは、「プログラムは main メソッドから始まる」ということです。

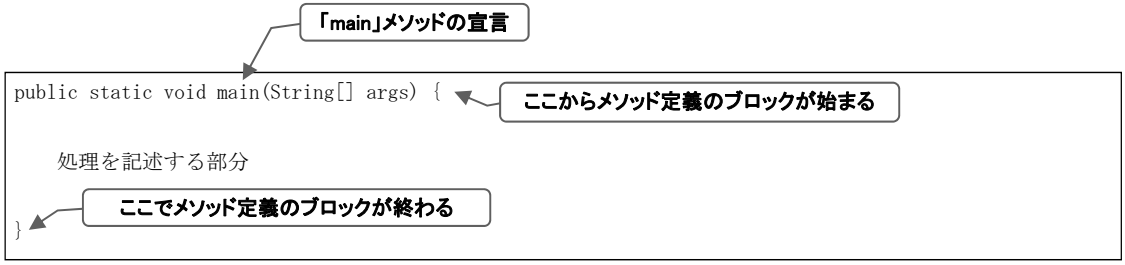
オブジェクト指向言語では、オブジェクトを「クラス」と呼ぶ「オブジェクトのひな型」で定義します。したがって、オリジナルのプログラムを作成するときは、必ず 1 つ以上の「クラス」を宣言・定義します。

例えば、上記のソースコードでは「WelcomeApp」という名前のクラスを宣言しています。また、クラスの中身は、{ から } までの間に定義し、この中括弧の括りのことを「ブロック」と呼びます。

そして、クラス定義の中には「フィールド」や「メソッド」と呼ばれる「クラスのメンバ」を宣言・定義します。フィールドは、オブジェクトのデータ、メソッドはオブジェクトの機能を表現します。

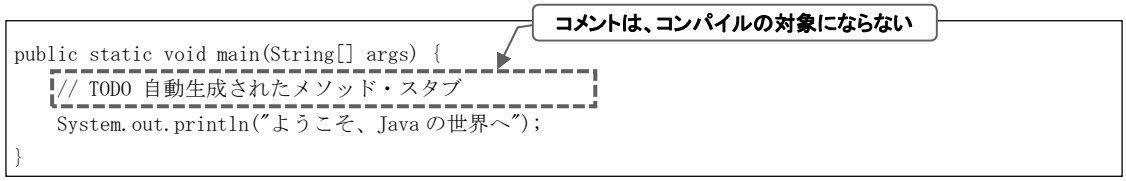
この中に、「main」という名のメソッドを宣言・定義します。Java のプログラムは、この「main メソッド」がプログラムの開始位置になります。

メソッドも、クラスと同じようにどのような処理を行うかをブロック内に定義します。



このとき、メソッドの宣言・定義がクラスのブロック内で行われていることが、とても重要です。このように、メソッドの宣言・定義はクラスの外ではできないことを覚えておきましょう。

次に、main メソッド内に「// TODO 自動生成されたメソッド・スタブ」という記述がありますが、この「//(ダブルスラッシュ)」で始まる行を「コメント」と呼びます。



コメントの部分は、Java バイトコードへの変換(この作業を「コンパイル」と呼びます)対象にならないため、どのような内容を記述してもかまいません。通常は、処理の説明文や注釈などを記述します。

また、// で始まるコメント以外に、/* から始まって */ で終了するコメントも書けます。両者の違いは、// で始まるコメントは行末までがコメント、/* で始まるコメントは複数行あってもかまわず */ までがコメントになります。

そして、実際に文字を表示しているのが、「System.out.println(″ようこそ、Java の世界へ″);」の部分です。このコードを正しく説明すると、「System(システム)」クラスにある「out(アウト)」フィールドが参照する「PrintStream(プリントストリーム)」オブジェクトの「println(プリントエルエヌ)」メソッドに「ようこそ、Java の世界へ」という文字列の参照を渡して呼び出す・・・となります。ややこしいですね。当然、今の段階では何のことかサッパリだと思います。

