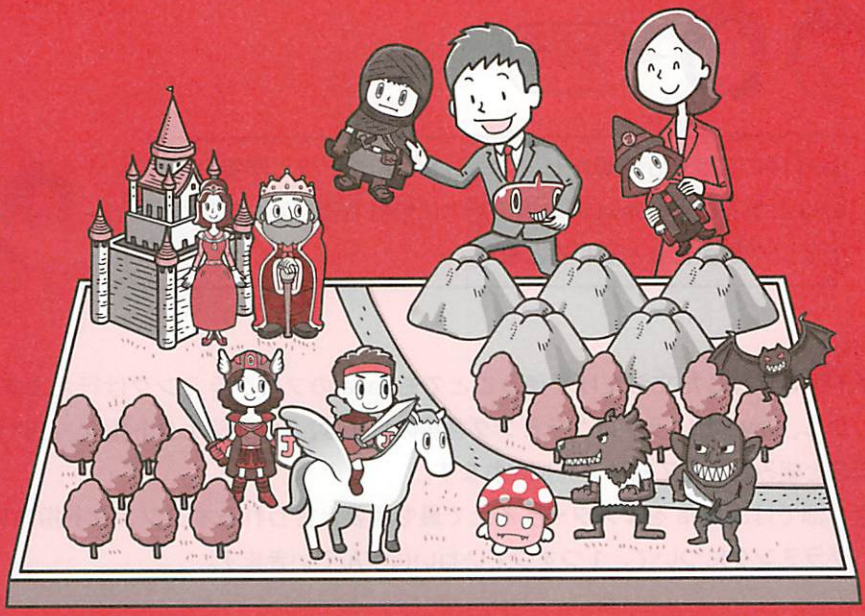
第 II 部

すっきり納得 オブジェクト指向



Java の本当のおもしろさ

ここまでお疲れさま。どうだい、Java とはいい友だちになれそうかい？

はい！　…いや、実はちょっと迷うこともあるけど…。でも基本的には main メソッドの中に処理を書いていけばいいんですよね？

専門家も使う Java って、超ムズかしくて、完全に意味不明なんじゃないかって想像してたのに、安心したっていうか、ちょっと拍子抜けっていうか…。

ははは、それはよかった。でも、2 人はまだ Java の本当の姿、本当の魅力にほとんど触れていないんだよ？

えっ？

Java はここからが大事だし、おもしろいんだ。少し学習のレベルは上がるけど、Java の本当の魅力に触れれば、2 人のプログラミング観がきっと変わると思う。

第 I 部で学習した文法だけを使うことでも Java のプログラミングは行えます。しかし、Java というプログラミング言語の真価は、「オブジェクト指向」という概念と組み合わせて初めて発揮されます。

第 II 部では、Java をマスターする上で最も重要と言われるオブジェクト指向プログラミングについて、1 つずつていねいに学んでいきます。

第7章

オブジェクト指向をはじめよう

この第7章からはJavaの根幹となる「オブジェクト指向」を学んでいきます。オブジェクト指向をラクに理解できるかどうかは、「学び方のコツを知っているか」「準備体操をしているか」によって変わってきます。

そこで本章では、オブジェクト指向に本格的に取り組む前段階として、「その全体像と学び方」を多くのイラストを交えてやさしく解説していきます。

CONTENTS

7.1 オブジェクト指向を学ぶ理由

7.1.1 ソフトウェア開発の新たな課題

菅原さん……助けてください！（泣）

どうしたんだい？ 先週までは、『ボクはもうどんな大きなプログラムだって書けるんです！』と自信満々だったじゃないか。

はい、文法はすべてわかりますし、必要な命令も自分で調べられます。「RPG:スッキリ魔王征伐」の開発も順調でしたが、ソースコードが 400 行を超えたあたりで、何というか……どこに何の処理を書いたのかわからなくなって、自分でも頭が混乱してしまいました。機能を修正しようとするたびに「頭がパンクしそうになって、開発が進まない」んです。

『わからないことはないのに書けない』んだね。それじゃ、その悩みを解決するための「オブジェクト指向」について学んでいこうか。

第 I 部を通して、私たちは Java の基本文法をひととおり学習しました。また、API リファレンスを調べることで、Java に用意されたさまざまな命令を利用できるようになりました。つまり、一部の特殊な例を除けば、理論上どのように大きなプログラムでも書くことができるようになったはずです。

しかし、実際に本格的なプログラムを開発し始めると、湊くんのように「ソースコードが長く複雑になりすぎて、開発者自身が把握しきれなくなる」という課題に直面します。

この課題は、第6章で学習した方法を用いてソースコードを複数のクラスやメソッドに分割することで、多少は緩和されます。ですが、それでもソースコードが数千行、数万行を越えると、結局は同じ問題に悩まされることでしょう。

実は、この課題は40年ほど前に世界中のプログラマたちがぶつかった壁そのものです。1970年代、さまざまなプログラム言語が登場し、これにより大規模なプログラムの記述が理論上は可能となっていました。

ですが、いざ大きなプログラムを記述しようとすると、人間の頭が追いつかず、開発に時間がかかったり、完成しても不具合だらけのプログラムになったりしてしまうことが少なくありませんでした。なぜなら、その原因はコンピュータの演算性能や記憶容量ではなく、図7-1にあるように**人間がプログラム開発のボトルネックになってしまっている**ことにあったからです。

図7-1　人間自身が巨大なプログラムを把握できなくなり「ボトルネック」になってしまった

7.1.2 オブジェクト指向プログラミングをマスターしよう

そこで、誕生したのが**オブジェクト指向プログラミング**(Object Oriented Programing = OOP)という考え方です。この考え方に従ってプログラムを書くと、前述のような課題に悩むことなく、**大規模なプログラムもラクに開発できる**ようになるのです。

第Ⅱ部(第7章〜第13章)では、このオブジェクト指向の考え方を学んでいきます。オブジェクト指向をマスターすることで、仕事などで携わる大規模で複雑なプログラムも、スッキリと開発できるようになるでしょう。

オブジェクト指向の目的

「人間が把握しきれない複雑さ」を克服するためにオブジェクト指向は生まれた。

7.1.3 オブジェクト指向を学ぶコツ

あぁ！もうダメだぁ！ ボク「オブジェクト指向」って聞いたことがあるんです。とにかく難しくて、挫折してしまう人が多いって……。

その半分は正解だが、半分は間違っているよ。確かにオブジェクト指向をマスターできずに挫折する人はいるし、その本質を理解せずに使っている人もいる。ただ、「**オブジェクト指向の難しさは『学び方』によって大きく変わる**」んだ。

