プログラミングのパターン
：スタイルの紹介

第 1 回ソフトウェアパターン勉強会

Dec. 12 2003

瀬戸川　教彦

スタイルとは…

- 従来から「イディオム」や「コーディングスタイル」「コーディング規則・規約」などの言葉で触れられてきた分野に関するパターン
  - イディオム
  - ベストプラクティス
  - コーディングスタイル／パターン
  - コーディング規則・規約
  - ノウハウ
  - 定石
- プログラミング言語に特化した内容が多い

スタイルとは…（つづき）

- プログラミングにおける日々の戦略について
  - オブジェクトや変数、メソッドの名前をどのように付けるのか
  - 論理を複数のメソッドに分割するには
  - ソースコードを通して明確に意図（そのクラスの作成者としての意図）を利用者に伝えるには
  - 保守性、拡張性をどう考えるか
- 経験豊かなプログラマは自然と実践しているけれど、初心者はなかなか実行できない／学びにくい事柄

スタイルとは…（つづき）

- 繰り返し生じる課題に対して、プログラミング言語を正しく用いて解決するのに役立つ
  - プログラムにおけるメモリ管理
  - オブジェクトの生成の方法
  - メソッドの命名
  - 読みやすさを向上させるためのソースコードの整形
  - 特殊なライブラリの効率の良い使用方法
- プログラミングガイドラインやその他のドキュメントで補ってきた分野でもある

誰にとっての／何時使うパターン？

- プログラマ個人や開発チームが使う
- 開発（プログラミング）をしながら
- 実装レベルのモデリング時
- コードレビューの時
- プログラムの保守計画、拡張計画の検討時

目的、期待する効果

- 自分のプログラミング技術の向上
- 問題の本質理解、解決策考案など洞察力向上
- プログラミング時意図した事を他者に伝え易くする
- 開発チーム内での意思疎通の活性化
- 開発チーム全体の技術力向上
- 開発ノウハウの継承
- 作成されたプログラムの品質向上
- プログラムの拡張性、保守性向上

どんなものがあるか

- Smalltalkベストプラクティスパターン [Bec97]
- Essential Java Style [Lan00]
- C++ FAQ [Mar00]
- C++プログラミングの筋と定石 [Cop94]

Smalltalkベストプラクティスパターン[Bec97]とEssential Java Style[Lan00]

- それぞれSmalltalkとJavaに特化したイディオム／スタイルガイド
- 振る舞い (28)
  - オブジェクトが実際に動き処理する部分をどのように定義すれば読み手に意図を明確に伝えられるか
  - メソッド
  - メッセージ

[Bec97]と[Lan00]（つづき）

- 状態 (24)
  - オブジェクトの振る舞いが完了した後、またはその途中を表す「状態」をどういった戦略でモデリングしていくのか
    - インスタンス変数
    - 一時変数
- コレクション (27)
  - 多用される「データ構造」の一つ「コレクション」について、どう利用していくのか
    - クラス
    - コレクションプロトコル
    - コレクションイディオム

[Bec97]と[Lan00]（つづき）

- クラス (4)
  - クラス（スーパークラスやサブクラス）にどんな名前を付ければ良いか
- フォーマット (7)
  - ソースコードの整形の方法について。どうすればコードの構造をわかり易くする事ができるか

パターンの例

名前	ENUMERATED CONSTANTS
問題	C言語にあるenumのような、安全な列挙型を作れるでしょうか？
解決策	列挙型を表現する専用のクラスを作ります。

例	public class MaritalStatus{
	　private MaritalStatus(){…};
	　publis static final MaritalStatus SINGLE = new MaritalStatus();
	　public static final MaritalStatus MARRIED = new MaritalStatus();
	　:
	}

関連パターン	CONSTANT

パターンの例（つづき）

- ENUMERATED CONSTANTSの動機
  - プログラムの中で状態や情報の種類を表す為に数値を用いるのは危険
  - クラスの利用者は思いがけない使い方をするもの

```
public class MaritalStatus{
    publis static final int SINGLE = 0;
    public static final int MARRIED = 1;
    :
}
```

```
Employee joe =
        new Employee(MaritalStatus.SINGLE);
    :
int bonus = rate * joe.getMaritalStatus();
    :
```

パターンの例（つづき）

名前	INTENTION REVEALING METHOD NAME
問題	メソッドにはどんな名前を付けるべきでしょうか？
解決策	そのメソッドが何（What）を達成するかを示す名前を付けます。どのように（How）実現するかという名前ではありません。
例	×：SomeClass#**hashedSearchFor(anObject)** ○：java.util.Collections#**contains(anObject)**
関連パターン	COMPOSED METHOD, QUERY METHOD, …

パターンの例（つづき）

名前	CONSTANT POOL
問題	複数のクラスで必要とする定数をどうやって定義しますか？
解決策	インタフェースを作成し、その中で定数を定義します。定数を必要とするクラスでは、そのインタフェースを実装します。

例	**public interface PayrollConstants**{
	final int HOURS_PER_WEEK = 40;
	final int HOURS_PER_YEAR = 2000;}
	public class Sarary **implements PayrollConstants**{
	public double calculatePay(double rate){
	return (rate * **HOURS_PER_WEEK** * …);}}

関連パターン	INDIRECT VARIABLE ACCESS, LAZY INITIALIZATION

C++プログラミングの筋と定石 [Cop94]

- C++における「イディオム」と、作成されるソフトウェアの性格（構造）を与えている「文体（スタイル）」を述べてプログラマ自身の熟達を導く解説書
- 記述はあまり形式化されていない
- 各項目に「いつこのイディオムを使うか」という情報がある