押さえておくべきポイントは、「このコードが「println」メソッドを実行しているということ。そして、この「println」メソッドの後ろにある「( )」の中に文字列を記述すると、コンソールにその文字列が表示される」ということです。

```
public static void main(String[] args) {
    // TODO 自動生成されたメソッド・スタブ
    System.out.println(″ようこそ、Java の世界へ″);
}
```

println メソッドで文字列をコンソールに表示

ここで、「文字列」という言葉が出てきましたが、プログラムでは「文字」と「文字列」は異なる意味で使われます。

通常「文字」とは、1 文字の「文字コード」のことで、「文字列」はその文字コードの並びです。Java 言語では、文字を表現するとき「'a'」のように、文字を「'(シングルクォーテーション)」で括ります。文字列は「″abc″」のように、「″(ダブルクォーテーション)」で括ります。

また Java 言語では、文字列は単なる文字コードの並びではなく「文字コードの並びを内部に持つオブジェクト」として扱われるので注意してください。

次に、注目して欲しいのは、「System.out.println(″ようこそ、Java の世界へ″);」の最後にある「;(セミコロン)」です。この ； は、「文の終端」を表します。改行は、文の終端ではないので注意しましょう。したがって、以下のように記述しても、正しくプログラムは動作します。

```
public static void main(String[] args) {System.out.println(″ようこそ、Java の世界へ″);}
```

最後に、1 行目にある「package sample;」の意味を簡単に説明します。

この文は、「パッケージ宣言」と呼ばれるもので、今回のコードの場合、「sample パッケージに WelcomeApp クラスを所属させる」という意味があります。

「パッケージ」とは、クラスの名前を補完するために用いるものです。例えば、同じプロジェクト内に「同じ名前のクラス」を複数作りたい場合、そのままでは「名前の衝突」が起きてしまいます。

そこで、同じ名前のクラスを、別の名前のパッケージに所属させることで、名前が同じでも異なるクラスとして判別できるようになります。

もし、パッケージを省略した場合、「デフォルトパッケージ」と呼ばれるパッケージにクラスは所属することになりますが、パッケージの省略は推奨されていません。したがって、Eclipse でクラスを作成する際には、何らかのパッケージ名を付けるようにします。また通常、すべて半角の小文字でパッケージ名を付けます。

全角文字と半角文字

日本語の文字をワープロやエディターなどで入力する場合、日本語変換ソフトを起動して入力する「全角文字」と、オフにして(直接入力状態)入力する「半角文字」があります。日本語のひらがなと漢字は全角文字ですが、アルファベットや数字は全角文字と半角文字の両方が存在します。

この全角と半角のアルファベットと数字は、見た目は非常によく似ていますが、コンピュータにとっては、まったく別の文字になります。

コンピュータは、文字を数値(番号)で識別しています。この文字の番号のことを「文字コード」と呼びます。人間にとっては全角の「A」も半角の「A」も同じ文字ですが、コンピュータにとっては、それぞれ異なる番号が与えられているため、違う文字として判断されます。

通常、Java のプログラムを記述する場合、全角文字は使いません。全角文字を使うときは、画面やコンソールに文字を表示する場合に限定されます。ただし、その場合でも必ず文字を「"(ダブルクォーテーション)」で囲む必要があります。

文字コードとは

文字コードとは、コンピュータの画面に文字を表示するため、各文字に割り当てられた「番号」や、「番号と文字の対応表」のことです。

コンピュータで、ディスプレイに文字を表示したり、プリンターで印字したりするには、文字を点(ドット)の集合(ビットマップ)で表現した「画像データ」が必要になります。そこで、文字に番号を割り当て、画像データとの対応表を作り、文字を番号で扱うようにしたものが「文字コード」です。

欧米ではアルファベットや記号を「ASCII」という文字コードで標準化していますが、日本では日本工業規格の JIS コード、UNIX 系で使われる EUC-JP、Windows などで使われる Shift_JIS、国際標準化機構が文字コードの統一を目指して標準化した Unicode など、多数の文字コードが存在しています。

