

Chapter

# 1

## プログラミングとJava言語

簡単なツールだけを使ってプログラムを作る時代は去り、今では、IDE（統合開発環境、Integrated Development Environment）という開発ツールを使うのが普通になりました。本書では、Java 言語用の IDE として最も有名な Eclipse を使うことを前提に解説します。

この章では、Java 言語の概要を解説した後、学習を始める準備として、Eclipse を手元のパソコンにセットアップします。また、プログラムの作成、保存、実行といった一連の開発過程についても解説します。

- 1.1 Java 言語とは
- 1.2 開発ツールの準備
- 1.3 Eclipse による初めてのプログラム
- 1.4 実行の仕組み
- 1.5 まとめ
- 1.6 通過テスト

# 1.1 Java言語とは

**Java言語**は、1995年にSun Micro Systems社のジェームズ・ゴスリン、ビル・ジョイ等が開発したオブジェクト指向言語です。Java言語は、C/C++の文法を参考にしつつも、後発の強みを生かして、より使い易く、より先進的な機能を備えた言語です。

メモリ管理の自動化(**ガベージコレクション**)や同時に複数の処理を実行できる**マルチスレッド**、そしてOSが違って同じプログラムをそのまま実行できるJavaバイトコードとJava仮想マシンなど、多くの先進的な機能を実現しています。

Java言語のスローガンは、"Write once, run anywhere" (一度プログラムを書けば、どこでも実行できる)です。それが可能なのは、プログラムを**Javaバイトコード**という中間言語に翻訳し、それを**Java仮想マシン** (JVM、Java Virtual Machine)が機械語に翻訳しながら実行する仕組みだからです。

直接、機械語に翻訳するC/C++よりも実行速度は劣りますが、さまざまな最適化技術により極めて高速に実行できます。現在では、JavaだけでなくJavaScript、Scala、Groovy、JRuby、Jythonなど数十ものスクリプト言語がJVMで稼働するようになりました。

また、Java言語は、初めてインターネットやウェブでのソフトウェア開発を目標とした言語です。現在、電子メールやSNSをはじめ、ネットショッピング、ネット予約、ネット銀行など、ウェブを介して利用するシステムが当たり前になりましたが、Java言語はこれらを開発するための最も信頼できる言語と考えられています。

Javaには、用途に応じて次の3つのエディションがあります。Jakarta EEは、Java EEと呼ばれていたエディションが、2017年にEclipse Foundationに寄贈され、オープンソース化されたものです。本書はJavaSEを対象としています。

名 称	用 途
Java SE (Standard Edition)	基本的なアプリケーション
Jakarta EE (Enterprise Edition)	ウェブシステムなどのサーバーアプリケーション
Java ME (Micro Edition)	モバイル機器、家電などに組み込むプログラム

## 1.2 開発ツールの準備

本書では開発ツールとしてEclipse (<http://Eclipse.org>)を使います。Eclipseは、非営利団体のEclipse Foundationにより開発・保守されている統合開発環境(IDE、Integrated Development Environment)で、プログラムの作成、コンパイル、実行、デバッグ、管理などを総合的に支援するツールです。OSS(オープンソースソフトウェア/内容が公開されていて自由に再頒布できる)なので、無償で利用できます。

本書では、オリジナルのEclipse IDE for Java Developerパッケージに、例題や練習問題を組み込んだオールインワンスタイルのEclipseを使って学習します。学習に先立ってWindows版、またはMacOS版をサポートウェブ(<https://k-webs.jp>)からダウンロードしてください。

