

目次

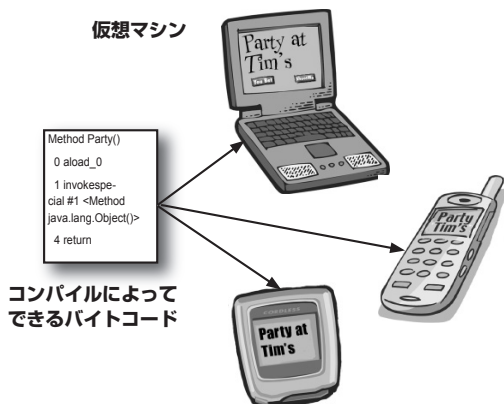
序章 この本の使い方 xxi

Javaを学ぶのは「脳」です。Javaに限らず、人間が何かを学ぶためには、脳を退屈させないことが重要です。脳は目の前に起きていることを重要でないとみなすと退屈し、もっと「重要なこと」を探し始めます。脳の構造は人間が野生に暮らしていた時のままなので、通常、肉食獣が近づいてくる、裸でスノーボードをするといった出来事以外、重要とは見なされません。Javaを学ぶには、脳に「Javaを学ばなければ命に関わる」と思わせる必要があります。

この本が対象としている読者	xxii
本当に真面目なJavaプログラムの本なのか?	xxiii
単に読むだけではなく「学習」できる本	xxiv
自分の思考について考える	xxv
この本で著者が工夫したこと	xxvi
この本を読むときに気を付けること	xxvii
用意するもの	xxviii
その他の注意事項	xxix
テクニカルエディタの紹介	xxx
謝辞	xxxi

1章 Javaの世界に飛び込もう ——ともかくまずは始めてみる 1

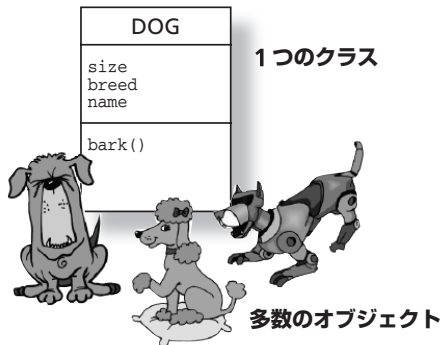
Javaは独自の特徴を持つプログラミング言語です。そのため、発表以来(発表当時のバージョンは1.02)、数多くのプログラマを惹きつけています。Javaには、文法がわかりやすい、オブジェクト指向である、メモリ管理が容易といった特徴もありますが、何と言っても重要なのは、その「移植性」です。一度書いてコンパイルしたコードがどこでも実行できるというのはJavaの最大の特徴と言っていいでしょう。1章ではこうしたJavaの特徴についておおまかに学びます。



Javaプログラムのできるまで	2
Javaプログラム作成の手順	3
Javaの歴史(概略)	4
Javaプログラム(ソースファイルの構造)	5
クラスの構造	6
mainメソッドを持つクラスの作成	7
mainメソッドの「中身」	8
分岐	11
本格的なJavaアプリケーション	12
PhraseOMaticプログラムの仕組み	15
特別座談会(コンパイラとJVM)	16
エクササイズとパズル	18

2章 クラスとオブジェクト —オブジェクト指向の国へ.....25

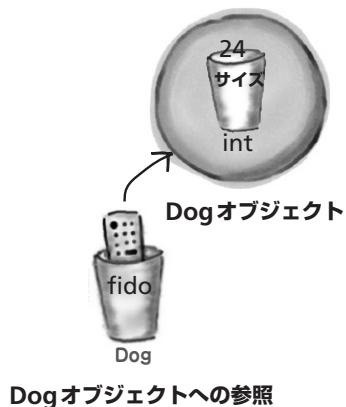
2章では、オブジェクト(クラス)について解説します。1章で紹介するJavaプログラムの例は、すべてmain()メソッドのみから構成されるものですが、これは正確には「オブジェクト指向」のプログラムとは言えません。オブジェクト指向でない言語のことを「手続き指向言語」と呼びますが、2章では、どのようにすれば手続き指向的ではない、オブジェクト指向的なプログラムが書けるのか、どうすればオブジェクトを作れるのかを具体的に学ぶことになります。Javaでのオブジェクト指向プログラミングの楽しさを知ってもらえればと思います。その他、この章では「クラス」、「オブジェクト」という言葉の違い、オブジェクト指向のメリットなどについても触れます。



対決！ オブジェクト指向対手続き指向	26
クラスを作る際に考えるべきこと	32
クラスとオブジェクトの違い	33
クラスの作成と使用	34
Movieクラスの作成とテスト	35
mainから外へ飛び出す	36
アプリケーションの実行	38
エクササイズとパズル	40

