デザインパターンとその基本構造を意識したプログラム視覚化システム

大城 正典[1,a)]　永井 保夫[1,b)]

概要：高度なオブジェクト指向プログラミング教育において，オブジェクトコンポジションや委譲などに代表されるオブジェクト指向の機能を利用した基本構造を理解することは重要であり，このような基本構造を持っている事例の多いデザインパターンはその教材に適している．一方で，デザインパターンの教育の側面から観ても，個々のデザインパターンの本質的理解と実際の問題に対して柔軟に適用させるためには基本構造の理解は欠かせない．そこで，デザインパターン中の基本構造を強調し意識させる視覚化システムを提案する．また，その利用例を実際の教材を題材に説明する．

キーワード：プログラミング教育，オブジェクト指向プログラミング，視覚化，デザインパターン，Java

Program Visualization System Conscious of Design Pattern and Its Basic Structure

Abstract: In advanced object oriented programming education, it is important to understand the basic structure using object oriented functions such as object composition and delegation. Design patterns with many cases with such basic structures are suitable for the teaching materials. On the other hand, also from the educational aspect of design patterns, in order to understand each design pattern inherently and to apply each design pattern flexibly to actual problems, such basic structures understanding is essential. Therefore, we propose a visualization system that emphasizes and makes conscious the basic structure in the design pattern. We will also explain how to use this system, using actual teaching materials.

Keywords: Programming education, Object oriented programming, Visualization, Design pattern, Java

1. はじめに

オブジェクト指向プログラミングの教育において，学習者を対象としたプログラミング学習環境の提案 [1], [2] や初中等教育におけるオブジェクト指向プログラミング言語を利用した教育に関する研究 [3] が精力的におこなわれている．このようなプログラミング教育においては，プログラムの理解を支援することが不可欠であり，静的な手法で解析したプログラムの振る舞い（プログラムのソースコードからフローチャートや実行パスの生成）やプログラムの実行履歴を取得することによる動的な振る舞いを視覚化情報として提供することで学習者の理解を支援する手法が提案されている [4], [5], [6].

われわれは，このようなオブジェクト指向プログラミングの教育を支援するために，プログラミング初学者を対象としたアプローチと，UML やオブジェクト指向による分析設計（モデリング）に関する知識を活かしたプログラミング学習者を対象としたアプローチに分けて研究をおこなっている．このような問題点に対応するために，視覚化機能を持つプログラミング学習支援システムを Eclipse プラグインとして開発してきた [7].

しかしながら，オブジェクト指向プログラミングでは，プログラムの設計・実装に際して，適切なオブジェクトを発見し，適切な粒度のクラスとしてまとめ，クラスの継承関係やインターフェースを定めることが必要となる．これらを有効に使いこなすためには，ある程度の経験が必要と

[1] 東京情報大学
Uniersity of Information Sciences, 4–1 Onaridai Wakaba-ku, Chiba, Chiba, Japan
[a)] ohshiro@rsch.tuis.ac.jp
[b)] nagai@rsch.tuis.ac.jp

なる．オブジェクト指向プログラミングにおいて，学習者が習得すべきプログラムの意味や適用条件を考慮した良い設計方法をまとめてパターン化したものがデザインパターンである [8], [9], [10]．

われわれは，オブジェクト指向プログラミング教育において，オブジェクト指向の特徴を利用した基本構造を理解することは重要であり，このような基本構造を持っている事例の多いデザインパターンはその教材に適していると考えている [11]．そこで，本論文では，デザインパターン中の基本構造を強調し意識させる視覚化システムを提案する．

