☞…本書の構成

Java はオブジェクト指向のプログラミング言語です。CPU の種類や OS の種類を問わずさまざまな環境で動作するマルチプラットフォーム対応という性質から，Web アプリケーションやスマートフォン用のアプリの開発に使われています。

本書は，プログラミング言語として Java を初めて学ぶという学習者の方を対象としていますので，学習の入り口にあたる基本事項の説明から始めて，重要な考え方となるオブジェクト指向の基礎についても紹介しています。また，基本情報処理技術者試験の午後試験で出題される Java の問題を解くために必要な考え方や知識を理解していただくため，例題の解説と演習問題も含めた構成にしています。

全体を通じて 15 章の章立てになっていますが，大きく分けると，次のようになります。

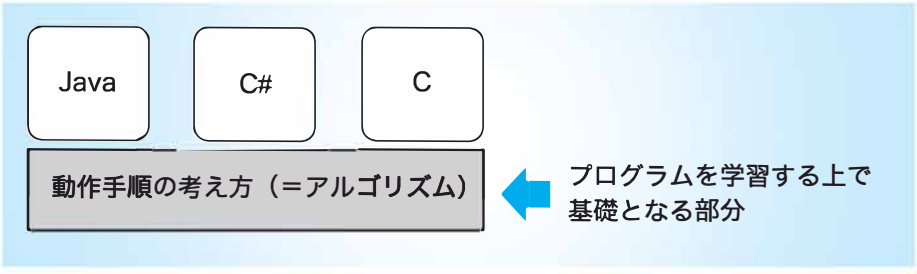
文法基礎	1 章〜6 章	命令文やコメントの書き方，条件判断，繰返しといった，基礎的な文法を紹介しています。
クラス，インタフェース	7 章〜10 章	オブジェクト指向を実現するために用意されたクラスやインタフェースに関して紹介しています。
試験によく出る技術要素	11 章〜14 章	基本情報技術者試験の問題を解く上で，必ず押さえておきたい，例外処理，スレッド，ジェネリクスについて紹介しています。
午後試験問題の演習	15 章	過去の情報処理技術者試験で出題された問題を使って，合格点を取るためのポイントを例題で解説し，演習問題を収録しています。

<プログラミング言語の初学者へ>

これまで，プログラミング言語に触れたことがない方は，本書を素直に先頭から読み進めてください。第 1 章〜第 6 章の内容は，Java だけでなく，C 言語や C++，C#，Perl，PHP でもほぼ同じ文法になるので，習得してしまえば，今後さまざまな場面で役立ちます。

プログラミング言語の文法は，プログラムの動作手順を構築するものであり，非常に重要です。

プログラムの動作手順の考え方は，Java に限らずほかのプログラム言語でも必要になります。プログラムを作成するときに基本となる部分です。



＜プログラムの学習は積み重ね＞

プログラムを学習していく途中で，なかなか理解できず頭を抱えてしまうこともあるかと思います。そんなときは投げ出さず，理解できない箇所を何度も読み返してください。必ず突破口が見つかるはずです。その突破口の先には「試験合格」が待っています。また，プログラムの学習は一夜漬けが効きません。日々の積み重ねが大事になります。

本試験１週間前になってあせって学習しても，なかなか習得することが難しいのです。計画を立てて学習してください。

作成したJavaプログラムは，実際にコンピュータで動作させて実行結果を確認してください。プログラムが間違っているときは，何度でも修正し，正しい結果が出るまで諦めないで学習してください。

また，プログラムの学習で間違えてしまった内容を大切にしてください。簡単に理解できるものは簡単に忘れてしまいます。理解するのが困難なものほど，なかなか忘れられず身に付きます。

プログラムの学習は，「暗記する！」のではなく「納得する！」ことが大事です。

＜C 言語や C++，C#，Perl，PHP を習得している方へ＞

Java は，C 言語や C++，C#，Perl などと基本文法が同じです。特に文の改行位置や，セミコロンを使った文末の指定，if 文や while 文などは，ほぼ同じ文法です。したがって，これらの言語の習得者であれば，第 1 章～第 6 章は確認程度に読み進めていただいてかまいません。

＜Python，シェルスクリプトを習得している方へ＞

また，Python，シェルスクリプトなど基本文法が異なる別のプログラミング言語の習得者は，第 1 章～第 6 章で紹介している Java の文法を，自身が習得しているプログラミング言語と比較しながら読み進めてください。いったん複数のプログラミング言語を覚えると，ほかのプログラミング言語の習得というのは，すでに覚えている言語の文法との照らし合せ作業になってくると分かるでしょう。ただし，非オブジェクト指向の言語しか知らない場合には，第 7 章をしっかり読んでください。

＜実践でも役立ててください＞

本書は基本情報技術者試験への対策書として，特に試験で問われる技術要素を重点的に紹介していますが，実際の開発業務においても，本書で取り上げている技術要素は頻繁に使われますので，試験対策書としてだけでなく，開発現場でもお役立てください。

特にスレッドや，ジェネリクスは慣れるまで，なかなか細かい文法を覚えきれない部分がありますので，本書を手元に置いていただくと安心できると思います。

＜章末問題と演習問題＞

各章の最後には，その章で学習した内容を確認するための章末問題が付いています。各章の学習が終わったら解いてみてください。正答率が低い章がある場合，その章が苦手であると考えられますので，解説文や，解説に掲載されたアドバイス文に沿って復習をしてください。

第 15 章には，実際に過去の基本情報技術者試験に出題された Java の問題を 6 問掲載しています。本書の内容が習得できていれば，必ず解ける問題です。6 問の演習問題を解いていくことで，自然と問題形式に慣れていきますので，自身の解答ペースをつかんでください。