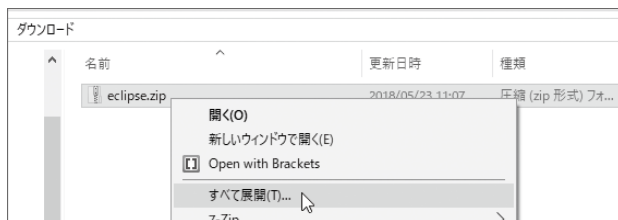
### 1. Windowsへのインストール

サポートウェブからダウンロードしたファイルを展開(解凍)するだけで利用できます。次に、展開した後、起動用アイコンを作るまでの手順を示します。

★この手順はサポートウェブの解説動画[ビデオ番号: 001]で見ることができます。

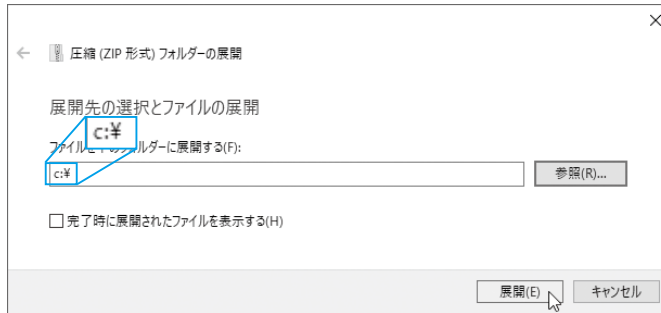
#### 【手順】

- ① エクスプローラーを開いて、ダウンロードしたeclipse.zipを表示する
- ② eclipse.zipファイルをマウスの右ボタンでクリックし、[すべて展開]を選ぶ



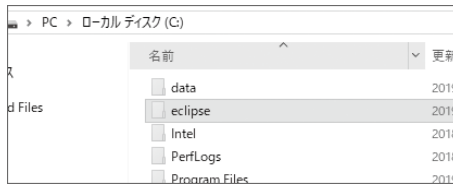
⇒ [圧縮フォルダの展開]ダイアログが開く

- ③ [ファイルを下のフォルダーに展開する]の欄に、c:¥と入力して、[展開]ボタンを押す



※Eclipseのフォルダ名やファイル名が非常に長いので、展開するとWindowsの制限を超えてしまうことがあります。そのため、ドライブの直下に展開することが推奨されています。d:¥のように、他のドライブが利用できる場合は、それを指定してもかまいません。

⇒ Cドライブの直下にeclipseフォルダができる



- ④ eclipseフォルダをダブルクリックして開く  
 ⑤ eclipse.exe を探してマウスの右ボタンでクリックし、[タスクバーにピン止めする]を選ぶ



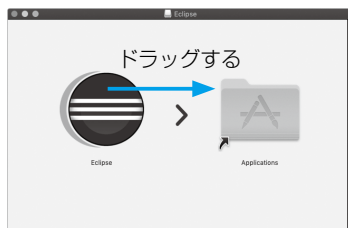
⇒ タスクバーに起動用のEclipseアイコン(🌐)ができる

## 2. MacOSへのインストール

サポートウェブからダウンロードしたファイル (eclipse.dmg) を、ダブルクリックしてインストールします。やり方は、MacOS用の他のプログラムと同じです。ただし、APPストアからダウンロードしたプログラムではないので、ターミナルで `xattr` コマンドを実行しておく必要があります。xattrコマンドは、プログラムを実行可能にするコマンドです。

### 【手順】

- ① サポートウェブから、MacOS用のEclipseをダウンロードする
- ② ダウンロードしたファイル (eclipse.dmg) をダブルクリックしてインストールする



- ③ ターミナルを起動し、xattrコマンドを実行する

```
tkx — -bash — 66x5
Last login: Sun Jun 23 15:09:52 on ttys000
KawabanoiMac:~ tkx$ xattr -c /Applications/Eclipse.app
KawabanoiMac:~ tkx$
```

`xattr -c /Applications/Eclipse.app`

# プログラムの書き方

Java 言語では、プログラムを「クラス (class)」という単位で作成します。クラスがどのような要素で構成され、どのように書けばよいのか、基本的な原則を知ることから始めましょう。この章では、コンソールに簡単な文を出力する例を通して、Java プログラムの基本的な書き方について解説します。

- 2.1 プログラムの成り立ち
- 2.2 プログラムのフォーマット
- 2.3 コメント文
- 2.4 コンソールへの出力
- 2.5 まとめ
- 2.6 通過テスト

## 2.1 プログラムの成り立ち

2

### 例題2-1 プログラムの構造

```
1 package sample;
2 public class Sample2_1 {
3
4     public static void main(String[] args) {
5         // ここにコードを挿入
6         System.out.println("Hello World");
7     }
8     実行結果のコンソール表示
9 }
```

```
Hello World
```