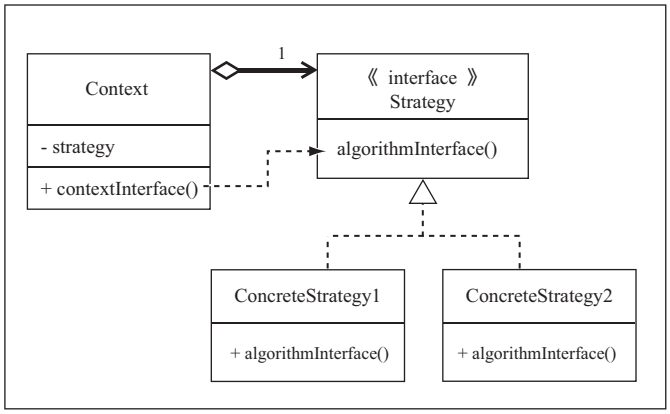
1.1 関連研究と本研究のねらい

デザインパターンを教育に利用したり，デザインパターンの理解を支援するシステムについては，以下の様な先行研究がある．小尻らは，代替設計との拡張性の比較に基づいたデザインパターン設計意図理解支援システムを提案している [12]．本システムでは，デザインパターンを用いたプログラムを基に, デザインパターンを用いない同じ動作をするプログラム (代替設計) を学習者に作成させ, さらにこれらのプログラムの問題設定を変更するシナリオを与えて学習者にデザインパターンを用いた設計と代替設計を拡張させることで, デザインパターンの設計の良し悪しを考察させている．

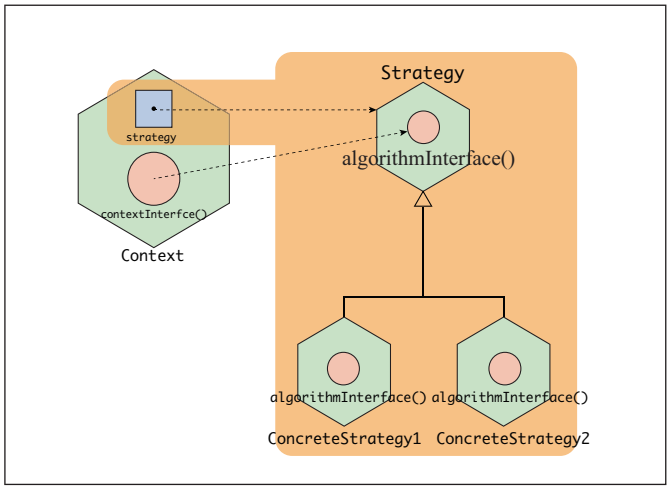
山下らは，オブジェクト指向プログラミングの初学者を対象に，デザインパターンの適用方法を示し，プログラミング段階におけるデザインパターン適用支援ツールの開発をおこなっている [13]．このツールは，プログラミングをおこなっている段階で，設計を考慮したプログラム構造の変更をおこなう場合に，デザインパターン適用のヒントをプログラマーに提供することを意図している．

佐藤らは，役割に基づく計算モデル Epsilon を提案している [14]．これにより，GOF パターンについて，協調動作を理解しやすい形でモジュール化でき，かつパターンに参加するオブジェクト間の関連を弱める形での実装を可能にしている．これにより，コードの再利用が促進され，ソフトウエア進化も支援されている．

筆者らは，視覚化機能を持つプログラミング学習支援システムを Eclipse プラグインとして開発してきた [11], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29]．本システムは Java のソースプログラム上の構造を視覚化する静的視覚化と，プログラムが実行される様子を視覚化する動的視覚化の機能を持つ．著者らは，この視覚化システムを利用した初等プログラミングから設計レベルまでの一貫したオブジェクト指向開発の教育を視覚化によって支援する教育システムを提案した [11]．ここでは，学習題材としてデザインパターン（Strategy パターンと State パターン）を取り上げて，視覚化によるオブジェクト指向設計・実装の学習を示してい



(1) A basic structure in Strategy Pattern represented by UML.



(2) A basic structure in Strategy Pattern represented in our visualizing system.