目 次

本書の構成
目　次

第1章　Java プログラム入門

1.1　Java とは……10
1.2　プログラムとは……14
1.3　Java プログラムの形式……16
1.4　改行位置とインデント（字下げ）……20
1.5　文の終了（；セミコロン）……21
1.6　コメント（注釈）……22
1.7　リテラル（定数）……23
1.8　予約語（キーワード）……27
1.9　実行の流れ……28
1.10　画面への出力……29
1.11　画面出力のバリエーション……30
理解度チェック……32
章末問題……33

第2章　変数

2.1　変数とは……38
2.2　変数の宣言（識別子とデータ型）……41
2.3　文字列の扱い……44
2.4　データ型のサイズ……45
2.5　変数の宣言についての注意……46
2.6　変数の初期化……47
2.7　変数の利用……48
理解度チェック……52
章末問題……53

第3章　式と演算子

3.1　式と演算子……56
3.2　算術演算子……58
3.3　インクリメント演算子／デクリメント演算子……59
3.4　ビット演算子……62
3.5　複合代入演算子……66
3.6　キャスト演算子（型変換）……69
3.7　演算子の優先順位……75
理解度チェック……77
章末問題……78

第4章　条件判断

4.1　条件式と評価……82
4.2　if 文……83
4.3　if, else 文……88
4.4　if 文のネスト……90
4.5　if, else if, else 文……92
4.6　switch-case 文……94
4.7　論理演算子……99
4.8　条件演算子……104
理解度チェック……105
章末問題……106

第5章　繰返し（ループ）

5.1　繰返しの考え方……110
5.2　for 文……111
5.3　while 文……115
5.4　do～while 文……116
5.5　ループ文のネスト……118
5.6　break 文……120
5.7　continue 文……123
理解度チェック……125
章末問題……126

第6章　配　列

6.1　配列の宣言……130
6.2　配列の利用……132
6.3　配列の初期化……135
6.4　配列の参照……137
6.5　多次元配列……141

理解度チェック …… 148
章末問題 …… 149

第7章　クラス

7.1　Java とオブジェクト指向 …… 154
7.2　クラス …… 156
7.3　メンバへのアクセス …… 161
7.4　メソッド …… 168
7.5　オーバロード …… 177
7.6　コンストラクタ …… 179
7.7　修飾子 …… 185
7.8　クラス変数，クラスメソッド …… 189
7.9　final 変数 …… 197
7.10　可変長パラメタ …… 198
7.11　クラスの配列 …… 199
理解度チェック …… 202
章末問題 …… 203

第8章　クラスの継承

8.1　クラスの継承とは …… 208
8.2　オーバライド …… 219
8.3　final …… 225
8.4　Object クラス …… 227
8.5　抽象クラス・抽象メソッド …… 232
理解度チェック …… 235
章末問題 …… 236

第9章　クラスライブラリ

9.1　クラスライブラリとは …… 240
9.2　String クラス …… 241
9.3　StringBuffer クラス …… 244
9.4　パッケージ（package） …… 247
9.5　static インポート …… 250
9.6　クラスリファレンスの読み方 …… 251
理解度チェック …… 252
章末問題 …… 253

第10章　インタフェースと Enum

10.1　インタフェース …… 256
10.2　インタフェースの多重実装 …… 261
10.3　インタフェースの継承 …… 263
10.4　Enum（列挙型） …… 266
理解度チェック …… 270
章末問題 …… 271

第11章　例外処理

11.1　例外処理とは …… 274
11.2　try-catch 文 …… 276
11.3　独自の例外クラス …… 281
11.4　複数の例外クラスの catch …… 284
11.5　例外の連鎖 …… 286
理解度チェック …… 288
章末問題 …… 289

第12章　スレッド

12.1　スレッドとは …… 292
12.2　スレッドの作成 …… 293
12.3　インタフェースを利用した
　　スレッドの実装 …… 297
12.4　マルチスレッドと排他制御 …… 299
理解度チェック …… 304
章末問題 …… 305

第13章　ファイル入出力と
キーボード入出力

13.1　ストリーム …… 308
13.2　バイトストリームを使った入出力 … 309
13.3　文字ストリームを使った入出力 …… 317
13.4　キーボードからの入力 …… 322
13.5　コマンドライン引数 …… 324

1.1 Java とは

オブジェクト指向型言語
プラットフォーム

(1) Java の特徴

ここ数年，Java はさまざまな方面で急激に注目を集めています。Java の歴史はほかの言語と比べ非常に浅く 1995 年にコンピュータベンダである Sun Microsystems 社（現 Oracle 社）で生み出されました。それまでのプログラム言語の弱点であった，CPU や OS に対する互換性をもたせた点を最大の特徴として登場し，広く利用されるようになりました。

Java の文法的な特徴としては「**オブジェクト指向型言語**」である点が挙げられます。また，Java の実行形態に関する特徴としては，「マルチプラットフォーム」という点が挙げられます。次に，それぞれの点について説明します。

(2) オブジェクト指向型言語

Java の特徴の一つに「オブジェクト指向」があります。オブジェクト指向の考え方では，ある種類の「モノ」を「オブジェクト」としてとらえます。

例えば人を「オブジェクト」として考えてみます。「人」には，氏名という識別のためのデータや，生年月日，住所といった属性を表すデータがあります。その一方で，「人」は，「食べる」，「歩く」，「寝る」などの動作を行います。このようにオブジェクト単位で，そのオブジェクトがもつデータや，動作を定義していく設計手法をオブジェクト指向の設計といい，それに即したプログラム言語をオブジェクト指向型言語といいます。Java は代表的なオブジェクト指向型言語です。