例題は、1章で作成したプログラムです。ここでは、プログラムの構造を説明します。まず、この中にある次の行は、プログラムの動作には何の影響も及ぼしません。

- ・空白だけの行 (3、8行目)
- ・// で始まる行 (5行目)

// で始まる記述は、**コメント文**とって、プログラムについての説明などを書いたものです。そこで、コメント文と空白行を取り去ってみると、プログラムは次のようになります。

```
1 package sample;
2 public class Sample2_1 {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5     }
6 }
```

これがプログラムの基本の形です。

また、書き方では、次のような特殊な書き方が見られます。

- ・行によって書き出しの位置が右にずれている
- ・{} の始まりと終わりが不自然な場所書かれている

このように、プログラムの形式や書き方には、特有の約束事があります。次に、それらを1つずつ見ていきましょう。

## 1. パッケージ文

まず、1行目は**パッケージ文**といいます。プログラムが含まれるパッケージの名前を書きますが、必ずプログラムの先頭行(第1行目)に書く必要があります。ただ、Eclipseではパッケージ文は自動的に挿入されるので、手書きすることはありません。

```
1 package sample; ← パッケージ文
2 public class Sample2_1 {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5     }
6 }
```

## 2. クラス宣言

Javaでは、プログラムを**クラス**という単位で作成します。2行目はクラスが始まる部分で、**クラス宣言**といいます。

`public class` はクラス宣言のキーワードで、`Sample2_1` がクラスの名前です。クラスの名前は、自由に決めることができますが、ただし、先頭は英字の大文字でなくてはなりません。

また、`{}` で囲まれた範囲を**ブロック**といい、その中に、クラスの具体的な内容が書かれています。

```
1 package sample;
2 public class Sample2_1 { ← クラス宣言
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5     }
6 } ← ブロック
```



## 3. メソッド

プログラムの中身にあたる3、4、5行の部分を**メソッド**といいます。クラスにはメソッドをいくつでも書くことができますが、しばらくの間は、メソッドをひとつだけ持つクラスを扱います。

3行目がメソッド宣言です。メソッドの名前は**main**です。mainは、プログラムの開始点になる特別なメソッドです。他の名前を使うことはできません。単語がいくつも並んでいますが、意味は後で説明しますので、当分の間はこのとおりに書いてください。

メソッドの定義内容も {} で囲まれたブロックの中に書きます。

```

1 package sample;
2 public class Sample2_1 {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5     }
6 }
```

メソッド宣言

ブロック

## 4. 命令文

メソッドの中で、具体的な仕事(処理)を実行するために書くのが**命令文**です。単に**文**ともいいます。命令文の最後には、**セミコロン (;)**が必要です。; は文の終わりを示す記号で、日本語の句点(。)にあたります。

命令文はいくつでも書くことができますが、このクラスでは4行目に1つだけ書いてあります。この命令文は、( ) の中に書いた "Hello World" という文字列を画面に表示します。

```

1 package sample;
2 public class Sample2_1 {
3     public static void main(String[] args) {
4         System.out.println("Hello World");
5     }
6 }
```

命令文、最後はセミコロン

このように、クラスの中にメソッドを書き、その中に具体的な命令文を書くのがJava言語の標準的な書き方です。

## 練習2-1

1. プログラムの構成要素は覚えましたか？

確認のため、次の問に答えてください。

問1 ①～⑤にあてはまる語は何ですか。

問2 プログラムの中にブロックはいくつありますか。

```
package exercise; ← ① 文
public class Ex2_1 { ← ② 宣言

    public static void ③ (String[] args) { ← ④ 宣言
        // ここにコードを挿入
        System.out.println("Hello"); ← ⑤ 文
        System.out.println("こんにちは");
    }
}
```

## 13.4 まとめ

13

この章では、配列の仕組みと作り方について解説しました。特に、配列変数に参照が入っていることと、参照を使って配列にアクセスする仕組みは重要なトピックです。Arraysクラスの使い方に関連して、ラムダ式とストリーム処理についても解説しました。「配列」の配列では、表示と入力の方法を解説しました。