図 1 基本構造としてのオブジェクトコンポジション．

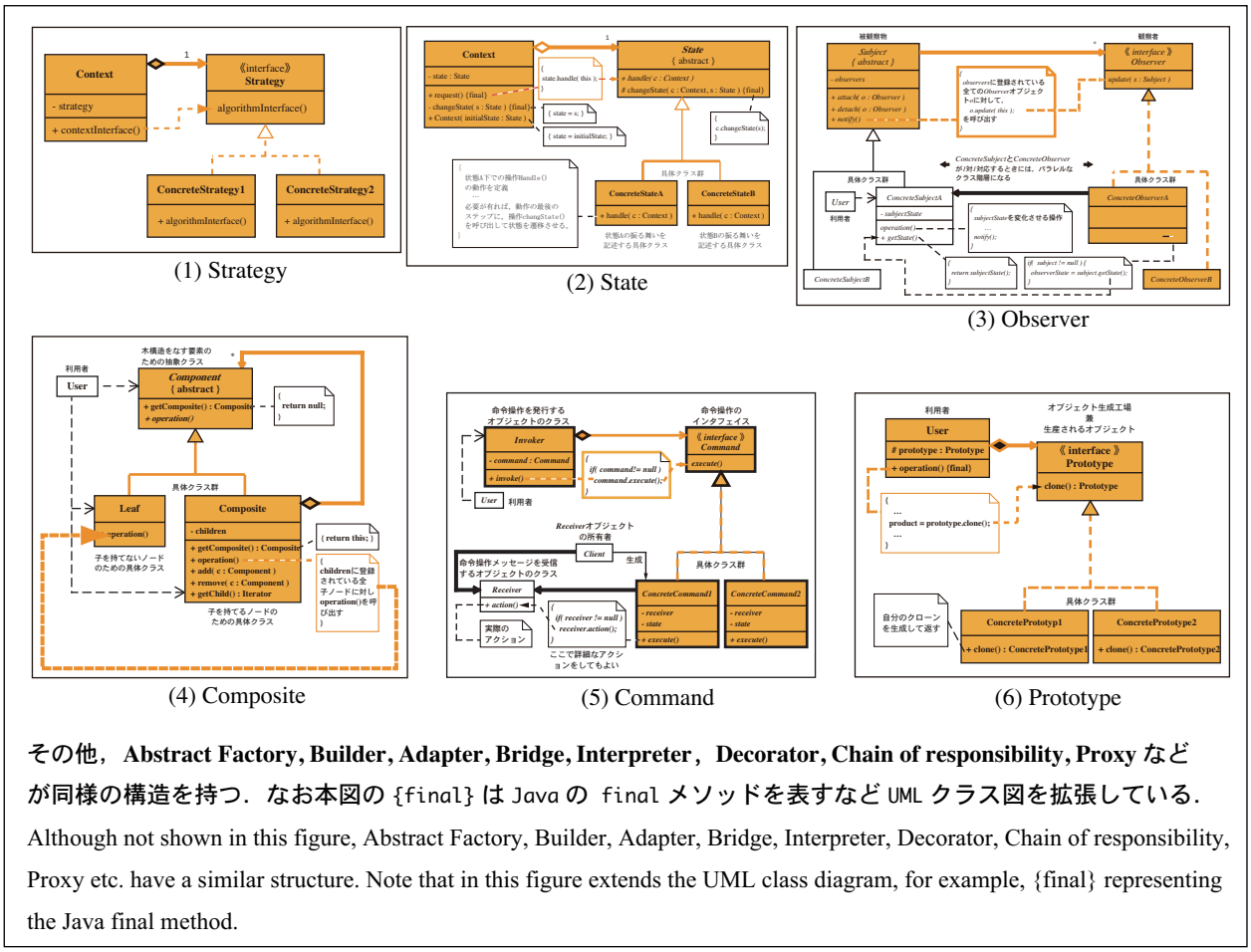
Fig. 1 Object composition as a basic structure.

る．本研究では，この視覚化システムに，多くのデザインパターンが共通して持つ基本構造を利用者に意識させる機能を導入した．

2. 強調する基本構造

本論文で提案するシステムの利用対象者は，オブジェクト指向の基本を理解したものの，まだ本格的な応用の仕方を知らない学習者と仮定する．具体的には，継承・ポリモフィズム・UML クラス図の基本的解釈ができ，多くのデザインパターンやすぐれた設計の中でクラスや継承が利用されていることは理解できても，どのような意図でそれらが組み合わされて各デザインパターンや設計の効果が生まれているかということに関しては本格的には理解できていない段階の学習者である．

多くのデザインパターンや有用な設計の中で，共通に現れる基本構造としては，委譲の形を形成するオブジェクトコンポジションの構造 (図 1) があり，代表的なデザインパ



その他，Abstract Factory, Builder, Adapter, Bridge, Interpreter，Decorator, Chain of responsibility, Proxy などが同様の構造を持つ．なお本図の {final} は Java の final メソッドを表すなど UML クラス図を拡張している．

Although not shown in this figure, Abstract Factory, Builder, Adapter, Bridge, Interpreter, Decorator, Chain of responsibility, Proxy etc. have a similar structure. Note that in this figure extends the UML class diagram, for example, {final} representing the Java final method.

