

グループワークによるソフトウェア開発に向けた オブジェクト指向教育実践の試み

松浦 佐江子 *

本学では2002年度から3年後期にグループワークによるソフトウェア開発の実験を行っている。本実験ではオブジェクト指向開発方法を取り入れ、モデリング記述言語 UML を用いて要求分析・システム分析・システム設計を行い、Java 等のオブジェクト指向言語を用いてプログラムを開発する。このような開発をグループで効率よく行うためには、オブジェクト指向開発の特長を十分理解していることが必要である。本稿ではグループワーク実験に先立って行われたオブジェクト指向開発の特長を理解し、そのメリットを実感することを目的としたプログラム開発実験について報告する。

An Object-Oriented Concepts Education for Experimental Software Development Group Work

Saeko Matsuura

As a part of software engineering education, we conducted a software development experiment with the junior students. They analyzed given requirements and designed a software system which implements the analyzed requirements along an Object-Oriented method. Requirements analysis and system design are performed using a modeling description language UML, and the program is developed using an Object-Oriented Programming language Java. In order to make a success of such development group work, it is required that the students understand the feature of object-oriented development enough. We planned an Object-Oriented concepts education for experimental software development group work. This paper reports a plan of the experiment and its results based on their reports.

1. はじめに

本学では2002年度から3年後期にグループワークによるソフトウェア開発の実験を行っている。2002年度は9グループ中8グループがオブジェクト指向開発を採用し、モデル記述言語としては UML(Unified Modeling Language)[1]を、開発言語としては Java を用いて開発を行った。この実験における問題点として、GUI(Graphical User Interface)・データベース接続等の設計・実装に関する知識の不足、分担して作業するために必要な約束事に関する知識の不足、オブジェクト指向の特長の理解不足、コミュニケーションツールの欠如、グループ作業に対する意識向上への工夫等が挙げられる[2]。

グループワークによるソフトウェア開発実験には、プロジェクトマネジメントの観点から2つの重要な要素がある。1つは開発方法論をソフトウェア開発の基盤とすることであり、もう1つは作業状況把握等のプロセス管理の支援方法を工夫することである。すなわち、前述の実験における問題点はこれらの観点から考察すると、開発方法論をソフトウェア開発の基盤とするための基礎教育に関する問題およびプロセス管理支援の問題と捉えることができる。

2003年度の実験に際しては、前者の問題に対処するために、3年前期に GUI 技術に関する知識の教育、オブジェクト指向における作業分担を体験すること、オブジェクト指向技術の特長を体験的に学習することを目的とした実験を行った。本稿ではこの実験の内容ならびにオブジェクト指向によるソフトウェア開発教育においては何をどのような手順で教えることが重要であるかを考察する。

[*] 芝浦工業大学 システム工学部 電子情報システム学科
Shibaura Institute of Technology Department of Electronic Information Systems

後者の問題に対するコミュニケーションツールや意識の向上のための工夫については、別途報告する予定である。

2. オブジェクト指向開発をグループワークに用いる上での問題点

オブジェクト指向は対象世界の「もの」や「抽象的な概念」をオブジェクトとして捉えて対象世界のモデリングを行う。オブジェクト指向開発の1つの特長は、要求分析から実装までのすべてのフェーズをオブジェクトとオブジェクト間の関係によって捉え、上流工程における概念と下流工程における概念をクラスによって一貫して定義することができることである。クラスとは具体的なオブジェクトに共通な性質を定義したものであり、オブジェクトの属性とオブジェクトが受け付ける操作から構成されている。この一貫性により、第一に変更に頑健なプログラムを構成できる可能性がある。第二には、作業分担可能なモジュールを構成することが可能である。第三には要求分析から実装までのフェーズ間でのフィードバックが行いやすくインクリメンタルな開発が可能である。しかし、変更に頑健であり、再利用性の高いプログラムを作成するためには、対象のモデリング方法（何をオブジェクトとするのか、オブジェクト間の関係をどのように定義するか）を工夫しなければならない。オブジェクト指向言語である Java の講義においては、オブジェクトのカプセル化によりモジュールの独立性を高めること、再利用性を高めるためのオブジェクトの構成、抽象クラスやインタフェースを用いた設計による拡張性や変更に対する頑健性といったオブジェクト指向の特長についても講義しているが、講義と単純な演習を行うだけではこれらの特長のメリットを学生が実感することは難しい。しかし、これらの特長を理解し、その意義を実感しないと、オブジェクト指向開発のメリットを「ある程度の規模のシステムを複数人で開発すること」に活かすことが難しくなり、グループによる共同作業の遂行を妨げることになる。