3章 プリミティブと参照 —変数の基礎知識.....47

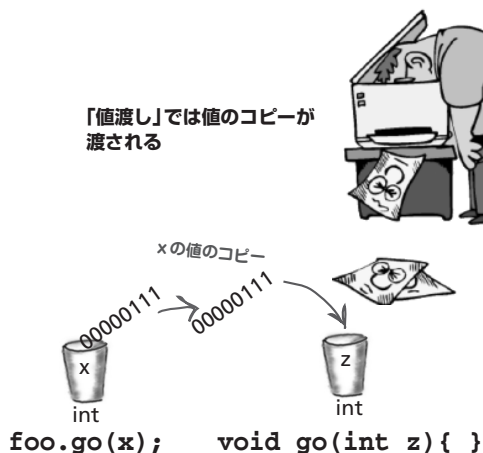
変数にはプリミティブ型、参照型という2つの種類があります。また、参照型の変数には、配列型、文字列(String)型、クラス型などの種類があります。こうした変数を使用すれば、単なる数値だけを使用するよりも複雑なプログラムが作れます。この章では、Javaにおける変数がどのようなもので、どのように使うのか、といったことを解説します。また、「ガーベジコレクション」についても詳しく説明します。



変数の宣言	48
「ダブルモカをください。intサイズで」	49
変数に値を代入する際の注意	50
名前付けのルール	51
オブジェクトを操作する	52
オブジェクトへの参照も変数の値である	53
Javaの真実(ゲスト：オブジェクトリファレンス)	54
オブジェクト、参照、ガーベジコレクション	55
オブジェクトの生と死	56
配列はコップを並べたトレイのようなもの……	57
Dog型の配列を作る	58
Dogオブジェクトの操作	59
エクササイズとパズル	61

4章 メソッドとインスタンス変数 —オブジェクトの「ふるまい」..... 69

ステートは、メソッドに影響を与え、メソッドはステートに影響を与えます。オブジェクトにはインスタンス変数(ステート)とメソッド(機能)があります。4章では、インスタンス変数とメソッドの関係がどのようになっているかということ学びます。インスタンス変数の値は、同じクラスのオブジェクトでも個々に変わります。そして、インスタンス変数の値は、メソッドの機能に影響を与えるのです。メソッドがインスタンス変数の値を利用して機能するとも言えます。たとえば、weight変数の値が「5」増えるごとに、makeNoise()メソッドが出す音が少しずつ大きくなる、というようなことが起きるのです。

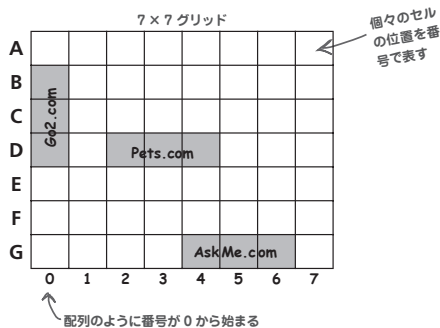


クラスにはインスタンス変数とメソッドがある	70
メソッドの機能に影響するインスタンス変数(ステート)	71
メソッドに値を渡す	72
メソッドは値を戻す	73
メソッドには、複数のパラメータを宣言することもできる	74
引数として指定された値の「渡され方」	75
パラメータ、戻り値の有効な使い方	77
カプセル化	78
データの保護	79
Javaの真実(ゲスト:オブジェクト)	79
Gooddogクラスをカプセル化する	80
配列でのオブジェクトの扱い	81
インスタンス変数の宣言と初期化	82
インスタンス変数とローカル変数の違い	83
変数の比較	84
エクササイズとパズル	86

5章 本格的なプログラムの作成 —本格的なプログラム(メソッド)を書く 93

この章では、実用に耐える本格的なプログラムを作成するのに必要ないくつかのことからについて解説します。実用的なプログラムを書くには、演算子、ループなどを使いこなせるようになる必要があります。ループはたとえば、乱数の生成や、Stringからintへのデータ変換などに使えます。この章では、ゲーム(バトルシップゲーム)のプログラムを例にとり、本格的なプログラムをゼロから作り、テストするという作業がどのようなものになるのか、ということについても学びます。

バトルシップゲーム



バトルシップゲームプログラムを作る	94
プログラムの設計	95
簡易版ゲームプログラム	96
クラスの作成	97
テストコードの作成	99
SimpleDotComクラスのテストコードを作成する	100
SimpleDotComクラスのテストコード	101
checkYoueself()メソッド	102
新しいテクニック	103
SimpleDotComクラスのコードとそのテストコードの最終形	104

SimpleDotComGame クラス (のmain()メソッド)の仮コード	106
SimpleDotComGame クラスのmain()メソッド	108
random()とgetUserInput()	109
GameHelper クラス	110
forループの基礎知識	112
ループの処理	113
String をintに変換する	114
プリミティブ型のキャスト	114
エクササイズとパズル	115