### 配列の作り方

- ・初期化リストで作成

```
int[] n = {1, 2, 3, 4, 5};
int[][] n = {
    {1, 2, 3},
    {4, 5, 6}
}
```

- ・無名配列

```
int[] n;
n = new int[] {1, 2, 3, 4, 5};
```

- ・newで生成する(既定の初期値で初期化される)

```
int[] n = new int[5];
int[][] n = new int[2][4];
```

### 配列の仕組み

- ・配列本体はヒープに、配列変数はスタックに作成される
- ・変数には参照が入っていて、参照による配列オブジェクトへのアクセスはJVMが代行する

### Arraysクラス

- ・配列を操作するユーティリティメソッドがある
- ・sort(ソート)、binarySearch(二分検索)、fill(一括代入)、copyOf(コピー)、toString(文字列化)、stream(ストリーム生成)
- ・streamメソッドは配列をストリームに変換し、filter、sorted、distinct、count、sum、forEachなどのメソッドでストリーム処理ができる
- ・30を超える要素だけを出力するストリーム処理



```
Arrays.stream(number).filter(n->n>30).forEach(n->System.out.println(n));
```

### 「配列」の配列

- ・int[][] drink では、drink[i][j] は、配列 drink[i] の [j] 番目の要素である
- ・2次元配列ともいうが、実質的に、1次元配列である
- ・出力には拡張for文を使い、入力には通常のfor文を使う

# 13.5 通過テスト

13

1. 次の表のデータを初期化リストにより配列として定義しなさい(Pass13\_1)。

A. 毎月の平均気温( double[] temp )

4月	5月	6月	7月	8月	9月
20.5	23.4	26.1	28.5	33.5	29.1

B. 月別商品売り上げ高( int[][] sales )

	6月	7月	8月	9月
ブドウ	120	130	100	110
メロン	250	230	230	240
バナナ	105	110	120	125

2. 1のAの配列 temp について、指示された処理を行うプログラムを作成しなさい。

問1 tempをArraysクラスのsortメソッドを利用して、昇順にソートし、実行結果のように表示しなさい(Pass13\_2\_1)。ただし、繰り返し処理には拡張for文を使います。

```
20.5    23.4    26.1    28.5    29.1    33.5
```

問2 ArraysクラスのcopyOfメソッドを使って、temp をコピーした配列 tempCopy を作成し、実行結果のように表示しなさい。ただし、配列要素の数を12個に増やします(Pass13\_2\_2)。

```
20.5 23.4 26.1 28.5 33.5 29.1 0.0 0.0 0.0 0.0 0.0 0.0
```

問3 Arraysクラスのstreamメソッドを使ってストリームに変換し、ストリーム処理により合計と件数をそれぞれ別に求めた上で、合計を件数で割って平均を表示しなさい(Pass13\_2\_3)。ただし、出力はprintfメソッドを使い、小数点以下は2桁目まで表示すること。また、合計はsumメソッド、件数はcountメソッドを使うこと。

```
平均=26.85
```

問4 streamに変換し、28度以上の気温だけを、実行結果のように表示しなさい (Pass13\_2\_4)。ただし、filterメソッドとforEachメソッドを使うこと。

```
28.5
33.5
29.1
```

3.1のBの配列 sales について、指示された処理を行うプログラムを作成しなさい。

問1 すべての要素を、for文を使って、実行結果のように出力しなさい (Pass13\_3\_1)。拡張for文を使わずに普通のfor文で作成します。

```
6月 7月 8月 9月
120 130 100 110
250 230 230 240
105 110 120 125
```

<ヒント>

- ・1行目の月は、"6月 7月 8月 9月" を出力します
- ・for文の書き方は例題13-8を参考にしなさい

問2 次のように商品別の合計を右端に表示しなさい(Pass13\_3\_2)。

```
6月 7月 8月 9月 合計
120 130 100 110 460
250 230 230 240 950
105 110 120 125 460
```