実際に 2002 年度の実験においても、以下のような状況が発生した。

- クラス構成が十分でなく、インタフェースやプログラミングの規約を決めなかったため

に、多くの不整合が生じ作業の後戻りが多く発生した。

- 分担が可能な状態にクラス構成を決定することができなかったために分担を諦め、少人数でプログラムを作成した。

このような状況を避けるためには、オブジェクト指向開発における対象のモデリングの特長を理解し、その意義を実感し、共同作業における注意点を認識することができるようなプログラミングを体験する必要がある。

本論文では、このような目的を達成するためにわれわれが実施した実験の計画、内容およびその評価について報告する。

3. 実験計画

本実験における受講した学生の前提知識、実験プロセスの設定、実施計画を以下に述べる。

3.1 前提知識

実験は、2 年後期にオブジェクト指向言語の講義（言語は Java）を履修した学部 3 年生 70 名で実施した。全員がオブジェクト指向開発ならびに UML に関する講義 1 コマも同時期に受講している。しかし開発方法の講義と本実験が同時進行であるため、要求分析・システム分析・システム設計についての経験はない。なお、オブジェクト指向言語では以下の内容を全 13 回で講義している。

- | |
|----------------------|
| ① オブジェクト指向の概要 |
| • なぜ、オブジェクト指向なのか？ |
| ② オブジェクトの定義 |
| • クラスとインスタンス |
| • フィールド・メソッド・コンストラクタ |
| ③ オブジェクトの関係と再利用性 |
| • 継承とポリモルフィズム |
| ④ オブジェクトの関係と再利用性 |
| • 抽象クラス・インタフェース |
| ⑤ オブジェクトの可視性 |
| • 修飾子・パッケージ |
| ⑥ 例外 |
| ⑦ ファイル操作と入出力 |
| ⑧ マルチスレッド |
| ⑨ デザインパターン |

3.2 実験プロセスの設定

オブジェクト指向開発教育のポイントの1つはオブジェクト概念の把握による妥当なクラス構成の習得にある。プログラムが一旦作られて、使われないあるいは変更されない場合にはその構造的欠陥は問題にはならない。大学におけるプログラミング演習は小規模であり、このような状況に相当するため、プログラムの構造を問題にすることは少ない。しかし、一般にはプログラムの構造的欠陥が保守や再利用の妨げとなっているのは事実である。このような構造的欠陥を減少させ、保守性・再利用性を高めることがオブジェクト指向の目的である。上流工程においてオブジェクト指向分析・設計を入念に行って、構造を十分に練った上でプログラミングを行うことが理想的ではあるが、要求が曖昧であったり、分析・設計の漏れを皆無とすることは難しいため、分析・設計の方法を学ぶだけでは十分ではない。一方、リファクタリング[3]はプログラムの外部的振舞いを保ったまま内部の構造を改善し、下流方向からオブジェクト指向プログラムの再利用性・保守性を高める技術である。このように、上流、下流両方向からの構造の改善を学ぶことにより、プログラムの構造的欠陥を減少させる方法を習得することがオブジェクト指向開発の教育には必要である。