もし、プログラムが文字コードを正しく判断できないと、表示は「文字化け」になってしまいます。

主な文字コード	
ASCII	英語圏、ヨーロッパなどで用いられる、最も使用されている文字コード。
ISO-2022-JP	通称「JIS コード」。JIS X 0211、JIS X 0201 のラテン文字、ISO 646 の国際基準版図形文字、JIS X 0208 など、複数の文字コードをまとめた規格。
EUC-JP	Linux などの UNIX 系の OS で利用される文字コード。
Shift_JIS	マルチバイト文字と 1 バイト文字の混在を可能とする文字コード。Windows-31J や CP932 などの派生文字コードがある。
Unicode	すべての言語で利用できるようにと考えられたマルチバイト構成の文字コード。Windows、Mac、Linux などがサポートする。Java も、内部は Unicode を利用。

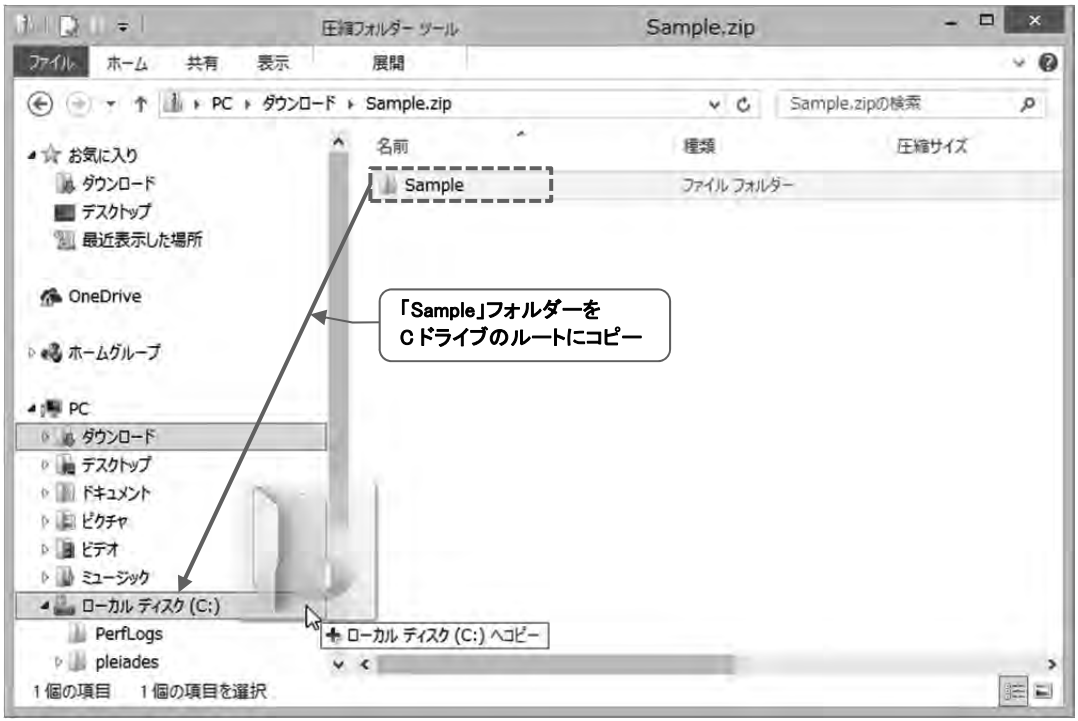
1.7 サンプルプロジェクトのインポート

本書で紹介するゲームプログラムのプロジェクトは、下記のサイトからダウンロードできます。