キーワード
Unix
ベル研究所で開発された小型コンピュータ用のオペレーティングシステム

キーワード
Linux
フィンランドの大学院生であった Linus Torvalds 氏が開発した Unix 互換のオペレーティングシステム

キーワード
Windows
Microsoft 社で開発されたパソコン用のオペレーティングシステム

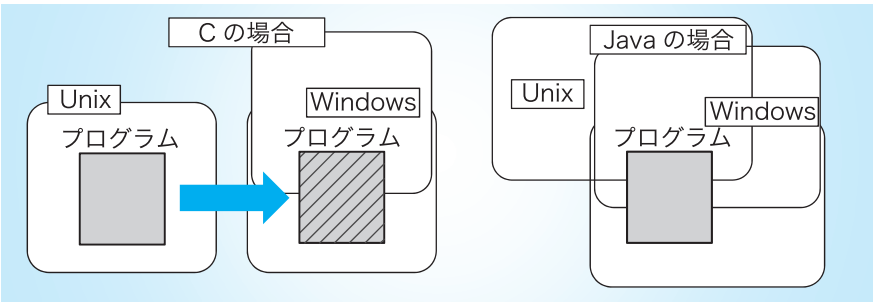
(3) マルチプラットフォーム

Java のもう一つの特徴として，プログラムを「**プラットフォーム**」に依存せずに実行できる点が挙げられます。

ここでいう「プラットフォーム」とは，Java プログラムを動作させるコンピュータのハードウェア・ソフトウェア環境をまとめた表現方法です。

例えば Unix や Linux で作成した C プログラムを，Windows 上で動作させる場合は，プログラムを Windows 用に作り直す必要があります。

一方，Java で作成したプログラムは，「プラットフォーム」に依存しないので，Unix や Linux で作成したプログラムをそのまま Windows 上で使用することが可能です。また，この逆も可能なので，例えば，開発は手元にある WindowsPC 上で行い，完成後は Unix や Linux のサーバ上で稼働させるということが可能です。



Java 仮想マシン
JVM
JRE
JDK
コンパイル
クラスファイル

(4) Java プログラム動作環境

Java のプログラムの実行には Java 仮想マシン（JVM：Java Virtual Machine）と呼ばれるプログラムが必要です。これは Java のバイトコード（Java プログラムをコンパイルすることによって生成されるコード）を読み込んで解釈・実行するインタプリタです。

また，JVM およびプログラムの実行に必要なクラスライブラリなどがまとめられたソフトウェア群のことを JRE（Java Runtime Environment）と呼びます。

Java プログラムを作成するためには，JDK（Java Development Kit）と呼ばれるソフトウェアが必要です。

JDK にはコンパイラ，インタプリタ，ライブラリなど，Java プログラムの作成に必要なソフトウェアが含まれています。

現在 JDK は，ライセンス形態の違いもあり，主に次の 2 種類が主流になっています。

① 米 Oracle 社が提供する JDK

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

（2020 年 6 月 10 日現在）

こちらは，米 Oracle 社が個人用途や開発用途など，限られた用途で利用可能なライセンス形態をとる JDK です。商用あるいは組織（企業や団体）で利用する場合には，米 Oracle 社との間で有償サポート契約が必要になります。

② Open JDK

https://openjdk.java.net/

最新版のバージョン 13 は https://jdk.java.net/13/ からダウンロード可能です。

（2020 年 6 月 10 日現在）

こちらは，同じく米 Oracle 社がフリーソフトウェアライセンス形式の 1 形態である，GPL（GNU Public License）として提供している JDK です。

＊Windows，Linux などプラットフォームごとにダウンロードすべきファイルが異なりますのでご注意ください。

JDK をコンピュータにインストールすると JRE も合わせてインストールされます。また，JRE だけインストールすることも可能です。

(5) IDE の活用

なお，(4)で見てきた Java プログラム動作環境の構築方法は，基本的なものになります。開発の現場では IDE（総合開発環境）を使うことが多いため，その代表例である Eclipse というオープンソースソフトウェアの IDE の環境構築手順を巻末に補足資料としてまとめました。

(6) Java プログラムの作成から実行まで

Java プログラムを作成するときは Windows アプリケーションソフトのメモ帳などのテキストエディタを使用します。作成したプログラムには拡張子として `".java"` を付けて保存します。これが，ソースプログラムになります。

作成したプログラムを実行するためには「コンパイル」作業が必要です。

コンパイルではプログラムの文法エラーのチェックが行われ，エラーがないとき拡張子として`".class"`が付いたクラスファイルが生成されます。

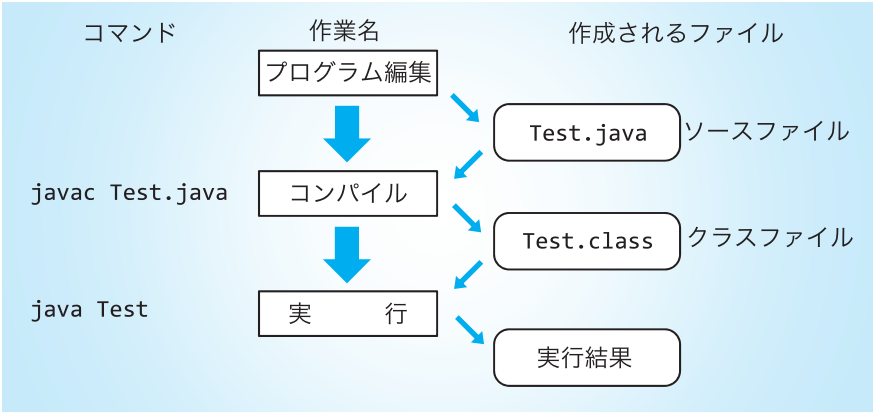
プログラムを実行するときは，JVM が，このクラスファイルを読み込んで実行します。