<ヒント>

- ・1行目は、"6月 7月 8月 9月 合計" を出力します
- ・商品別の合計を入れる変数 sum を作成し、1つの商品の売上高を合計します

問3 次のように、商品名を左端に表示しなさい(Pass13\_3\_3)。

```
        6月 7月 8月 9月 合計
ブドウ 120 130 100 110 460
メロン 250 230 230 240 950
バナナ 105 110 120 125 460
```

<ヒント>

- ・1行目は、" 6月 7月 8月 9月 合計" を出力します
- ・商品名の配列 String[] name = {"ブドウ", "メロン", "バナナ"}; を作成します
- ・1つの商品の売上高を表示する時、最初に商品名を出力します

4. 要素を5つ持つString型の配列 `name` を作成し、キーボードから名前を入力して、`name` にセットするプログラム (Pass13\_4) を作成しなさい。ただし、名前は、すべてひらがなで入力します。また、配列に代入した後、配列をストリームに変換し、名前順にソートした上で、実行結果のように表示しなさい。

```
String>わだひろし☑
String>さいとうたけし☑
String>かわばたいちろう☑
String>むらたしょうじ☑
String>たなかこうじ☑
かわばたいちろう
さいとうたけし
たなかこうじ
むらたしょうじ
わだひろし
```

<ヒント>

- ・ストリームでソートするには、`sorted()` メソッドを適用します
- ・出力には `forEach` メソッドを使います

5. `String[][] meibo = new String[2][3];` を定義し、この配列に、次の表データをキーボードから入力するプログラム (Pass13\_5) を作成しなさい。また、入力後、すべてのデータを実行結果のように表示しなさい。

名前	所属	住所
田中佳子	総務部	東京都
鈴木一郎	営業部	神奈川県

```
String>田中佳子☑
String>総務部☑
String>東京都☑
String>鈴木一郎☑
String>営業部☑
String>神奈川県☑
氏名 所属 住所
田中佳子 総務部 東京都
鈴木一郎 営業部 神奈川県
```

<ヒント>

- ・1行目は、"氏名 所属 住所" を出力します
- ・出力するデータは、半角空白を2文字、右に連結して出力します

Chapter

16

## オブジェクトの作り方

オブジェクト指向は、オブジェクトを定義して作成するところから始まります。ここでは、データの集まりをオブジェクトにするという例を元に、オブジェクトを定義し、インスタンスを作成し、インスタンスメソッドを使ってみるまでの手順を解説します。また、ソースコードの作成には、実際のプログラム開発と同様に、Eclipse のソースコード自動生成機能を使います。

- 16.1 オブジェクトを定義する
- 16.2 インスタンスを作る
- 16.3 インスタンスメソッドを使う
- 16.4 まとめ
- 16.5 通過テスト

# 16.1 オブジェクトを定義する

**?** オブジェクト指向というのは、どういうことですか？  
 パッと、わかるような説明があるとうれしいのですが。

オブジェクト指向とは、クラス、継承、多態性を使うプログラミングスタイル、ということですが、これだけでは内容までは無理ですね。

でも、安心してください。この本を最後まで読むと、必ず理解できるはずですよ。

クラス、継承、多態性(ポリモーフィズム)は、オブジェクト指向の3大特徴と言われています。これから、それらを1つずつ、プログラムを作りながら見ていきます。プログラムは、Eclipseで大部分、自動生成します。思ったよりも簡単なことがわかるでしょう。

## 1. データの集まりをオブジェクトにする

一番シンプルなオブジェクトは、データの集まりです。いろいろな項目を持つデータを、まとめて1つのオブジェクトにします。次の表を見てください。

### ▼ 商品売り上げ明細

商品コード <code>String code</code>	品名 <code>String name</code>	価格 <code>int price</code>	個数 <code>int quantity</code>	在庫の有無 <code>boolean stock</code>
SY-200	冷蔵庫	50,000	20	あり
TB-100	洗濯機	30,000	15	あり
AX-551	テレビ	15,000	32	なし