図 2　代表的なデザインパターンに含まれる基本構造としてのオブジェクトコンポジション．

Fig. 2　Object composition as a basic structure included in typical design patterns.

ターンである Strategy パターンはこの形そのものといっていい構造を持つ [8], [30], [31]．たとえば，Gammma らはその著書 [8] で 23 種の代表的なデザインパターンを紹介したが，そのうち 14 種がこの構造を持つ．その中から代表として 6 種のデザインパターンの構造を UML クラス図で図 2 に示す．

そこで，本システムの静的視覚化に，ソースコード内にこのオブジェクトコンポジションの基本構造が含まれている場合には，その部分を強調表示する機能を追加することした．基本的にはソースコード全般に対して分析を行い，本システムの通常の可視化表示はもちろんこの構造を持つ部分に対しては同様の強調表示が行われる．学習者にはこの基本構造を持つデザインパターンの使用例を教材用ソースコードとして示す．この強調表示によって，利用者は Strategy パターン同様の汎用性をもたらすこの構造を意識させられることになる．さらに，その他のクラス間関係から GOF パターンのうち，この構造を持つ 14 パターンに合致すると判断された場合は，そのパターンを構成するクラス群を囲むパターン名付きの枠を表示することとする．

また，オブジェクト指向の中心的な機能である継承を使う場合，スーパークラスが属する一般化された層と，具体クラスが属する特殊化された層 (図 3,[31]) を意識することが重要である．通常，一般化層に関係を閉じ込めることによって，一般化層に再利用可能な構造を作ることができる．もし，この境界をまたぐような参照関係を作ってしまうと，この再利用可能性を阻害することになってしまう．そこで，本システムの静的視覚化にはこの一般化層と特殊化層の境界線を表示し，利用者にこの境界線をまたぐような参照関係が無いこと，もしくはこの境界線をまたぐような参照関係を作ってはいけないことを意識させるようにする．前述した基本構造とともに一般化層と特殊化層の境界線を表示すると (図 4) の様になる．なお，複数のデザインパターンを組み合わせる場合には，実装の都合でそれぞれの一般化層と特殊化層が相対的になる可能性がある．たとえば，実装中のあるデザインパターン構造の特殊化層がもうひとつのデザインパターン構造の一般化層と重なってる，などである．また，多段階の継承を行ってる場合も同様に一般化層と特殊化層は相対的なものとなる．本システムでは，この境界線は継承を起点として発生し，スーパークラス側でのクラス間参照の関係とサブクラス側でのクラ

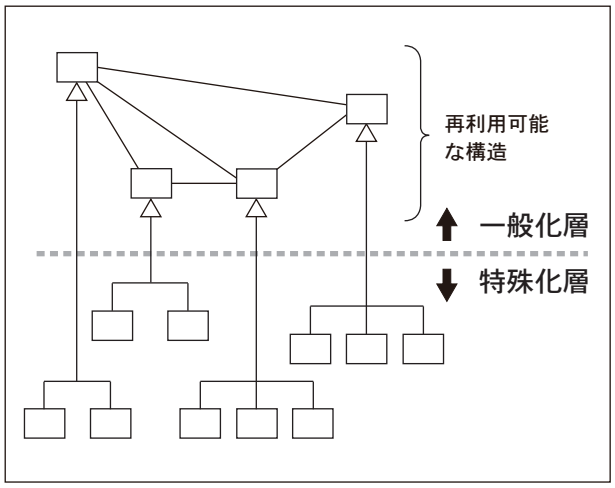


図 3　一般化層と特殊化層.

Fig. 3 Generalized layer and specialized layer.

