オブジェクト指向の入門的な開発スタイルを意識させる初級 Java プログラミングのグループ演習の構想

玄馬史也[†1]　　富永浩之[†1]

オブジェクト指向の入門的な開発スタイルを意識させる初級 Java プログラミングのグループ演習を提案する. 各メンバは，出題コードのうち，分担したメソッドを実装する．支援サーバ上で模範コードに組み込まれ，個人テストとして実行される．また，コードの品質を確認するような試練コードによる単体テストも行われる．最終的には，グループとしてのコードとして採点され，個人の貢献度も加味される．これにより，協調作業によるソフトウェア開発の手法を体験する．本論では，その指針を述べる．

A Group Exercise of Preliminary Java Programming for Introduction to Object-Oriented Developing Style

FUMIYA GEMBA[†1]　HIROYUKI TOMINAGA[†1]

We propose a group Exercise of preliminary Java programming for introduction to Object-Oriented developing style. A member of a group selects a method in a problem code. It is embedded into the exemple code and executed in a support server as a personal test. A unit test by an exam code is also performed in order to check non-functional requirement.Finally, collection of answer codes of all members is performed as a productbof the team. A point of a member is calculated by the team score and personal contribution. The educational purpose is to make learners aware of cooperation works in OOP style. In this paper, we discuss issues of educational approach and examples.

1. はじめに

本研究室では，これまで，大学情報系学科の学生を対象にしたプログラミング演習の教育支援システムを幾つか開発し, 実際の授業で運用してきた. これらのシステムでは，C 言語や Java 言語の手続的なゲームを題材として興味と関心を呼び起こしたり，競争型学習のアプローチを取り入れて対抗意識を刺激し，教育効果を高めている．これらのプログラミング演習では，C 言語と Java 言語の手続的な側面である制御構造，データ構造，関数定義に関する文法の理解と語法の活用が主である．また，例題として提示したテンプレートやスケルトンコードからスパイラル的な改良スタイルを誘導している．

しかし，情報系学科のプログラミング関連のカリキュラムのもう 1 つの柱とすべき，オブジェクト指向プログラミングの要素が不十分である．本研究では，これまでの教育支援システムを補完し，オブジェクト指向の開発スタイルを意識させる Java プログラミング演習を提案する. このようなアプローチとしては，やはり，教授者による直接的な指導，学習者同士の相互評価の比重が大きい．しかしながら，本研究では，プログラムの機能要件の外部評価である入出力サンプルでの実行結果の確認を前提としつつ，その拡張として，非機能要件の内部評価に，システムによる自動的な検証や支援を導入していくことを目指す.

2. プログラミング演習の支援システム

情報系学科の 1 年次および 2 年次の必修である C プログラミングの科目では，小コンテスト形式の演習支援システム tProgrEss を実施している. 学生が作成した解答コードをサーバにアップロードさせ，入出力サンプルを与えて実行し，正誤を判定する．ここで，複数の予備テストと 1 つの最終テストからなる実行テスト系列を用意する．予備テストは，中間目標となる部分仕様を設問として，部分点を与えるものである．コーディングへの糸口を与えたり，早期からの動作検証を促したり，段階的詳細化に基づく開発手順を誘導する．最終テストは，完全な仕様の実装を求めるものである．時間調整点と誤答減点を導入し，迅速な解答と慎重な確認を意識させる．また，理解度の大きく異なる受講者が，各自の進捗に応じた実行テストを選択し，それぞれのペースで演習を進められるようにする．ただし，正答となるコードが，適切な書法に沿っているとは必ずしも言えず，教師による目視での確認も必要となっている.