ということは、うまく勉強すれば難しさを感じずにマスターできるってことですか？

そう。君たちの先輩には「オブジェクト指向って、思っていたより簡単ですね」と話す人もいたよ。しかも、その人はプログラミングの経験がなくて、文系出身の人だったんだ。

オブジェクト指向を学ぶ人々からは、「学習が難しい」あるいは「挫折してしまった」、「理解できていない部分もあるが、なんとなく使っている」という声を聞くことも少なくありません。しかし安心してください。「Java の基本文法は理解できたのにオブジェクト指向はとても難しく感じる」という人には共通点があって、彼らは次のような**たった１つの簡単な前提知識を知らずに、いきなりオブジェクト指向の学習を始めてしまっているだけ**なのです。

基本文法とオブジェクト指向とでは、そもそも学ぶものも学び方もまったく違う

本書の第Ⅰ部で学んだ内容は、「こう書けば、こう動く」「こう書かないと正しく動かない」という文法や記述のルールでした。書き方の「**正解自体を学ぶ**」のですから、単に丸暗記して習ったとおりに使えばよいだけです。算数にたとえれば計算問題（足し算や引き算の方法を知る）、料理でいうならレシピ（カレーの材料と作る手順を知る）のようなものです。

一方、この第Ⅱ部では、「プログラムを作るときには、このようにプログラム全体を捉えて、こういう文法の組み合わせ方をすると、ラクにプログラムを作れる」という考え方を学びます。つまり「**正解に辿り着くための考え方**」**を学ぶ**のです。算数ならば文章題（どう問題を捉えてどう計算していくと答えが出るかという考え方を学ぶ）、料理ならもてなし方（どのような状況で、どのような料理を組み合わせて出せば喜ばれるかという考え方を学ぶ）に相当するでしょう。

このように、第Ⅱ部では「捉え方」「考え方」を学んでいきます（図 7-2）。つまり、**オブジェクト指向の学習においては、理解することやイメージすることを、より大事にする必要がある**のです。

本書の部・学ぶ趣旨	第Ⅰ部 基本文法 （定義・ルールなど）	第Ⅱ部 オブジェクト指向 （考え方）
算数にたとえると？	計算問題を学ぶ （四則演算の意味、書き方など）	文章題を学ぶ （文章の内容を、どう捉えるか）
料理にたとえると？	料理のレシピを学ぶ （いつ、何の材料を、どれだけ入れるか）	料理での「もてなし方」を学ぶ （お祝い事を、どのように捉え、料理の食材で表現していけばよいか）

図 7-2　本書の第Ⅰ部では Java の基本文法、第Ⅱ部では「オブジェクト指向の考え方」を学ぶ

丸暗記やひたすら練習みたいな「計算問題の学び方」でがんばっても、文章題はなかなか解けるようにならなくて当然ですね。

そう。オブジェクト指向本来の「考え方」を理解する前に、いきなり小難しい文法を頭に詰め込もうとするから挫折しちゃうんだよ。

オブジェクト指向を学ぶにあたり、まず必要なことは、その「考え方」の概要や全体像を理解することです。この第7章はそのためだけにあります。

しかし、オブジェクト指向という「考え方」には形がなく、把握しづらいところがあるのも事実です。これは「出世する人の仕事に対する考え方」や「恋も仕事もうまくいく人の考え方」などと似ていて、学んですぐに隅々まで100%理解できるようなものではありません。

完璧主義の人にはやや気持ちが悪いかもしれませんが、初めは「こういう感じで考えるんだな」という、多少あいまいでぼんやりとした理解で構いません。繰り返し学習したり、たくさんプログラムを組んだりしていくうちに、徐々に明瞭なイメージになっていくでしょう。

むしろ、一部の章だけを切り出して完璧に理解しようとしても、思ったように理解は進みません。なぜなら、私たちが今から学ぼうとしているオブジェクト指向には、さまざまな「発想」「着眼点」「テクニック」そして「関連する文法」が含まれており、それらは相互に、しかも密接に関係しているからです。第Ⅱ部の学習は、図7-3のように少しずつ繰り返し学んでいくのがコツなのです。

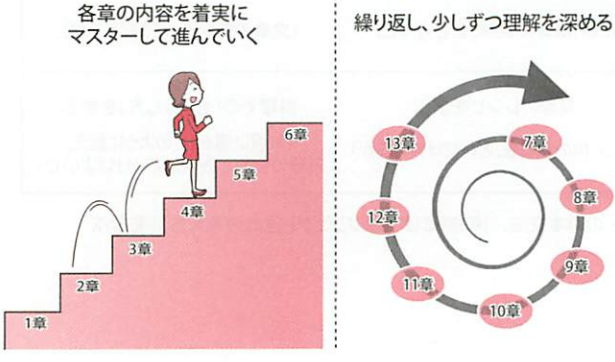


図7-3　Javaの基本文法（第Ⅰ部）は着実に、オブジェクト指向（第Ⅱ部）は繰り返し学んで徐々に理解しよう

7.2 オブジェクト指向の定義と効果

7.2.1 オブジェクト指向の定義

オブジェクト指向を初めて学ぶ人が最初にぶつかる壁があります。それは「オブジェクト指向とは何か？」という問いに明確な答えが得られないことです。

先輩プログラマに質問しても、それぞれ答えが異なり、あいまいで長くてわかりづらい説明をされることもあるでしょう(図 7-4)。正確な答えを知ろうとして、教科書的な定義を持ち出されると、ますます混乱していきます。

図 7-4 「オブジェクト指向とは」という質問が一番難しい?

オブジェクト指向を初めて学ぶ私たちには、小難しい学問的な定義よりも、以下のようなシンプルな定義のほうが理解しやすいでしょう。

オブジェクト指向の定義

オブジェクト指向とはソフトウェアを開発するときに用いる部品化の考え方のこと。

「部品化」という言葉は第6章で学びましたね。1つの巨大なmainメソッドを作る代わりに、複数のメソッドやクラス（ソースファイル）に分割したり、複数の部品を組み合わせたりして、全体として1つのプログラムを作る手法のことでした（図7-5）。

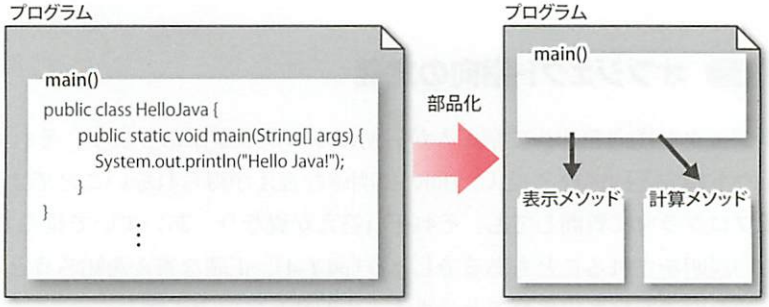


図7-5　巨大な1つのmainメソッドを複数のメソッドやクラスに分割してプログラムを作る

しかし第6章では、「どういう基準で部品を分けるべきか」という**部品化のルールについては触れていません**。プログラムを開発していて、ある程度大きくなって読みづらくなってきたから分割する、なんとなく意味が似ている単位で分割するなど、その程度の主観的な判断にすぎず、「**こう分割したほうが良いプログラムが作れるという確固たる根拠に基づいた部品化**」ではありませんでした。

