GSLetterNeo Vol.21
2010年4月

オブジェクト指向設計原則（１）

オブジェクトモデリングスペシャリスト
土屋　正人
Masato Tsuchiya
m-tsuchi@sra.co.jp

オブジェクト指向言語を使って開発したのにオブジェクト指向の効果が得られない、ということが起きています。原因は様々でしょうが、オブジェクト指向の考え方が理解されていないことが大きいのではないかと思います。

オブジェクト指向の考え方を設計に展開するためのガイドラインとして、オブジェクト指向設計原則なるものがありますが、ロバート・C・マーチンがその著書「アジャイルソフトウェア開発の奥義」でコンパクトにまとめています。取り上げられているのは次の 11 の原則です。

- 単一責任の原則（SRP）
- オープン・クローズドの原則（OCP）
- リスコフの置換原則（LSP）
- 依存関係逆転の原則（DIP）
- インタフェース分離の原則（ISP）
- 再利用・リリース等価の原則（REP）
- 閉鎖性共通の原則（CCP）
- 全再利用の原則（CRP）
- 非循環依存関係の原則（ADP）
- 安定依存の原則（SDP）
- 安定度・抽象度等価の原則（SAP）

この中から代表的な原則を、何回かに分けて紹介していきたいと思います。

◆単一責任の原則（SRP）

SRP（Single Responsibility Principle）と呼ばれます。一番シンプルな原則で、定義は「**クラスを変更する理由は１つ以上存在してはならない**」です。

あるクラスを変更するのに 2 つの理由がある場合、そのクラスは 2 つの責任を持っていることになります。オブジェクト指向の教材でよく使われる、Rectangle（矩形）クラスを例にするとその責任は、

- 画面上に自分を描画する
- 自分の面積を計算する

といったものが考えられます。そこで Rectganle クラスにこの 2 つの仕事を割り当てます。すると、Rectangle クラスは次の要求が出てきたときに変更が必要になります。

- 描画方法を変更する
- 面積の計算方法を変更する

Rectangle クラスは描画するといった GUI の責任と、面積計算といった幾何学的責任の 2 つを持つことになり、SRP に違反します。幾何演算だけを必要とするアプリケーションが GUI という余分な要素を含んでしまうわけです。このことは変更に対する影響範囲／修正対象を広げ、修正個所を特定する時間が余計にかかります。この問題は、描画専門クラスと幾何専門クラスに分けることで解消できます。

責任という言葉は、役割という言葉に置き換えると理解しやすいかも知れません。「クラスは役割を 1 つだけ持ちましょう」ということになります。

SRP の考え方は、オブジェクト指向特有なものではなく、昔からあるものです。構造化設計では内的品質を測る 10 数種類の指標がありますが、その中で最もよく知られているものに、モジュール凝集度とモジュール結合度があります。モジュール凝集度はモジュールを構成する要素間の関係を 7 段階で分

類したもので、最も評価の高いものは機能的凝集度と呼ばれ、SRP に相当します。自分の作ったモジュールの凝集度が、機能的凝集度かどうかを判断する最も簡単な方法は、「**モジュールの機能を 1 センテンスで説明できるか**」というものです。「機能」の部分を責任や役割に置き換えてみると、クラスに適用しやすいと思います。クラスの粒度は様々なので、責任の粒度も様々ということになりますが、「**責任あるいは役割を 1 センテンスで説明できるか**」という問いは、粒度に関係なく有効でしょう。

◆リファクタリング

コードを書き上げてから改良する手段として、リファクタリングがあります。リファクタリングとは「**外から見えるふるまいを変えずに、ソフトウェアを分かりやすくし、安いコストで変更できるようにするためにソフトウェアの内部構造に加えられる変更**」あるいは、これを行なうことです。

マーチン・ファウラ等の著書「リファクタリング」にはリファクタリング定石が満載ですが、それらは、たとえオブジェクト設計原則を知らなくとも原則を遵守するようにコードを導いてくれます。その中に「オブジェクト間でのメンバの移動」というカテゴリがあり、次のような定石が説明されています。

- メソッドの移動
- フィールドの移動
- クラスの抽出
- クラスのインライン化
- 委譲の隠蔽
- 横流しブローカの除去

例えば**メソッドの移動**は、メソッドが自分のクラスよりも、他クラスの機能を使ったり他クラスから利用されたりする、といった事実が動機となり、

- メソッドを最もよく使っているクラスに同じ内容の新メソッドを作る
- 古いメソッドはこのメソッドに処理を委ねるか、完全に取り除く

という施策をとることで単一責任の原則に適うように設計実装を改良していきます。

設計原則を遵守しようとするあまり、設計に時間をかけすぎてしまうのも問題です。最初から完全遵守を目指すのではなく、原則を意識しつつも早めに実装して動きを確認し、リファクタリングによって改良する。このような設計と実装の繰り返しが望ましい形でしょう。設計は UML を使ったダイアグラムだけで行なうものではなく、**コーディング、デバッグも設計の一過程である**と考えることが必要だと思います。

◆◇◆◇　いつでも心地よく　◆◇◆◇

コンサルタントファシリテータ　野島勇

『奇跡の脳』という書籍をご存知ですか？

この書籍では、脳の専門家である著者が脳卒中に襲われ、左脳が壊れてしまったときの体験を綴っています。脳卒中になったその日に著者が体験したこと、脳卒中から回復する過程で体験したことが主に書かれています。

脳卒中になったその日のことを読んでいると、右脳の世界と左脳の世界は違うのだなと思えてきます。左脳が働いているときは自己を知覚でき、電話の番号を識別でき、痛みを知覚することができます。しかし、ひとたび左脳の働きが鈍ると、自分と周囲の世界の境界線がなくなってしまい、数字は識別できずただの模様となり、痛みはなく心地よさに包まれたとあります。

私はこれを読み、自分のなかにはいつでも心地よさが存在しているんだと思いました。痛みを感じているその時にも、心地よさの源は同時に存在している。ただ、その心地よさの存在に気付いていないだけなんだと思いました。

少し話は違いますが、牢獄のなかにいても幸せを感じられる人々がいるという話もあります。牢獄に囚われるという同じ状況でも、自分が意識を向ける対象に応じて体験は異なってきます。

自分のなかにいつでも心地よさの源があるのだとしたら、自分の意識を心地よさに向けて、探してみるのがいいのかもしれません。

GSLetterNeo Vol.21
2010 年 4 月 20 日発行
発行者●株式会社 SRA 産業開発統括本部
編集者●土屋正人、柳田雅子、小嶋勉、野島勇

ご感想・お問い合わせはこちらへお願いします●gsneo@sra.co.jp

株式会社SRA
〒171-8513 東京都豊島区南池袋２－３２－８