# Java言語演習科目における対戦型ゲーム課題の設計と実践

水口 充[1]

**概要**：オブジェクト指向プログラミングの導入として Java 言語の基礎を扱う授業における，学習者の理解度を評価するために実施するプログラミング課題について，プログラムの自由度を適度に持たせ，また学習者のモチベーションを維持させることを目的としてプログラミングゲーム形式の課題を実施した．具体的には，学習者は与えられた親クラスを継承して，与えられたメソッドやオブジェクトを利用して思考ルーチンを実装した自分のクラスを作成する．そして，作成したプログラム同士で対戦を行うゲームとした．課題を実施した結果，学習者の幅広い理解度に対応可能かつ十分に自由度が高く，意欲的に取り組むことによって課題を通じたプログラミングの理解および開発経験を促すことができた．

# Design and practice of a programming game exercise for a Java course

Minakuchi Mitsuru[1]

***Abstract:*** We designed and practiced a programming game exercise in order to evaluate understanding of students in a basic Java language course, which was organized as an introduction to the object-oriented programming. We designed the game so that it had adequate variations in programming, and so that it was able to keep giving motivation to the students. In particular, the students made their own classes that implemented intelligent algorithm by inheriting a given super class and by using given methods and objects. The students' programs played each other in the game. As a result, we confirmed that the game was enough flexible to be applicable to a wide range of students' understanding. We also confirmed that it was able to improve programming knowledge and practical skill of students through developing programs actively.

## 1. はじめに

演習系のプログラミングの授業において，学習者の理解度を評価するためにプログラミング課題を実施する必要がある．しかし，特に受講生の多い授業の場合，課題内容の設定が難しい．例えば自由にプログラムを作成，という内容にすると，受講生はどの程度のプログラムを作成すればよいか困惑するであろうし，出題者側が理解して欲しいと考えているポイントを押さえたプログラムになっていない可能性も高くなる．書籍やインターネット上に公開されているプログラムをコピーするケースもあり，検出が手間であるという問題もある．逆に，詳細な仕様を与えてプログラムを作成する内容とすると，受講生間で似たようなプログラムになるため理解度の評価が画一的になってしまう．また，受講生間でのコピーの検出も難しくなる．適度に自由度を持たせつつ，理解度を測りたいポイントを盛り込んだ課題内容とする必要がある．

一方，授業の進行と独立して長めの提出期限を設定するならば，学習内容の復習と応用を兼ねたプログラミング経験を積む内容とすることで学習効果を期待できる．更に，成績に反映される課題となると学習者は義務的に取り組むことになるが，楽しめるような課題内容にすることで意欲的に取り組むことも期待できる．

そこで，ゲーミフィケーションの考えに基づき，思考ルーチンをプログラムして戦わせる形式の課題を実施した．本稿では，授業の特性および課題の目的に沿って設定した要件と，それらを満たす課題内容の設定，および実施結果について述べる．

[1] 京都産業大学コンピュータ理工学部
Kyoto Sangyo University, Faculty of computer science and engineering

## 2. 背景

### 2.1 授業の概要

実践した授業は 2012 年度秋学期に開講した，本学部 2 年次配当の必修科目であった[*1]．この授業ではオブジェクト指向プログラミングの導入として，Java 言語の基礎を学習することを目的とした．具体的な内容としては，基本的な文法を C 言語との違いを中心に解説した後，オブジェクト指向の基礎（クラスとオブジェクトの概念，クラスメンバ，継承），例外処理，クラスライブラリの使い方（コレクション，入出力，GUI，ネットワーク）を一通り扱った．

授業は週 2 回 15 週の全 30 回あり，各回は 1.5 時間であった．履修登録者は 169 名であった．教室には教員による解説用のモニターが 2 人に 1 台設置されていて，受講生は各自のノート型パソコンを使用して演習を進めた．

授業は文法などの新規の内容を説明し，練習問題を示して各自でプログラムを完成させた後，解説を行う形式で進めた．また，提出課題を適宜設定し，授業時間外に取り組むよう指示した．