1件のデータは、商品コード、品名、価格、個数、在庫の有無という5つの項目からできています。これらをまとめて、1個のデータとして扱えると便利です。


ただ、商品コードのデータ型はStringですが、価格はintです。このように、項目のデータ型が違っていると、1つの配列にすることはできません。

そこで、これを1つのオブジェクトにします。配列の拡張版みたいなものです。オブジェクトにするには、まずは、定義が必要です。次はそれをやってみましょう。



オブジェクトの名前をSalesとすると、定義は次のようにします。


```
public class Sales {  
    private String code;        // 商品コード  
    private String name;       // 商品名  
    private int price;         // 価格  
    private int quantity;      // 個数  
    private boolean stock;     // 在庫  
}
```

 オブジェクトを定義するということですが、これは、クラスを作っているのですか？

そうです。これは、クラスの定義です。クラスでオブジェクトを定義します。変数宣言を並べるだけなので簡単です。ただ、変数は非公開にするため、`private`を付けて宣言します。

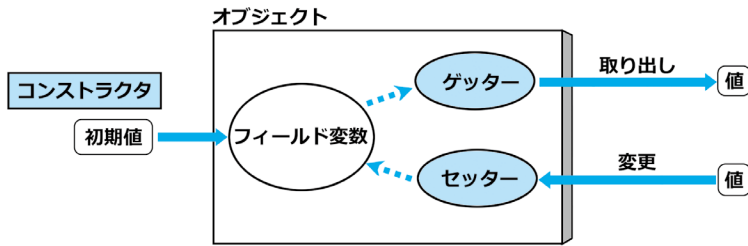
オブジェクトを定義するには、クラスの中に変数宣言を並べるだけです。クラスの中で宣言した変数をフィールド変数(クラス変数)とかフィールドといいます。

フィールド変数は、オブジェクトの内部情報ですから、`private`を付けて非公開にします。`private`は、宣言したクラスの中でだけ使用できるという意味でアクセス修飾子といいます。なお、これまで使っていた`public`もアクセス修飾子ですが、制限なく公開するという、全く逆の意味です。

 オブジェクトを定義したので、早速、使ってみたいのですが、使い方を教えてください？

いえ、使うためにはあと少し準備が必要です。定義から、オブジェクトを生成したり、フィールド変数の値を操作する仕組みが必要です。ただ、それは、Eclipseで自動生成できるので、手書きすることはありません。

オブジェクトを利用するには、次の図のような仕組みが必要です。



まず、左端の**コンストラクタ**は、オブジェクトに初期値を設定します。**ゲッター**は、オブジェクトの中から、特定のフィールド変数の値を取り出します。また、**セッター**は、オブジェクトの特定のフィールド変数の値を変更します。

<b>コンストラクタ</b>	オブジェクトを作る時、フィールド変数に初期値を代入する
<b>ゲッター</b>	オブジェクトからフィールド変数の値を取り出す
<b>セッター</b>	オブジェクトのフィールド変数の値を変更する


これらの仕組みを、オブジェクトの定義の中に組み込みましょう。

組み込みはEclipseで自動生成できるので、次の手順で実行します。後の練習問題で、やっ  
てもらおう予定ですから、ざっと手順を見ておいてください。

## 2. 基本機能の組み込み

フィールド変数を定義する前の、クラス作成のところから基本機能の組み込みまで、一連の手順を通して見ていきましょう。

### 手順1 クラスの作成

- ① sampleパッケージをマウスでクリックする(=パッケージの選択)
- ② [新規Javaクラス]ボタン(  )を押して、[新規Javaクラスダイアログ]を開く
- ③ [名前]欄にSalesと入力する
- ④ [完了]を押す(mainメソッドは作成しません)

The screenshot shows the 'New Java Class' dialog box. The 'Name (N):' field contains 'Sales'. The 'Superclass (S):' field contains 'java.lang.Object'. The 'Access modifier' section has 'public(P)' selected. The 'Complete' button at the bottom is labeled '完了(F)'. Callout boxes with numbers 3 and 4 point to the name field and the '完了(F)' button respectively.