6章 Java APIの基礎

—Java ライブラリを使用する..... 121

Javaの開発環境にはあらかじめ数多くのクラスが用意されています。そうしたクラスの集まりをJavaライブラリ (Java API) と呼びます。作りたいプログラムに合うクラスをJavaライブラリで見つけることができれば、自分と同じようなクラスを改めて作る必要はなくなります。その分の時間や労力を別のことに振り向けることができるのです。Java APIを使用すれば、それぞれのプログラムの、他のプログラムにない特徴的な部分だけを書けばよくなります。通常は、プログラムのかなりの部分が、Java APIの中でも特に重要な「コアライブラリ」に用意された大量のクラスを「ブロック」のように使うだけでできあがります。

「java.utilパッケージにArrayListというクラスがあることがわかったのはよかったと思います。でも、自分の力でこのようなクラスを見つけるにはどうしたらいいのですか？」

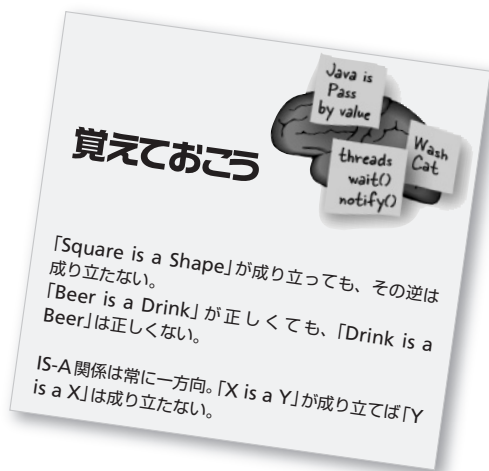
— Julia(31才)、職業ハンドモデル



簡易版バトルシップゲームプログラムのバグ	122
バグの原因は？	123
バグの修正方法	124
方法 1 はあまりに複雑	125
Java APIのクラスを使う	128
ArrayListクラスの使い方	129
Javaの真実(ゲスト: ArrayListクラス)	131
ArrayListと配列の比較	132
ArrayListを使用したバグの修正	134
ArrayListを使って修正したコード	135
「完全版」バトルシップゲームプログラムの作成	136
簡易版から完全版への変更ポイント	137
DotComBustクラスの処理	138
DotComBustクラスの仮コード	140
DotComBustクラスのリアルコード	142
DotComクラスのリアルコード	146
高度な条件判定式	147
GameHelper クラス	148
ライブラリ (Java API) の使用	150
APIの使い方	154
エクササイズとパズル	157

7章 継承とポリモーフィズム —オブジェクト指向をより深く知る..... 161

プログラムを作る際には、将来のことを考えて設計をしなければなりません。たとえば、自分以外の人が簡単に改造できるようにしておくことは重要です。また、「もうすぐ完成」という段階になつての突然の仕様変更などにも柔軟に対応できるようにすべきでしょう。この章では、そのために役立つ、オブジェクト指向言語の「ポリモーフィズム」、「継承」といったことがらについて解説します。ポリモーフィズムと継承について十分に理解すれば、より柔軟性のある、改造の容易なプログラム(クラス)を設計することができるようになります。



2章の復習	162
継承について理解する	164
動物シミュレーションプログラムを構成する	166
重複を防ぐのに役立つ継承	167
動物の「食べ方」はみな同じか	168
継承関係を再検討する	169
呼び出されているメソッドは？	171
継承関係を図に表す	172
IS-A関係とHAS-A関係	173
IS-A関係に関する注意事項	174
クラスの継承関係が正しいことを確認する方法	175
サブクラスが継承できるもの、できないもの	176
継承の適切な使用	177
継承は何のために行う？	178
オーバーライドに関するルール	186
メソッドのオーバーロード	187
エクササイズとパズル	188

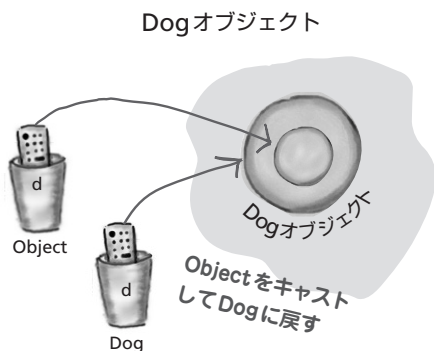
8章 インターフェースと抽象クラス —ポリモーフィズムの高度な利用方法..... 193

継承とポリモーフィズムがどのようなものかをただ学んだだけではあまり意味がありません。この章では、その継承やポリモーフィズムをより有効に利用するために必要な、「インターフェース」というものについて学びます。インターフェースをうまく利用すれば、単にクラスを継承しただけの場合より、さらに柔軟性、拡張性の高いプログラムが書けます。インターフェースは、「抽象クラス」と呼ばれる種類のクラスに似ています。抽象クラスというのは、インスタンス化ができないクラスのことです。この章を読めば、インターフェースのメリットがよくわかるはずです。