URL　http://www.scc-kk.co.jp/scc-books/support/B-376/support.html

ダウンロードするファイルは、Sample.zip です。Sample.zip のダウンロードが完了したら、エクスプローラーで表示してダブルクリックします。

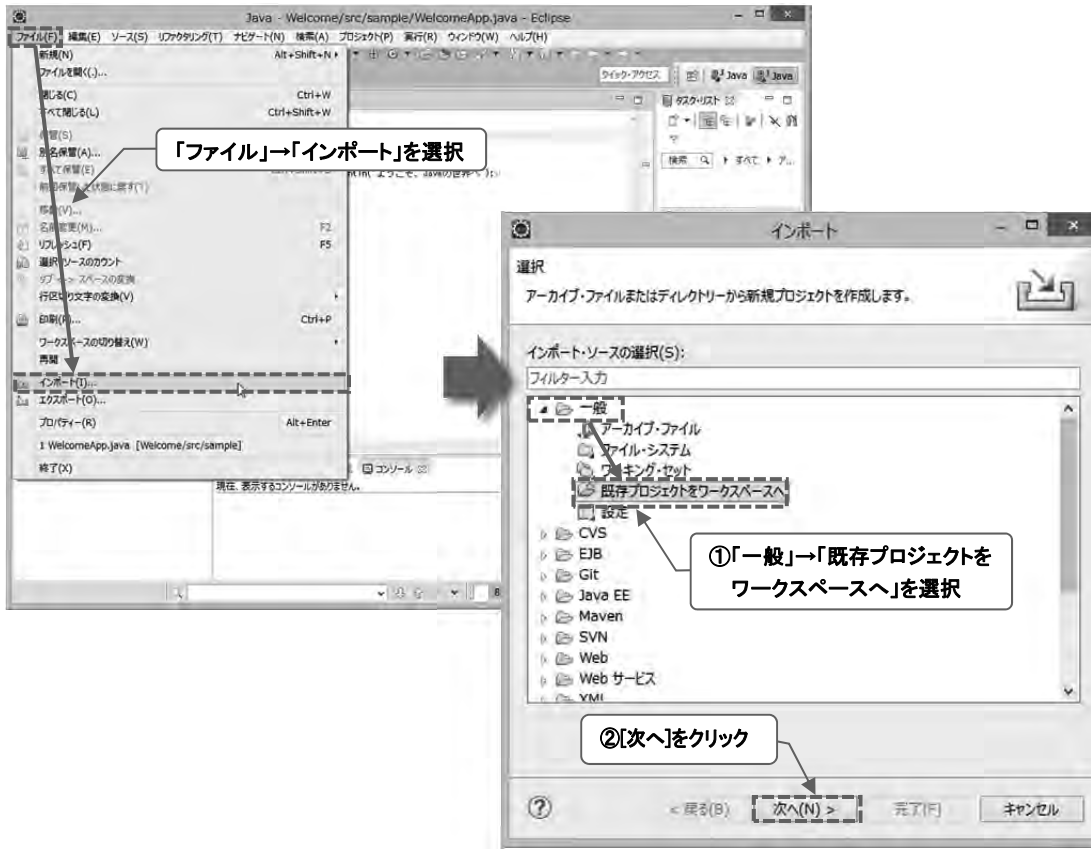
圧縮ファイルの中には「Sample」というフォルダーが入っているので、マウスで C ドライブのルートにドラッグアンドドロップしてコピーしておきます。



Sample フォルダーの中には、本書で紹介するゲームプログラムのプロジェクトが入っています。このプロジェクトを、Eclipse の「インポート」機能を使って取り込みます。

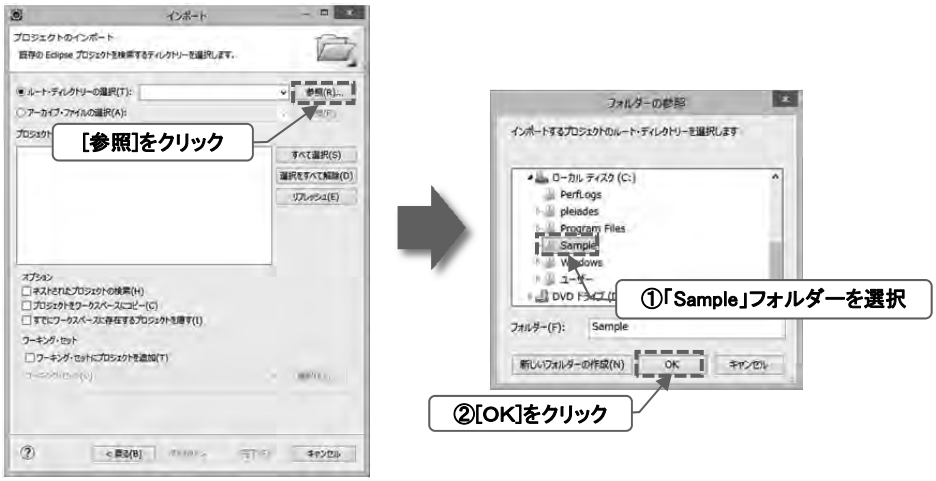
まず、Eclipse を起動します。Java パースペクティブの画面が開くので「ファイル」メニューをクリックします。さらに表示される項目から「インポート」をクリックします。