Java アプリケーション
Java アプレット
サーブレット
JSP

アドバイス

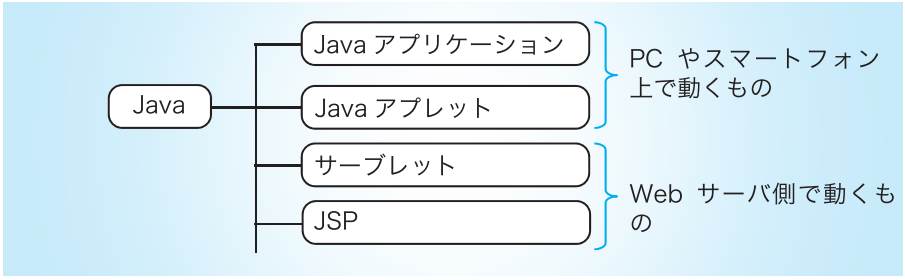
コマンドとはコンピュータに対して与える命令のことです。各 OS にはコマンドラインと呼ばれる，コマンドを入力するためのプログラムがあり，Windows ではコマンドプロンプトというプログラム名になっています。

【プログラム名　Test の作成から実行までの手順】



(7)　Java の利用形態

Java は現在，さまざまな場面で利用されています。最も標準的な「Java アプリケーション」，Web ブラウザ上で動的なコンテンツを実現する「Java アプレット」や，Web サーバの側で動作する「サーブレット」や「JSP」などです。ほかにも「Android アプリ」などに代表されるようなスマートフォンで動くものや，家電製品などでの利用も盛んになっています。



キーワード

コマンドライン
コマンドを入力するためのプログラム

①　Java アプリケーション

デスクトップ環境で動作し，OS にインストールされた JVM によって実行されるプログラムです。

試験で出題される Java プログラムは，GUI などを使用せずコマンドラインから実行するような最も基本的な Java アプリケーションが対象になっています。

Java と JavaScript の関係

Java と似た名前のプログラミング言語に JavaScript があります。Web ページ上で様々な機能を持たせるために利用されていることをご存知の方も多いでしょう。Java が米 Sun Microsystems 社（現 米 Oracle 社）が開発したプログラミング言語である一方，JavaScript は Web ブラウザの黎明期を担った米 NetScape 社が開発したプログラミング言語です。両社が業務提携していたこともあり，Java という名称や文法として類似点がいくつもあります。

しかし，Java はコンパイル型のプログラミング言語ですが，JavaScript はインタプリタ型の言語です。また，変数の型を JavaScript では意識しないなど，大きな違いもあります。このように，文法はほぼ同じですが，動作環境が全く異なる言語だということを認識しておいてください。

第1章 理解度チェック

ポイントは頭に入っていますか？
理解できているか Check してみましょう。

check

- ☑ マルチプラットフォーム
- ☑ クラス
- ☑ ブロック
- ☑ コメント
- ☑ インデント
- ☑ 文
- ☑ リテラル
- ☑ メソッド
- ☑ エスケープシーケンス
- ☑ 予約語（キーワード）

第1章 章末問題

問1 － 1 次のプログラムの実行結果を解答群から選びなさい。

```
// Java プログラム問題
class Test1_1 {
  public static void main(String[] args) {
    System.out.print("Java の");
    System.out.println("問題");
    System.out.println("です。");
  }
}
```

解答群

ア **実行結果▶**
```
Javaの問題です。
```

イ **実行結果▶**
```
Javaの
問題です。
```

ウ **実行結果▶**
```
Javaの問題
です。
```

エ **実行結果▶**
```
Javaの
問題で
す。
```

答え

問1 － 2 次のprintln文は文字列“問題”を出力しているものです。正しい記述には○，誤っている記述には×を付けなさい。

(a) `System.out.println("問題")` (a) ［　　］

(b) `System.out.println  ("問題");` (b) ［　　］

(c) `System out println("問題");` (c) ［　　］

(d) `System.out.println(問題);` (d) ［　　］

問1 － 3　次のプログラムには誤りがあります。誤っている箇所を答えなさい。

```
// Javaプログラム問題
class Test1_2
  public static void main(String[] args) {
    System.out.println("エラーがあるプログラムです");
  }
```

答え

問1 － 4　次のリテラルと対応するリテラルの種類を解答群から選びなさい。

(a)　12　　(a)

(b)　"abc"　　(b)

(c)　'a'　　(c)

(d)　"b"　　(d)

(e)　"12"　　(e)

(f)　3.0　　(f)

解答群

ア　整数リテラル　　イ　浮動小数点数リテラル
ウ　文字リテラル　　エ　文字列リテラル

Java Programming

第15章 午後試験問題の対策と演習

この章ではまず 15.1 で基本情報技術者試験の午後の言語問題の解法についてのテクニックを例題で解説し，その後で演習問題を掲載しています。考え方や解き方は別冊の解答・解説で詳しく解説しました。

演習方法としては，単に解説を最初から読むということはせず，問題文を見て，自分でまず処理内容を調べ，結果を考えてから解説を読むことをおすすめします。この繰返しで解法力を付けることができます。

本試験では 40 分程度の時間で解答を出すことが要求されますが，最初は多くの時間をかけて考えて解いても構いません。試験が近づいてきたらこの時間内で解くように心がけてください。

注　問題文や解説中の'\'は'¥'と同じコードです。
'\n'や'\0'は，'¥n'や'¥0'と解釈してください。

15.1 午後の言語問題の解法

(1) 解き方のテクニック

午後の言語問題でJavaを選択する人は，すでに教育機関や企業でJavaを学んできた方が多いと思います。また，これから就職する人で，企業でJavaを使うので学習するという学生の方も多いかと思います。

過去に出題されたJavaの問題を見ると，クラス定義におけるスーパクラスの指定（extends），インタフェースの実装(implements)，例外に関する記述（try-catchや，throws，throw）など，文法にまつわる設問や，if文の穴埋め，変数の型を問うもの，代入する変数・定数を問うものといった，処理内容にまつわる設問の他，プログラムコードとして示されたプログラムを実行した際の結果をトレースさせる設問に大きく分けられます。

今までに出題された問題の一部を演習問題として15.2以降に掲載していますが，中には一筋縄ではいかない難易度の高い問題も含まれています。これらの問題は，文法を学習しただけの知識では簡単には解けません。それではどのようにすれば合格ラインに達することができるでしょうか。