前章の例のおさらい	194
Animalオブジェクトはどんな動物を表す？	196
抽象クラスのインスタンスは作成できない	197
抽象クラスと具象クラス	198
抽象メソッド	199
抽象メソッドの本体はサブクラスで必ず作る	200
ポリモーフィズムの応用例	202



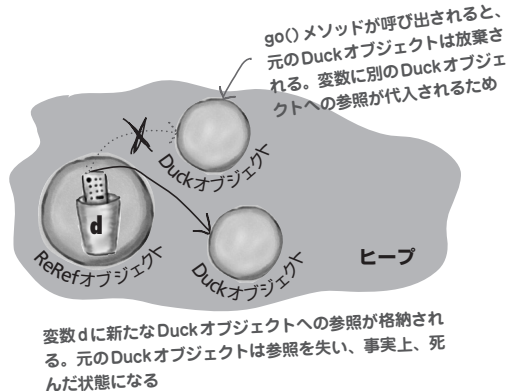
	クラスの異なるサブジェクトを混在させる	203
	Animalのサブクラス以外のオブジェクトを 格納するにはどうするか	204
	Objectクラスのメソッド	205
	オブジェクトは自分の型を「忘れる」ことがある	207
	DogオブジェクトがDogクラスのオブジェクトでは なくなるとは？	208
	元のクラスのメソッドは使えない	209
	すべてのクラスはObjectクラスの特長、機能を受け継ぐ	210
	「ポリモーフィズム」は元来「多くの形」を意味する	211
	利用できるメソッドを変更する	214
	ペットショッププログラムのための修正方法	215
	インターフェース	220
	Petインターフェースの作成とインプリメント	221
	エクササイズとパズル	225



9章 コンストラクタとガーベージコレクション —オブジェクトの生と死.....231

オブジェクトには、「生まれて、やがて死ぬ」というライフサイクルがあります。そのライフサイクルをどのようなものにするか決めるのはプログラマの仕事です。あるオブジェクトがいつ作られ、いつ削除されるか、ということを決めるのはプログラマの仕事です。ただし、オブジェクトを削除するためのコードをプログラマが直接書くわけではありません。オブジェクトを削除するのは、非情なガーベージコレクタです。この章では、オブジェクトをどのように作るか、作ったオブジェクトはどこに置かれるか、どうすればオブジェクトを効率的に消滅させることができるか、といったことについて解説します。具体的には、ヒープ、スタック、スコープ、コンストラクタ、スーパーコンストラクタなどについて詳しく触れることになります。

	オブジェクトはどこに存在するか	232
	メソッドはスタックに格納される	233
	ローカル変数が参照型のものである場合は？	234
	インスタンス変数の格納場所	235
	オブジェクト誕生の神秘	236
	コンストラクタのコード	238
	オブジェクトのインスタンス変数の設定	239
	コンストラクタを使用して オブジェクトのインスタンス変数の初期設定をする	240
	複数のコンストラクタを作る	241
	引数のないデフォルトコンストラクタは どのような場合でも必ず作られるのか	242
	コンストラクタの復習	245
	スーパークラスのコンストラクタとサブクラスの コンストラクタは同時に実行されるのか	246
	オブジェクトのコンストラクタにおける スーパークラスのコンストラクタの役割	247



Hippoオブジェクトを作るには、Animalオブジェクト、Objectオブジェクトとしての部分も必要	248
スーパークラスのコンストラクタを意図的に呼び出す場合は？	249
親より先に子供は産まれるか？	250
スーパークラスのコンストラクタに引数がある場合	251
コンストラクタの中でコンストラクタを呼び出すには？	252
オブジェクトの「寿命」はどのくらい？	254
参照型変数の場合	256
特別座談会(インスタンス変数とローカル変数)	260
エクササイズとパズル	262

10章 数値の扱いとスタティック変数、メソッド —数値の扱い..... 269

この章ではJavaにおける数値の扱いについて解説します。「数値の扱い」には、絶対値を求める、丸め(概算)を行う、2つの数値の大小を比較する、などさまざまなことがらが含まれますが、Java APIにはそうしたことを容易に行えるメソッドが用意されています。金額や日付といった数値を扱う際には、そのための独自の処理が必要です。通貨の単位を付けることや、時には小数点以下の桁数を定めることなども必要になるでしょう。日付には、国によって表示の仕方が異なるという問題があります。その他、String(文字列)から数値、あるいは数値からStringへの変換などについても知っておく必要があるでしょう。注意すべきなのは、そうしたことに使うメソッドの多くが「スタティック」なものであるということです。この章ではJavaにおいて変数(定数も含む)やメソッドがスタティックであるとはどういうことであるか、ということについても解説します。