オブジェクト指向開発の初学者である学生がオブジェクト指向開発の特長を学ぶためには、自己のプログラムの構造的欠陥に気づくことが重要である。このためには一旦作成したプログラムを作り直す作業を通じて、自己の作成したプログラムの構造的欠陥を認識することを体験する必要がある。さらに、プログラムの作成にデザインパターン[4]に基づく制約を与えることにより、保守性・再利用性の高いプログラム構成を学習することができる。

また、オブジェクト指向開発プロセスの標準化を目指すUP(Unified Process)[5]においても、対極的なXP(eXtreme Programming)[6]に代表されるアジャイルな開発においても、要求からプログラムまでのプロセスを反復的に行うことが推奨されている。これは前述のように一方向からだけでは構造的欠陥の少ない保守性、再利用性の高いプログラムを作成することが難しいことによるものである。

本実験における受講生の前提は、前述のとおりJavaプログラミングの初学者であり、オブジェクト指向分析、設計に関しては未修得である学生を対象としている。そこで、1つのプログラムをリファクタリングしながら洗練していくプロセスを実験に取り入れるることにより、学生にプログラムの構造的欠陥を認識させることとした。

3.3 実施計画

スケジュールは以下のように計画した。各授業時間は2コマ(計180分)である。

第1回	実験内容の説明(オセロゲームをオブジェクト指向でプログラミングする) 基本プログラム(Ver.1)の考え方の解説
第2回	基本プログラム(Ver.1)の設計 テスト項目の抽出
第3回	javadoc 用コメントの定義によるドキュメントの作成 テストプログラムの定義
第5回	デモンストレーション審査① レポート提出: Ver.1
第6回	Applet の講義・演習
第7回	AWT の講義・演習
第8回	インタフェースの変更(アプレットの定義等の機能追加) クラス構成規約の追加① プログラムの入れ替えの実験①
第9回	戦略に関する講義 インタフェースの定義 クラス構成規約の追加②
第10回	デモンストレーション審査② レポート提出: Ver.2
第11回	プログラムの入れ替えの実験② コード・リーディング(2人1組) 問題点の報告 対戦型への変更(コンピュータ対コンピュータ)
第12回	対戦用プログラムの配付 対戦準備、動作確認作業
第13回	トーナメントによる対戦 ディスカッション(全体) 最終レポート提出: Ver.3

本実験の受講生に対する評価は以下の3つの方法で実施し、総合的に評価を行った。

A) 学生は、第5回、第11回にそれぞれVer.1,

Ver.2 のプログラムと考察を中間レポートとして提出し、提出プログラムのデモンストレーションを行いながら質問に答える。

- B) 学生は、第 13 回に Ver.3 のプログラムを提出し、トーナメント方式で対戦を行う。トーナメント方式による対戦では、教員が配付した対戦用プログラムに対し、各自が定義した戦略のクラスのみを追加してコンパイル・実行を行う。コンパイル・実行できることで要求された規約を満たしていることが判断できる。対戦の成績も合わせて評価する。
- C) 学生は最終レポートとして機能追加とプログラムの変更についての考察を行う。
- D) 学生は最後にアンケート形式の評価レポートを提出する。

ここでデモンストレーション審査は以下の要領で行った。

- 教員が用意した「テスト項目」について学生が実際にプログラムを動作させ、その実行結果を教員が審査する。
 - 1 グループ (19 から 20 名) 毎に担当教員が審査する。
 - 1 人当たりの持ち時間は 8 から 9 分とする。
- 本審査の目的はつぎのとおりである。
- 要求したプログラムが作成されているかを「テスト項目」に従ってプログラムを実行させ、プログラムの機能を検査する。
 - プログラムの構造を javadoc によって生成されたドキュメントに従って説明させることにより、学生のプログラム構造に対する考え方を把握すると同時にそのプレゼンテーション能力を訓練する。

なお、javadoc はドキュメンテーションコメントを記述することにより、Java のソースプログラムから API 仕様を生成し、Java の標準 API 仕様と同じ形式のドキュメントを生成する仕組みである。ソースから生成することでソースと仕様書の不整合を減らすことができることから、レポートの提出ではこの仕組みにより、ソースコードにコメントを記述して、審査時にこのドキュメントを使用して説明を行うこととした。