(2) 試験対策の要点

実際に試験を受けて合格に結び付けるための対策の要点は次の2点です。

① Javaはオブジェクト指向の言語なので，プログラムコードもクラスかインタフェースという単位に分かれていること
② List，Set，Mapがかなりの頻度で登場すること

文法に関連する設問については，どうしても暗記が必要です。過去問題を解いたり，自分でも簡単なプログラムを書いてみるといった手法で，覚えましょう。

if文の穴埋め，変数の型を問うもの，代入する変数･定数を問うものといった，処理内容に関連する設問を解く上では，基本的なアルゴリズムのパターンを覚えておくことが非常に重要です。これはトレース力の増強と併せて鍛えることができます。典型的なアルゴリズムである整列処理や探索処理をJavaで実際にプログラミングし処理結果を自分で理解することによって，考える力を養いましょう。

キーワード
アルゴリズム
問題を解決するための手順

(3) 問題文の解読方法

次に問題文を解読するよい方法を紹介します。実際に試験場に行って問題を読んだときには，時間の制限もあり，十分に問題を理解することはできません。特に何ページにも渡る長い問題では読むだけで時間がかかってしまいます。

その場合はまず設問を先に読み，設問で問われていることは何なのかを早く理解するようにします。穴埋め問題ならその空欄に相当する〔プログラムの説明〕の部分があるはずです。空欄の前後をよく読み，〔プログラムの説明〕の該当箇所にアンダーラインを引いておきましょう。

選択肢を一つ一つ検討し，何が当てはまるかを調べます。選択肢の中には紛らわしいものも含まれているので，問題で示されたプログラムとよく対比して選択を間違わないことが大切です。

選択肢の中に添字を含む問題では，添字をiとすると，選択肢の中にi，i+1，i-1が入っていることがあります。このような問題で正解を導くためには，添字の値をトレースすることも必要です。

(H25 秋-FE 午後問 11)

問 11 次の Java プログラムの説明及びプログラムを読んで，設問1～3に答えよ。
（Java プログラムで使用する API の説明は，この冊子の末尾を参照してください。）

〔プログラムの説明〕

ディレクトリパス（以下，パスという）から木構造を生成するプログラムである。

パスは，木構造をもつファイルシステムにおいてディレクトリを特定するために利用される文字列であり，ディレクトリの名前を“/”で区切って並べて表す。“/”で始まるパスを絶対パスという。絶対パスはルートディレクトリを起点として表したパスである。“/”で始まらないパスを相対パスという。相対パスは任意のノードを起点として表したパスである。

このプログラムが生成する木構造中の各ノードは，それぞれが一つのディレクトリを表し，ルートノードはルートディレクトリを表す。

図1に木構造の例を示す。図1中の楕円(だえん)一つはノード一つに対応し，“ と ”で囲まれた文字列はノードの名前を表す。ルートノードの名前は空文字列とする。例えば，ノード usr を特定する絶対パスは “/usr” であり，ノード usr を起点とする相対パス “local/lib” が特定するノードは，絶対パス “/usr/local/lib” が特定するノードと同じノード lib である。

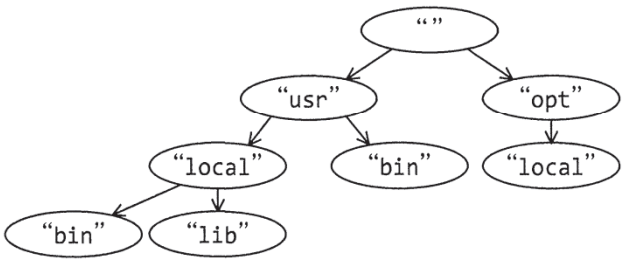


図 1 木構造の例

クラス DirectoryNode は木構造を構成するノードを表すクラスであり，一つのインスタンスが一つのノードを表す。フィールド name はノードの名前を，フィールド parent は親ノードへの参照を，フィールド children は子ノードのリストを保持する。引数を取らないコンストラクタはルートノードを生成する。

クラス DirectoryNode は次のメソッドをもつ。

(1) public DirectoryNode add(String path)

引数 path で与えられたパスが，このノードを起点に一つのノードを特定できるように木構造を拡張し，パスが特定するノードを返す。引数 path が空文字列又は絶対パスを表すなら，IllegalArgumentException を投げる。

パス中の連続する“/”は一つの“/”として扱い，末尾の“/”は無視する。つまり，パス“local//lib/”は“local/lib”とみなす。

(2) public String path()

このノードを特定する絶対パスを表す文字列（末尾は常に“/”）を返す。

例えば，図 1 の最下段のノード lib でこのメソッドを呼ぶと，“/usr/local/lib/”を返す。

(3) public List<DirectoryNode> find(String name)

このノードが保持する各子ノードを頂点とする全ての部分木から，引数 name で与えられた名前をもつノードを全て探し，見つかったノードをリストで返す。

クラス DirectoryNodeTester はテスト用のプログラムである。実行結果を図 2 に示す。

```
usr
opt
/usr/local/
/opt/local/
IllegalArgument
```