子供
インスタンス1

スタティック変数：
アイスクリーム

子供
インスタンス2



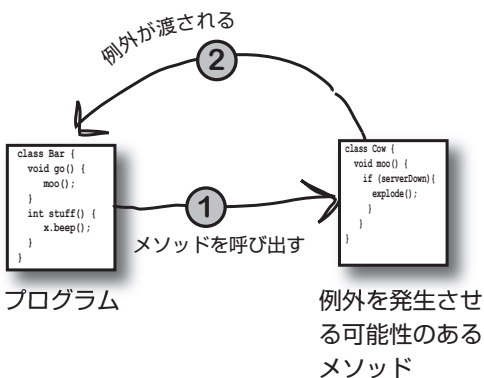
スタティック変数は、あるクラスのすべてのインスタンスによって共有される

Mathクラスのメソッド：「グローバル的」なメソッド	270
通常のメソッドとスタティックメソッドの違い	271
スタティックメソッドを持つクラスを作る	272
スタティックメソッドではスタティックでない変数(インスタンス変数)は利用できない	273
スタティックメソッドの中で通常のメソッドを呼び出すことはできない	274
スタティック変数： クラスのすべてのインスタンスで値が同じになる変数	275
スタティック変数の初期化	277
static finalと宣言した変数は定数のようになる	278
finalは必ずしもstaticと同時に使用しなくてよい	279
Mathクラスのメソッドの概要	282
プリミティブのラッピング	283
ラップに用意されているユーティリティメソッド	284
数値を数字に変換する	285
数値のフォーマット	286
特別座談会(インスタンス変数とスタティック変数)	289
エクササイズとパズル	291

11章 例外処理

—危険をともなうプログラムの作成 297

プログラムにはトラブルはつきものです。原因は、必要とされているファイルが実は存在しない、サーバがダウンしている……などさまざまです。プログラマがいかに優秀でも、あらゆる物事を完全にコントロールすることはできません。トラブルの起きそうな処理を行うプログラムを書く時には、あらかじめそのトラブルに対処するコードを組み込んでおく必要があります。ただ、問題は、トラブルが起きそうかどうかをどのように判断するか、ということです。トラブルに対処するコードをどこに入れるかということも知っておかねばなりません。この章では、JavaSound API を利用した音楽演奏プログラムを例に、そうしたことを解説していきます。



音楽演奏プログラムを作る	298
基礎的なことからについて学ぶ	299
まず必要なのは Sequencer オブジェクト	300
正しく動作しない恐れがある (他人が書いた)メソッドを呼び出す	301
Javaのメソッドでは、実行中に何か例外が起きたことを 呼び出し側のコードに知らせるために「例外」が発生 させる	302
try/catch ブロック	303
例外はすべて Exception 型を継承するクラスのオブジェクト	304
例外を発生させるメソッドを作る	305
try/catch ブロックでの処理の流れ	308
finally ブロック：例外が発生した場合も しなかった場合も行うべき処理	309
1つのメソッドが複数種の例外を発生させることもある	311
例外とポリモーフィズム	312
catch ブロックを複数作る場合は、 対応する例外クラスが下位のものから順に並べる	314
上位の例外クラスに対応する catch ブロックを 先にはできない	315
例外処理をしたくない時は……「回避！」	317
例外処理の回避は実は「先送り」	318
例外についてのまとめ	319
音楽演奏プログラム	320
プログラムキッチン(音楽演奏プログラム)	321
音楽演奏プログラムの基本的な作り方	322
簡単な音楽演奏プログラム	324
MidiEvent オブジェクト(MIDIデータ)の作成	325
ShortMessage オブジェクトの内容	326
setMessage()メソッド	327
楽器と音程を変更できるプログラム	328
12章以降で作成するプログラムの概要	329
エクササイズとパズル	330

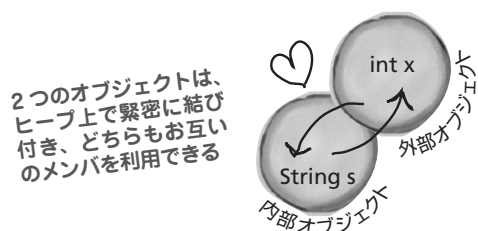
12章 GUIの基礎

—グラフィカルなインターフェース 335

GUI(Graphical User Interface)は今や、プログラム(アプリケーション)にとって不可欠なものと言えるでしょう。たとえ、作るのがWebブラウザによって利用されるサーバサイドのプログラムであったとしても、その点に変わりはありません。12、13章では、このGUIについて、そしてイベント処理、内部クラス、といったJava言語にとって重要な概念について解説していきます。12章ではまず、画面上にボタンを作る方法を学ぶことになります。その他、画面上に絵を描く方法、JPEG画像を表示する方法、アニメーションを作る方法なども学んでいきます。

```
class MyOuterClass {
    class MyInnerClass {
        void go() {
        }
    }
}
```

