A Philosophy of Software Design (一人輪講)

古殿直也 2025

輪講でやること（復習）

- 担当者を持ち回りでやる（今回はぼく）
- 担当範囲が決まっている（今回は第一章）
- 担当範囲の解釈を説明する（以降のスライドで説明します）
- 参加者はツッコミとか質問とかをする。すべての参加者が少なくとも一回は質問すること
    - いつでも質問してOK。担当者の発言を遮ってください
- 終わらなかったら次回に繰り越し

このように本のトピックを議論して理解していきましょう

Chapter 1-4 はすでに記事に書いてしまったので後回し

気が向いたらスライドに起こします。スライドの方が主張の構造を練ることになりやすいと感じているのでした。

https://dev.nfurudono.com/posts/read-a-philosophy-of-softfare-design/

Chapter 5. Information Hiding (and Leakage)

- Chapter 4 では深いモジュール (deep module) が望ましいことを主張した。
- この章は、モジュールを深くするためのテクニックの一つとして情報隠蔽を説明する

5.1 情報隠蔽

情報隠蔽では知識を隠蔽してインターフェースから隠しつつ、実装に落とし込む

- モジュールにはインターフェースと実装がある
- モジュールの機能を実現するためには知識が必要
- 情報隠蔽は、機能をインターフェースで提供しつつ、実現に必要な知識をモジュールの利用者が忘れられるようにするテクニック
    - このとき必要な知識は実装として埋め込まれる
    - 実装はアルゴリズムとデータ構造で表現される

情報隠蔽は何の役に立つだろうか

情報隠蔽はインターフェースを簡潔にし、システムの変更を簡単にする

情報隠蔽は二つの観点から複雑さを減らすと言える

1. モジュールのインターフェースを簡潔にする

- 知識が隠蔽されるので、それがインターフェースに染み出さなくなる
- 結果としてインターフェースが簡潔になる

1. システムの変更を簡単にする

- 情報を隠蔽で切れていれば、それへの外部からの依存はなくなる
- そのためシステム全体の複雑さは減る
    - Chapter 2か3で述べた、システムの複雑さは依存の量に相関がある話を思い出したい
    - 直感的にも依存が少ない方がシステムの変更は簡単

情報隠蔽の程度は 0 / 1 ではない

- 完全に隠し切るのがベストではある
- とはいえ普段は使わないで済むけど、必要なら詳細に突っ込める、みたいなインターフェースがあっても価値がある
  - 必要なら詳細に突っ込むインターフェースがあるということは、隠蔽し切れてはいない
  - それでも多くのクライアントコードではその詳細を忘れられる

5.2 Information leakage

情報が漏れるとはどういうことか

情報隠蔽の反対に、漏れるとはどういうことか

- 実装上の選択が複数のモジュールに影響する
  - モジュール間の依存を生む
- インターフェースに実装上の選択が反映されていれば、やはりその選択は漏れる
- 暗黙的な除法の漏洩もある
  - ファイルフォーマットを仮定している
  - 実行時刻に依存している

複数のモジュールが共通の知識を知っているとしたら、それは情報が漏洩しているということ

情報漏洩への対処

漏れている知識にまつわる変更を加えたくなったときに、単一のモジュールの変更に留まるようにするために、どんな設計をするとよいかを考えよう

- 一つのモジュールにまとめる
  - 実はその知識は、それらのモジュール群が一緒に扱うべき難しい知識だった
  - 依存し合うモジュールが小さめであるときに有効
- その知識を扱う新しいモジュールを定義する
  - 良いインターフェースを定義できるときにだけ効く
  - 暗黙的だった依存関係がインターフェースに出てくるだけに留まることになるので

5.3 Temporal decomposition

処理の順番的に近いものをまとめるようなモジュールを作りがちだけど、それが正解とは限らないので要注意

タイミングの異なる別の箇所で同じデータに触る箇所とか同じドメイン知識が必要になることは多々ある。それらがバラバラにならないようにしたい

5.4 Example: HTTP server

content-lengthをパースしないとどこまで読んでいいかわからない（ストリームのreadを終わらせるタイミングが判定できない）のに、readとparseを分けると、readで結局parseの再実装をすることになる

5.8 Information hiding within class

コードの細かいレベルでも、情報隠蔽はしたい

外部APIで情報隠蔽するメリットを語ったが、それに限らず詳細でも隠蔽したい

- プライベートメソッドに詳細を切り出す
- インスタンス変数を減らす

5.9 Taking it too far

意味がある情報隠蔽はモジュール外で不要な情報を隠すものだけ

モジュール外が知っておくべき情報は隠してはいけない

- 特に、非機能要件のための情報が隠れそうか
- その他にもないと不便を強いるような情報も隠し過ぎてしまうかも

5.10 Conclusion

情報隠蔽はモジュールを深くするのに効く

- たくさんの情報を隠せれば、それだけ機能を有することと、インターフェースを簡潔にすることにつながる
  - これはモジュールを深くすることそのもの
- モジュール分割するときには実行順序に引きずられないように注意
  - 処理に必要な知識に着目して分割し隠蔽すると良い