しかし、これから学習するオブジェクト指向という考え方に沿って1つのソフトウェアを複数に部品化すると、プログラムが把握しやすくなり、**「人間の頭が追いつかない状況」を避けることができる**ようになるのです。

7.2.2 オブジェクト指向のメリット

先輩、私、オブジェクト指向の目的やメリットは本で読んだことがあります。大規模なプログラムを作る際に、「柔軟性が上がる」「再利用性が上がる」「保守性が上がる」と書いてありました。賢い部品化で「複雑なプログラムを人間が把握しやすくなる」から、その結果、柔軟で保守・再利用しやすいプログラムが作れるんですね。

オブジェクト指向とは「何のため」にある考え方なのでしょうか。利用すると「何が嬉しい」のでしょうか。

その根底にある目的は、「**人間に把握できるプログラム開発を実現する**」というものです。この考え方を利用した「賢い部品化」を行うと、把握しやすさの向上のほかにも次のようなメリットが生まれるといわれます。

- **プログラムを容易に変更しやすくなる（柔軟性・保守性の向上）**
- **プログラムの一部を簡単に転用できる（再利用性の向上）**

しかし、Java を学び始めたばかりのみなさんは、このメリットを**一度忘れてください**。これら柔軟性・保守性・再利用性のメリットは、厳しい予算や納期の中で大規模なプログラムを何度も開発・修正するようになって初めて実感できるものです。

つまり、本書を学習しているみなさんは、先ほど述べた「保守性」や「再利用性」の必要性や大切さを心の底から理解し、**「保守性や再利用性のために、オブジェクト指向を絶対マスターするぞ！」とは思えないはず**です。

オブジェクト指向は「一度マスターしてしまえば、二度と手放せないぐらい便利な一生モノの技術」です。しかし、「鼻歌交じりにナメてかかって理解できる」ほど生やさしいものではありません。これからオブジェクト指向を学ぶみなさんは、今この章で**「マスターできたら嬉しい！」「絶対マスターしたい！」と心底思えるようなメリットをイメージしなければなりません**。

今の私たちが抱くべき「オブジェクト指向のメリット」は、次のひとことで十分です。

オブジェクト指向を用いるメリット

「ラクして、楽しく、良いもの」を作れる

同じ開発をするのなら、毎晩のように徹夜をしたり休日出勤したりして開発するのと、毎日定時に帰れるよう効率的に開発するのと、どちらがよいでしょうか？

頭をかきむしりながら画面とニラメッコして開発するのと、まるで絵でも描くように創造力を発揮しながらプログラムを作っていくのでは、どちらがよいでしょうか？

聞くまでもありませんね。みなさんはこの第Ⅱ部の内容をマスターすることによって「ラクして、楽しく、良いものを」作れるようになるのです。

「考え方」「捉え方」の違いが世界を変えることもある

私たちは日常生活でも「考え方」を変えることによって生活を便利にしています。たとえば「ゼロ」や「マイナス（負数）」の考え方です。

天気予報の気温などでは「○○地方の最低気温はマイナス 10 度」などと表記されます。日常では当然のように見かけるマイナスの数字ですが、原始時代の人類にとって、自然界の何かを数える場合は常に 1 以上でした（そのため 1 以上の整数を「自然数」といいます）。

「なにもないこと（ゼロ）」や「なさすぎること（負数）」を数字として扱う「考え方」を導入したのは、人類の長い歴史の中でもわずか 1400 年ほど前からです。ゼロや負の数という考え方を導入する前と後で世界が変わったわけではありません。**人間が「考え方」「捉え方」「概念」を変えただけ**なのです。

しかし、この新たな「考え方」の導入によって、人間は世界のさまざまなものを数字として把握、制御することが可能になり、IT のみならず社会生活を発展させる基盤となっています。

7.3 オブジェクト指向の全体像と本質

7.3.1 オブジェクト指向と現実世界

でも菅原さん、たかが「ある考え方」を利用するだけで、なぜラクして楽しく良いものが作れてしまうんでしょうか？ その「なぜ」がわからないとスッキリしません。

そうだね。それはオブジェクト指向の全体像と、その本質を知れば理解できるよ。

オブジェクト指向という考え方を採用し、部品化をするだけで、「ラクして楽しく良いもの」が実現できてしまうなんて、何だか悪徳商法の宣伝文句のような印象を持たれる方もいるかもしれません。この節では、その根拠となるオブジェクト指向の本質を探っていきましょう。

そもそも、我々は何のためにプログラムを開発するのか、考えたことはあるかな？

普段あまり考えることはないかもしれませんが、私たちが開発するプログラムやシステムは、**「現実世界における何らかの活動を自動化するためのもの」**です。さらにわかりやすく言うなら、「人がやってきたことを機械にやらせて人がラクをするためのもの」と言い換えることもできるでしょう。

たとえば、銀行のATMシステム（ATMの機械と、その中で動いているプログラム）を想像してください（次ページ図7-6）。江戸時代のように、コンピュータがなかった頃には手作業で行っていた作業（依頼の受け付け、残高の検査、引き出し、記帳、受け渡し）を、機械に肩代わりさせていると考えることができます。