4. 実験内容

4.1 課題

オセロゲームのプログラムを作成することが課題である。基本プログラム (Ver.1) はよく知られた基本ルールを満たすオセロゲームを実現するプログラムであり、以下の要件を満たすものである。

- 人が石を置く位置は標準入力から入力する。
- 盤面の初期状態を標準出力に出力する。
- 1 手打つごとに、盤面の状態とスコアを標準出力に出力する。
- コンピュータの戦略は「打てる位置で空いているところにランダムに置く」とする。

基本プログラムへの要求の特徴は、標準入出力をユーザインタフェースとすることで、それまでの講義の知識を用いて十分にプログラムを作成できることである。

4.2 実験のプロセス

GUI の知識を習得し、オブジェクト指向の特長を認識するために、基本プログラム (Ver.1) から Ver.2、Ver.3 へと 3 回の開発を行った。各バージョンのプログラムは動作するものであり、デモンストレーション審査の対象となる。以下に、各バージョンに対する変更要求を示す。

基本プログラムから Ver.2 への変更点

- 人が石を置く位置は盤面の図上のマウスクリックによるものとする。
- 盤面の初期状態および 1 手毎の盤面の状態をアプレットを使って図示する。
- 人对コンピュータだけではなく、コンピュータ対コンピュータの対戦も可能とする。

Ver.2 から Ver.3 への変更点

- コンピュータの石を置く戦略の切り替えを可能とする。

基本プログラムは前述の要求を満たすものであればよく、クラスの構成に関する制約はない。

Ver.2 に対する要求は「基本プログラムにおける標準入出力を使用したインタフェースの GUI 化」である。この要求を段階的に実現するために以下のステップでクラス構成に関する制約を与えた。

第 1 ステップ

- 基本プログラムから盤面のデータと入出力だけを取り出し、初期状態の表示・位置の入力・入力した位置に石を置く・盤面の更新、

が行えるようにする。

- ゲームのルールはまだ考慮せず、単に交互に白と黒が置くことができればよいものとする。

ただし、プログラムはつぎの条件を満たすものとする。

- インタフェースの拡張にも共通に利用できるように Board, Constant, Position のクラスを定義する。
- 標準入出力を使うプログラムのパッケージを Standard とする。
- Standard には DisplayBoard, InputBoard, TestBoard (main を含むクラス) を定義する。

ただし、指定したクラスにはその責務の説明とそれぞれのフィールドおよびメソッドのシグネチャ¹を与える。

第2ステップ

- 盤面データから盤面を描画する。内部データと描画の連携を図る。
- MouseListener (Java API) を implements することにより、マウスイベントを盤面データに反映する。

ただし、プログラムはつぎの条件を満たすものとする。

- Board, Constant, Position のクラスを利用する。
- GUI を使うプログラムのパッケージを Applet とする。
- Applet には DisplayBoard, InputBoard, AppletBoard (Applet を継承したクラス) を定義する。

第3ステップ

- クラスの責務の分割や、抽象クラス・インタフェースの導入により、クラス構成の整理を行う。
 - 入出力インタフェースを定義する。
 - スコアに関する責務をもつクラス Score を定義し、スコアを表示するために各パッケージの DisplayBoard クラスを変更する。
 - 人とコンピュータを同等に扱うためのプレイヤークラスを定義する。抽象クラスと

しての Player クラスに対し、Human クラスと Computer クラスを定義する。

- ゲームの進行の責務をもつクラス GameProgress を定義する。

Ver.3 に対する要求は「コンピュータの戦略として、基本プログラムにおける『打てる位置で空いているところにランダムに置く』以外の戦略を定義する。」である。この要求を実現するために、戦略の切り替えが可能であるようなクラス構成の変更 (Strategy インタフェースの定義とその実装) をプログラムの制約とした。

