

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	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

理解度チェック.....	326
章末問題.....	327

第14章 ジェネリクス

14.1 ジェネリクスとは.....	330
14.2 ジェネリクスとラップクラス.....	335
14.3 ジェネリクスと拡張 for 文.....	338
14.4 オートボクシング/ オートアンボクシング.....	339
理解度チェック.....	341
章末問題.....	342

第15章 午後問題の対策と演習.....

15.1 例題.....	346
15.2 例題解説.....	351
15.3 演習問題.....	356

巻末資料

A.1 Eclipse による開発環境の構築.....	392
A.2 Java プログラムの作成方法.....	396
A.3 デバッグ実行.....	401
A.4 ペインとは.....	404
B 修飾子のいろいろ.....	405
C Java プログラムで使用する API の説明	406

索引

メンバ

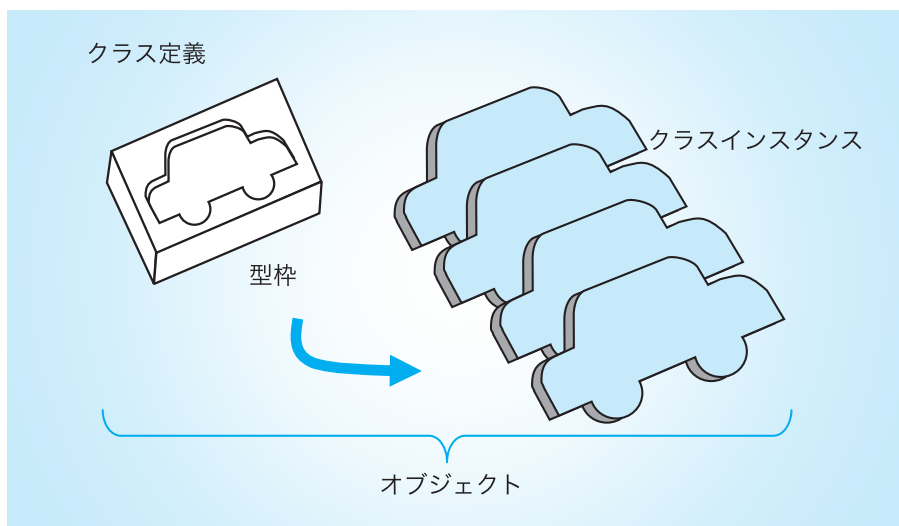


キーワード インスタンス

実体のこと

(1) クラス定義とクラスインスタンス

漠然と「クラス」といった場合、これはクラスの定義を指す場合と、クラスの実体を指す場合の両方があり得ます。厳密にいい分ける場合、前者をクラス定義、後者をクラスインスタンスといいます。クラス定義はチョコレートに例えると型枠、クラスインスタンスは固まったチョコレートになります。そしてチョコレートを作るのと同じように、一つの型枠があれば、実体であるクラスインスタンスはいくつでも作ることができます。



なお、単に「オブジェクト」といった場合も「クラス」と同様に、クラス定義とクラスインスタンスのどちらかをあいまいに示す表現として使われる場合があります。

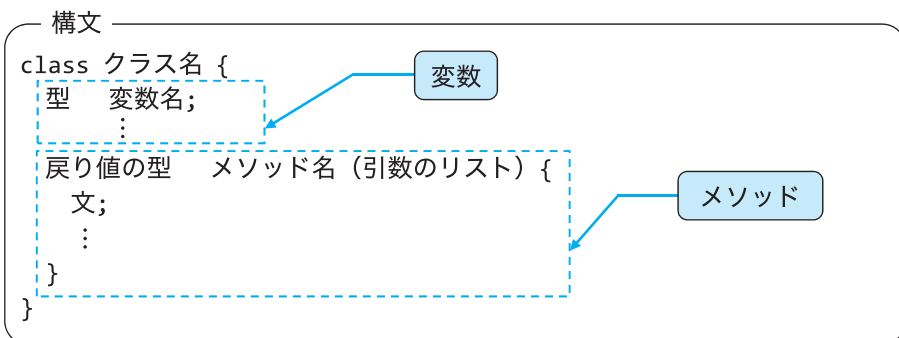
(2) クラス定義（変数とメソッド）

クラスの中には**メンバ**（変数、メソッド）が定義できます。メソッドの定義については7.4の「メソッド」で説明しますので、ここでは概要を理解してください。



アドバイス

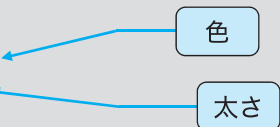
クラスの中の変数やメソッドをメンバといいます。



では、実際に簡単なクラスを定義してみます。

次のプログラムでは「Pencil」というクラスに、色の値をもつ「color」、太さの値をもつ「size」という変数を定義します。これがPencilクラスの変数となります（とりえず、Pencilクラスには変数だけを定義することにします）。

```
class Pencil {  
    String color;  
    double size;  
}
```