3 年次の必修である応用プログラミングの科目では，ゲーム戦略を AI の題材とする大会形式の演習支援システムとして，WinT と WinG を実施している．WinT は，C プログラミングで，ポーカーなどの独り用のカードゲームで得点を競う．高い役を作るため，手札のどれを捨てるかという着手を実装する．大量のランダムな山札を用意し，平均的な得点を結果とする．予備大会の期間を設け，試作した

†1 香川大学
Kagawa University

戦略コードを大会サーバにアップロードし，順位や得点を公開して，試行錯誤的なフィードバックを行わせる．他の受講者および過去の自分の戦略との比較で，持続的な改良への意欲を促進する．最終大会では，各自の最高得点の戦略が選択される．

WinG は，Java プログラミングで，対戦式のボードゲームの勝敗を競う．現在は，五目並べにオセロのような石取りのルールを加味した五五というゲームを採用している．アップロードした戦略は，多数の他の戦略との対戦が実行される．前者と同様，予備大会の期間を設け，順位と戦績を公開する．ここで，重み付きの勝点度を導入し，間引対戦による結果反映の迅速化，各戦略のレーティングによる戦績補正を実現する．最終大会では，受講者が選択した戦略で改めて総当り対戦を行い，その勝敗を最終結果とする．

上記のプログラミング演習におけるプログラムの評価では，入出力データとの照合，大量サンプルによる実行結果としての得点，プログラム同士の対戦結果など，その外面的な振舞いのみに依存している．しかし，高成績のコードが，必ずしも質が良いとは言えない面がある．そこで，プログラムの設計やコードの書法など内面を評価する方法も重要である．この点については，戦略の設計，コードの解説，各自の実験評価やデバッグ状況をまとめた課題レポートを提出させ，その評価も加味することで補っている．さらに，幾つかのコードメトリクスに着目し，質の良いコードを定量的かつ自動的に評価する手法を検討している．

3. オブジェクト指向の教育の現状

本学科におけるオブジェクト指向プログラミングの教育の現状について述べる．まず，ソフトウェア工学やオブジェクト指向言語の科目が用意され，概念的な事項の学習や Java の基本的なプログラミング手法の教育は行っている．しかし，Java プログラミングの初級的な演習では，マウスによるイベント駆動，オブジェクトの内部状態と遷移，Applet 上での GUI 部品のデザイン，CG やデータ構造に関する既存の API の利用などで，授業回数が満ちてしまい，手一杯である．自分でクラスを設計したり，責務の委譲を検討したりする機会は少ない．そのため，デザインパターンやリファクタリングなど，コードの品質を意識したプログラミングの教育は十分とはいえない．

その後の応用的なグループ演習では，PONG や TRON をベースとする GUI のゲームの改良の課題も実施している．こちらは，グループ間の格差が大きく，クラスの再設計を行うところもあれば，与えられたメソッドの改良のみとなることも多い．WinG においても，ボードゲームの着手を決める各局面での if-then ルールの実現が主で，盤面の状況へのパターンマッチやヒューリスティックな評価値の実装の練習に留まっている．そのため，オブジェクト指向の開発手法を強く意識したプログラミング演習が求められている．

4. アンケートの実施

本研究を提案する上で，オブジェクト指向の用語やそれらを実現する Java の予約語などについて，アンケートを実施した(表 1)．対象は，本大学の情報系学生の 3 年生 42 名である．彼らは，2 年次後期に「ソフトウェア工学 I」を履修済みで，3 年次前期に「オブジェクト指向言語」を履修中である．後期には，応用的なグループ演習に取り組む予定である．「オブジェクト指向言語」の授業では，代表的なオブジェクト指向言語である Java 言語でプログラムを作成するために，必要ないくつかの概念（継承・動的束縛・カプセル化・イベント駆動・スレッドなど）を理解することを目的としている．アンケートは，選択形式とした．選択肢は，(4)理解している，(3)知っている，(2)聞いたことがある，(1)知らない，の 4 段階である．

質問は，大きく 3 つの分野に分かれている．1 つ目は，カプセル化，継承，オーバーライド，インスタンスなどのオブジェクト指向の用語についての質問である．2 つ目は，Redmine，Git，UML，ユースケースなどの，ソフトウェア開発工程や，それらに関するツールについての質問である．3 つ目は，public，extends，interface，static などの Java の予約語についての質問である．

