2024 年 8 月 6 日
オブジェクト指向プログラミングレポート

デザインパターン

情報経営システム工学分野 B3
学籍番号 ： 24336488
氏名 ： 本間 三暉

1　デザインパターン

私は State パターンを選択した.

State パターンは，オブジェクトが持つ状態によって異なる振る舞いを提供するためのデザインパターンである．このパターンは，状態が変わることによってオブジェクトの動作が変化するシステムに適している．State パターンを使用することで，状態遷移を明確にし，状態に依存するコードを整理することができる.

1.1　どのようなところに応用できるか

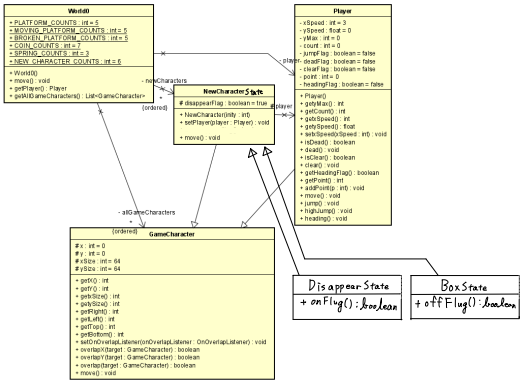
State パターンは以下のような場面で使えると考える.

- ゲーム開発におけるキャラクターの動作管理
- GUI アプリケーションにおけるウィジェットの状態管理
- 通信プロトコルの実装における接続状態の管理
- ワークフローシステムにおけるタスクの進行状態の管理

1.2　適用例

図 (a) に State パターン適用前の NewCharacter を含むクラス図を示す．また，図 (b) に State パターン適応後の NewCharacter を含むクラス図を示す.

(a) 適用前



(b) 適用後

図 1: デザインパターンの適用例

State パターンを適用すると，状態ごとの振る舞いを独立したクラスに分離することができる．状態ごとの処理を分離することで，コードの可読性とメンテナンス性が向上する事がわかる．

2 考察

State パターンを選択した理由は，状態遷移の複雑さを軽減し，状態ごとの振る舞いを分離することでコードの可読性と拡張性を向上させるためである．状態遷移が明確になることで，バグの発見と修正が容易になる．

2.1 メリット

- 可読性の向上: 状態ごとの振る舞いが独立したクラスに分離されるため，コードが読みやすくなる．

- 拡張性の向上: 新しい状態を追加する際に，既存のコードを変更する必要が少なくなる．
- メンテナンス性の向上: 状態ごとの処理が独立しているため，バグの発見と修正が容易になる．

2.2 デメリット

- クラスの増加: 状態ごとにクラスを定義するため，クラスの数が増加する．
- 複雑さの増加: 状態遷移の管理が複雑になる場合がある．

3 感想

State パターンを学ぶことで，オブジェクト指向プログラミングの設計原則を深く理解することができた．状態ごとの振る舞いを明確にすることで，コードの品質を向上させることができると感じた．次年度以降も，このようなデザインパターンの学習を通じて，より良いソフトウェア設計を目指していきたいと思う．