［増補改訂］

良いコードを書く技術

読みやすく保守しやすいプログラミング作法

Agata Toshitaka
縣俊貴
［著］

技術評論社

●初出
本書は、小社刊『WEB+DB PRESS』Vol.44～Vol.49の連載「良いコードへの道 —— 普通のプログラマのためのステップアップガイド」をもとに、大幅に加筆と修正を行い書籍化したものです。

支えてくれたみなさんへ

はじめに

あなたはプログラマーですか？ もしもプログラマーなら何のためにプログラムを書いていますか？ 生活のため、それとも楽しいからでしょうか。理由は人それぞれでしょう。だけど、もしあなたがプログラマーなら、「良いコード」が書けるようになることで仕事がもっと楽しくなるはずです。

本書は良い仕事をしたい普通のプログラマーが、読みやすく保守しやすい良いコードを書けるようになるための解説書です。ここでの普通のプログラマーとは、初級プログラマーや初級から中級になりかけのプログラマーのことを指します。私が考える良いコードの具体例は本書の中で解説していきますが、一番大切なのは「良い仕事をしたい」「良いソフトウェアを作りたい」という気持ちです。この本を手に取ってくれたあなたの中にも、きっとそういう気持ちがあると思います。

良い仕事をすると仕事にやりがいを感じられるようになる、つまり仕事が楽しくなります。そして職人として仕事に誇りが持てるようになります。良い仕事をするプログラマーは仲間や顧客からの信頼が厚くなり、仕事を依頼される機会も増えていきます。ということは収入もアップすることでしょう。

と、なれば理想ですが、実際には職人として良いコードが書けるプログラマーは、あくまでも私の実感ですが業界の2割ぐらいしかいません。あとの8割はいわゆる普通のプログラマーです。その中には、良い仕事をしたいと思っているものの、なかなかそのコツをつかめなかったり、何から手を付けてよいかわからない人が多いようです。

プログラミングはとても創造的でエキサイティングで楽しいものなのに、これではもったいないです。たとえば、泳ぎ方がよくわからないままプールに行ってもおもしろくありません。むしろ苦痛なぐらいです。しかし、25メートル、50メートルと泳げるようになるとだんだん楽しくなります。さらに思いのまま泳げるようになると、快感へと変わっていきます。

私が良いコードを書けるようになったのは、良い師匠や書籍と巡り合えたおかげです。本書が良い仕事をしたい普通のプログラマーのみなさんの道しるべとなることを願っています。

2021年4月　縣 俊貴

謝辞

本書は多くの方から支えていただいて執筆することができました。

まず、地元福岡のプログラマー仲間である桜井雅史さん、上津竜太郎さん、きしだなおきさんには、プログラミングや設計、教育や勉強会についてたくさんの議論をしていただきました。その議論での気付きが本書にふんだんに盛り込まれています。山本竜三さん、染田貴志さん、南里範子さん、鳥谷部昇さん、城和典さんには、校正時に地道な作業を手伝ってもらい、たいへん助けていただきました。本書に彩りを添えている素敵なプログラマーのイラストは、高田美奈子さんに描いていただきました。馬場保幸さんからは、オープンソースのWebアプリケーションフレームワークCubbyを一緒に開発していく中で、本書のヒントをいただきました。株式会社ヌーラボのみなさんからは、日々のシステム開発、サービスの運営を通して、ソフトウェア技術についてあらゆることを教わりました。この場を借りてみなさんに感謝の言葉を贈ります。

遅筆な筆者を時には励まし、時には尻を叩いて支えてくれた技術評論社の稲尾尚徳さんには、どれだけ感謝の言葉を言っても足りないぐらいです。本当にありがとうございました。

増補改訂での更新点

増補改訂では、コード例を執筆時点で最新のJava 15に対応しました。古いライブラリやフレームワークも、すべて新しいものに置き換えています。

また、良いコードの実現に重要なデータ構造についての理解を深めるために、第7章「データ構造」を書き下ろしました。加えて第11章「メタプログラミング」では、メタプログラミングの題材を、外部DSLから、より導入が手軽で保守性が高い内部DSLに変更しました。これらの加筆、改訂により、より実践で役立つ内容となっています。

コード例について

本書には多くのコードが例として登場します。コード例で使用する言語は基本的にJavaです。RubyやJavaScript、TypeScriptなどほかの言語を使用する場合のみ、本文中にその言語名を明記します。ただし、言語に特化した話はあまりありませんので、ほかの言語の人でも十分に参考になる構成になっています。