「インポート」ダイアログの「選択」画面が開くので、「インポート・ソースの選択」欄にある「一般」フォルダーの左側の「三角アイコン」をクリックします。展開した「既存プロジェクトをワークスペースへ」をクリックして選択します。最後に[次へ]ボタンをクリックします。



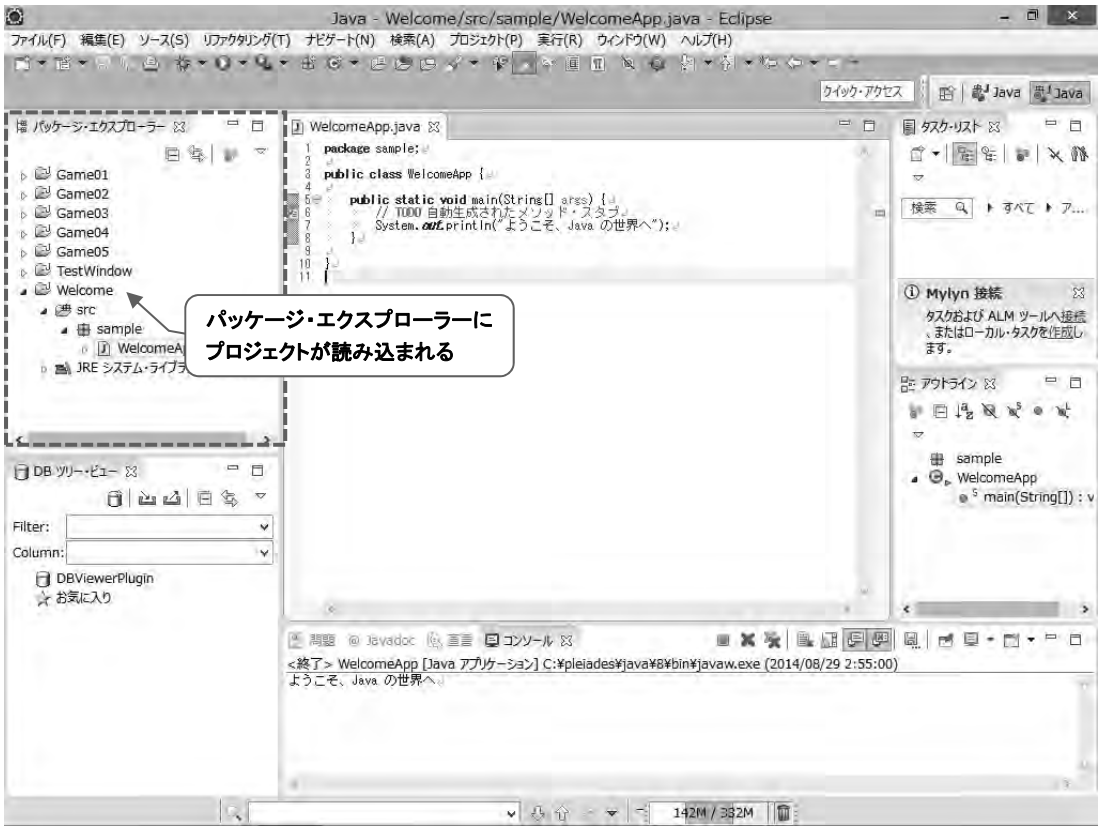
「インポート」ダイアログが「プロジェクトのインポート」画面に切り替わるので、画面の右上にある［参照］ボタンをクリックします。