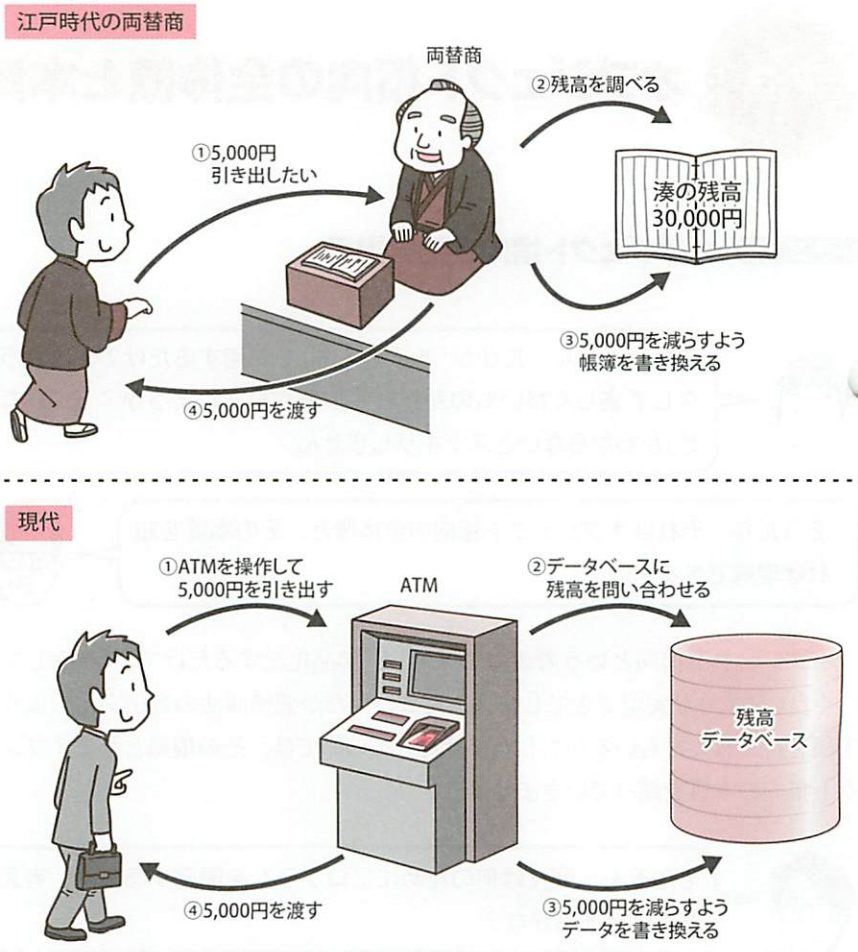


図 7-6　ATM システムは、江戸時代における「両替商」の代わりであり、人間がやっている作業をプログラムに置き換えたもの

同様に、現実世界での「切符販売の駅員の仕事」を機械とプログラムに置き換えたものが、「自動券売機とそのプログラム」です。現実世界での「みんなに公開する日記帳」をプログラムにしたものが「ブログ」です。また、本書の湊くんが作っている RPG も現実世界ではありませんが、ファンタジーの世界のさまざまな人物の冒険や戦いをコンピュータ上で実現しているものです。

このようにプログラムやシステムは、現実世界のある活動を人間に代わって機械にやらせるために作られるものであって、**現実世界とは無関係に単独で存在しているものは、ほとんどありません。**

7.3.2 手続き型プログラミングとの違い

第Ⅰ部で私たちが行ってきた従来のプログラミング手法は、**手続き型プログラミング**(procedural programming)と呼ばれています。プログラマは頭を捻り、コンピュータがどのように動けばよいかという手順を考え、プログラムの先頭から順番に命令として記述していく方法です。

一方、オブジェクト指向で開発を行う場合、プログラマはいきなりコードを書き始めることはしません。まず、プログラムで実現しようとする部分の「現実世界」を観察します。たとえば銀行振込の手続きをプログラム化する際には、それを観察して図 7-7 のようなイメージ図(設計図)を描きます。

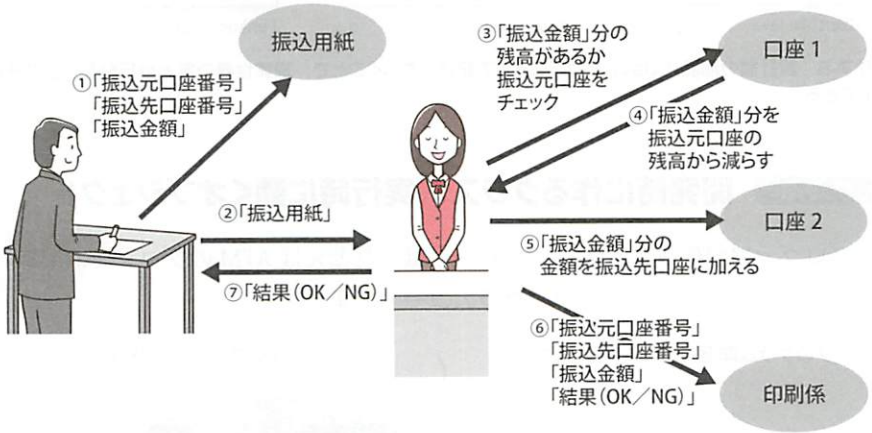


図 7-7　オブジェクト指向では、まず現実世界を観察し、それを設計図に落とし込んでいく

ここで着目してほしいのは、この設計図は IT の知識がない一般の人に見せても理解できる「現実世界における銀行取引の構図そのもの」である点です。オブジェクト指向の開発では、設計図の中の登場人物や物の 1 つひとつを部品と捉え、それを「クラス」という Java における部品で記述していくのです(次ページ図 7-8)。