なお、石を置く場合の戦略としては、「一番多く相手の石をひっくり返せる位置に置く」や「相手が最善の手を打ち返してくると想定して次の手を考えるとといった先読みを行う」方法がある。これらを実装するためには、プレイの状態を表すゲーム木を作成し、ミニマックス法等により着手を決定する。効率の問題も考慮する必要があり、これらのアルゴリズムについての解説を行った。

4.3 共同作業に関する実験

本実験では、GUI 化が終了した第8回、インタフェースの定義等のクラス構成の規約の実装が終了した第11回に、以下のような2人1組の共同作業を行った。

プログラムの交換実験

第8回には、以下に示すプログラムの交換を行った。

- Standard および Applet パッケージ以外のクラスファイルを交換して、コンパイル・実行を行う。

この結果、パッケージ宣言の相違、メソッド名の相違、フィールドのアクセスメソッドの定義の有無による障害が発生した。

第11回には、以下に示すプログラムの交換を行い、その時に生じた問題点を2人で検討し、検討結果をレポートとして報告する実験を行った。

- Applet パッケージを交換してコンパイルする。
- コンピュータが石を置く戦略を定義した RandomStrategy クラスを交換する。

インタフェースの定義が正しく行われていた学生は、問題なく交換した結果のプログラムをコンパイル実行することができた。

¹ メソッドのシグネチャとはメソッドの名前とパラメータの組のことである。

コード・リーディング

学生は他の学生が作成したプログラムを読んで理解するという作業の経験がほとんどない。作成者同士で、プログラムの内容を比較することにより、自己のプログラムの説明能力を高める・他人のプログラムと比較し評価する経験をすることが可能になる。

4.4 GUI の学習

本実験における GUI に関する講義の目的はつぎのとおりである。

- ① 1つのアプリケーションに対して、ユーザインタフェースの切り替えを可能とする方法を学習する。
- ② オセロゲームに必要な、描画、マウスクリック等の技術を習得する。

①の学習を行うために標準出力への文字列の出力をグラフィックス・オブジェクトへの描画に置き換える以下のような課題を用意し、段階的に②の技術を習得できるようにした。

(ア) 時計クラスの定義

標準出力に時刻を時間：分：秒の単位で文字列表示する。

(イ) アプレットを継承したテキスト時計クラス

ブラウザに時刻を時間：分：秒の単位で文字列表示する。

(ウ) アナログ時計クラス

ブラウザに時刻を目盛盤・短針・長針・秒針を円や線を描画して表示する。

(エ) デジタル時計クラス

ブラウザに時刻を画像の配置により表示する。

この課題では、(ア)の基本機能をベースに下線部を変更する。この時ベースとなるクラスはそのまま利用できるようにすることにより、ユーザインタフェースの切り替え方法を学習する。

オセロゲームの基本プログラムを GUI 化するためにはつぎの項目を学習する必要がある。

- 「標準入力から人が石を置く位置を入力する」を「マウスでクリックした位置をとる」に変更する。このためには、盤面を描画する位置やどの枡がクリックされたかを知る必要があり、座標計算を行わなければならない。

- 「標準出力に盤面を文字列で表示する」を「盤面の矩形や線による描画またはイメージの描画」に変更するためには、石の円の塗りつぶしによる描画またはイメージの描画、文字列の表示、色の設定を行わなければならない。

前者はマウスイベントの例題により、後者は前述の「時計の課題」によって学習する。

5. 実験の評価

5.1 評価方法

本実験に対する学生の取り組みを調査し、本実験の効果を評価するために、実験終了時に全受講生に対して評価レポートの提出を義務付けた。評価レポートは、56の質問から構成され、選択方式（複数回答可もあり）と自由記述形式で回答を行うものである。評価レポートを実施するに当たって調査の目的を明記し、事実を正確かつ丁寧に記入するように指導した。質問は大きく分けてつぎの5つに分類できる。それぞれの質問の概要および意図は以下のとおりである。