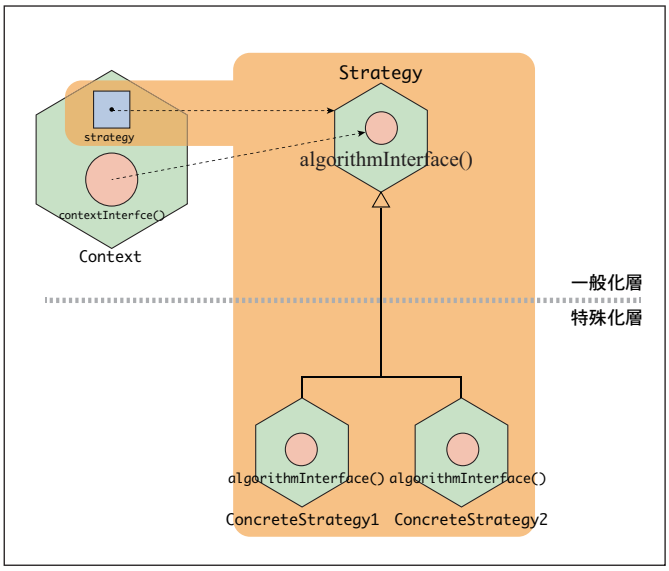


図 4　一般化層と特殊化層の境界線が表示された基本構造.

Fig. 4 Basic structure with borderline between generalized layers and specialized layers displayed.

ス間参照の関係を分離するように描画されるが，現時点では，単一のデザインパターンを用いている教材用のソースプログラムを前提としているため，前述したような複雑な状況による表示上の問題は起こらない．

3. 実際の表示例

ゲームステージにキャラクターを登場させる例で，実際の表示例を示す．まず，最初の段階のプログラム例を図 5 に示す．このソースプログラムに対して表示される静的視覚化の例が図 6 である．Stage クラスは自身が参照している Character クラスとは同じ一般化層に属していないことがこの静的視覚化からわかる．これは，Stage クラスの start メソッドが特殊化層の具体的なクラス名 Knight を参照している，つまり，Stage クラスが特殊化層に依存しているためである．このままでは，Stage クラスは再利用し難い状態，すなわち Stage クラスのサブクラスを作成してバリエーションを得ることが難しいことを意味している．これらを強調するため，Stage クラスの輪郭線と Knight クラスへの参照線が赤く強調表示される．また，委譲の形を形成するオブジェクトコンポジションの基本形は望ましくない状態であることを表すために赤くハイライトされている．これは，一般化層にあるスーパークラス Character が特殊化層から参照されてしまっているためである．

そこで，図 6 の画面上で Stage クラスを本来有るべき一般化層側にドラッグして移動させると，図 7 の様に，Stage クラスから Knight クラスへの参照線が一般化層・特殊化層の境界線に交差し，そのために一般化層・特殊化層の境界線自体も赤く表示され，有ってはならない一般化層から特殊化層への依存が存在することを強調する．そして，オブジェクトコンポジションの基本形は，Character クラスが同一層から参照されている形になるため正常色でハイライトされていることになる．もちろんこれはドラッグした状態の一時的なものであるため，ソースコードを改善して直す必要がある．

最終的に，図 5 のプログラムを Prototype パターンを使って改善したのが，図 8 である．この改善されたプログラムを静的視覚化機能によって表示した例が図 9 となる．オブジェクトコンポジションの構造部分が正常色でハイライトされ，関係クラスが Prototype パターンのラベルがついたフレームで囲まれる．そして，Stage クラスが Character クラスと同一の一般化層に属し，もはや特殊化層に依存していないことがわかる．したがって，Stage クラスは再利用性が高まったことが明らかとなる．例えば，異なる多彩なステージを Stage のサブクラスとして定義しても，この Character 型のサブクラスのオブジェクトを生成する構造は再利用することができる．

4. おわりに

本研究では，オブジェクト指向の基本を理解したものの，まだ本格的な応用の仕方を知らない学習者を対象とした，デザインパターンの基礎構造を意識した視覚化表示の提案を行った．このような視覚化と教材により，多くのデザインパターンなどに共通する有益な効果をもたらす基本構造を学習者に意識させることができると思われる．デザインパターン修得の側面から観ても，各デザインパターンの全体構造をただそのまま覚えるだけではなく，その本質的な構造を理解し，自身が解決しなければならない問題に適用させるために典型的なデザインパターンに必要な変更を加えたり，逆にそのように実際に応用・適用されたデザインパターンを読み取り，理解するためには，このような基本

```
abstract class Character {
  int money;
  public Character( int money ) {
    this.money = money;
  }
  abstract void attack( Character target );
}
class Knight extends Character {
  public Knight( int money ) {
    super( money );
  }
  void attack( Character target ) {}
}
class Samurai extends Character {
  public Samurai( int money ) {
    super( money );
  }
  void attack( Character target ) {}
}

class Stage {
  Character c;
  void start() {
    c = new Knight( 10 );
    // ゲーム処理開始
  }
}

public class Game {
  public static void main( String [] args ) {
    Stage s = new Stage();
    s.start();
  }
}
```

図 5　改善前のプログラム例.