コード例はサポートページからダウンロードできます。このコードはパブリックドメインですので、仕事や趣味のプログラムに自由に利用していただいてかまいません。

サポートページ

本書のサンプルコードは、下記サポートページから入手できます。正誤情報なども掲載します。

URL https://gihyo.jp/book/2021/978-4-297-12048-1/support

［増補改訂］**良いコードを書く技術** 読みやすく保守しやすいプログラミング作法●目次

第1章

良いコードとは何か

1.1 良いコードの定義と価値

良いコードとはいったいどのようなものでしょうか。良いコードを書くと何かうれしいことがあるのでしょうか。そもそも良いコードを書く必要があるのでしょうか。

本章では序論として、「良いコード」がどのようなものか定義し、その価値について考えていきます。

1.2 良いコードの定義

一口に良いコードと言っても、組織やプロジェクト、プログラマーか管理者かなど、状況が異なると定義も変わってきます。本書では次の4つを満たすものを良いコードと定義します。

- 保守性が高い
- すばやく効率的に動作する
- 正確に動作する
- 無駄な部分がない

定義した良いコードを実現するために必要な知識を、本書では**図1.1**の構成で解説していきます。

保守性が高い

私たちが書いたコードは、私たちが想像するよりも長く利用されます。あとから見て何をやっているのか理解不能なコードは良いコードとは言えません。将来の自分は記憶力において他人と同然です。つまり、他人が見て理解できるコードであれば、将来の自分が見ても理解できる良いコードであると言えます。

本書では第3章「名前付け」にて、変数やメソッドの適切な名前の付け方を

解説します。続く第4章「スコープ」にて、依存性が低く保守性が高いコードの書き方を紹介します。そして第5章「コードの分割」と第6章「コードの集約」、第7章「データ構造」にて、長いコードの分割や重複コードの集約、データ構造の改善を行うことで、読みやすく保守しやすいコードを実現します。

すばやく効率的に動作する

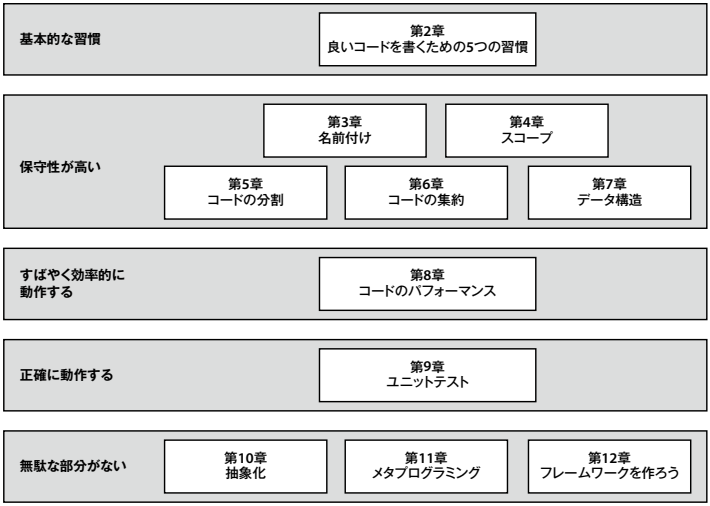
似たような実装がいくつか考えられるとき、あきらかに効率の悪いものを選択する必要はありません。良いコードは適切なパフォーマンスで動作します。

本書では第8章「コードのパフォーマンス」で、すばやく効率的に動作するコードについて解説します。

正確に動作する

確実に動作し、信頼性が高いことは良いコードの条件です。「防御的プログラミング」という言葉がありますが、これは「正常な値が来るはず」という

図1.1　本書の構成



決め付けをせずに、不正な値が来ても被害を受けないように防御的にプログラミングを行うことです。良いコードは防御的で、不測のバグを生み出しにくい作りになっています。

本書では第9章「ユニットテスト」で、バグが少なく正確に動作するコードを実現するために、テストの自動化について解説します。

無駄な部分がない

無駄がないコードは理解するのも修正するのも簡単で時間がかからないため、良いコードと言えます。コード内に繰り返し現れるパターンを劇的に短くする方法として抽象化があります。