(3) クラスインスタンスの生成

クラスの定義はあくまでもそのクラスでもつメンバを定義しているだけです。定義したクラスをプログラムで使用するときは、そのクラスのインスタンス（クラスインスタンス）を生成しなければなりません。

クラスインスタンスを生成するときは、次のように記述します。

構文

```
クラス名 変数名;  
変数名 = new クラス名();
```

では、実際に先ほどの `Pencil` クラスを使ってクラスインスタンスを生成してみます。

Question

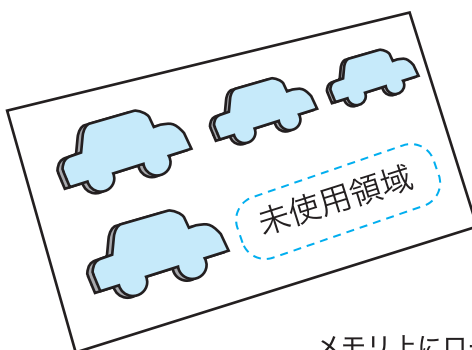


インスタンスの生成のイメージは型枠を作って1個1個のチョコレートを作るようなものですが、コンピュータ上では具体的にどういうことが行われるのですか？

Answer



クラスインスタンスを生成すると、クラス定義で定義されたクラスのメンバ変数やメンバメソッドがメモリ上にロードされます。つまり、メモリを消費するわけです。



メモリ上にロードされた
クラスインスタンス



アドバイス

クラスごとにソース
コードファイルを分
けます。

■Test701 クラスを使用したプログラム

```
1 class Pencil {  
2     String color;  
3     double size;  
4 }
```

Pencil.java

```
1 class Test701 {  
2     public static void main(String[] args) {  
3         Pencil pen;  
4         pen = new Pencil();  
5     }  
6 }
```

Test701.java

Pencil 型の変数の宣言

オブジェクトの生成

【プログラムの説明】

このプログラムは、Pencil クラスと main メソッドがある Test701 クラスの二つのクラスで構成されています。

(4) Pencil 型の変数の宣言

Test701 クラスの 3 行目で Pencil クラスインスタンスを扱うために変数 pen を宣言します。この変数のことを **オブジェクト変数** といいます。オブジェクト変数を宣言するときは、クラス名の後に変数名を記述します。

この場合、変数 pen は Pencil 型のオブジェクト変数となります。int 型や double 型の変数を宣言する記述を思い出してください。int 型の変数 a を宣言するときは、「int a;」と記述しました。Pencil 型の変数 pen を宣言するときは、「Pencil pen;」と記述します。

```
int 型変数 a の宣言 ..... int a;  
Pencil 型変数 pen の宣言 ..... Pencil pen;
```

(5) クラスインスタンスのための構文生成

クラスインスタンスを生成するときは **new 演算子** を使用して「new クラス名()」と記述します。プログラムでは Test701 クラスの 4 行目で Pencil クラスのインスタンスを生成しています。

```
Pencil クラスインスタンス生成 ..... new Pencil()
```

Test701 クラスの 4 行目では生成したクラスインスタンスを Pencil 型変数 pen に代入しています。

```
Pencil 型クラスインスタンスの代入 ..... pen = new Pencil()
```

理解度チェック



この章はたくさんの用語が出てきましたね。
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

- ☒ クラス
- ☒ オブジェクト
- ☒ クラスインスタンス
- ☒ オブジェクト変数
- ☒ メンバ
- ☒ メンバ変数, メンバメソッド
- ☒ メソッド
- ☒ ローカル変数
- ☒ 実引数
- ☒ 仮引数
- ☒ オーバロード
- ☒ コンストラクタ
- ☒ this
- ☒ 修飾子
- ☒ インスタンス変数, インスタンスメソッド
- ☒ クラス変数, クラスメソッド
- ☒ オブジェクトの配列

問7 - 3 次のプログラムの説明を読んで設問に答えなさい。

【プログラムの説明】

Test クラスは、Pencil クラスのインスタンスを生成して、color と size に値を設定する。

```
1 class Pencil {  
2     String color;  
3     double size;  
4 }
```