オブジェクト指向による部品化のルール

現実世界に出てくる登場人物の単位で、プログラムをクラスに分割する。

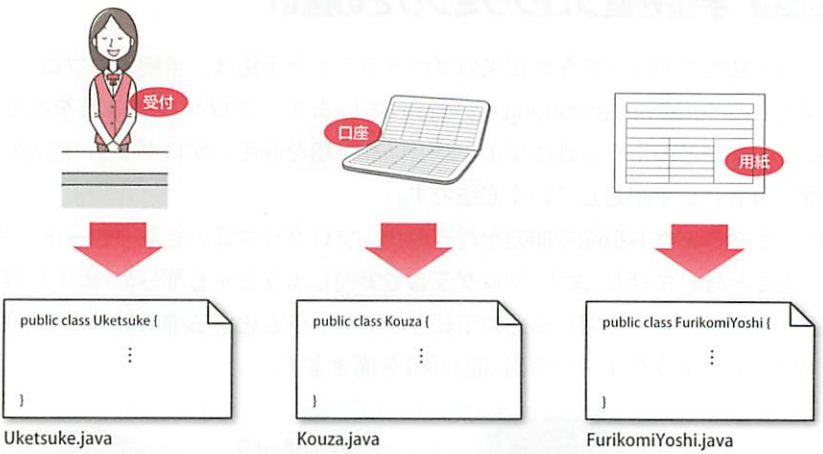


図 7-8　設計図の部品を Java のクラスとして記述していくことで、現実世界の定義や行動をプログラム化できる

7.3.3 開発時に作るクラス、実行時に動くオブジェクト

プログラマが作成する部品（クラス）とは、たとえば ATM のプログラムであれば「Uketsuke.java」のようなプログラムコードです。

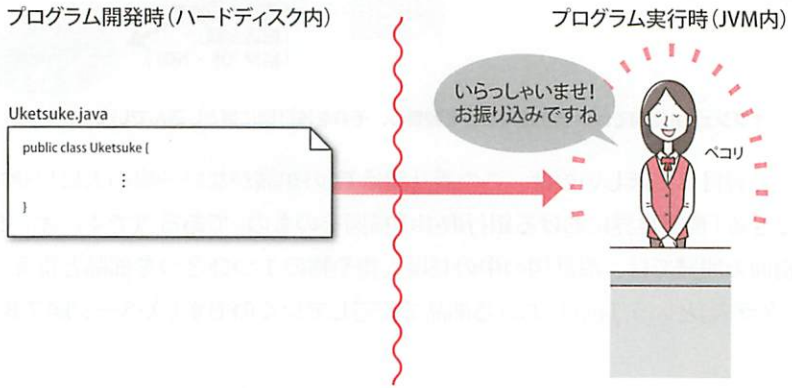


図 7-9　Java で作られた Uketsuke クラスから、JVM 内に「仮想的な受付係」が生み出され、現実世界の動作をまねて動き出す

開発時に作られたクラスは、プログラムとして実行される際、それぞれ「仮想的な登場人物」のオブジェクトとして JVM の中にその存在が生み出されます（図 7-9）。

そして図 7-10 のような「仮想的な口座」「仮想的な受付」「仮想的な印刷担当」などが、コンピュータ（JVM）という「電子的な仮想世界」の中に作られ、現実世界をそっくりまねた「Java 仮想世界」とでもいえるような世界を形成します。

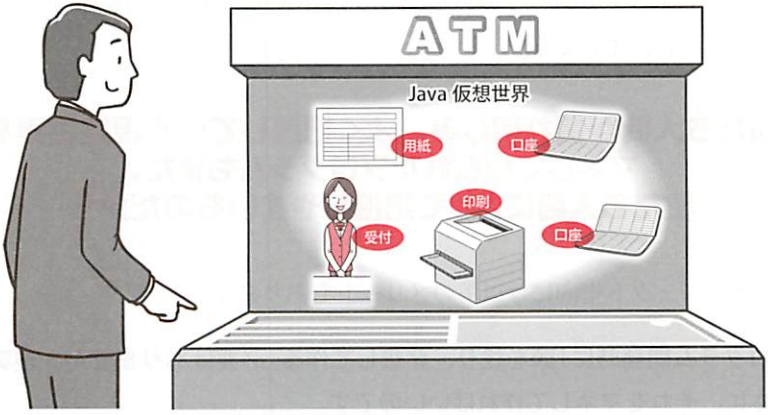


図 7-10　ATM は銀行の各種業務をプログラムに置き換えた「電子的な仮想世界」

7.3.4 オブジェクト指向におけるプログラマの役割

オブジェクト指向プログラミングにおいて、プログラマはまるで「Java 仮想世界における神様」のような存在です。なぜなら仮想世界にどんな登場人物や物を生み出し、それらをどのように連携させるかを決め、それぞれの部品を作っていく立場だからです。

たとえば国内の複数の倉庫に在庫のある書籍をネットで販売するプログラムを作るなら、本の販売業務を観察した上で、「必要な物（オブジェクト）は『本』『倉庫』『顧客』『購入記録』…」と判断し、図 7-7 のような設計図を書いていきます。

手続き型のプログラマのように「コンピュータが一行一行、何を実行するかという手順を定める」のではなく、「オブジェクトをどう作るか、どのように連携させるか」を第一に意識しながら開発していきます。このことが「オブジェクト指向プログラミング」という名前の由来になっています。

「～指向」というのは、「～を大切にした」「～を中心に据えた」という意味だよ。

7.3.5 オブジェクト指向の本質

それでは、「なぜオブジェクト指向の考え方を使うと、大規模で複雑なプログラムも把握しやすくなり、その結果、ラクして楽しく良いものを作れるのか？」という問いに答えましょう。

**私たち人間が慣れ親しみ、よく把握している現実世界を
マネして作られたプログラムもまた、
私たち人間にとって把握しやすいものだから**

さらにオブジェクト指向には以下のメリットもあります。

- プログラム開発時に「頭を捻り、発想して作る」必要はありません。現実をお手本に、それをマネして作ればいいのです。
- 現実世界の登場人物に変化があった場合、対応する部品（クラス）を修正、交換すれば簡単にプログラムを修正できます。

このようなメリットは、「現実世界をマネる」からこそ生まれてきます。つまり、現実世界の登場人物たちを、コンピュータの中の仮想世界にオブジェクトとして再現し、現実世界と同じように連携して動くようにプログラムを作ることこそがオブジェクト指向の本質なのです。

いくらクラスをたくさん使っていても、後述する「継承」などの機能を利用していても、オブジェクト指向の本質を正しく理解しておかなければ、「人間が把握しにくく、修正や交換が困難なプログラム」しか作成できないでしょう。

オブジェクト指向の本質

現実世界の登場人物とそのふるまいを、コンピュータ内の仮想世界で再現する。

7.4 オブジェクトと責務

7.4.1 サッカーで考えるオブジェクト指向

手続き型とオブジェクト指向との違い、そして「オブジェクト」をより深くイメージするために、サッカーの試合を題材に考えてみましょう。

監督であるあなたは、11名の選手に的確に指示を出して彼らを動かし、相手チームからゴールを奪わなければなりません。これは、「プログラマであるあなたが、コンピュータに的確に指示を出し、目的どおりに動かさなければならない」状況と似ています。

次の図7-11は、「手続き型プログラミング」をサッカーにたとえて表現した図です。

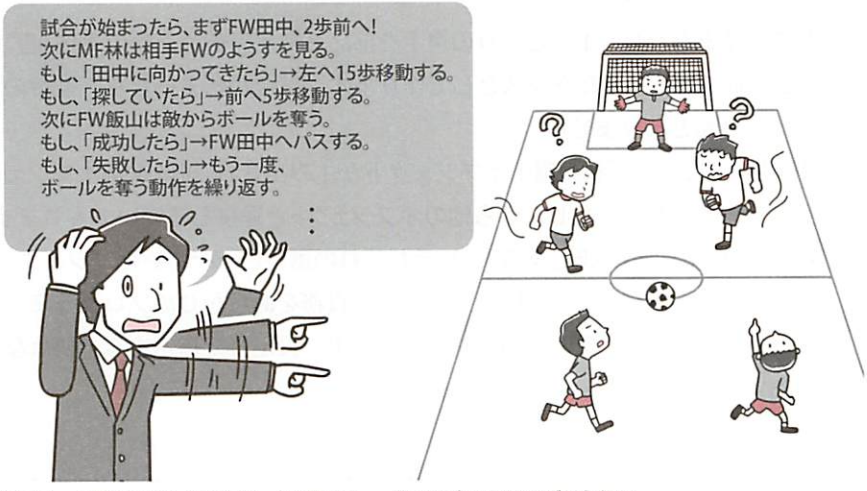
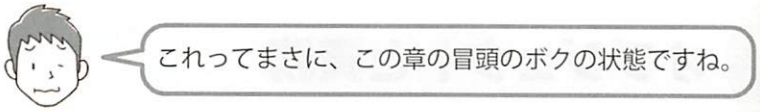