外部オブジェクトと内部オブジェクトは
緊密に結び付く



まずウィンドウを作る	336
ボタンを備えたウィンドウ(フレーム)を作る	337
前ページのプログラムの場合、	
ボタンをクリックしても何も起きない……	338
イベント処理	339
イベントリスナーとイベントソースのコミュニケーション	341
ActionEventの扱い	342
リスナー、ソース、イベント	343
再びGUIの話……	345
描画パネルをフレームに組み込む	346
paintComponent()メソッドの例	347
Graphics2DをGraphicsの代わりに使用する	348
描画する円の色をグラデーションにするためのコード	349
イベントと図形の描画	351
フレームに複数のウィジェットを組み込む	352
ボタンのクリックによって円の色を変化させる	353
ボタンを2つにする	354
内部クラス	358
内部オブジェクトを作る	360
Javaの真実(ゲスト:内部オブジェクト)	362
内部クラスをアニメーションに利用する	364
アニメーションプログラムのコード	366
プログラムキッチン(ミュージックビデオプログラム)	368
GUI部品以外で発生するイベントに対応するリスナーの作成	369
MidiEventオブジェクトの作成	370
ユーティリティメソッドmakeEvent()の使用例	371
第2段階: Controllerイベントのリスナーの作成、登録	372
第3段階: 音楽に合わせて図形を描画する	373
エクササイズとパズル	376

13章 Swingの基礎知識 —Swingを利用する.....381

Swingはさほど難しいものではありません。ウィジェットが画面上のどこに表示されるかを気にしなければ、の話ですが……。Swingを利用するコード自体は比較的簡単に書けるのですが、プログラムをコンパイルして実行してみると「あれ、なぜこれがこんなところに……」ということが起きやすいのです。実は、コードが簡単に書けるのも、ウィジェットが思いがけない位置に表示されるのも、その理由は「レイアウトマネージャ」というオブジェクトにあります。レイアウトマネージャは、少し手間をかければ思いどおりにコントロールすることができます。この章では、レイアウトマネージャをはじめ、Swingに関する基本的なことがらについて解説します。

east, west領域のコンポーネントは、横の長さを必要に応じてのばせる

north, south領域のコンポーネントは、縦の長さを必要に応じてのばせる



Swing コンポーネント	382
レイアウトマネージャ	383
レイアウトマネージャの機能	384
レイアウトマネージャの分類	385
さまざまなSwing コンポーネント	395
プログラム・キッチン (BeatBox プログラム)	400
エクササイズとパズル	406

14章 シリアライゼーションとファイルI/O —オブジェクトの保存.....411

オブジェクトは、ファイルに保存することができます。オブジェクトは大きく「機能」と「状態(ステート)」という要素に分けられるのですが、オブジェクトの機能は、そのオブジェクトが属するクラスのコードによって決まります。一方、「状態」はオブジェクトごとに異なります。この「状態」、つまりインスタンス変数の値さえ保存すれば、いつでも同じオブジェクトを再現できます。とはいえ、インスタンス変数の値を逐一確認し、それをファイルに保存するコードを書くのは大変です。そこで、Javaには、そうしたことを自動的に(オブジェクト指向言語らしく)行う「シリアライゼーション」という機能が用意されています。

シリアライズした状態



何かご質問は？

デシリアライズ後

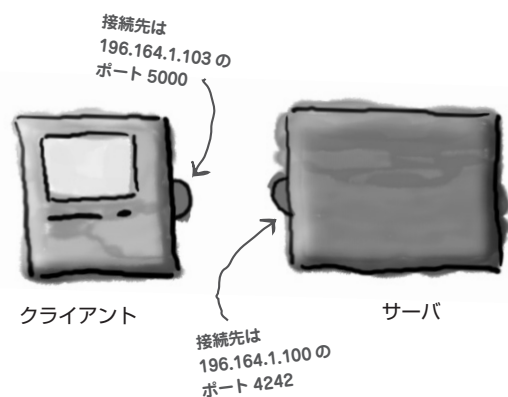


ドラムパターンを保存する	412
オブジェクトの保存方法	413
オブジェクトをシリアライズしてファイルに保存する	414
データの通り道となる「ストリーム」	415
シリアライゼーションとは	416
インスタンス変数の値はどのようなかたちで保存されるか	417
クラスにシリアライズの機能を持たせるには	
Serializable インターフェースをインプリメントする	419
インスタンス変数をシリアライズの対象から外すには	
「transient」宣言する	421
デシリアライゼーション：保存したオブジェクトの再現	423
デシリアライズはどのように行われるか	424
ゲームキャラクターオブジェクトのシリアライズと	
デシリアライズ	425