Fig. 5　Example of program before improvement.

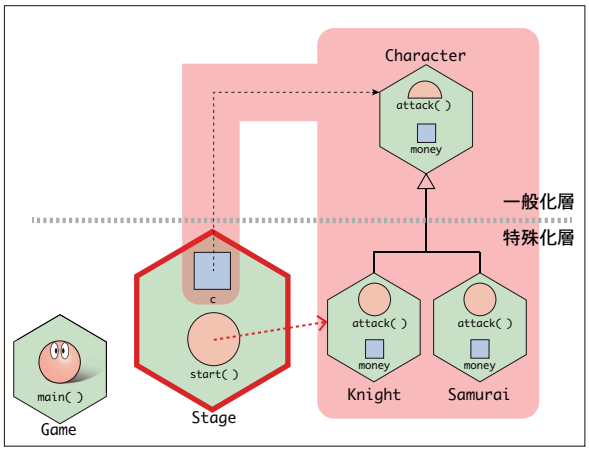


図 6　改善前のプログラムを静的視覚化機能で表示した例

Fig. 6　Example of static visualization before improvement.

構造をしっかり理解するための視覚化・教材は重要であると思われる．そして，このような基本構造を理解してこそ，

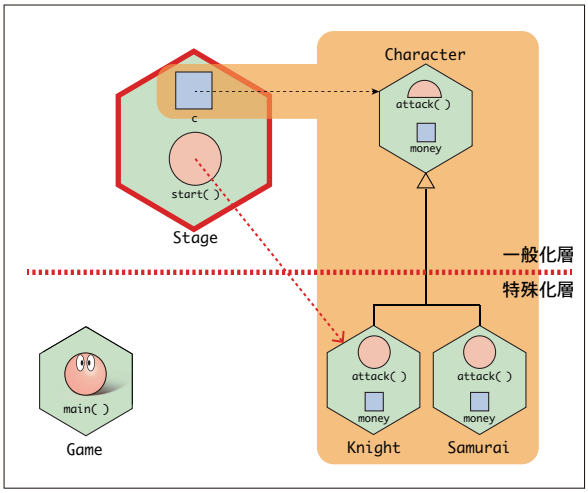


図 7　改善前のプログラムを別の観点から静的視覚化機能で表示した例

Fig. 7　An example of displaying the program before improvement with the static visualization function from another viewpoint.

デザインパターンとしてカタログ化されていない有効なオブジェクト指向設計による構造を自ら生み出せるようになるはずである．今後，本システムの実装を行って実際に授業などで使用して効果を検証したい．

参考文献

[1] 萩庭　崇，永田守男：オブジェクト指向言語のための視覚的プログラム支援環境，情報処理学会ソフトウエア工学研究，Vol. 96, No. 4, pp. 25–32 (1996).

[2] 長　慎也，甲斐宗典，川合　晶，日野孝昭，前島真一，筧　捷彦：Nigari-Java 言語へも移行しやすい初学者向けプログラミング言語，情報処理学会研究報告コンピュータと教育，Vol. 103, No. CE-071, pp. 13–20 (2003).

[3] 兼宗　進，中谷多哉子，御手洗理英，福井眞吾，久野　靖：初中等教育におけるオブジェクト指向プログラミングの実践と評価，情報処理学会論文誌，Vol. 44, No. SIG 14 (PRO18), pp. 58–71 (2003).

[4] 喜多義弘，川添貴議，片山徹郎：初心者を対象にした Java プログラム自動可視化ツールの実現に向けて，信学技報 SS，Vol. 104, No. 570, pp. 19–24 (2005).

[5] 谷口孝治，石尾　隆，神谷年洋，楠本真二，井上　克：プログラム実行履歴からの簡潔なシーケンス図の生成手法，コンピュータソフトウェア，Vol. 24, No. 3, pp. 153–169 (2007).