図7-11 「手続き型」の監督は、選手に逐一、指示を出さなければならない

監督であるあなたは、全選手の一挙手一投足に対して、すべての指示を細かく出す必要があります。これでは監督の負担が非常に大きく、体がいくつあっても足りません。作戦変更・選手交代などをしようとしても、監督自身が大混乱してしまうでしょう。



そして次の図 7-12 は、「オブジェクト指向プログラミング」をサッカーで表現した図です。

図 7-12　「オブジェクト指向型」の監督が事前に選手の役割と責任を割り当てれば、試合では選手自身が自分の役割を果たすために行動する

監督であるあなたは、1 人ひとりの選手を部品と考え、**それぞれの責務（役割・責任）**を事前に割り当てたクラスとして作ります。試合が始まったら、監督のすることはほとんどありません。

仮想世界には、それぞれの選手オブジェクトが生み出され、**後は選手オブジェクト自身が自分の役割を果たしながら他のオブジェクトと連携して動いてくれます。**

監督（＝プログラマ）であるあなたは、それぞれの選手（＝クラス・オブジェクト）に、「この状況下で、どう行動すべきか」という責務をあらかじめプログラミングしているため、試合中にそれぞれの選手の一挙手一投足まで指示する必要はなくなるのです。

責務の割り当て

オブジェクト指向プログラミングでは、プログラマはそれぞれの部品に「責務」をプログラムとして書き込む。

7.4.2 オブジェクトの姿

これまでの例に挙げた「サッカー選手」「口座」「受付」など、仮想世界で動くそれぞれのオブジェクトは、すべて何らかの責務を仮想世界の神様たるプログラマから与えられます。たとえば、「サッカー選手」オブジェクトは「ボールを受けたら前に走る」「シュートする」など、あらかじめ設定された役割を果たす**行動責任**を負っています。

同様に、ATM の「受付」オブジェクトにも図 7-7 (p.281)のような行動責任があります。振り込み依頼を受け付けたら、「口座」オブジェクトが管理する 2 つの口座間でお金を移動し、その結果を「印刷係」オブジェクトに渡して ATM 利用控えの印刷を依頼するという一連の流れが受付の行動責任です。

では、「口座」オブジェクトは、いったいどんな責任を負っているのでしょうか？「口座」は行動責任を負っていませんが、「残高をしっかり覚えておく」という**情報保持責任**を負っています。

確かに、気づいたら中身が消えているような「無責任な口座」では困りますね。

このような「情報保持」と「行動」の責任を果たすために、それぞれのオブジェクトは「**属性**」と「**操作**」を持っています。

【属性】 その登場人物に関する情報を覚えておく箱
【操作】 その登場人物が行う行動や動作の手順

たとえば、湊くんが開発中の RPG における「勇者」という登場人物は、図 7-13

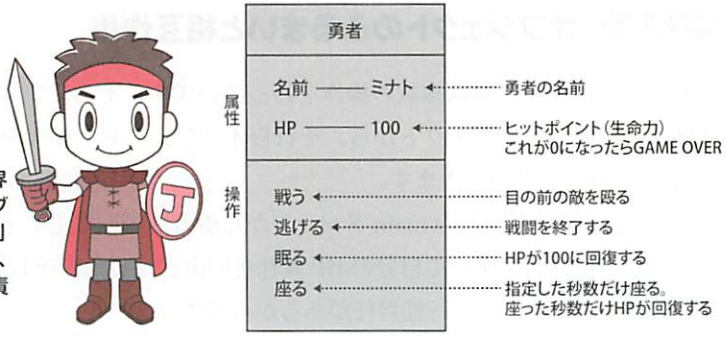


図 7-13 仮想世界を冒険する勇者オブジェクトは、「属性」と「操作」を持ち、情報保持と行動の責任を果たす

のようなオブジェクトとして考えることができるでしょう。

勇者オブジェクトは、自分の名前やHPをしっかり覚えておかなければなりません（＝情報保持責任）。そして、もし「戦え」と命令されれば勇敢に目の前の敵と戦い、「眠れ」と命令されれば眠って自分のHPを回復させる責任（＝行動責任）があるのです。

そのオブジェクトがどんな属性や操作を持つかは、プログラマが部品を作成する際に決定します。そのためには現実世界の登場人物をよく観察し、どのような属性を持ち、どのような操作ができるかを忠実に再現する必要があります。

では、勇者同様に、「お化けキノコ」という登場キャラクター（これもオブジェクトです）を考えてみましょう（図7-14）。

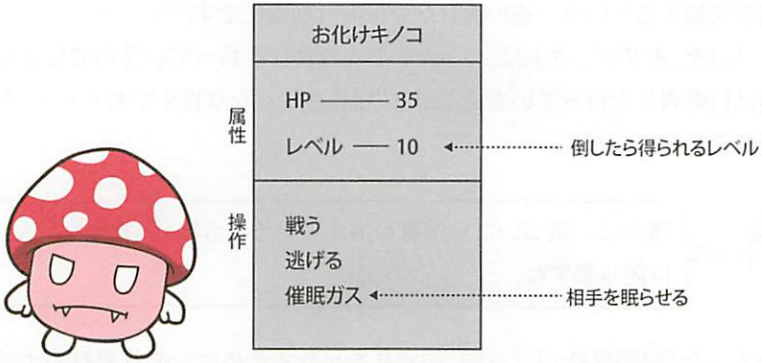


図7-14　仮想世界で「勇者」オブジェクトの敵となる「お化けキノコ」オブジェクトは、勇者と同じく「属性」と「操作」を持っている

お化けキノコは特に重要でないモンスターなので名前は不要と考え、「名前」属性は持っていません。また、このモンスターは「催眠ガス」という技が使えるという設定に従って、その操作を持っています。

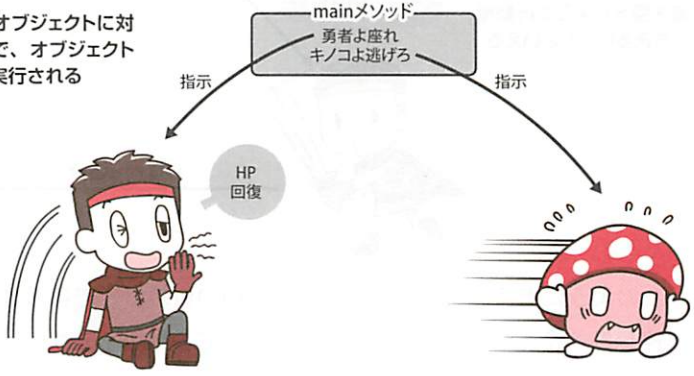
7.4.3　オブジェクトのふるまいと相互作用

勇者やお化けキノコは複数の操作を持っています。そしてプログラムのmainメソッドや他のオブジェクトから、それらオブジェクトの操作を呼び出す（＝行動指示を送る）ことができます。