オブジェクトのデータを ブレンテキストファイルに書き込む	427
暗記カードプログラム	428
QuizCardBuilderクラスの概要	429
java.io.Fileクラス	432
バッファを使うメリット	433
テキストファイルからデータを読み込む	434
QuizCardPlayerクラスの概要	435
StringTokenizerの使用法	438
クラスのバージョン	440
バージョンID	441
プログラム・キッチン (BeatBoxプログラムにドラムパターンを保存する)	442
エクササイズとパズル	446

15章 ネットワークとスレッド —ネットワーク接続のためのプログラムを書く.....451

Javaではネットワーク接続のためのプログラムも簡単に書くことができます。複雑な、いわゆる「低水準の」処理は、java.netライブラリのクラスによってすべて自動的に行われるからです。その他、ネットワーク経由でのデータ送受信を単なるI/Oのように扱うことができる、というのもJavaの大きな特徴でしょう。ただ、同一マシン上のI/Oとは使用する接続ストリームが若干異なるだけです。この章では、ネットワーク経由でのデータ送受信、そして、複数の処理を同時に行うために必要な「スレッド」という概念について解説します。サンプルプログラムとして、ドラムパターンをサーバとの間でやりとりできる一種の「マルチスレッドチャットクライアント」も紹介します。



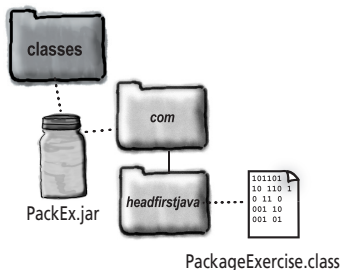
チャットクライアントとチャットサーバ	452
チャットプログラムの概要	453
接続、送信、受信	454
ネットワーク(ソケット)接続を確立する	455
TCPポートもアドレスの一部	456
ソケット接続経由でのデータの読み取りにも BufferedReaderが使用できる	458
ソケット接続経由でのデータ書き込みには PrintWriterを使用する	459
簡単なクライアントプログラム	460
クライアントプログラムのコード	461
サーバプログラムの作成	463
サーバプログラムのコード	464
簡単なチャットクライアントの作成	466
スレッドを表すThreadクラス	470
「マルチスレッド」の意味	471
スレッドを起動するためのコード	472
スレッドには「ジョブが必要」	473
Runnableインターフェースをインプリメントする	474

スレッドの状態の変化	475
通常は実行可能／実行中という	
2つの状態の間で切り替えが行われる	476
スレッドスケジューラ	477
スレッドをスリープさせる	481
Sleepメソッドを使用すれば	
スケジューラをある程度コントロールできる	482
ユーザースレッドを2つ作る	483
スレッドの問題点「競合」	484
Dr.Steveショー「離婚の危機? このカップルは修復可能か」	485
複数のスレッドによるオブジェクトの共有	486
「RyanとMonica」プログラム	487
makeWithdrawal()メソッドに	
同時にアクセスできるスレッドを1つに限定する	490
すべてのオブジェクトが備える「ロック」	491
「更新が無効になる」という問題	492
コードを実行してみる……	493
increment()メソッドをsynchronized宣言する	494
synchronized宣言の問題点	496
改良型チャットクライアント	498
チャットサーバのコード	500
プログラム・キッチン	
(BeatBoxプログラムの完成バージョン)	503
エクササイズとパズル	504

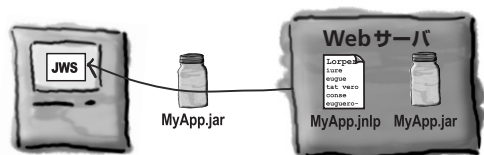
16章 プログラムの提供 —プログラムが完成したら…… 509

プログラムが完成したら……。何度も修正をし、さまざまな苦勞を重ねて(途中、「もう、やめたい」などと言ったりしながら)素晴らしいプログラムを完成させても、人に使ってもらえなければ意味がありません。しかし、どうすれば使ってもらえるのでしょうか。16章と17章では、プログラムの「パッケージング」などについて触れる他、プログラムの提供方法の種類(ローカル、リモート、その中間、など)などにも触れていきます。具体的には、実行可能JARファイル、Java Web Start、RMI、Java Web Startなどについて解説することになります。一見、難しそうですが、実際にはそれほどでもないので安心してください。

Webサーバ



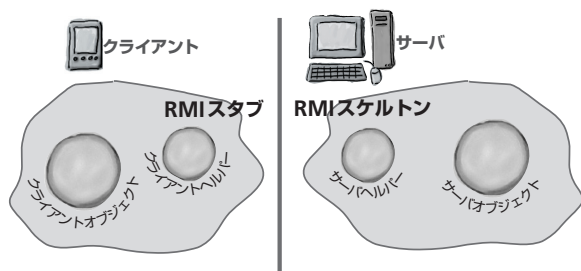
Javaプログラムの提供	510
プログラムができた。その後は……	511
ソースファイルとクラスファイルを分けて保存する	512
プログラムに必要なファイルをJARファイルにまとめる	513
JARファイルの実行	514
クラスをパッケージにまとめる	515
パッケージ名の重複を避ける	516
クラスをパッケージに所属させる	517