[6] 竹下彰人，片山徹郎：シーケンス図を用いた実行履歴の可視化による Java プログラムの理解支援に関する考察，信学技報 SS，Vol. 106, No. 426, pp. 43–48 (2006).

[7] 大城正典，永井保夫：初心者から上級者までを対象とした視覚機能を持つプログラミング学習支援システム，東京情報大学研究論集，Vol. 22, No. 1, pp. 22–38 (2018).

[8] Gamma, E., Helm, R., Johnson, R. and Vlissides, J. M.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, Reading, Massachusetts (1994).

[9] 結城　浩：増補改訂版 Java 言語で学ぶデザインパターン入門，ソフトバンククリエイティブ (2004).

[10] 鷲崎弘宜，坂本一憲，大杉直樹，権藤克彦，服部　哲，久保

```
abstract class Character implements Cloneable {
  int money;
  public Character( int money ) {
    this.money = money;
  }
  abstract void attack( Character target );
  public Character clone() {
    Character c = null;
    try {
      c = (Character)super.clone();
    }
    catch ( CloneNotSupportedException e ) {}
    money = c.money;
    return c;
  }
}
class Knight extends Character {
  public Knight( int money ) {
    super( money );
  }
  void attack( Character target ) {}
}
class Samurai extends Character {
  public Samurai( int money ) {
    super( money );
  }
  void attack( Character target ) {}
}

class Stage {
  protected Character prototype;
  Character c;
  public Stage( Character prototype ) {
    this.prototype = prototype;
  }
  void start() {
    c = prototype.clone();
    // ゲーム処理開始
  }
}

public class Game {
  public static void main( String [] args ) {
    Stage s = new Stage( new Knight( 10 ) );
    s.start();
  }
}
```

図 8 Prototype パターンを使って改善されたプログラム例.

Fig. 8 Improved program example using Prototype pattern.

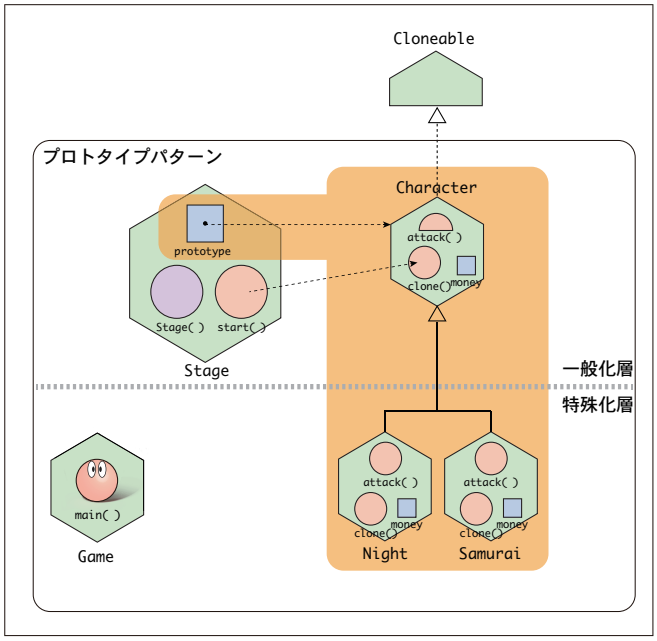


図 9 改善したプログラムを静的視覚化機能で表示した例

Fig. 9 Example of static visualization after improvement.

淳人，小林隆志，大月美佳，丸山勝久，榊原　彰：デザインパターンへのソフトウェア工学的取り組み，コンピュータソフトウエア， Vol. 29, No. 1, pp. 130–146 (2012).

[11] 大城正典，永井保夫：初等プログラミングから設計レベルまでを対象としたオブジェクト指向教育のための支援システムの提案，情報処理学会 情報教育シンポジウム 2011 論文集 SSS2011，pp. 59–66 (2011).

[12] 小尻智子，大江洋希，瀬田和久：代替設計との拡張性の比較に基づいたデザインパターン設計意図理解支援システムとその評価，信学技報，Vol. 114, No. 53, pp. 31–36 (2014).

[13] 山下純司，林　雄二：プログラミング段階のデザインパターン適用支援ツール，北海道情報大学紀要，Vol. 13, No. 2, pp. 67–74 (2002).