たとえば、プログラムのmainメソッドから勇者に「座れ」という指示を送れば、勇者は仮想世界内で座って自分のHP属性を回復させる動きをします（図7-15）。なぜなら勇者には、その行動責任があるからです。

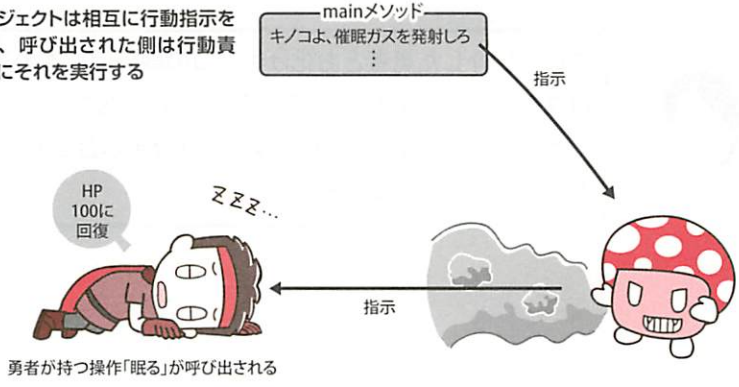
同様に、お化けキノコに対して「逃げろ」と指示を送れば、仮想世界内のお化けキノコは逃げ出して戦闘が終わります。

図 7-15　それぞれのオブジェクトに対し行動指示を送ることで、オブジェクトの操作が呼び出されて実行される



また、あるオブジェクトから別のオブジェクトへ操作の指示を送ることも可能です。main メソッドからお化けキノコに「催眠ガス」という指示を送ると、お化けキノコは勇者が持っている操作の中から「眠る」を呼び出します。すると勇者は眠ってしまいます（図 7-16）。

図 7-16　オブジェクトは相互に行動指示を送ることができ、呼び出された側は行動責任を果たすためにそれを実行する



オブジェクトは別のオブジェクトが持つ操作を呼び出すだけでなく、他のオブジェクトの属性を取得したり書き換えたりもできます。

たとえば main メソッドから勇者に「戦う」という指示を送ると、次ページの図 7-17 のように勇者は戦い、結果としてお化けキノコの HP 属性を書き換えて減らす動作をするでしょう。

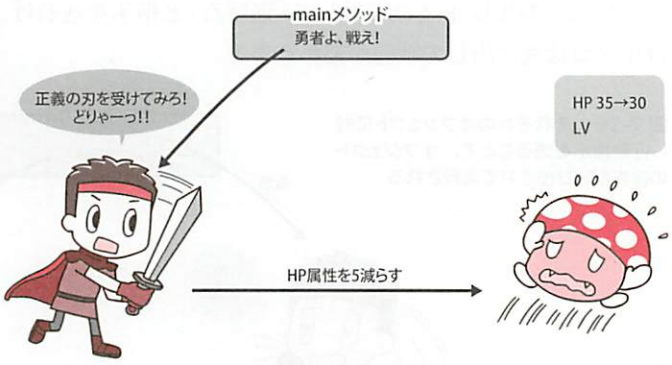


図 7-17　勇者がお化けキノコを攻撃するということは、勇者がお化けキノコに対して HP 属性を書き換えるように行動指示を送る行為だといえる

このように考えると、コンピュータの中の仮想世界で各オブジェクトが互いの属性を書き換えたり操作を呼び合ったりして、物語を繰り広げていく姿が目に浮かびませんか？

このようにして仮想世界内のオブジェクトは互いに「属性」を読み書きしたり、「操作」を呼び出したりして連携し、全体では 1 つのプログラムとして動きます。

そして、仮想世界の中で「仮想的な受付」や「仮想的な口座」が現実同様に正確に動いてくれるからこそ、現実世界の「本物の受付係」や「紙の口座帳簿」は仕事から解放され、コンピュータによる自動化が可能になるのです。

ここで紹介した勇者とお化けキノコの戦いは、次の第 8 章でプログラミングして動かしてみるよ。今は「仮想世界の中でオブジェクトたちが互いの属性や操作を呼び出し合って動作する」姿をイメージできれば十分だ。

7.5 オブジェクト指向の3大機能と今後の学習

7.5.1 3大機能とその位置付け

「オブジェクト指向といえば継承」と聞いたことがありますけど、継承は大事じゃないんですか？

さすが朝香くん。もうそんな専門用語を知っているんだね。もちろん、まだ解説していない内容だから知らなくても問題ないけどね。

その「継承」って何ですか？

オブジェクト指向の本質に沿った開発をするための手伝いをしてくれる、便利な機能のひとつだよ。

オブジェクト指向の本質は、あくまでも「現実世界を仮想世界内に再現すること」です。よって、現実世界を観察し、「口座」や「受付」という単位でクラスを分割して適切な責務を与えたプログラムであれば、それだけで十分にオブジェクト指向の考え方に沿ったプログラムと言えます。

さらに、Javaのような**オブジェクト指向言語**（Object Oriented Programming Language＝オブジェクト指向の考え方に沿ってプログラムを作りやすい配慮がなされているプログラミング言語）には、開発者が「より便利に」「より安全に」現実世界を模倣できるよう、文法などに専用の機能が準備されています。それが次ページの図7-18に示した**オブジェクト指向の3大機能です**。

図 7-18　オブジェクト指向の 3 大機能「カプセル化」「継承」「多態性」を利用することにより、便利で安全なプログラムを作ることができる

この 3 大機能については、それぞれ第 10 ～ 13 章で詳しく解説していきます。今は「オブジェクト指向の実践を支援する、このような 3 つの機能があるんだな」と、あいまいなイメージで捉えておいてください。

7.6.2　以降の章の学び方

オブジェクト指向の全体を一軒の家になぞらえると、第Ⅱ部の章（7 ～ 13 章）は、それぞれ次の図 7-19 のような構造に相当します。

この第 7 章と次の第 8 章がすべての基礎であり、その上に「さまざまなクラス機構」（第 9 章）と、「オブジェクト指向の 3 本柱」（第 10 ～ 13 章）が載っています。

本章の冒頭でも紹介しましたが、これから本格的にオブジェクト指向を学ぶにあたっては、次の点に注意してください。

図 7-19　第Ⅱ部の各章で解説するオブジェクト指向の構造

● 1 章ずつ完璧に理解して次に進む必要はない

第Ⅱ部で学ぶ内容は、それぞれの章が密接に関係しています。そのため「後の章を学ぶことで、前の章の内容をより深く理解できる」ということもよくあります。それぞれの章を 1 度読んだだけでは完全に理解できなかったとしても、とりあえず疑問点は置いて次の章に進んでください。