- ① 技術の理解度：Java 言語および GUI ライブラリの理解度が実験により高まったか否かを択一式で回答する。
- ② 実験への取り組み：各段階において授業時間以外にプログラム作成にかけた時間を択一式で、各段階の課題によって生じた問題点の把握状況を記述式で回答する。
- ③ 実験計画：課題の難易度、例題の有効性、講義内容に関する評価を択一式で回答する。
- ④ グループワーク：プログラムの交換における問題点の把握状況を記述式で回答する。
- ⑤ オブジェクト指向開発方法：オブジェクト指向開発の有効性、プログラム開発に対する意識の変化について記述形式で回答する。

5.2 評価結果の考察

前述の質問に対する回答は以下のとおりである。回答者数は62名であり、表の数値の単位は「人」である。

① 技術の理解度：

表1のとおり、Java 言語および GUI 関連のライブラリに対する理解度はかなり向上していることがわかる。

表1 理解度の変化

(人)	Java		GUI	
	開始前	終了時	開始前	終了時
a	1	9	0	3
b	11	43	4	36
c	10	7	3	17
d	29	3	18	6
e	11	0	37	0

a. 十分理解していた b. ある程度理解していた
c. どちらともいえない d. あまり理解していなかった
e. 全く理解していなかった

② 実験への取り組み：

学生が各バージョンのプログラムを作成するのに授業時間以外に費やした作業時間は表2のとおりである。基本プログラムの作成は実験開始時までの知識で可能なことから、プログラミング技術に関する注意は与えていない。また、Ver.2では制約を満たすようにVer.1のプログラムを変更しなければならず、特に制約が多かった第3ステップでは、責務の明確ではないクラス構成からの変更は苦労が多かったと考えられる。このことから、これらのフェーズに多くの時間がかかっている。

表2 授業時間以外の作業時間

時間	Ver.1	Ver.2 Step1	Ver.2 Step2	Ver.2 Step3	戦略	対戦 修正
0～1	1	1	4	2	2	7
1～3	8	8	7	5	13	15
4～6	8	20	24	14	17	18
7～9	21	13	11	16	18	12
10～	24	20	16	25	12	10

各ステップにおいては、自分のプログラムと与えた制約間の違いの修正にかなりの時間を費やしたようである。記述形式の回答からは、クラスの関連がうまく整理できていないため、規定されたクラスにこれまでのメソッドを分割することに苦労した様子が伺える。

「この実験によってプログラム開発に対する意識の変化はあったか？」という質問に対して60人(97%)が「はい」と答えている。

コーディングの前に設計を行うことの重要性を感じた・制約のもとにプログラムを作ることは難

しいが共同作業するためには必要である・プログラムは個人が理解できればよいというものではなく他人も理解できるということが重要である・オブジェクト指向によって共同開発がうまくいきそうである・汎用性の高いプログラムを書くということを意識するようになった・抽象クラスやインタフェースの使い方がわかった・オブジェクト指向は作りながら改良することでより洗練されたプログラムを作成できるといった意見があり、本実験の目的はおおよそ達成できたと考える。

③ 実験計画：

GUIの講義における「時計」の例題は58人(94%)がオセロゲームのGUI化に役に立ったと答えており、GUIの導入の例として適切であったという意見が多かった。われわれの意図したクラスの切り替えができるようなクラスの分割の意義を理解したことを記述した学生も5人(8%)いた。

表3 難しかった作業

どの作業が難しかったか？(複数回答)	
基本プログラムの作成	24
アナログ・デジタル時計の作成	19
オセロゲームのGUI化	39
クラスの整理(リファクタリング)	39
戦略の定義	30
対戦への準備	16

表3より未経験のGUI化やリファクタリング作業を難しく感じたようであるが、「実験を行って自分にとって得るものはあったと思うか？」という質問に対して、「強くそう思う」「そう思う」が合わせて61人(98%)であった。得たものとしては、Javaが使えるようになった・GUIが作成できるようになった・オブジェクト指向が理解できた・複数人で開発するときの問題点がわかったといった回答が多数であった。