アンケートを実施した結果，平均が 1.79 であり，全体的に理解度が低い傾向にあることが分かった．オブジェクト指向の用語については，カプセル化，継承，インタフェースなどの授業で扱う用語の理解度が，2.17 以上と比較的高かった．しかし，実際に実装する上で必要となってくる，コンストラクタ，デストラクタ，オブジェクトの寿命などについては，1.5 以下と理解度が低かった．その他の用語については，UML，スパイラルモデル，ユースケースなどの理解度が，2.1 以上と比較的高かった．その反面，ソフトウェア開発の実際の現場で必要となる，チケット管理システムの Redmine や，バージョン管理システムの Git などに対する理解度は，1.42 以下と低かった．また，Java の予約語については，public，extends，static などの理解度が，2.0 以上と比較的高かった．しかし，授業ではあまり扱わない，protected，abstract，interface，try～catch などの理解度は，1.5 以下と低かった．

以上のことから，基本的な概念や用語は，ある程度理解していることが分かった．しかし，実際に使う上で必要な知識に関しては，理解度が低かった．座学を中心とした，通常のオブジェクト指向の授業や演習では，その知識を具体的にどう使うのか，実感が伴っていない可能性が考えられる．よって，オブジェクト指向の品質確認を含む，入門的な Java プログラミングのグループ演習として，本研究を提案する．

表 1 アンケートの質問項目と回答の平均値

1-1	オブジェクト指向の用語について	1.68
1	カプセル化	2.17
2	継承	2.17
3	抽象クラス	1.78
4	具体クラス	1.75
5	インタフェース	2.28
6	多重継承	1.53
7	オーバーライド	1.78
8	オーバーロード	1.69
9	コンストラクタ	1.53
10	静的コンストラクタ	1.31
11	デストラクタ	1.33
12	インスタンス変数	1.72
13	インスタンスメソッド	1.72
14	クラス変数 (静的変数)	1.97
15	クラスメソッド (静的メソッド)	1.89
16	イベント駆動 (イベントドリブン)	1.25
17	変数のスコープ	1.50
18	変数の有効期間 (生存期間)	1.56
19	オブジェクトの寿命	1.31
20	例外処理	1.69
21	スレッド処理	1.42

1-2	その他の用語について	1.95
1	Redmine	1.42
2	Git	1.86
3	UML	2.06
4	スパイラルモデル	2.14
5	プロトタイピング	2.11
6	テストコード	1.78
7	ユースケース	2.31

2-1	Java の予約語について	1.73
1	public	2.50
2	protected	1.44
3	private	1.72
4	abstract	1.25
5	extends	2.14
6	implements	1.97
7	interface	1.47
8	new	1.78
9	super	1.89
10	@Override	2.25
11	final	1.36
12	static	2.03
13	try～catch～finally	1.31
14	runnable	1.11

表 2 tProgrEss と本提案の比較

	tProgrEss	本提案の演習
対象者	初級(B1～B2)	中級(B3～B4)
プログラム言語	手続型 C 言語	オブジェクト指向 Java 言語
参加形態	個人	グループ(3～4 人)
学習目的	文法や書法 算法や語法	クラス継承 メソッド構成
手法	実行テスト系列	実行テスト(内部) 検証テスト(外部)
解答の進め方	時間調整点に 対する個人戦略	リーダーの下の 分担と計画

5. 本提案の OOP 開発演習の出題構成と意図

本研究では，小コンテスト形式による初級 C のプログラミング演習支援システム tProgrEss を受けて，初級 Java のプログラミング演習支援システムを提案する．制御構造やデータ構造などの手続型の基本的なプログラミング技能は前提とし，オブジェクト指向プログラミングのコーディングスタイルを習得させる．対象者は，情報系学科の 3 年次程度とし，参加形態も，個人ではなく，3～4 人程度のグループとする．

本提案の演習では，グループ単位での穴埋め方式による，ソースコードの共同作成としてのプログラミング課題を出題する. 課題は, メソッドやクラス単位で分割されており，グループのメンバは, 分担してコーディングを行っていく. 課題では，教員は模範コードを用意し，適当な箇所を隠蔽して穴空きのコードとして学生に提示する(図 1). 学生には，表 3 のような種別に基づき，問題の一部または全てを隠蔽し，公開する．