「フォルダーの参照」ダイアログが表示されるので、C ドライブのルートにある「Sample」フォルダーをクリックして選択し、［OK］ボタンをクリックします。



「プロジェクトのインポート」ダイアログに「Sample」フォルダー内のプロジェクトが表示されます。すべてにチェックが入っていることを確認して、［完了］ボタンをクリックします。

Eclipse の「パッケージ・エクスプローラー」に、「Sample」フォルダーのプロジェクトが読み込まれます。

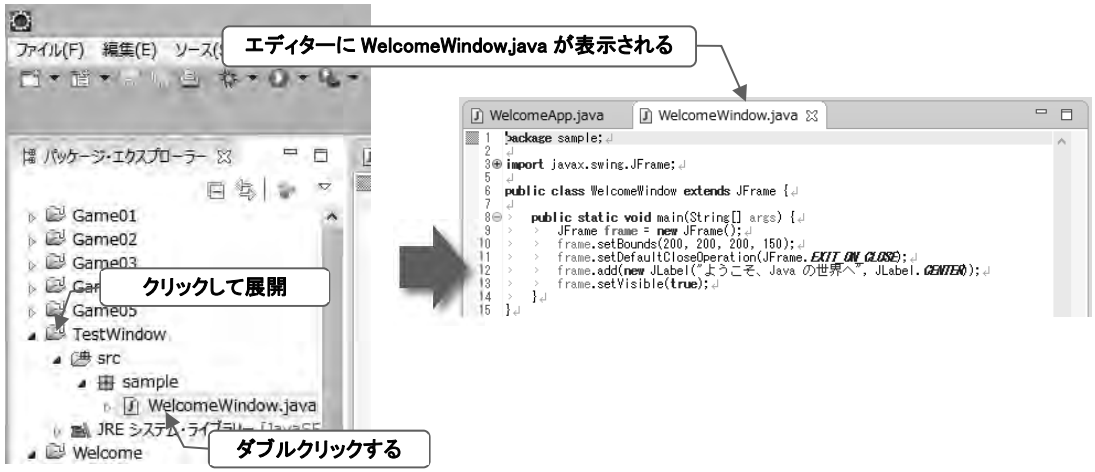


1.8 ウィンドウを表示してみよう

本来、各章の最後で学習した文法を利用するゲームプログラムを紹介しますが、第 1 章ではまだ Java 言語の文法を学んでいません。そこで、ゲームプログラムを作る土台となる、ウィンドウを表示するプログラムを紹介します。

パッケージ・エクスプローラーに表示されている「TestWindow」プロジェクトを展開します。すると中に「src」フォルダーがあり、その下に「sample」パッケージ、さらに「WelcomeWindow.java」というファイルを確認することができます。

表示された「WelcomeWindow.java」をダブルクリックすると、エディターにソースコードが表示されます。



ソースコードが確認できたら、[実行]ボタンをクリックして起動してみましょう。

ウィンドウが表示されて、「ようこそ、Java の世界へ」と表示されます。通常のウィンドウと同じように、サイズ変更や最大化、最小化が可能です。終了するときは、表示されたウィンドウの右上にある[×]ボタンをクリックしてクローズすると、プログラムは終了します。

通常は、このように[実行]ボタンをクリックすることで実行できますが、[実行]ボタンが実行しているのは、「パッケージ・エクスプローラー」で選択されているファイルです。

もし複数のソースコードが「パッケージ・エクスプローラー」に読み込まれている場合は、「パッケージ・エクスプローラー」から直接起動することもできます。

「パッケージ・エクスプローラー」の「WelcomeWindow.java」で右クリックしてみましょう。多くの項目が表示されますが、その中に「実行」という項目があり、クリックするとさらに「1 サーバーで実行(1) Alt+Shift+X,R」「2 Java アプリケーション(2) Alt+Shift+X,J」「実行の構成(N)...」という 3 種類の表示が現れます。この中の「Java アプリケーション」のほうをクリックしても、実行できます。