④ グループワーク：

他人とプログラムを交換してコンパイルした結果、多くのエラーが生じることにより、プログラムを統合することが容易ではないこと、および、単純な名前の違いから、メソッドの意味の相違まで、「同じクラス構成でも約束事なしに他人のプロ

グラムと統合する場合、何が問題を生じさせているのか」を体験できたようである。

⑤ オブジェクト指向開発方法：

表4 基本プログラムのクラス数

1 個	2 個	3 個	4 個	5 個
0	3	3	16	12
6 個	7 個	8 個	9 個	10 個～
12	6	2	2	6

基本プログラムの設計におけるクラス数は表 4 のとおりである。また、基本プログラムにおいて抽象クラスを1個以上使用した学生は12人(19%)であり、インタフェースを1個以上使用した学生は4人(6%)であった。なお、配付した対戦用のサンプルは、抽象クラス1、インタフェース3、クラス12の構成である。

基本プログラムの作成にあたり、「クラスの抽出作業が基本プログラムの作成に活かされたか？」という質問に対して43人(69%)が「そう思う」と答えている。しかし、21人(34%)が活かされたとはいえないと考えており、その理由としてオブジェクトの捉え方とJavaでの実装の方法がわからなかったため、プログラムの構成と食い違いが生じたことを挙げている。この段階ではオブジェクトの概念をまだうまく捉えられていないことがわかる。

「オブジェクト指向開発はプログラム作成に役立ったと思うか？」という質問に対して、57人(92%)が「はい」と答えている。クラス構成をうまく行くと変更範囲が限定されることや、クラスの役割を明確にすることで、プログラム全体の見通しがよくなり、変更点を見つけることが容易になった、というように、リファクタリングプロセスを経ることによって、プログラムの保守性、再利用性についてオブジェクト指向の特長を理解し、そのメリットを実感できたと考えられる。

6. まとめ

本実験により、受講者は以下の項目について学習を行うことができた。

① プログラムの設計

- 要求文を満たすオセロゲームプログラムの設計と実装
- プログラム規約の設定(変数名、定数名、

メソッド名)

- ユーザインタフェース(GUI)の設計

② アルゴリズムの設計

- オセロゲームの基本ルールの定義
- 戦略の定義

③ 置ける場所に置いている、正しく反転している、正しくスコアを計算している、置けないときはパス等のプログラムのテスト項目を抽出し、そのテストプログラムを定義する。

④ javadoc 用のコメントを作成することにより、プログラムの可読性について考察する。

⑤ 機能追加に従い、抽象クラスやインタフェースを利用した定義を行うことにより、プログラムの保守性・再利用性を考察する。

⑥ アプレットによるGUIの実装方法を習得する。

現在、本実験の受講生は「はじめに」で述べたグループワークによるソフトウェア開発の実験を行っている。実験の経過を観察すると、「言葉の使い方の統一を考慮して、要求分析、システム分析、システム設計を行っている」「早い時期から、テスト仕様の詳細やプログラミング規約の設定を考慮している」「インタフェースを意識した設計を行っている」といった傾向が見受けられる。これは今回の実験の質問「今回の実験を踏まえて、複数人でソフトウェア開発を行う場合に注意しなければならないことは何だと思えるかを記述せよ。」に対して回答したことを実践していると考えられ、本実験がオブジェクト指向の特長を理解させることに寄与したと言える。

参考文献

- [1]UML : <http://www.omg.org/uml/>
- [2] 松浦, 相場, グループワークによるソフトウェア工学教育の試み, 情報処理学会, CE68-1, pp.1-8, 2003.
- [3]M.Fowler, Refactoring : Improving The Design of Existing Code, Addison-Wesley,1999.
- [4]E.Gamma,R.Helm,R.Johnson,J.Vlissides, Design Patterns, Addison-Wesley,1995.
- [5] I.Jacobson, G.Booch, J.Rumbaugh, The Unified Software Development Process, Addison-Wesley, 1999.
- [6]K.Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley, 1999.