本書では第10章「抽象化」で簡単な抽象化について触れ、第11章「メタプログラミング」でより高度な抽象化に挑戦します。第12章「フレームワークを作ろう」では簡単なフレームワークの作成を通してフレームワークの動作原理を知り、本格的な抽象化と柔軟で再利用可能なフレームワーク作成のコツを学びます。

これらの定義を満たす良いコードを書くためには、基本的な日々の習慣を身に付けている必要があります。そこで第2章では、「良いコードを書くための5つの習慣」を紹介します。

1.3 良いコードの価値

では、私たち開発者が良いコードを書けるようになると、具体的にどんなメリットがあるのでしょうか。

プロジェクトを強力に推し進める

良いコードは、プロジェクトを推し進めて成功へと導くための基本的な

要素となります。

達人たちは、シンプルで保守性が高く、安定したコードを、ものすごいスピードで書き上げていきます。場合によっては、単純作業を自作のDSL（*Domain Specific Language*、ドメイン特化言語）[注1]に置き換えたり、テストが難しいレガシーなコードをテスト可能で検証できるコードに変更したりすることで、品質や生産性を数百倍に高めることさえあります（おおげさではなく、本当に数百倍の場合もあるのです！）。

これはプロジェクトの成功にとって大きなアドバンテージです。もちろん、良いコードがあれば必ずプロジェクトが成功するわけではありません。実際は、開発プロセスやマネジメント、コミュニケーションなどほかの要素により左右されることも多いのですが、それを差し引いたとしても良いコードの持つ力は大きいと言えます。

プログラマーとしての評価が高まる

良いコードを書くプログラマーは、総じてプログラマーとして信頼され、評価されます。プログラマーとしての評価が組織としての実際の評価や収入に結び付くかどうかは、所属する組織の評価制度やプログラミング以外の仕事ぶりも含めて決まるのが現実です。ですが、「良いコードを書けること」がマイナス評価につながることはないでしょう。

仕事に満足感や自信が持てるようになる

もう二度と触りたくない保守が不可能なコードを書いたことはありませんか？ そのような低いクオリティの仕事をしてしまったときは、仕事に対する満足感を得ることは難しいでしょう。

逆に、自分の意志で適切に良いコードを書き、品質の高い安定したソフトウェアを開発したときは、満足感も高く、自信を持って仕事に取り組めたはずです。長いプログラマー人生を考えると、良いコードが書けるレベ

注1　ある特定の問題に対応するための言語のことです。DSLについては第11章で詳しく取り上げます。

ルを目指すのは合理的なことです。

1.4 代表者の声

本書の対象読者は次の3グループを想定しています。それぞれの代表者にコメントしてもらいましょう。

●良い仕事をしたい普通のプログラマー

別にハッカーを目指すわけでもないし、休日はパソコンに向かうより家族との時間を大切にしたいです。でも、だからといってプログラムが嫌いなわけではありません。良いコードは書けるようになりたいし、高いクオリティの仕事をしたいです。

特段に「達人プログラマー」を目指しているわけではないが、良い仕事と成果を出したいと考えているプログラマーの人は、本書により普段知ることのない新しい概念を知り、興味関心の対象を広げることができるでしょう。

●達人プログラマーを目指す中級プログラマー

新しい技術は大好きっす。書籍やWebでの情報収集は常識なので、良いコードは書けていると思うっす。ただ、最近はインプットが多すぎて、ちょっと消化不良気味。ときにはアウトプットしていきたいなぁ。

向上心が高く達人を目指すプログラマーにとって、本書が良いドキュメントとしての役割を果たすはずです。

●達人プログラマー

良いコードが書けるようになるには時間が必要じゃ。焦らず積み上げていけば必ずや誰でも良いコードが何かわかるようになるはずじゃぞ。すべては、やるか、やらないかじゃ。

すでに達人な人には、「この書籍を新人などに見せれば教育に使えるな」という観点で見ていただくとよいでしょう。

1.5 まとめ

本章では良いコードの定義を行いました。「どのようなものが良いコードか」というのは、それぞれのプロジェクトの特性やチーム構成、技術的な要素、1回きりしか使われないシステムなのか長期的に使用されるシステムなのかなどにより、優先事項が変わります。ただ、ここで紹介した良いコードの定義は基本かつ普遍的なものばかりです。ぜひ、コードを書き終えたら「良いコードの定義を満たせているか」と自分に問いかけてみてください。