Pencil.java

```
1 class Test {  
2     public static void main(String[] args) {  
3           
4         p.color = "black";  
5         p.size = 0.5;  
6         System.out.println();  
7         System.out.println();  
8     }  
9 }
```

Test.java

実行結果▶

COLOR:black
SIZE:0.5

設問1 プログラムの空欄に入る正しい答えを解答群から選びなさい。

aの解答群

- ア p = Pencil();
- イ Pencil p = new Pencil;
- ウ Pencil p = new Pencil();
- エ Pencil() p = new Pencil();

(a)

b, cの解答群

- ア "COLOR:" + color
- イ "COLOR:" + p.color
- ウ "COLOR:" + p.color()
- エ "SIZE:" + size
- オ "SIZE:" + p.size
- カ "SIZE:" + p.size()

(b)

(c)

演習問題① あみだくじ

(H25 春-FE 午後問 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〕

あみだくじの作成と結果の表示を行うプログラムである。

あみだくじとは、複数の平行に並ぶ縦線と、2 本の縦線だけに水平に接続する複数の横線から成るくじである。横線は縦線の上端及び下端には接続せず、縦線の同一箇所複数の横線が接続されることもない。くじをたどる手順は次のとおりである。

- (1) 1 本の縦線を選択し、上端から下方向にたどり始める。
- (2) 途中に、接続する横線があるときは必ず曲がり、もう一方の縦線までたどる。
- (3) 縦線に到達したら、また下方向にたどる。
- (4) 下端に到達するまで(2)と(3)を繰り返す。
- (5) 下端に到達したときの縦線の位置が、くじを引いた結果である。

あみだくじの例を図 1 に示す。図中の太線は、左端の縦線を選択したときにたどった経路である。

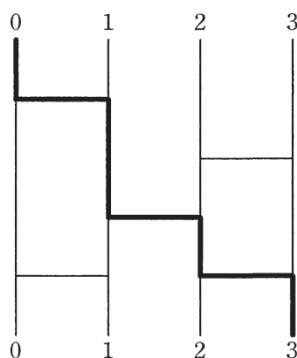


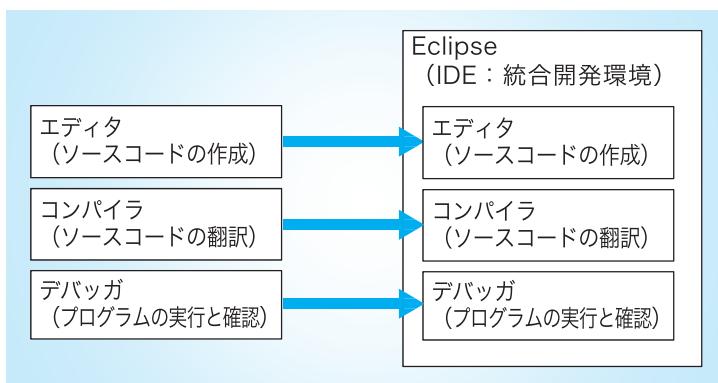
図 1 あみだくじの例

クラス GhostLeg は、あみだくじを作成し、作成したあみだくじをたどるプログラムである。

- (1) 縦線を左からの順番で保持する。左端を 0 番目とする。
- (2) 縦線の長さを 1.0 とし、上端の縦軸座標を 1.0、下端の縦軸座標を 0.0 とする。
- (3) 横線の両端の縦軸座標は等しく、0.0 よりも大きく 1.0 未満である。

ここでは、Java のプログラムコード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、GUI から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。