演習は，大きく前半と後半に分ける．前半では，主に基礎的な内容を出題する(表 4). 後半では，主に応用的な内容を出題する(表 5). 基礎的な内容とは，具体クラスの各メソッドの実装，変数やメソッドへのアクセス修飾子の記述などである．応用的な内容とは，抽象クラスの実装，新たな下位クラスの実装などである．

プログラムの外部評価は，単体と統合の 2 種類の実行テストで行う．単体の実行テストでは，メンバ各自の提出したコードと，模範コードの他の空欄部分を結合し，実行結果を判定する．統合の実行テストでは，グループ全員の提出したコードを結合し，実行結果を判定する．その際，単なる入出力データの照合だけではなく，GUI プログラムにおいては，様々なイベントに対する振舞いを確認する．

さらに，実装の適切さを確認するため，試練テストを行う．学生の提出したコードと試練コードを結合し，試練プログラムとする. 試練コードは, テスト駆動開発における，テストコードおよびその拡張である．正誤だけではなく，非機能用件としての処理速度や応答性など，パフォーマンスも測定する．後者では，提出されたコードを内部的に検証するための内容を記述し，内部評価も行う．例えば，カプセル化について，オブジェクト変数へのアクセス修飾子を正しく設定しているかどうか，試練コードからの攻撃的なアクセスで確認する．また，継承について，具体クラスが，与えられた抽象クラスから正しく特化されているかどうか確認する．

6. クイックソートを題材とする出題例

ソートのアルゴリズムを例題とした，演習の流れについて説明する．前提として，学生は，Java の文法事項などは既に知っており，C 言語ではソートのアルゴリズムを実装

したことがあることとする．

学生の実装するクラスは，クイックソートを実現する QuickSort クラス(図 2)，挿入ソートを実現する InsertSort クラス(図 3)，QuickSort クラスと InsertSort クラスを汎化した SortTypeAbstract 抽象クラス(図 4)，高度なクイックソートを実現する，QuickSort クラスを特化した ExtremeQuickSort クラス(図 5)である．ExtremeQuickSort クラスは，クイックソートで分割した配列の要素数が，ある一定以下になると挿入ソートへ切り替わるソートアルゴリズムを実現するものである．

前半では，SortTypeAbstract クラスを継承していない，QuickSort クラスと InsertSort クラスを分担して実装する．後半では，SortTypeAbstract 抽象クラスと ExtremeQuickSort クラスを実装する．SortTypeAbstract 抽象クラスを実装させる問題では，SortTypeAbstract 抽象クラスを継承した，完成された BubbleSort クラスを学生に提示する．学生は，BubbleSort クラスを参考に，汎化を満たすような SortTypeAbstract 抽象クラスを実装する．また，汎化に関する問題を解いた後に，ExtremeQuickSort クラスを実装させる．以上のような流れで演習することにより，学生は，具体クラスだけの実装や使い方という下流工程から，理想的なクラス設計，抽象クラスの概念，継承の意味や有用さ，クラスの汎化，特化などのオブジェクト指向的なソフトウェア開発工程を体験することができる．