### 2.2 課題の要件

本授業の成績評価のために，オブジェクト指向と Java 言語に関する基礎的な知識を問う筆記試験と，実践的なプログラミング能力を問うプログラム課題（最終課題）を課した．最終課題の内容は授業の終盤に提示して，授業終了後の一定期間の後を提出期限とした．提出期限については，平常時の提出課題は速やかにフィードバックを行うために短く設定（内容に応じて数日〜1 週間程度）したのに対し，最終課題はプログラミングに突き詰めて取り組む体験をさせることも狙って提出期限を長く設定した（本稿で報告する課題では 6 週間）．

授業の特性および最終課題の目的より，以下の項目を課題の要件として設定した:

**要件 1. オブジェクト指向プログラミングの理解を測る・深める**

本学部のカリキュラムにおいて，本授業はオブジェクト指向の概念を学ぶ最初の科目となっている．基本的な理解度は筆記試験で評価するが，実践的な理解度を測り，かつ課題を通じて理解を深めさせるために，基本的なオブジェクト指向プログラミングの要素を盛り込む．具体的にはクラスの書き方，フィールド・メソッド・コンストラクタの定義，オブジェクト変数の使い方，継承の使い方，などである．

**要件 2. デバッグ，テスト作業を含めたソフトウェア開発を経験させる**

受講生の多くは自主的なプログラム開発をほとんど行っていないため，デバッグやテスト作業の経験が少ない．そこで，試行錯誤的にアイデアを実装し，実行させて確かめることを繰り返すような内容とする．

**要件 3. モチベーションを維持させる**

課題の提出期限を長く設定しているため，長期間に渡ってモチベーションが維持できる内容とする．また，手間を掛けるほど良いプログラムとなることが実感できるようにする．

**要件 4. 幅広い理解度に対応可能とする**

受講生のプログラミングの理解度の幅は大きい．そこで，見よう見まねでも最低限のプログラムは完成でき，一方でプログラミングに習熟しているほど高度なプログラムが作れる内容とする．

**要件 5. 自由度を高くする**

誰が書いても似たようなプログラムになるような内容の課題では習得度の細かな評価ができない．またコピーによる提出課題の検出が困難となり，誘発される恐れもある．プログラムの自由度が高いことが望ましい．

## 3. 方針

先に挙げた課題の要件の 3.（以下，要件 3. のように記す）を満たすためには達成感を継続して与えることが重要となる．そのためには目標が明確であり，かつ段階的に達成感が得られる内容とするのがよいと考えた．そこでゲームを題材にすることにした．これは近年盛んになっているゲーミフィケーションの流れを取り入れたものでもある．

プログラミング教育におけるゲーミフィケーションの方法として，ゲーム形式で学習するものと，ゲーム自体を作るものが考えられる．

例えば CodeSpell[1] [*2]はゲームを進めるために Java 言語のプログラムを書くもので，前者の方式にあたる．ゲーム形式の学習は継続的な学習には適しているが，正解が存在するため要件 5. を満たすことが難しく，本稿の目的である最終課題には向いていないと考えた．また，準備に多大な労力を要するという欠点もある．

後者のゲーム自体を作る方式としては，まったく自由にゲームを制作させる方式から，ルールや条件などの仕様を与えて制作させる方式，プログラムの一部を与えて残りを完成させる方式までが考えられる．自由に制作させる方式では，要件 4. を満たすことができず，要件 1.〜3. も不十分になる可能性が高い．逆に，残りを完成させる方式では自由度を持たせることが困難で要件 5. を満たすことができない．要件 3. の，課題に対するモチベーションもあまり高まらない恐れもある．

[*1] 2011 年度にも本稿と同様の方針で，別のルール設計のゲーム課題を実施した．

[*2] 配布サイトは https://sites.google.com/a/eng.ucsd.edu/codespells/home（2013 年 7 月 8 日確認）

中間の，仕様を与えて制作させる方式の場合は，仕様の規模に応じて自由度に幅がある．そこで，ゲーム本体のプログラムは提供し，そのゲームに出場するプログラムを制作させる，プログラミングゲームの形式を採ることにした．プログラミングゲームの歴史は長く[*3]，1961 年に開発された Darwin が最初のプログラミングゲームと記録されている．最も有名なプログラミングゲームの一つには Core Wars がある．プレイヤーはアルゴリズムを考える必要があるためプログラミング教育の効果も高い．例えば Robocode[3] は Java か.NET のプログラムを対戦させるものである．チェスや将棋などのボードゲームの思考プログラムもこの部類に入るとも言えるが，既存のゲームの場合はゲーム自体の分析の観点が強く，プログラミング教育にそのまま適用するのは難しい．尾崎らは独自のルールのボードゲームを設計し実践している [2].