[14] 佐藤忠義，玉井哲雄：役割に基づく計算モデル Epsilon を用いたデザインパターン再利用化の促進，日本ソフトウエア科学会第 20 回大会論文集，pp. 25–25（オンライン），DOI: 10.11309/jssstconference.2003.0.25.0 (2003).

[15] 大城正典，永井保夫：オブジェクト指向プログラムの視覚化によるプログラミング教育システム，電子情報通信学会 2009 総合大会講演論文集 情報システム講演論文集 1，p. 212 (2009).

[16] 大城正典，永井保夫：オブジェクト指向プログラムの静的・動的側面を視覚化するプログラミング教育支援システム，情報処理学会第 71 回全国大会講演論文集 (4)，pp. 4–413, 4–414 (2009).

[17] 大城正典，永井保夫：オブジェクト指向プログラムの視覚化によるプログラミング教育システムの改良，電子情報通信学会 2010 総合大会講演論文集 情報システム講演論文集 1，p. 170 (2010).

[18] 大城正典，山川裕子，J., M. K.，松下孝太郎，布広永示：プログラミング学習支援システム CAPTAIN における学習状況把握機能の開発，日本教育工学会「教育実践を指向した学習支援システム／一般」研究会，pp. 81–84 (2010).

[19] 永井保夫，大城正典：モデリングを考慮したソフトウエア教育のためのオブジェクト指向プログラミング教材の検討，電子情報通信学会 2010 総合大会講演論文集 情報システム講演論文集 1，p. 169 (2010).

[20] 大城正典，永井保夫：オブジェクト指向プログラム視覚化教育システムにおけるデザインパターンの視覚化サポート，電子情報通信学会 2011 総合大会講演論文集 情報システム講演論文集 1，p. 139 (2011).

[21] 大城正典，永井保夫：情報視覚化を活用したオブジェクト指向プログラミング教育支援システムの提案，信学技報教育工学， Vol. 110, No. 453, pp. 131–136 (2011).

[22] 大城正典，永井保夫：Eclipse を用いたオブジェクト指向プログラミング教育支援視覚化システムの設計と実装，信

学技報教育工学， Vol. 112, No. 500, pp. 185–188 (2013).
[23] 大城正典，永井保夫：モニタ機能と可視化機能を持った構造指向による漸次的なプログラム作成学習システム，信学技報教育工学， Vol. 113, No. 482, pp. 31–34 (2014).
[24] 大城正典，永井保夫：段階的コーディングガイド機能およびモニタ機能を持つオブジェクト指向プログラミング教育のための視覚化支援システムの提案，信学技報教育工学， Vol. 114, No. 260, pp. 53–58 (2014).
[25] 大城正典，永井保夫：Eclipse 視覚化プラグインによる総合的なプログラミング教育支援システム，情報処理学会情報教育シンポジウム 2015 論文集 SSS2015，pp. 23–30 (2015).
[26] 大城正典，永井保夫：視覚化機能を持つ Eclipse プラグインによる段階的コーディングから動作検証までをサポートするオブジェクト指向プログラミング学習システム，信学技報教育工学， Vol. 115, No. 492, pp. 61–66 (2016).
[27] 大城正典，永井保夫：プログラミング初学者を対象としたオブジェクト指向プログラミング教育システムの提案-オブジェクト指向の基本概念の理解に基づいたプログラムの作成・実行支援機能を中心として-，情報処理学会情報教育シンポジウム 2016 論文集 SSS2016，pp. 114–121 (2016).
[28] 大城正典，永井保夫：シームレス性と文脈依存性を重視した視覚化機能を持つ Eclipse プラグインによるプログラミング学習支援システム，情報処理学会情報教育シンポジウム 2017 論文集 SSS2017，pp. 137–144 (2017).
[29] 大城正典，永井保夫：デザインパターンの基本構造を意識したプログラム視覚化システム，信学技報教育工学，Vol. 118, No. 510, pp. 1–5 (2016).
[30] Pree, W.: *Design Patterns for Object-Oriented Software Development*, Addison-Wesley (1994).
[31] 大城正典：Java と C++によるデザインパターン入門 総括編，pp. 67–76, ソフトバンク (1999).