- ⑤ Salesクラスにフィールド変数を入力して保存します。

```
package sample;
public class Sales {
    private String code;    // 商品コード
    private String name;   // 商品名
    private int price;     // 価格
    private int quantity;  // 個数
    private boolean stock; // 在庫
}
```

次は、基本機能の組み込みです。コンストラクタの組み込みと、ゲッター・セッターの組み込みの2つに分けて実行します。

## 手順2 コンストラクタの組み込み

自動生成では、現在のカーソル位置にコードが追加されるので、フィールド変数を書いた次の行(下図8行目)にカーソルを置いてから、作業を始めてください。

```

Salse.java
1 package sample;
2 public class Sales {
3     private String code; // 商品コード
4     private String name; // 商品名
5     private int price; // 価格
6     private int quantity; // 個数
7     private boolean stock; // 在庫
8
9

```

改行し、ここにカーソルを置いてから、組み込みの操作を実行します

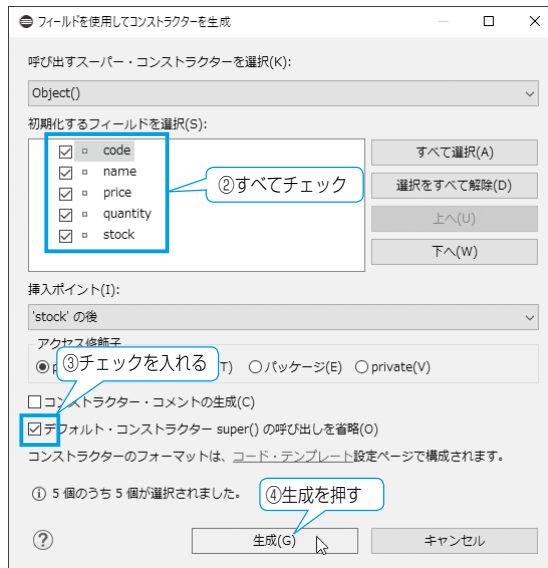
### 【注意】

カーソルの位置に注意してください。カーソルがクラスの外にあると、「この操作は現在の選択には適用できません」というエラーが表示されます。

16

①Eclipseのメニューで、[ソース]⇒[フィールド変数を使用してコンストラクターを生成]と選択する

⇒[フィールドを使用してコンストラクターを生成]のダイアログが開く



② ダイアログで、すべてのフィールドがチェックされていることを確認する

③ [デフォルト・コンストラクター super()の呼び出しを省略]にチェックを入れる

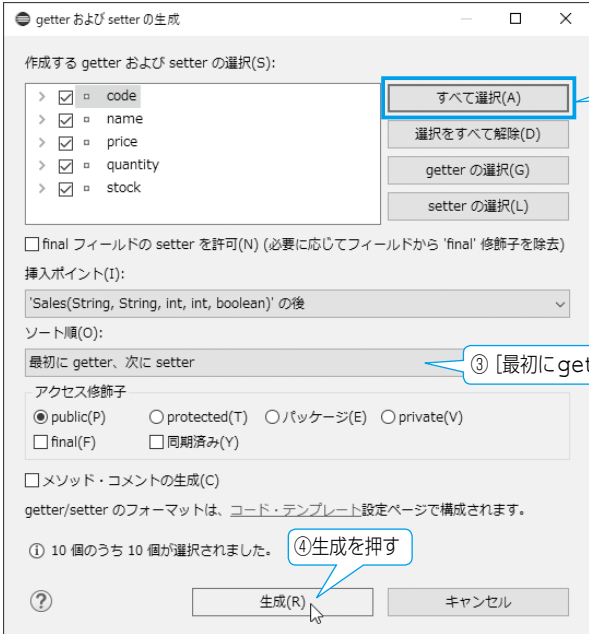
④ [生成]を押す

以上でコンストラクタが生成されるので、ゲッター・セッターの生成へ進みます。

## 手順3 ゲッター・セッターの生成

カーソルがコンストラクタの次の行にあることを確認して、次の手順を実行します。

① Eclipseのメニューで[ソース]⇒[getterおよびsetterの生成]と選択する  
⇒[getterおよびsetterの生成]ダイアログが開く



② [すべて選択] を押す

③ [最初にgetter、次にsetter] を選択

④ 生成を押す

② [すべて選択] ボタンを押して、すべてのフィールドにチェックを入れる

③ [ソート順]で[最初にgetter、次にsetter]を選択する

④ [生成]を押す

## 完成したクラスについて

完成したクラスは次のようになります。

一挙に40行以上のソースコードになってしまいましたが、1つずつは、どれもよく似た単純なコードです。代入文やreturn文しかないことに気付くと思います。

細かな説明は後でしますので、ひとまず、全体を眺めてみてください。

## 例題16-1 Salesクラス

```
1 package sample;
2 public class Sales {
3     private String code;    // 商品コード
4     private String name;    // 商品名
5     private int price;      // 価格
6     private int quantity;   // 個数
7     private boolean stock;  // 在庫
8     public Sales(String code, String name,
9                   int price, int quantity, boolean stock) {
10        this.code = code;
11        this.name = name;
12        this.price = price;
13        this.quantity = quantity;
14        this.stock = stock;
15    }
16    public String getCode() {