本授業ではオブジェクト指向プログラミングの基礎的な位置づけであることと，要件 4. で述べたように受講生の理解度に幅があることから，探索アルゴリズムなどの高度な解法を求めることはできない．そこで，簡単な処理，例えば条件分岐の組み合わせだけでも最低限の思考ルーチンが作成可能なゲームルールを設計することにした．

また，要件 1. を満たすために継承やオブジェクト変数を使わせる必要がある．そこで，基本となるクラスを提供し，受講生はそのクラスを継承したサブクラスを実装する方法を採った．さらに，ゲームプレイの中で別クラスのオブジェクトを操作するように設計することにした．

ゲームのルールは，先に挙げたプログラミングゲームの多くと同様に，2 本のプログラムを戦わせる対戦型とした．これにより，課題プログラムを作成する過程で受講生同士が対戦させてみる必要が生じるので，対戦自体が実行テストになりデバッグ作業を促す，より強くなるよう改良するモチベーションを与える，学習者同士でプログラミング技術を教え合う，といった効果を見込める．

以上の設計方針をまとめると次の通りである（括弧内は対応する要件）：

- 学習者が作成したプログラム同士が対戦するゲーム内容とする．（要件 2, 3）
- ゲーム本体のプログラムは提供する．（要件 4）
- 学習者は提供されたクラスを継承して自分のクラスを作成する．（要件 1, 4）
- 学習者が作成するプログラムでは，別クラスのオブジェクトを操作させる．（要件 1）
- 簡単な処理でも最低限の思考ルーチンを作成可能とする．（要件 4, 5）

[*3] プログラミングゲームの歴史については，例えば Programming Games Wiki http://programminggames.org (2013 年 7 月 8 日確認) が詳しい．

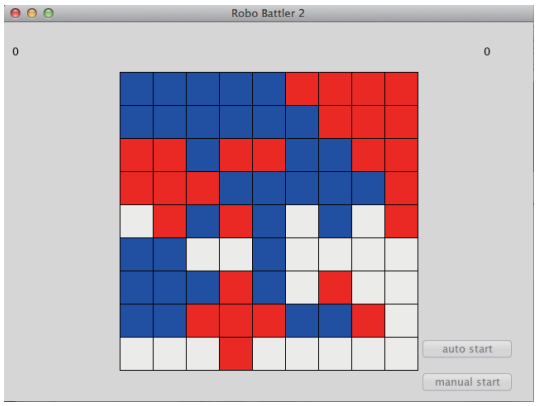



**図 1** ロボバトラー 2 の実行例.
**Fig. 1** Screenshot of Robobattler2.

## 4. ロボバトラー 2

### 4.1 ルール概要

以上の設計方針を基に，ゲーム「ロボバトラー 2」[*4]のルールを設計した．

ロボバトラー 2 は，2 体のロボプログラム（以下，単にロボと呼ぶ）が対戦する陣取り型のゲームとした．2 体のロボは交互に，与えられたミサイルの爆撃座標を決めて撃つ．ミサイルには 5 種類があり，種類ごとに爆撃範囲が決まっている．撃ったミサイルの爆撃範囲のマス目は自分の陣地となる．双方がすべてのミサイルを撃ち終わった時点で陣地の多い方が勝ちとなる．

図 1 はゲームの画面例である．赤と青のマス目は双方のプレイヤーの陣地で，白はどちらのプレイヤーの陣地でもないマス目である．

プレイヤー（学習者）は自分のロボを，与えられた Robo クラスを継承して作成する．Robo クラスには自分の手番にゲーム進行管理クラスから呼ばれる action メソッドを抽象メソッドとして定義してあるので，プレイヤーはこのメソッドを必ず実装しなければならない．また，Robo クラスには相手ロボが直前に撃ったミサイルに関する情報を取得するメソッドを用意した．このようにしてクラス継承の書き方と使い方を再確認させる仕組みを導入した．

action メソッドの引数として，次に撃つべきミサイルのオブジェクトが渡されるようにした．プレイヤーはミサイルを撃つために，ミサイルオブジェクトに対して目標座標を引数とする fire メソッドを呼ぶ必要がある．このようにして，オブジェクト変数の扱いとメソッド呼び出しのコーディングを強制するようにした．更に，ゲーム開始時は相手ロボが直前に撃ったミサイルの座標を得るメソッドの返値が null となるようにすることで，正しく対処しておかなければ NullPointerException が発生するようにしておき，