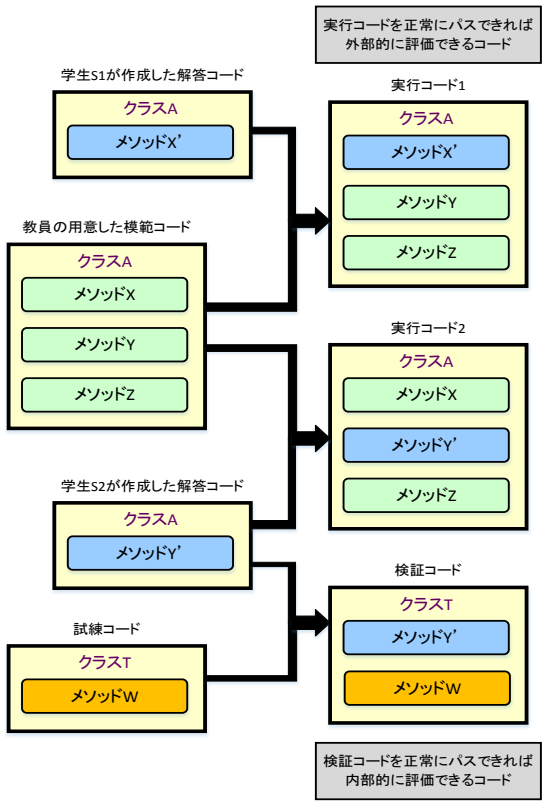

図 1 グループ解答の概念図

表 3 学生に対する公開条件に基づく問題の種別

●	完全な答えを公開
◎	変数の定義やヒントを公開
○	概略としてスケルトンコードのみ公開
△	完成されたメソッドの概略のみ公開 (使用可)
×	存在を隠す

表 4 前半に公開される問題例とその種別

QuickSort	クイックソートクラス	●
QuickSort()	コンストラクタ	●
sort()	内部のソートを呼出し	●
quickSort()	クイックソートで整列	◎
getPivot()	ピボットを取得する	○
swapArr()	配列の要素を交換する	○
InsertSort	挿入ソートクラス	●
InsertSort()	コンストラクタ	●
sort()	内部のソートを呼出し	●
insertSort()	配列を挿入ソートで整列	◎
swapArr()	配列の要素を交換する	△

表 5 後半に公開される問題例とその種別

SortTypeAbstract	ソートの抽象クラス	◎
swap_count	交換回数	◎
SortTypeAbstract()	コンストラクタ	○
sort()	抽象メソッド	●
getSwapCount()	交換回数を取得する	○
swapArr()	配列の要素を交換する	●
BubbleSort	交換ソート	●
sort()	内部のソートを呼出し	●
bubbleSort()	配列を交換ソートで整列	●
InsertSort	挿入ソート	◎
sort()	内部のソートを呼出し	●
sort(hig, low)	ソートを範囲指定で呼出し	○
insertSort()	配列を挿入ソートで整列	●
QuickSort	クイックソート	◎
sort()	内部のソートを呼出し	●
insertSort()	配列を挿入ソートで整列	◎
getPivot()	ピボットを取得する	●
ExtremeQuickSort	高度なクイックソート	◎
sort()	内部のソートを呼出し	●
getSwapCount()	交換回数を取得する	○
quickSort()	クイックソートで整列	◎

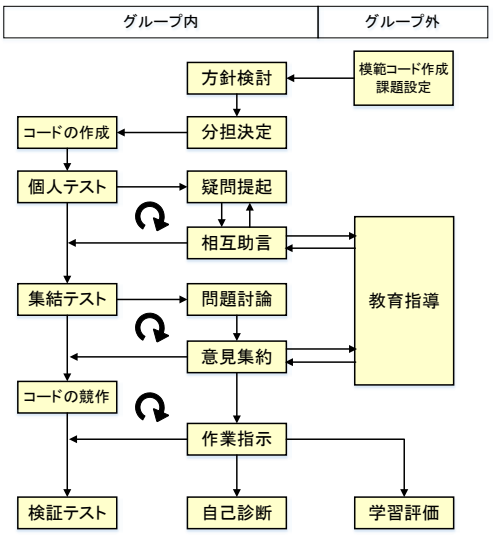

図 6 グループ演習の進行モデル

7. グループ演習の進行モデル

本提案でのグループ演習は，図 6 のような流れで進行する．演習の開始前に，教員は模範コードを用意し，課題を設定する．具体的な演習内容をコメントとして課題コードに付加し，各グループに出題する．また，課題コードが満たすべき仕様として，サンプルの入出力も提示する．

学生は，教師からの課題提示に対し，グループ内で相談して作業分担を決め，リーダーが各自に指示を出す．課題によっては,データ型を揃えるなどの全体の方針も定める．各メンバがコーディング作業を進め，個人テストに至る．個人テストでは,メソッドやクラス単位でコードを提出し，模範コードと結合して実行することで入出力の検証を行う．