注意 「1 Java アプリケーション(1) Alt+Shift+X,J」「実行の構成(N)...」だけの場合もありますが、どちらの場合でも「Java アプリケーション」の方を選択します。

この方法なら、実行するソースファイルを選択すると同時に、右クリックで実行ができるので、[実行]ボタンまでマウスを移動させる必要がありません。

注意 メニューから実行した際、ポートの解放を要求するダイアログが出る場合があります。その場合は[キャンセル]ボタンでキャンセルしてください。

それでは「WelcomeWindow.java」のコードを見てみましょう。コードは、次のようになっています。

```
package sample;

import javax.swing.JFrame;
import javax.swing.JLabel;

public class WelcomeWindow extends JFrame {

    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.setBounds(200, 200, 200, 150);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.add(new JLabel("ようこそ、Java の世界へ", JLabel.CENTER));
        frame.setVisible(true);
    }
}
```

※2 行目にある import 文が Eclipse では 1 行しか表示されない場合、import 文の先頭にある(＋)マークをクリックしてください。展開されたすべての import 文が表示されます。

このソースコードがウィンドウと、その中に「ようこそ、Java の世界へ」という文字を表示させているプログラムです。

もし、ソースコード全体が表示されない場合は、エディターのサイズを変更することができます。

また、ディスプレイ解像度の低いパソコンの場合、「DB ツリービュー」「タスク・リスト」「アウトライン」などのウィンドウタイトルの横にある[×]印をクリックして、ウィンドウを非表示にすると見やすくなります。

非表示にしたウィンドウを、再び表示したい場合は、Eclipse の「ウィンドウ」メニューをクリックして、表示された「ビューの表示」から選択すれば再表示できます。

ここで、このソースコードを変更して、自分の名前が表示されるようにしてみましょう。エディターに表示された「WelcomeWindow.java」のソースコードから次の記述を探してください。

```
frame.add(new JLabel("ようこそ、Java の世界へ", JLabel.CENTER));
```

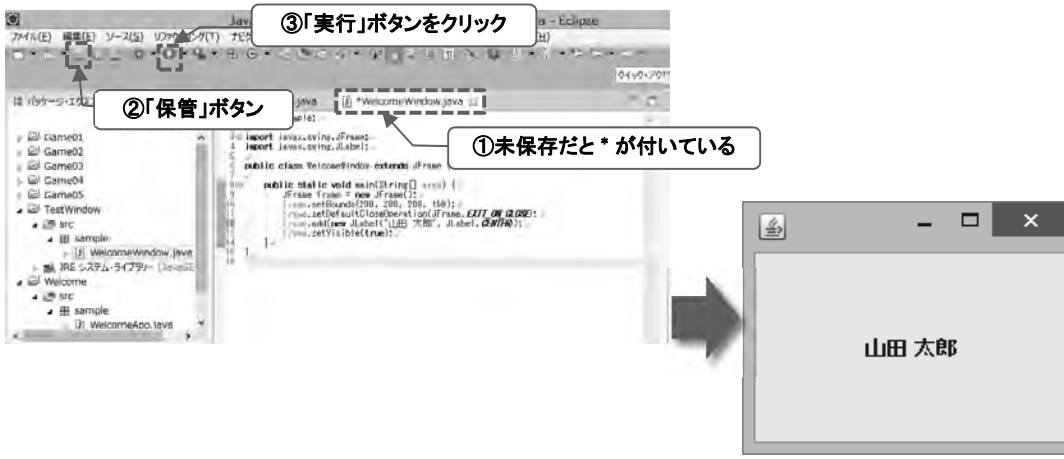
この中に記述されている、「ようこそ、Java の世界へ」の部分が、表示する文字を決めているところです。この部分を、あなたの名前で書き換えてみましょう。例えば、あなたが「山田太郎」さんなら、次のように変更します。

```
frame.add(new JLabel("山田 太郎", JLabel.CENTER));
```

変更が終了したら、ソースファイルを上書き保存します。未保存のソースファイルには、エディターのソースファイル名の先頭に、*(アスタリスク)マークが付くので確認できます。もし、保存だけをしたい場合は、[保管]ボタンをクリックします(＊が消えてソースコードが上書き保存されます)。