[*4] 本ゲームではロボット自体は現れないが，去年度に実施した課題の名称を引き継いでこのように命名した．

未テストのプログラムを検出できるようにした．

なお，ゲームのルールには不確定な要素を導入することにした．ゲームの種類としては二人零和有限不確定不完全情報ゲームとなる．これは，ルールを簡単にしながら多様な思考ルーチンのアイデアを可能にさせること，確実に強いプログラムを作成するには数多くのテストが必要となることを狙ったものである．

ルールの詳細については付録のとおりである．

### 4.2 想定した戦略

ロボバトラー 2 では，両プレイヤーが使えるミサイルの種類と数は同じであるので，勝つためには効率的に陣地を獲得する必要がある．自分の手番に撃ったミサイルによって既に相手の陣地となっているマス目を奪い返すと差し引き 2 マスの得，どちらの陣地でもないマス目を獲得すると 1 マスの得となる．また，既に自分の陣地であるマス目および盤外に対しては無駄になることになる．

もし相手が撃ったミサイルと同種のミサイルを同じ座標に撃てれば確実に相手の陣地を奪い返すことができ，最も効率が良いことになる．しかし，相手が撃ったミサイルの種類は確実に知ることができるが，ミサイルの目標座標を正確に知ることはパワーサーチを使わなければできないようになっている．また，Line 型ミサイルに関しては相手が縦方向か横方向かどちらの向きに設定したかを知る方法は提供していない．プレイヤーはこれらの不確実性に配慮してアルゴリズムを考える必要がある．例えば，パワーサーチの使用回数は 1 ゲーム中 5 回に制限しているので，使う状況を考える必要がある．

本授業の受講生は大学 2 回生および再履修生であり，多くはプログラミング経験，特に，自由な発想のプログラム作成の経験は少ない．これを踏まえて，受講生は次のように考えてアルゴリズムを作成すると予想した：

まず，最初に思いつくのは何も考えずに次に撃つミサイルの座標を決める方法である（無思考型）．例えば乱数で決める方法や固定的に決めておく方法である．当然この方法では効率が悪く，少し高度な作戦をとるプログラムにはまず勝てないことに気付く．

次に思いつくであろう作戦は，相手の撃ってきたミサイルと同じ座標に撃つ作戦である（単純迎撃型）．最も単純な方法は，通常の探索で得られる座標と同じ座標に，誤差を無視して撃ち返す作戦である．しかし，通常の探索で正しい座標が得られる確率は 1/4 と，比較的低く設定されている．パワーサーチを使えば 5 回は正確な座標を得ることができるが，いつ使うかが問題となる．例えば最後の 5 回など適当なタイミングで使用すると予想した．

ところが単純迎撃型では，撃つミサイルの種類は自分では選ぶことができない点が問題となる．相手のミサイルと自分のミサイルが同じ種類ならば，同じ座標に撃ち返せばすべてのマス目を奪い返すことができる．しかし，BigPlus 型と BigX 型の組み合わせの場合は同じ座標に撃ち返してもまったく奪い返すことができない．また，Line 型の場合は向きが異なれば 1 マスしか重ならないことになる．このように，ミサイルの種類には相性があり，これを配慮した作戦を考えると予想した（相性考慮型）．例えば，直前に相手が撃ったミサイルと次に自分が撃つミサイルが Line 型以外の同種ならば，パワーサーチを使えば確実にすべてのマス目を奪い返すことができる．が，そのようになる確率は 5/32 であるから 16 発中の平均値は 2.5 発となり，パワーサーチの使用可能回数 5 回と比べると多くはない．そこで，Plus 型と BigPlus 型，X 型と BigX 型の組み合わせでもパワーサーチを使うことにすればより効率的となる．