●最初から、すべての章を理解できなくてもよい

第 7 章から第 13 章までのすべてを理解しなければ、オブジェクト指向をまったく使えないわけではありません。この第 7 章と第 8 章を理解しておけば、最低限のオブジェクト指向プログラミングは可能になります。第 10 ～ 13 章で紹介する 3 大機能は、マスターできたものから少しずつ使っていけばよいのです。

とはいえ、カプセル化や継承の基本を理解していないとプログラミングの実務に支障があるため、本書では次ページの図 7-20 のような学び方をお薦めします。

まずは次ページ図 7-20 にある 1 周目の内容までを理解していれば、とりあえずオブジェクト指向を用いた開発ができるでしょう。初めて Java を学習する人は、まず 1 周目の部分を理解することを目標にしましょう。

2 周目では、「オブジェクト指向の山場」である 12 ～ 13 章に挑みますが、第 12 章に入る直前に、この第 7 章をもう一度読み返しておくと、より理解が深ま

ります。

最後の３周目では、第７章と第８章を復習してオブジェクト指向の全体像と本質を振り返った上で、第 12 ～ 13 章を読み返すことにより、オブジェクト指向に関する深く実践的な理解を得られるだけでなく、「さまざまなものを自由自在にプログラムで表現できるまでに成長した自分自身の可能性」に気づくでしょう。その段階になれば、きっとオブジェクト指向の楽しさの虜になっているはずです。

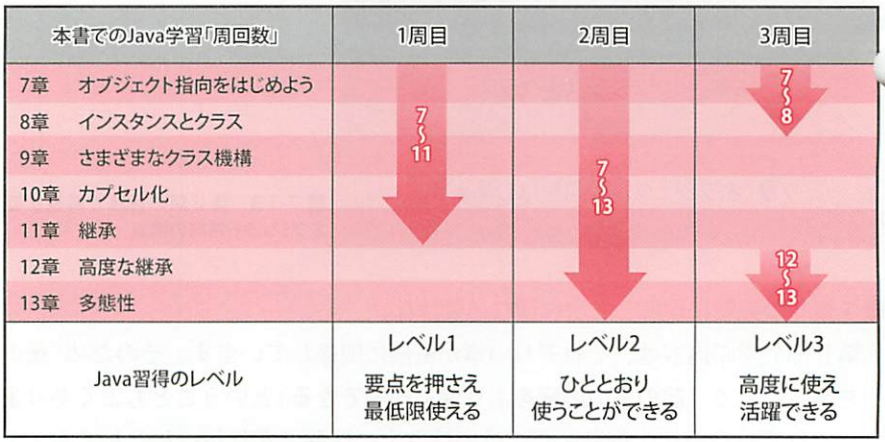


図 7-20　本書が推奨する学習の流れ（1 周目から 3 周目の学び方）

7.6 第7章のまとめ

この章では、次のようなことを学びました。

学習方法の違い

- 第II部ではオブジェクト指向という「考え方」を学ぶため、「文法」を学んだ第I部とは学び方を変える必要がある。
- 第II部ではイメージを重視し、各章を繰り返し学んでいくことで、ぼんやりとした理解を少しずつ明確にしていく。
- 初めからオブジェクト指向のすべてをマスターする必要はない。まずは第11章までの「1周目」の達成が目標。

オブジェクト指向の概要と本質

- オブジェクト指向とは、ソフトウェアを開発する際に用いる部品化の考え方。
- オブジェクト指向を用いると、大規模で複雑なソフトウェアであっても、ラクして、楽しく、良いものを開発できる。
- オブジェクト指向の本質は、現実世界における「登場人物とそのふるまい」を、仮想世界においても「オブジェクトたちとそのふるまい」として再現すること。
- オブジェクトは属性と操作を持つことによって、現実世界と同様の責務を果たす。
- オブジェクト指向の本質に沿った開発を支援するために準備されている「カプセル化」「継承」「多態性」などの機能を用いたプログラム開発は、第10章〜第13章で学ぶ。

7.7 練習問題

練習問題 7-1

ATM、券売機、ブログなどの他に、「現実世界の人間の活動をプログラムで機械化・自動化しているもの」の例を考えて書き出してみましょう。

(ヒント 1) 現代では「機械がやることが当たり前」のことも、昔は人が手作業でやっていたことがほとんどです。

(ヒント 2) ATM や券売機のように「見た目がコンピュータではないもの」の中でもプログラムは動いています。

練習問題 7-2

次のプログラムを作る場合に登場するオブジェクト（現実世界の登場人物）にはどのようなものがあるか、自由に考えて書き出してみましょう。

①現在航行中のすべての飛行機と空港を管理する、航空管制システム
②国内の映画館を選択すると、その映画館での上演映画と、その主演俳優の一覧を表示してくれるプログラム
③余っている食材を入力すると、膨大なレシピの中からその食材を使う料理を検索してくれるプログラム

練習問題 7-3

ある都市の観光案内所には、タッチパネル式の「観光案内端末」が設置されています。利用者が画面から希望条件を入力すると、オススメのお店や名所旧跡の名前・住所・電話番号・解説を提示してくれます。

この観光案内端末の中で動くプログラムの内部では、さまざまなオブジェクトが動作しています。そこで以下2つのオブジェクトが持つであろう「行動責任」「情報保持責任」を自由に考え、操作と属性として書き出してみましょう。

①現実世界の案内係を再現した「案内係」オブジェクト
②現実世界のお店や名所旧跡を再現した「観光地」オブジェクト

(ヒント) オブジェクトには、操作や属性だけを単独で持つものもあります。

7.8 練習問題の解答

練習問題 7-1 の解答

この問題に対する解答は無数に考えることができますので、以下に示したのは解答の一例です。

電卓の中に入っているプログラムは、「指示したとおりにすばやく計算をしてくれる人」を機械化・自動化したものです。

- 電子メールは、現実世界の「手紙」を電子化したものであり、そのメールを配送するインターネットのシステムは「郵便配送のしくみ」を機械化したものです。
- ネットショッピングサイトは、現実世界の「商店」を電子化したものであり、そのプログラムは従来、店員が受け持っていた「商品の検索依頼・注文依頼・決済」などを自動化したものです。

練習問題 7-2 の解答

この問題の解答も無数に考えることができますので、以下に示したのは解答の例です。

①「飛行機」オブジェクト、「空港」オブジェクト
②「映画館」オブジェクト、「映画」オブジェクト、「俳優」オブジェクト
③「食材」オブジェクト、「レシピ」オブジェクト、「料理」オブジェクト

練習問題 7-3 の解答

この問題の解答も無数に考えることができますので、以下に示したのは解答の例です。

①「指定条件に基づいて観光地を検索する」操作
②「名所の名前」属性、「名所の住所」属性、「名所の電話番号」属性、「名所の解説」属性