C++ FAQ [Mar00]

- C++のオブジェクト指向的な実践的使い方を解説
- プログラミングのうえでの注意点、個々の言語要素についてそれが何であり、C++の意味論に従ってどのように利用すれば良いか説明
- 様式
  - 問題
  - 解決策（一文）
  - 解説文（サンプルコード、問題の詳細や解決策の理由）
  - ※ 個々のFAQには「名前」が無い

C++ FAQ [Mar00]（つづき）

- C++の言語要素に特化
  - newとdelete
  - インライン関数
  - 派生クラス
  - staticの使い方
  - コンストラクタとデストラクタ
  - 仮想関数
  - エラーハンドリングの戦略
  - 正しい継承
  - テスト戦略
  - etc. etc. …

様式の比較

	[Bec97]	[Lan00]	[Cop94]	[Mar00]
名前	◎	◎	×	×
問題	◎	◎	○	○
解決策	◎	◎	○	○
文脈	◎	○	○	○
例	◎	○	○	○
理由	◎	◎	◎	◎

◎：あり ○：項目は無いが文中から読みとれる ×：なし

様式の比較（つづき）

- パターンをコミュニケーションの道具とするなら「名前」は大切
- 解決策が有効な理由や、そもそもの問題の本質を記載する事も重要
  - 単なるＱ＆Ａ集では初心者が学習しにくい（理由が分からない）
- プログラムコード（の断片）があると直感的な理解を助ける
  - 回りくどい文章よりも一目で分かるコードを
- 文章を書くなら「意図」を書くべき
  - 理由、動機なども同じ

組織で利用していくには

- 少人数の開発チーム（２～５人程度）に導入
- 「コーディング規則」ではなく、チームで共有する「ノウハウ集」に
- 作成したプログラムコードを調べ、使われているパターンを洗い出してみる
  - あえて複雑な処理にした点、分かりにくくなった点をパターンに書いてみる（「意図」の伝達）
- その為のテンプレートを用意しておく
  - Wikiのようなツールと一緒に活用
- コードレビューをしよう
  - つっこみのネタとしてもパターンは使える

パターンテンプレート

名前	効果を表す端的な名前
問題	解決したい問題を質問形式で
解決策	簡潔に、要点を押さえて説明。詳細は別途説明文を書いても良い
例	具体的なサンプルコード
動機・理由	そのプログラミングに至った経緯や表したかった意図を詳しく
メモ	議論したい点があれば覚え書きとして。パターンを推敲するのに役立てる

パターンテンプレート（例）

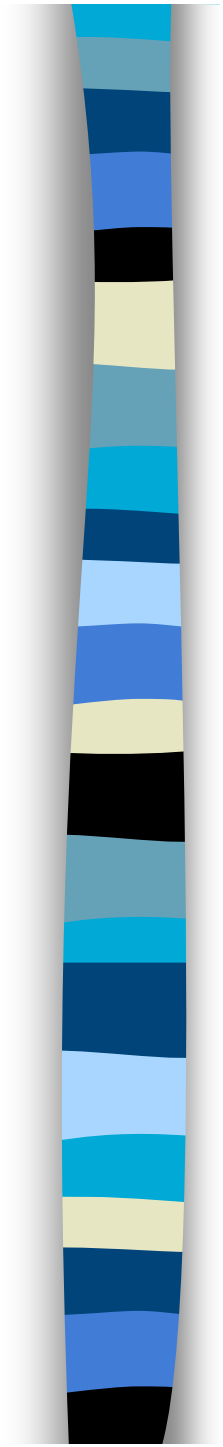
名前	PLUGGABLE QUERY
問題	DBから値を取り出す為の似たような処理をするメソッドの実装をどうやって統一する？
解決策	テーブルのメタデータを参照しながら値を取り出す専用のクラスを作る
例	省略
動機・理由	複数の異なるテーブルから値を取得する為の似たような処理が複数必要だが、コードの統一をしつつ、尚かつ個々のテーブルの定義の変更にも対応できるようにしたい
メモ	個別のSQL文をメンテナンスする方が効率いい？

参考文献

- [Bec97] Kent Beck. Smalltalk Best Practice Patterns. Prentice Hall PTR, 1997. ISBN 0-13-476904-X（邦訳「ケント・ベックのSmalltalkベストプラクティス・パターン シンプル・デザインへの宝石集」ピアソン・エデュケーション, 2003）
- [Lan00] Jeff Langr. Essential Java Style - Patterns for Implementation -. Prentice Hall PTR, 2000. ISBN 0-13-085086-1
- Langr Software Solutions - Resources, http://www.langrsoft.com/resources.html
- [Cop94] J. O. コプリエン. C++プログラミングの筋と定石, アジソンウェスレイ・トッパン, 1994. ISBN 4-8101-8063-8

参考文献（つづき）

- [Aok97] 青木淳. Smalltalkイディオム. ソフト・リサーチ・センター, 1997. ISBN 4-915778-81-9. http://www.sra.co.jp/people/aoki/SmalltalkIdioms/index.htm
- [Bus99] F. ブッシュマン他. ソフトウェアアーキテクチャ ソフトウェア開発のためのパターン体系. トッパン, 1999. ISBN 4-8101-9007-2
- [Mar00] マーシャル・クライン, マイク・ギルウ, グレッグ・ロモウ. C++ FAQ 第 2 版. ピアソン・エデュケーション, 2000. ISBN 4-89471-194-X



ありがとうございました

第１回ソフトウェアパターン勉強会

プログラミングのパターン
：スタイルの紹介

平成１５年１２月１２日

瀬戸川教彦