図 2 クラス DirectoryNodeTester の実行結果

〔プログラム 1〕

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class DirectoryNode {
   public final DirectoryNode parent;
   public final String name;
   public final List<DirectoryNode> children =
      new ArrayList<DirectoryNode>();

   public DirectoryNode() {
      this( a );
   }
```

Java Programming

巻末資料

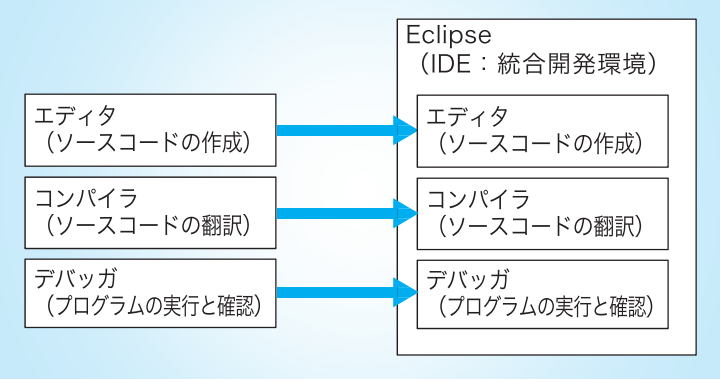
1.1　Eclipse による開発環境の構築
1.2　Java プログラムの作成方法
1.3　デバッグ実行
1.4　ペインとは
2　　修飾子のいろいろ
3　　Java プログラムで使用する API の説明

商標表示　各社の登録商標および商標，製品名に対しては，特に注記のない場合でも，これを十分に尊重いたします。

1.1 Eclipse による開発環境の構築

ここでは，Java のプログラムコードの作成からデバッグ作業までを行うことができる，オープンソースの IDE（統合開発環境），Eclipse（エクリプス）の環境構築方法を紹介します。

Eclipse を利用すれば，見やすく操作しやすい専用の画面上で，簡単にソースコード作成やコンパイル，デバッグ実行をすることが可能になります。



ここでは，Eclipse の入手と環境構築を行う手順を説明します。

1.1.1 JDK（Java Development Kit）の入手とインストール

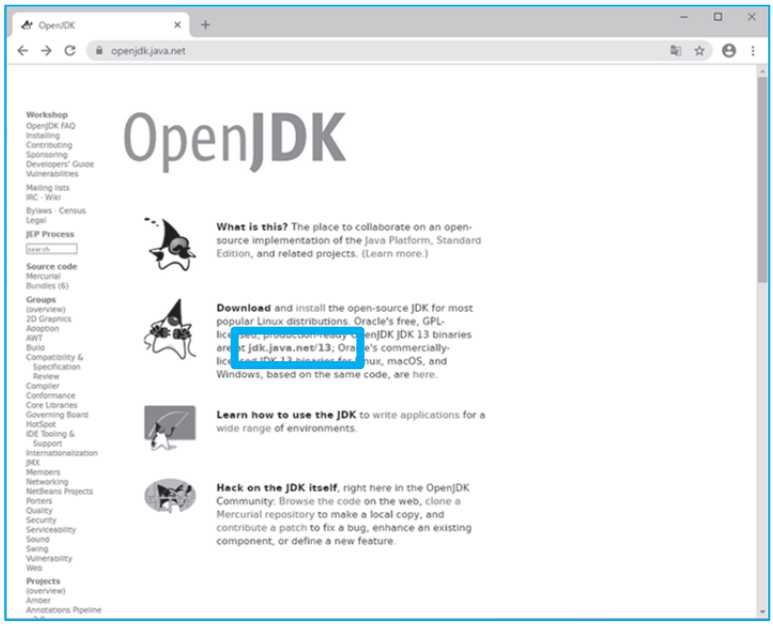
IDE（統合開発環境）である Eclipse のインストールに先立ち，Java のコンパイラや VM（Virtual Machine；仮想マシン）を備えた開発用キットの JDK を入手し，インストールします。

※ここでの手順は 64bit 版の Windows を前提にしています。

> JDK には，以下でご案内している Open JDK の他に，米 Oracle 社が提供している「Oracle JDK」などもありますが，無償で利用する場合には，個人利用（デスクトップ PC やノート PC 上での利用）や，開発目的での利用等に限られる，といったライセンス上の制約がありますので，本書では GPL ライセンスで提供される Open JDK を紹介します。
>
> Oracle JDK のライセンスに関する詳細は以下をご参照下さい。
>
> https://www.oracle.com/downloads/licenses/javase-license1.html

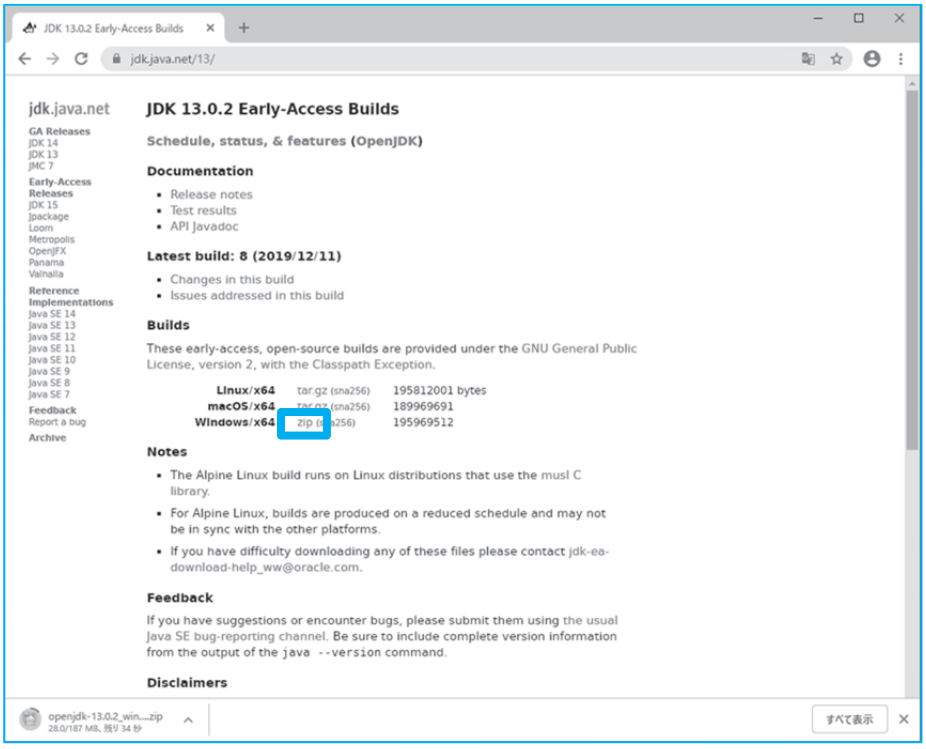
JDK は次の URL から入手します。

`https://openjdk.java.net/`



2 段落目の中盤にあるリンク「jdk.java.net/13」（2020 年 4 月現在）をクリックします。

Windows/x64 用の ZIP ファイルのリンク（zip の部分）をクリックします。



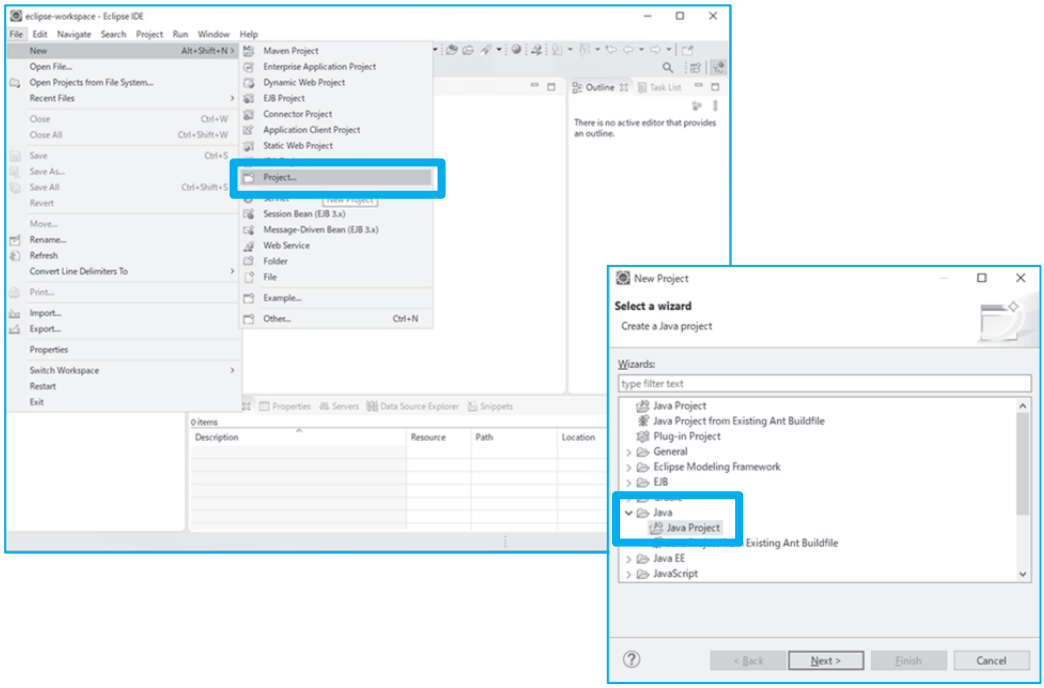
1.2 Java プログラムの作成方法

Eclipse を使った Java プログラムの作成は次の手順で行います。

① 新しい Java プロジェクトの作成
② 新しいクラスの作成
③ 自動コンパイル機能
④ 入力候補自動表示機能

1.2.1 新しい Java プロジェクトの作成

Eclipse を使って Java プログラムを作成する際には，最初に「Java プロジェクト」という箱を作成する必要があります。Java では一つのクラスファイルが一つのクラスを表現しますが，多くの場合，複数のクラスがまとまって，一つの意味をもったプログラムとなります。Java プロジェクトはこれを格納するための箱なのです。逆にいうと一つの Java プロジェクト内には複数のクラスを格納することができます。



Eclipse 上で，Java のプログラムの作成を開始するには，画面上部のメニューから「File」→「New」→「Project」と選択し，「New Project」ダイアログを表示し，そこから「Java Project」を選択します。

2 修飾子のいろいろ

final
private
アクセス指定子なし
protected
public
static
abstract
public なし

1. 変数の修飾子

修飾子	意　味
final	定数として使用できる変数
private	同じクラスからしかアクセスできない変数