パッケージに属するクラスのコンパイル、実行	518
-dオプションの優れた機能	519
パッケージに属するクラスを 実行可能JARファイルにまとめる	520
JARファイルの内容確認、内容の抽出	521
Java Web Start	525
.jnlpファイル	527
JWSを利用するプログラムを提供する手順	528
エクササイズとパズル	529

17章 RMIによるプログラムの提供 —分散コンピューティング 535

プログラムの構成要素は一か所にまとまっていなくてはならないわけではありません。すべてが同じコンピュータ上の同じヒープにあり、使用するJVMも1つで済めば、これほど楽なことはありません。しかし、それがいつでも可能とは限らないのです。また、そうすべきでないこともあります。たとえば、プログラムは大変な処理能力を必要とするものなのに、ユーザーが携帯用の小型デバイスしか持っていないとしたらどうでしょうか。セキュリティ上の理由から、データベースアクセスをサーバ上のコードにのみ許可したい、ということもあるでしょう。この章では、主に、RMI(Remote Method Invocation)という技術について解説します。その他、サーブレットや、EJB(Enterprise Java Beans)、Jiniといった技術にも簡単に触れます。

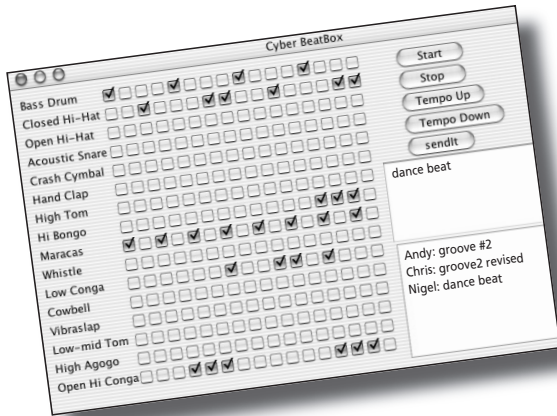


メソッドの呼び出しは 通常、同一のヒープ上のオブジェクト間で行われる	536
他のマシン上のオブジェクトのメソッドを呼び出すには？	537
小型デバイス上のオブジェクトAからパソコン上の オブジェクトBのメソッドを呼び出すには？	538
RMIを設計してみる	539
ヘルパーの役割	540
RMIのクライアントヘルパー、サーバヘルパー	542
クライアントがスタブを取得するには	548
スタブのクラスファイルをクライアントに提供するには	549
クライアントマシン、サーバマシンに必要なクラスファイル	550
RMIは実際にどのように応用されているのか？	552
サーブレット	553
サーブレットの作成・実行	554
簡単なサーブレットの例	555
PhraseOMaticプログラムを利用した サーブレットを作ってみる	557
サーブレットで利用しやすいよう改良した PhraseOMaticクラスのコード	558
EJB(Enterprise JavaBeans)：大規模システムのためのRMI	559
Jiniとは	560
サービスの自動検索機能	561

ルックアップサービスの自動修復機能	563
ユニバーサルサービスブラウザ	564
ユニバーサルサービスブラウザの動き	565

付録A BeatBoxプログラム完全版 577

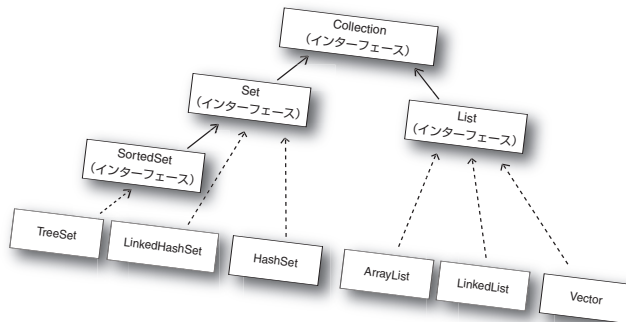
BeatBoxプログラム完成版。付録Aには、本書の後半で紹介したBeatBoxプログラム(ドラムマシンプログラム)の完成版のコードを載せてあります。このプログラムがあればロックスターも夢ではありません。



BeatBoxプログラム完成版(クライアント)のコード	578
BeatBoxサーバプログラム完成版	585

付録B 本文で触れなかった重要事項ベスト 10 587

本文で触れなかった重要事項ベスト 10。本文で学んだことだけではまだ一人前のJavaプログラマとは言えません。付録Bでは、本文では触れなかった重要なことについて解説します。これで、この本での勉強は本当に終わりです。



索引	607
訳者あとがき	615