Java を用いたデザインパターンコーディングの必要性

文理学部情報科学科
5419045 高林 秀

2021 年 11 月 4 日

概要

本稿では、今年度発展プログラミングの課題研究として「デザインパターン」に準拠したコーディングスタイルの必要性について、実際に自身でコーディングを行い、論ずる。本演習には Java を利用した。

1　目的

本稿は今年度発展プログラミングの課題研究として、「デザインパターン」に準拠したコーディングスタイルの必要性について論ずることを目的とする。本稿前半では、デザインパターンの利点・効果・必要性などについて説明し、後半では、実際にコーディングを行う。その際、デザインパターンを使用する前のコードとデザインパターンを使用した際のコードを比較・考察していく。

2　デザインパターンとはなにか

この章では、プログラムコーディングおけるデザインパターンとはどのようなものなのか、歴史的背景や利点・効果、その必要性について述べる。

2.1　歴史的背景

デザインパターンは、オブジェクト指向型プログラミング言語において、記述したコードを様々なプログラムで再利用できるようにするために考案された「プログラムの設計ルール」のようなものである。1955 年に出版された書籍「オブジェクト指向における再利用のためのデザインパターン [1]」にて、初めて「デザインパターン」と呼ばれる用語が使用された。その書籍の広がりにより、デザインパターンの考え方が広く知られるようになった。

その書籍の著者ら（※参考文献の原著 4 名）は、23 種にもおよぶデザインパターンを取り上げており、デザインパターンとはなにか、以下のように述べている。以下、`https://ja.wikipedia.org/wiki/%E3%83%87%E3%82%B6%E3%82%A4%E3%83%B3%E3%83%91%E3%82%BF%E3%83%BC%E3%83%B3_(%E3%82%BD%E3%83%95%E3%83%88%E3%82%A6%E3%82%A7%E3%82%A2)` より引用する。

- 原文

  [Design patterns] solve specific design problems and make object-oriented designs more flexible, elegant, and ultimately reusable. They help designers reuse successful designs by basing new

designs on prior experience. A designer who is familiar with such patterns can apply them immediately to design problems without having to rediscover them.

- 訳文 (Deepl[*1]による翻訳)

  特定の設計上の問題を解決し、オブジェクト指向設計をより柔軟に、エレガントに、そして最終的には再利用可能にします。デザインパターンは、新しいデザインを過去の経験に基づいて行うことで、成功したデザインを再利用するのに役立ちます。このようなパターンに精通している設計者は、パターンを再発見することなく、すぐに設計問題に適用することができます。

この書籍の著者たちは「Gof（Gang of Four)」と呼ばれデザインパターンの名前の 1 つにもなっている。

Gof のデザインパターンはかなり前のデザインパターンである。そのため、現代では賛否両論あるようで、一部では批判的な意見も挙げられている。というのも、後に説明するが、デザインパターンは Java や Ruby などオブジェクト指向型言語で使用される考え方だ。しかし Gof のデザインパターンが発表されたのは Java がリリースされる 1995 年よりも前、つまりオブジェクト指向プログラミングというものが未熟な時代に発表されたからだ。したがって、現代のプログラミングと合致しない場合もあるという。この意見は `https://qiita.com/irxground/items/d1f9cc447bafa8db2388` より参考にさせていただいた。

2.1.1 デザインパターンの利点・効果

2.1.2 デザインパターンの必要性

3 実際のコーディングでデザインパターンを利用する

3.1 課題説明

■問題 任意のデザインパターンを 1 つ選択し、そのパターンを用いる前と用いた後のコード両方を作成して示し、そのパターンを用いたことの効果を説明せよ。

- 選択したデザインパターン：

3.1.1 演習環境

今回の演習は仮想マシン上で Java を使用し行った。下記に演習時の環境を示す。

- ホスト OS：Window10 Home 20H2
- 仮想 OS：Ubuntu 20.04.2 LTS
- CPU：Intel(R)Core(TM)i7-9700K @ 3.6GHz
- GPU：Nvidia Geforce RTX2070 OC @ 8GB
- ホスト RAM：16GB
- 仮想 RAM：4GB
- 使用言語：Java
  - バージョン情報は下記に示す。

```
openjdk version "11.0.11" 2021-04-20
```

[*1] ドイツに拠点を置く DeepL GmbH によって開発された、ニューラルネットワークによる翻訳を行うサービス。Google 翻訳よりも精度が高く、微妙なニュアンスのある翻訳ができると話題。

```
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)
```

3.2 制作物

3.2.1 デザインパターン前のソースコード

3.2.2 デザインパターン後のソースコード

3.3 考察

4 まとめ

5 巻末資料

参考文献

[1] Erich Gamma (原著), Ralph Johnson (原著), Richard Helm (原著), John Vlissides (原著), 本位田 真一 (翻訳), 吉田 和樹 (翻訳)「オブジェクト指向における再利用のためのデザインパターン」改訂版（ソフトバンククリエイティブ,1999/10/1)