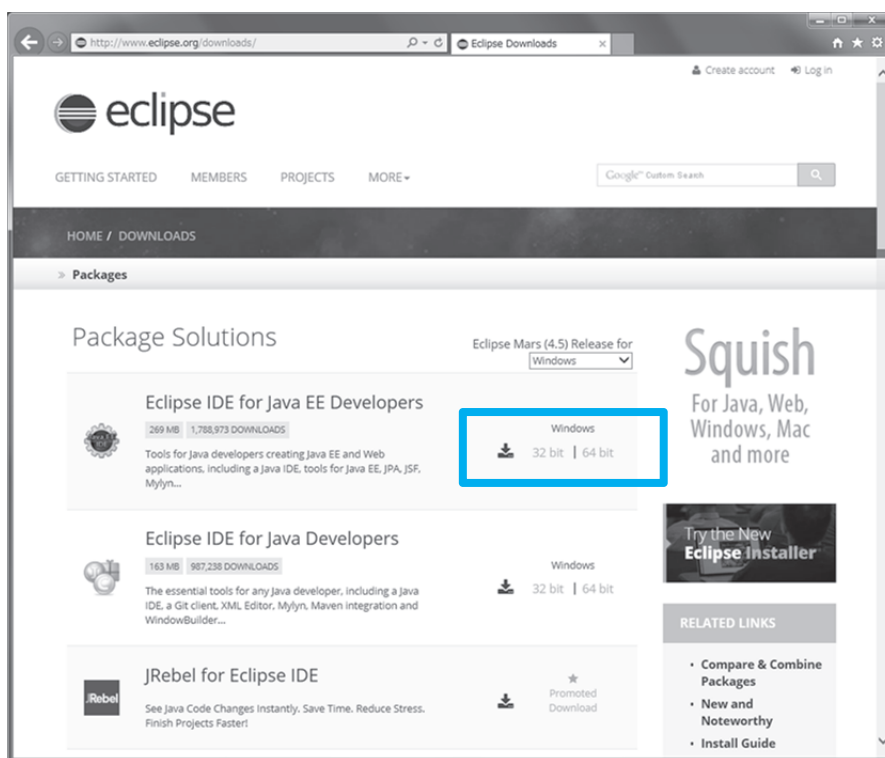
ここでは、Eclipse の入手と環境構築を行う手順を説明します。

1.1. Eclipse の入手先

Eclipse は、次の URL から入手可能です。

ダウンロード先 URL

<https://eclipse.org/downloads/>



本内容は、2015 年 8 月 20 日時点の最新情報に沿っています。

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, 405
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

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くて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。

演習問題 解答・解説

第 15 章

演習問題① あみだくじ

(H25 春-FE 午後問 11)

【解答】

[設問 1] ウ

[設問 2] a-イ, b-ウ, c-イ, d-ウ

[設問 3] e-ア, f-イ

【解説】

あみだくじの作成と結果の表示を行うプログラムの問題です。プログラムではコレクションフレームワークのインタフェース `List`, `SortedMap`, 及びクラス `ArrayList`, `TreeMap` が使われており、問題文と合わせて、問題冊子末尾の「Java プログラムで使用する API の説明」を確認する必要があります。

プログラムでは、クラス `GhostLeg` (`GhostLeg` はあみだくじの英訳) と、その入れ子クラスで、縦の線を表すクラス `VerticalLine` が定義されています。

設問 1 はプログラムの穴埋め問題になっていますが、本問ではプログラムの内容に関する記述の穴埋めとなっています。

設問 2 は、プログラムの穴埋め問題です。空欄 a, b はコンストラクタ部分の穴埋めです。Java の文法の基礎を押さえていれば容易に解答できます。空欄 c, d は、メソッド `trace` の処理です。問題文のくじをたどる手順の説明及びメソッド `trace` の説明を読み、プログラムでどのように実装されているか確認して解答します。

設問 3 は、クラス `GhostLeg` のメソッド `addHorizontalLine` に、例外を `throw` する処理を追加する場合における、挿入する適切な位置と、`throw` する条件を問う問題です。

[設問 1]

クラス `GhostLeg` の内部クラス `VerticalLine` において、横線の表現方法に関する記述の穴埋めです。〔プログラムの説明〕(5)に横線を追加するメソッド `addHorizontalLine` の説明があります。この説明と、プログラムの実装から読み取っていきます。メソッド `addHorizontalLine` のプログラムを見ます。まず、縦線を表すクラス `VerticalLine` のインスタンスを格納する `List` 型フィールド `verticalLines` に対し、`x1` を引数としてメソッド `get` を呼び出し、戻り値を変数 `v1` に代入しています。同様に、`x2` を引数としてメソッド `get` を呼び出し、戻り値を変数 `v2` に代入しています。〔プログラムの説明〕(5)①に「左から `x1` 番目の縦線と `x2` 番目の縦線」という記述があり、引数 `x1`, `x2` は、縦線の左から何番目であるかを表していることが分かります。メソッド `get` はインタフェース `List` のメソッドで、問題冊子末尾の「Java プログラムで使用する API の説明」から、引数で指定された要素を返すことが分かります。ここでは、`x1`, `x2` 番目の縦線を表すクラス `VerticalLine` のインスタンスが戻り値となります。次の処理の、`if` 文の条件式を見ます。すると〔プログラムの説明〕(5)①の「`x1` と `x2` が等しいか、2 本の縦線のいずれか又は両方に、既に縦軸座標 `y` で接続する横線があるときには追加しない」という説明の、反転した条件であることが分かります。つまり、