次の段階では，個人テストを行い，文法事項やエラーなど，個人作業で生じた問題点をグループ内で互いに助言し合う．ここでは「分かる人が分からない人に教える」という 1 対 1 のコミュニケーションが中心となる．グループ内で解決できない時は，教員に質問し，回答を得る．

各々の担当した部分において，個人テストが終了した場合，メンバ全員のコードを合わせて集結テストを行う．集結テストでは,メンバが解答したメソッドやクラスの中で，使用するものを決定し，1 つのプログラムとする．個人テストでエラーが生じなくとも，全体の整合性が満たされなければエラーが生じたり，実行結果が正しくない状態が発生したりする．ここでは，各自の解答コードを 1 つの答案コードとして共有する．これを題材として，メンバ全員で討論を行い，リーダーが意見を集結して，最終的なプログラムを完成させる．教員は，各グループを巡回し，適切なアドバイスを与える．

個人テストや集結テストで，外部評価として入出力の検証を終えた場合は，検証テストに挑戦する．検証テストでは，「この変数は private であるべき」や「このようなクラス関係であるべき」といった内部評価として，非機能要件を満たしているかどうかを検証する．

課題の評価としては，各グループの共同作業の内容を，個人テスト，集結テスト，検証テストにより，内外から評価する．また，教員は，最終的なプログラムを目視で確認し，採点する．その他にも，プログラムに関する理解度を図るために，各グループに対してコードレビューや口頭試問などを行う．学生は，教員からの指摘を受けるだけではなく，他のグループと議論し，互いに問題点や良かった点を挙げ，グループ外からも幅広い知識を得る．

8. システムの検討

本提案のシステムの検討にあたっては, 稲見 iProgrEss を参考とする．iProgrEss は，C 言語プログラミングのグループ演習において，協調学習のアプローチを取り入れたシステムである．問題として，幾つかの穴埋め箇所を設定したコードを提示し，グループで分担して解答する．グループのリーダーを中心にして，メンバの分担を決める．各自のコードをサーバにアップロードすると，その部分だけ模範コードに埋め込んで実行される．最終的には，全員の実装部分を集めて，模範コードの該当箇所を置き換えたものがグループとしての解答コードとなる．このような仕組みを実現するため，クライアント側に，出題コードを共有表示するホワイトボード付チャットを取り入れたエディタを開発し，サーバ側に，コードの一部を差し替えて実行する機能を開発した．これらを参考にして，支援システムの仕様を確定させる．

9. おわりに

本提案の演習支援では，オブジェクト指向プログラミングの特性をシステムの機能にも活かし，クラスの継承や派生などで，模範コードの一部の差替えなどを実装する予定である．また，JSON 形式を用い，演習の進捗過程のスナップショットを保存し，各自の貢献度の把握や外部からの助言の誘導などの機能を取り入れたい．実際の演習での実践を行い，教育効果を検証する．

参考文献

1) 富永浩之，西村智治: 実行テスト系列を取り入れた小コンテスト形式の初級 C 演習－学生の得点状況の時系列分析による活性度の推定－，情処研報, Vol.2012-CE-116, No.15, pp.1-8(2012).
2) 玄馬史也，吉田亜未，大川昌寛，山田航平，富永浩之: ポーカー戦略を題材とする応用 C プログラミング演習の支援と実践－最終大会の提出コードの特徴分析－，信学技報, Vol.114, No.121, pp.17-22(2014).
3) 玄馬史也，吉田亜未，大川昌寛，山田航平，富永浩之: ポーカー戦略を題材とする応用 C プログラミング演習の支援と実践－最終大会の提出コードの特徴分析－，信学技報, Vol.114, No.121, pp.17-22(2014).