アクセス指定子なし	同じパッケージからだけアクセスできる変数
protected	同じパッケージまたはサブクラスからしかアクセスできない変数
public	ほかのクラスからアクセスできる変数
static	クラス変数

private 変数は，同じクラス内からだけアクセスでき，ほかのクラスからはアクセスできません。public 変数はほかのクラスからアクセスできます。

protected 変数については，第 8 章 8.1 のクラスの継承で説明しています。final 変数については第 7 章 7.9 の final 変数で，static 変数については第 7 章 7.8 のクラス変数，クラスメソッドで説明しています。

2. クラスの修飾子

修飾子	意　味
abstract	抽象クラス
final	継承できないクラス
public	ほかのすべてのクラスから参照できるクラス
public なし	同じパッケージからだけアクセスできるクラス

クラスの修飾子の abstract については，第 8 章 8.5 の抽象クラス・抽象メソッドで，final については，第 8 章 8.3 の final で説明しています。

3. メソッドの修飾子

修飾子	意　味
abstract	実装しないメソッド
final	オーバライドできないメソッド
private	同じクラス内からしか呼び出せないメソッド
アクセス指定子なし	同じパッケージからだけ呼び出すことができるメソッド
protected	同じパッケージ内またはサブクラスからしか呼び出せないメソッド
public	ほかのクラスから呼び出せるメソッド
static	クラスメソッド

4. コンストラクタの修飾子

修飾子	意　味
private	同じクラスからしか呼び出せないコンストラクタ
アクセス指定子なし	同じパッケージからだけ呼び出すことができるコンストラクタ
protected	同じパッケージまたはサブクラスからしか呼び出せないコンストラクタ
public	ほかのクラスから呼び出せるコンストラクタ

java.util
  public interface List<E> extends Collection<E>
    リスト（順序付けられたコレクション）のためのインタフェースを提供する。

メソッド

  public boolean add(E e)
    指定された要素をリストの最後に追加する。
    引数：e－リストに追加する要素
    戻り値：true

  public boolean addAll(Collection<E> c)
    指定された要素コレクションの全ての要素をリストの最後に追加する。
    引数：c－リストに追加する要素コレクション
    戻り値：true

  public E get(int index)
    リスト内の指定された位置にある要素を返す。
    引数：index－返される要素のインデックス
    戻り値：リスト内の指定された位置にある要素

  public int indexOf(Object obj)
    指定された要素がリスト内で最初に検出された位置を返す。
    引数：obj－検索する要素
    戻り値：指定された要素がリスト内で最初に検出された位置
            リストにこの要素がない場合は－1

  public iterator<E> iterator()
    型 E の要素セットの反復子を返す。
    戻り値：型 E の要素セットの反復子

  public boolean isEmpty()
    リストに要素がなければ true を返す。
    戻り値：リストに要素が一つもなければ true，それ以外は false

  public int size()
    リスト内の要素数を返す。
    戻り値：リスト内の要素数

  public void clear()
    全ての要素をリストから削除する。

java.util
  public class ArrayList<E>
    インタフェース List の配列による実装である。

コンストラクタ

  public ArrayList()
    空のリストを作る。

  public ArrayList(int n)
    指定された大きさの空リストを作る。
    引数：n－リストの大きさ

メソッド

  public boolean add(E e)
    インタフェース List の add と同じ

public int size()
    リスト内の要素数を返す。
    戻り値：リスト内の要素数

章末問題　解答・解説

第１章

問 1−1

【解答】

ウ

【解説】

print 文と println 文は( )の中で指定した値を画面に表示する文です。どちらも同じような働きをしますが，print 文は末尾に改行を付けないで文字列を表示する一方，println 文は末尾に改行を付けて文字列を表示します。