17        return code;
18    }
19    public String getName() {
20        return name;
21    }
22    public int getPrice() {
23        return price;
24    }
25    public int getQuantity() {
26        return quantity;
27    }
28    public boolean isStock() {
29        return stock;
30    }
31    public void setCode(String code) {
32        this.code = code;
33    }
34    public void setName(String name) {
35        this.name = name;
36    }
37    public void setPrice(int price) {
38        this.price = price;
39    }
40    public void setQuantity(int quantity) {
41        this.quantity = quantity;
42    }
43    public void setStock(boolean stock) {
44        this.stock = stock;
45    }
46 }
```

フィールド変数

コンストラクタ

ゲッター

セッター



**this**. ~ となっているところがいくつもありますが、これは何ですか？

thisが書かれているところでは、フィールド変数と同じ名前の引数があります。同じだと区別がつかなくなるので、フィールド変数の側に this を付けるのです。

引数をフィールド変数と同じ名前にするのは、対応関係をわかりやすくするためです。例えば、コンストラクタでは、どの引数をどのフィールド変数に代入するのか、わかりやすくなります。

そこで、**フィールド変数であることを示すためにキーワードthisを付けます**。thisがないと、コンパイラはどれも引数と判断してしまうからです。

・同じ名前の引数がある時、フィールド変数には**this**を付ける

thisを付けるのは、同じ名前の引数がある時だけです。ゲッターのように、フィールド変数を使っても引数がない場合は、thisを付ける必要はありません。



コンストラクタには、戻り値が書かれていませんが、これで、いいのですか。

コンストラクタは、メソッドと似ていますが、メソッドではありません。役割はオブジェクトの初期化に限定されています。値を返せないなので、戻り値はありません。voidと書くのも間違いです。

**コンストラクタ**は、メソッドではなくオブジェクトを初期化するための専用のパーツです。メソッドとよく似ていますが、**名前もクラス名と同じ**で、値を返せないので**戻り値型を書かない**という規則になっています。もちろんstaticも付けません。

#### コンストラクタの特徴

- ①クラス名と同じ名前
- ②メソッドではないので値を返せない⇒戻り値型を書かない

それから、ゲッターとセッターはメソッドですが、クラスメソッドではなく、**インスタンスメソッド**なので、**static を付けない**ことに注意してください。staticを付けるのはクラスメソッドだけです。

・ゲッター、セッターはインスタンスメソッド  
⇒ static を付けない

### 練習16-1

1. これまでの手順は、オブジェクトを作る時に必ず行う手順です。ここで復習して、しっかり覚えておきましょう。

例題にならって、次のパソコン用インクカートリッジの注文表から、オブジェクトのクラスを作ってください。クラス名はOrderです。クラスには、フィールド変数、コンストラクタ、ゲッター、セッターを作成します。

型番 <code>String id</code>	受注日 <code>String date</code>	価格 <code>int price</code>	個数 <code>int quantity</code>	納品済みか <code>boolean delivery</code>
ICBK61	2020-07-11	2100	5	true
ICBK62	2020-09-02	1050	10	false
ICBK63	2020-07-15	1050	12	true

<ヒント>

・フィールド変数は、表の1行目に書いてある型と変数名を使います