4) 山田航平, 富永浩之: ボードゲームの戦略プログラミングを題材とした Java 演習支援－間引き対戦の導入と提出戦略の詳細分析－, 情処研報, Vol.2013-CE-124, No.10, pp.1-6(2014).
5) 玄馬史也, 富永浩之: ポーカー戦略を題材とする応用 C プログラミング演習の支援と実践－大会運営サーバ WinT の提出状況とコード比較の機能の追加－, 情処研報, Vol.2015-CE-128, No.9, pp.1-6(2015).
6) 稲見望, 富永浩之: プログラミング演習のためのグループチャット型 CSCL, 信学技報, Vol.101, No.706, pp.107-114(2002).

```
01  public class QuickSort extends
02    SortTypeAbstract {
03    //----  ソート
04    @Override
05    public void sort(int[] arr) {
06      quickSort(arr, 0, arr.length-1);
07    }
08    //----  ソート本体
09    private void quickSort(int[] arr,
10      int low, int hig) {
11      int i = low;
12      int j = hig;
13      int pivot = getPivot(arr, low, hig);
14      if (low >= hig) { return; }
15      while (i <= j) {
16        while (arr[i] < pivot) { i++; }
17        while (arr[j] > pivot) { j--; }
18        if (i > j) { break; }
19        swapArr(arr, i, j);
20        i++; j--;
21      }
22      if (low < j) { quickSort(arr, low, j); }
23      if (hig > i) { quickSort(arr, i, hig); }
24    }
25    //----  ピボットの決定
26    protected int getPivot(int[] arr,
27      int low, int hig) {
28      return arr[low];
29    }
30  }
```

図 2 QuickSort クラス

```
01  public class InsertSort extends
02    SortTypeAbstract {
03    //----  ソート
04    @Override
05    public void sort(int[] arr) {
06      insertSort(arr, 0, arr.length-1);
07    }
08    public void sort(int[] arr,
09      int low, int hig) {
10      insertSort(arr, low, hig);
11    }
12    //----  ソート本体
13    private void insertSort(int[] arr,
14      int low, int hig) {
15      for (int i = low+1; i <= hig; i++) {
16        int j = i;
17        while (j >= low+1 &&
18          arr[j-1] > arr[j]) {
19          swapArr(arr, j, j-1);
20          j--;
21        }
22      }
23    }
24  }
```

図 3 InsertSort クラス

```
01  public abstract class SortTypeAbstract {
02    private int swap_count;    // 交換回数
03    //----  コンストラク
04    public SortTypeAbstract() {
05        swap_count = 0;
06    }
07    //----  ソート
08    public abstract void sort(int[] arr);
09    // 交換回数の取得
10    public int getSwapCount() {
11        return swap_count;
12    }
13    //----  要素の交換
14    protected void swapArr(int[] arr,
15      int p1, int p2) {
16      int tmp = arr[p1];
17      arr[p1] = arr[p2];
18      arr[p2] = tmp;
19      swap_count++;
20    }
21  }
```

図 4 SortTypeAbstract 抽象クラス

```
01  public class ExtremeQuickSort extends
02    QuickSort {
03    private final int limit = 5;
04    private InsertSort insert_sort;
05    //----  コンストラクタ
06    public ExtremeQuickSort() {
07      insert_sort = new InsertSort();
08    }
09    //----  ソート
10    @Override
11    public void sort(int[] arr) {
12      quickSort(arr, 0, arr.length-1);
13    }
14    //----  交換回数の取得
15    @Override
16    public int getSwapCount() {
17      return super.getSwapCount() +
18        insert_sort.getSwapCount();
19    }
20
21    //----  ソート本体
22    private void quickSort(int[] arr,
23      int low, int hig) {
24      int i = low;
25      int j = hig;
26      int pivot = getPivot(arr, low, hig);
27      if (low >= hig) { return; }
28      if (hig-low <= limit) {
29        insert_sort.sort(arr, low, hig);
30        return;
32      }
33      while (i <= j) {
34        while (arr[i] < pivot) { i++; }
35        while (arr[j] > pivot) { j--; }
36        if (i > j) { break; }
37        swapArr(arr, i, j);
38        i++; j--;
39      }
40      if (low < j) { quickSort(arr, low, j); }
41      if (hig > i) { quickSort(arr, i, hig); }
42    }
43  }
```

図 5 ExtremeQuickSort クラス