また、エディターは同時に複数のソースコードを編集できるので、複数の変更済みファイルを一気に保存する場合は、その横の[すべて保管]アイコンをクリックします。

保存ができたら、[実行]ボタンをクリックして実行します。



どうですか、正しく自分の名前が表示できたでしょうか。動作を確認できたら、表示されたウィンドウの[×]ボタンをクリックして、プログラムを終了してください。

1.9 第１章のまとめと練習問題

第 1 章で説明してきた内容の「まとめ」で復習し、正しく理解しているか簡単な三択のテストをしてみましょう。

1.9.1 第１章のまとめ

1. Java は、サン・マイクロシステムズ社(現在、オラクル社)が開発した、オブジェクト指向プログラミング言語です。
2. Java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、「JRE」を利用して動作します。
3. JRE には、「Java API」や「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、OS が読み込むネイティブコードに変換しながら実行していきます。
4. Java でプログラムを作るには、「JDK」と呼ばれるツールセットが必要です。
5. Javaコンパイラとは、Javaで記述されたソースファイル(拡張子は.java)を、JVM上で実行できるJavaバイトコードに変換するソフトウェアのことです。
6. Java バイトコードでできた「クラスファイル」の拡張子は、「.class」です。
7. 統合開発環境とは、アプリケーション開発に必要なさまざまなツールが 1 つのアプリケーションのように統合された環境のことで、「IDE」と呼ばれます。
8. Eclipse とは高機能な IDE のことで、プラグインによって機能を拡張することができます。
9. オブジェクト指向言語では、オブジェクトをひな型である「クラス」で定義します。
10. クラスやメソッドは、{ から } までの「ブロック」内に定義を行います。
11. Java のプログラムは、main メソッドから始まります。
12. ソースコード内に、注釈として「コメント」が記述できます。コメントは、// から行末までをコメントにするものと、/* から始まって */ までをコメントにするものがあります。
13. コンソールに文字列を表示するには、「System.out.println」メソッドを使用します。
14. Java 言語では、文字を「″」で括り、文字列を表現します。

15. Java 言語の文の終端は、改行ではなく「;」で表現します。
16. クラスは名前の衝突を避けるため、パッケージ宣言によってパッケージに所属させることができます。
17. パッケージ宣言は省略できますが、その場合は「デフォルトパッケージ」に所属することになります。ただし、パッケージ名の省略は推奨されていません。

1.9.2 練習問題

■ 問題 1

Java の特徴となる言語仕様は、次のうちどれですか。

1. プラットフォーム
2. Java API
3. オブジェクト指向

■ 問題 2

JVM 上で動作するコードは、次のうちどれですか。

1. Java バイトコード
2. ソースコード
3. ネイティブコード

■ 問題 3

JVM や Java API を含むものは、次のうちどれですか。

1. JRE
2. IDE
3. Java アプリケーション

■ 問題 4

JVM を使う利点を説明しているのはどれですか。

1. JVM を使うほうが、コンピュータが小型になるから
2. JVM のほうが、OS よりも処理速度が速いから
3. JVM が、OS の違いを吸収してくれるから

■ 問題 5

Eclipse のように、アプリケーション開発に必要なさまざまなツールを、同じアプリケーションから利用できるようにしたものをなんと呼びますか。

1. テキストエディター
2. 統合開発環境
3. 実行環境

■ 問題 6

Java のソースファイルの拡張子は、次のうちどれですか。

1. .java
2. .class
3. .zip

■ 問題 7

Java のプログラムは、次のどれから始まりますか。

1. sample パッケージ
2. main メソッド
3. System クラス

■ 問題 8

次のうち、ブロックの記述に使用する記号ではないものはどれですか。

1. {
2. }
3. //

■ 問題 9

次のうち、コメントの記述に使用する記号ではないものはどれですか。

1. *
2. /
3. ;

■ 問題 10

Java 言語のソースコードで、文の終端に使う記号はどれですか。

1. ;
2. {
3. }