```
System.out.print("Java の");       //改行なし
System.out.println("問題");        //改行あり
System.out.println("です。");      //改行あり
```

よって，結果は次のようになります。

実行結果▶

```
Javaの問題
です。
```

【アドバイス】

「1.10 画面への出力」と「1.11 画面出力のバリエーション」を参照してください。

問 1−2

【解答】

(a) ×　　(b) ○　　(c) ×　　(d) ×

【解説】

(a) 文の最後にセミコロン（;）がないので×です。

(b) println と（"問題"）の間が空白になっていますが，Java はフリーフォーマットで記述できるので○です。

(c) println 文のピリオド（.）がないので×です。

(d) println 文で表示する文字列を" "で囲んでいないので×です。

【アドバイス】

「1.4 改行位置とインデント（字下げ）」，「1.5 文の終了（;セミコロン）」，「1.10 画面への出力」を参照してください。

索 引

記号・数字

! …… 99
!= …… 83
!演算子 …… 100
% …… 58
%= …… 66
%演算子 …… 58
& …… 62
&& …… 99
&&演算子 …… 99
&= …… 66
* …… 58
*= …… 66
/ …… 58
/= …… 66
\ …… 24
| …… 62
|| …… 99
||演算子 …… 100
|= …… 66
~ …… 62
¥ …… 24
+ …… 58
+= …… 66
< …… 83
<< …… 63
<<= …… 66
<= …… 83
- …… 58
-= …… 66
== …… 83
> …… 83
>= …… 83
>> …… 64
>>= …… 66
>>> …… 64
>>>= …… 66
^ …… 62
^= …… 66
2次元配列 …… 141

A

abstract …… 232, 426
append …… 244, 245
args …… 324
ArithmeticException …… 275
ArrayIndexOutOfBoundsException … 275, 276
available …… 309

B

Boolean …… 335
boolean …… 42
boolean型 …… 82
break …… 94, 120
BufferedReader …… 317, 318, 323
BufferedWriter …… 319, 320
Byte …… 335
byte …… 42

C

capacity …… 244
case定数 …… 94
catch …… 276
char …… 42
Character …… 335
charAt …… 241, 242, 244, 245
ClassCastException …… 275
ClassNotFoundException …… 275
close …… 309, 310, 317, 319
continue …… 123

D

DataInputStream …… 313, 315
DataOutput …… 314
DataOutputStream …… 313, 314
Date …… 240
default …… 94
Double …… 335
double …… 42
do～while …… 116

E

Eclipse ……400, 407, 409
endsWith …… 241, 243
Enum …… 266
equals …… 227, 241
equalsIgnoreCase …… 241
equalsメソッド …… 229
Exception …… 275
extends …… 208, 263, 265

F

false …… 82
FileInputStream …… 310, 312
FileOutputStream …… 310, 311
FileReader …… 317
FileWriter …… 319
final …… 426
finally …… 279, 280
finalクラス …… 225
final変数 …… 197, 225
finalメソッド …… 225
Float …… 335
float …… 42
for …… 111

G

getClass …… 227
getPriority …… 293

I

IDE ……400
if …… 83
if, else …… 88
if, else if, else …… 92
IllegalAccessException …… 275
implements …… 257, 265
import …… 249
indexOf …… 241, 243
InputStream …… 309
InputStreamReader …… 322
insert …… 244, 245, 246
InstantiationException …… 275
int …… 41, 42
Integer …… 335
interface …… 256
InterruptedException …… 275
IOException …… 275

J

Javaアプリケーション …… 12
Javaアプレット …… 12, 13
Java仮想マシン …… 11
JDK …… 11, 400, 409
join …… 293, 296
JRE …… 11, 409
JSP …… 12, 13
JVM …… 11

L

lastIndexOf …… 241
length …… 147, 241, 243, 244, 245
Long …… 335
long …… 42

M

mainメソッド …… 18

N

NegativeArraySizeException …… 275
newLine …… 320, 321
new演算子 …… 130, 158
NoSuchFieldException …… 275
NoSuchMethodException …… 275
null …… 26, 318, 319
NullPointerException …… 275
nullリテラル …… 23
NumberFormatException …… 275, 276