オブジェクト指向的アプローチについて

渡辺大地

1 オブジェクト指向とは

1. オブジェクト指向とは、基本要素が **オブジェクト** であり、単一の実体の中に **データ構造** とその **振る舞い** の両方を持っている 概念のことである。
2. 別の述べ方をすると、オブジェクト指向では **目的の実現** を体系化することから始まる。それに対し、構造化プログラミングでは**手段である部品** を体系化することを念頭に置く。
3. オブジェクト指向による最大のメリットは、その**汎用性**、**拡張性**、**柔軟性** である。なぜなら、システムの目的が変更されることは稀だが、手段は修正や変更が頻繁に行われるからである。
4. オブジェクト指向を念頭に置いた開発を、言語レベルでサポートした言語のことを**オブジェクト指向言語**と呼ぶ。
5. オブジェクト指向言語とは、次のような機能の全部あるいは一部をサポートしている言語のことを言う。
   - データと振る舞い (関数、メソッド) からなる**クラス**の概念。
   - データの**カプセル化** (**隠蔽化**)。
   - **抽象クラス**。
   - **継承**によるクラスの階層構造の定義。
   - **多相性** (**ポリモーフィズム**)。

2 データカプセル化

1. 従来型のデータアクセス

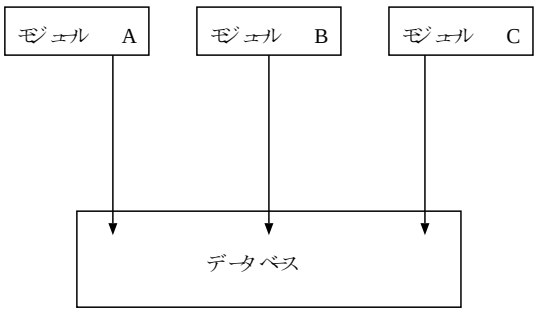

図 1: 従来型のデータアクセス

2. 従来型の欠点

- データ部分に**変更**があると、依存する全てのモジュールの**修正**が必要となる。
- **全て**のモジュールによってデータが破壊される可能性がある。
- 不具合修正時に、全てのモジュールについて**把握**しなければならない。
- あるモジュールの大幅な変更のために、他のモジュールが**影響**を受ける可能性がある。

3. アクセスメソッドによるデータカプセル化

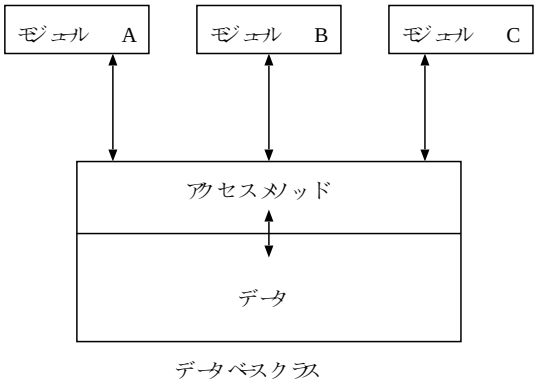

図 2: アクセスメソッドによるデータカプセル化

4. データカプセル化によるメリット
   - データ部分の変更が各モジュールに影響**しない**。
   - アクセスメソッドによる**チェック機能**によって、データの破壊を避けることができる。
   - データアクセスの**履歴**を取ることで、不具合修正が容易となる。
   - アクセスメソッドの**互換性**を保つことで、モジュールとデータの機能拡張を他のモジュールに依存せずに行うことができる。
5. データカプセル化は、設計を誤ると次のような問題が生じる。
   - 大きな**パフォーマンス低下**。
   - 機能の**縮小**や**制限**。
   - モジュールの**自由度**の制限。
   - **行数**や**工数**の増加。

3 継承の利用

1. オブジェクト指向言語の最大の特徴は、クラスの**継承**をサポートしていることである。しかし、継承の概念は初心者には使う場面に悩む場合が多い。
2. 継承は、次の三つの概念を直接表現するのに適している。
   - クラス中の異なる概念の分離。

- **階層構造**による**抽象クラス**の構築。
- **差分プログラミング**の実現。

3. 継承を用いると、クラス中の異なる概念を分離させることができる。

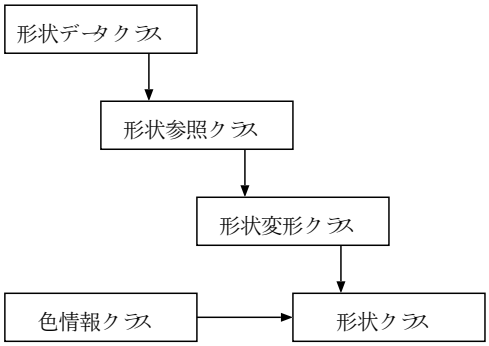

図 3: 継承による概念の分離

4. **抽象クラス**を用いれば、複数のクラス間に異なる実装を持つ共通の**インターフェース** (**仮想関数**) を実現できる。

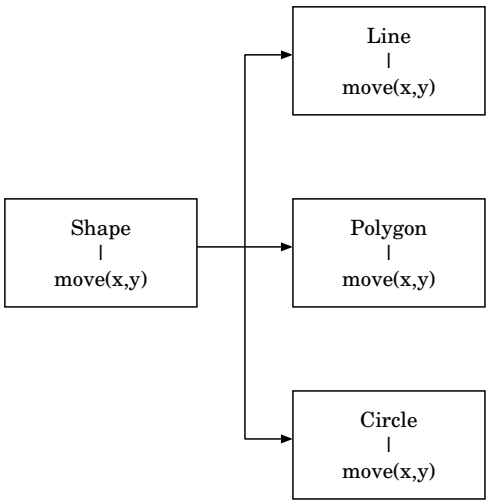

図 4: 抽象クラスと仮想関数

5. **差分プログラミング**とは、完成度の高いクラスを継承して、小規模な変更、修正を行うことで機能を完成させる手法である。例として以下のようなものがある。
   - ウィンドウクラスを継承し、描画部分を付け足す。
   - GUI ボタンクラスを継承し、押されたときの挙動を付け足す。
   - 戦闘シミュレーションなどのユニットクラスを継承し、個別の戦闘アルゴリズムを付け足す。