


## デザインパターンって何?

 絶版に伴い、校正前の原稿テキストを公開したものです。基本的に原稿そのままをHTML形式に変換したものですので、誤字/脱字、説明不足の箇所もあるかも知れませんがご了承ください。初出:「PHPによるデザインパターン入門」(下岡秀幸/畑勝也/道端良 著, 秀和システム, ISBN4-7980-1516-4, 2006年11月23日発売)

そもそも「パターン」とは何でしょうか？

英語の「pattern」が日本語に翻訳される場合は、「模範」「手本」「模様」「体系」などになりますが、日本語で「パターン」というと、すぐに思い浮かべるのは、毎回同じ手を使う事を表す「ワンパターン」や、幾何学模様など繰り返し模様を指す「パターン」がありますね。電子分野では「回路パターン」や「パターン認識」、少し離れると、ファッションデザイン界では、デザインの原型やその原型をおこす型紙のことも、「パターン」と呼ぶようです。

このように「パターン」とは、

- ・ とある一定の事象が繰り返される規則的なもの
- ・ 模範や手本となるもの

を指す言葉である、ということになると思います。

それでは、本書で扱う「デザインパターン」とは何でしょうか？

「デザインパターン」という言葉は、ソフトウェア界発祥の言葉ではありません。もともとは、建築家であるChristopher Alexander氏により発表された論文「A Pattern Language: Towns, Buildings, Construction」(邦題:「パタン・ランゲージ—環境設計の手引き」/鹿島出版会/1984年)に由来するものです。Alexander氏は、建物や市街の設計・建築における様々なトレードオフや問題と、それに対する典型的な解決の方法をまとめました。そして、誰もがわかりやすいよう名前を付け、「パターン」として示しました。これにより、他の設計者が「パターンのカタログ」から必要なパターンを選択して、より要求にかなった建築がおこなえる足がかりとしていく、という方法を提唱しました。

そして、1987年、OOPSLA(Object-Oriented Programming, Systems, Languages & Applications)というオブジェクト指向に関するカンファレンスで、Kent Beck氏とWard Cunningham氏が発表した[Using Pattern Languages for Object-Oriented Programs](#)という論文によって、この「パターン」という考え方がソフトウェア界に持ち込まれました。これから本書で扱っていく「デザインパターン」のルーツは、このAlexander氏の「パタン・ランゲージ」です。

ソフトウェア設計における「デザインパターン」も、建築におけるパターンと同様です。ソフトウェアを何度も設計していると、あちらこちらで共通する機能や問題があることに気づくことがあります。その問題を解決するために毎回ゼロから設計しているのは、非常に効率が悪くなります。このような状況の場合、経験を積んだ開発者であれば、「こういった問題を解決するには、こういう構造でプログラムを書けばうまくいく」といったノウハウを持っているものです。しかし、「ノウハウ」というものは、なかなか人に伝えることが難しいものです。もし、これらのノウハウを他の開発者と共通のイメージとして共有することで、コミュニケーションが円滑に行われるとしたら、どんなに良いでしょうか。

このノウハウを、他の開発者が再利用できるようカタログ化したものがデザインパターンです。もっと簡単に言ってしまうと「設計のコツの虎の巻」といった感じでしょうか。その「虎の巻」を手にする事で、「良い設計」を何度でも再利用することが可能になります。

ここまでの話でお分りの通り、「デザインパターン」は何か具体的なクラスライブラリの設計方法ではありません。ましてや、クラスライブラリそのものでもありません。デザインパターンは、あくまで、

「ある問題を解決するには、こうすればうまくいく」

という設計における「考え方」や「アイデア」を抽象化したものです。つまり、これから扱うデザインパターンに出てくる「クラスの形」だけを丸暗記しても、あまり意味がありません。少なくとも、具体的なデザインパターンを使ったプログラムを理解する手助けになるかもしれませんが…。これまでも何度か出てきましたが、それぞれのデザインパターンはある「問題点」を解決します。それには、いろいろな役割を持つ部品(クラスやインターフェース)が存在し、お互いに連携し合います。

また、デザインパターンには、次に挙げるオブジェクト指向の3要素がふんだんに盛り込まれています。

- ・ カプセル化
- ・ 継承
- ・ ポリモーフィズム(多態性)

この視点からデザインパターンを見ることで、デザインパターンをより深く理解でき、メリットをより一層感じられるのでは

ないかと思います。みなさんも是非、この視点からパターンを眺めてみてください。

少々前置きが長くなりました。次節では、本書で扱う代表的なデザインパターン、GoFパターンについて見ていきましょう。