さて，ミサイルの相性に気付くと，直前に相手が撃ったミサイルの種類と位置のみから次の自分のミサイルを効率よく撃つことには限界があることがわかる．そこで，相手の撃った履歴を残す方法を考えるであろう（履歴活用型）．つまり，次に撃つミサイルと同種のミサイルを過去に相手が撃っていたら，それに重なるように撃つ作戦である．ただし，途中の経過によっては自陣に塗り変わっている可能性もあるので自分の撃った履歴も考慮する必要がある．また，過去に遡ってパワーサーチを使うことはできないが，BigPlus 型と BigX 型は爆撃範囲が広いので，これらに対して使うことが有効であろう．

更に進めると，盤全体の状態を記録して次のミサイルを撃つ最善の位置を探す方法が考えられる（盤面状態予想型）．但し，通常サーチによる誤差と，相手の撃った Line 型の向きは分からない，という不確定要素があるので，盤の状態を完全に知ることは不可能である．これらの不確定要素については，考慮しない方法や確率値で扱う方法など，様々な方法が考え得る．また，パワーサーチを使うタイミングも色々な可能性がある．

なお，チェスや将棋で用いられる局面探索型の思考アルゴリズムは，不確定要素のためあまり有効ではないと予想した．上述以外のアルゴリズムも可能であるが，大半はこれらのバリエーションであると予想した．

### 4.3 戦略とプログラミングスキル

前節で挙げた戦略は，必要とするオブジェクト指向のプログラミングスキルが段階的に異なっている．無思考型はクラス継承の基本だけ理解していればよい．単純迎撃型は親クラスのメソッドを使用する必要がある．相性考慮型はミサイルクラスオブジェクトのメソッドを使用する必要がある．履歴活用型や盤面状態予想型はフィールドを使用する必要がある．また，処理が複雑化するにつれ，内部メソッドを作ったり追加のクラスを作る必要も生じる．

このように，より強いプログラムを作成するには，オブジェクト指向プログラミングをよりよく理解している必要

```
import robobatter2.*;

public class RandomRobo extends Robo {
    protected void action(Missile missile) {
        int x = (int)(Math.random() * Field.SIZE);
        int y = (int)(Math.random() * Field.SIZE);
        fire(x, y);
    }
}
```

**図 2** 学習者に提示したプログラム例.

**Fig. 2** Sample code that provided to the students.

があるようにルール設計を行った.

## 5. 実践

以上の課題内容を，次のようにして受講生に指示した：最初にゲーム本体の jar ファイルとルールは演習資料として提示し，完成形を示した．そして GUI プログラミングとしてゲームの実行画面（図 1）を作成する方法を 3 回に分けて演習形式で段階的に説明した．その内容は，ウィンドウの表示方法，コンポーネントの配置，描画処理，イベント処理，マルチスレッドであった．途中，ゲームの内部処理については説明を省略し，ソースコードおよび API リファレンスの配布のみとした.

また，プレイヤーが作成するロボプログラムの作り方について例を示して説明した．具体的には，親クラスとなる Robo クラスを継承し，抽象メソッドである action メソッドを実装することを説明した．更に，サンプルとして座標を乱数で決定して撃つだけのロボプログラムを提示し（図 2），コンパイルと実行方法を説明した.

GUI の内容の 3 回目の講義 (2012 年 12 月 24 日) で，課題内容について説明した．その際ヒントとして，プレイヤーが作成するプログラム中で使用可能な Robo クラスと Missile クラスの API，およびゲームの基本的な考え方（相手の陣地を奪うことが得策であること，不確定要素にどう対処するかがポイントとなること）を説明した．また，テスト用の対戦ロボとして，図 2 のプログラムを改良し撃ったミサイルの爆撃範囲が盤外に出ないように補正するロボ，および通常サーチで得られた座標に撃ち返すだけのロボのクラスファイルを提供した．これらのロボについては挙動を見てプログラム内容を推測させるよう，ソースコードは非公開とした.

課題の提出締切は 2013 年 2 月 3 日に設定した．課題内容の説明から約 6 週間あったが，期間中，他のトピックの演習と筆記試験を行い，もう一つの別の内容の最終課題を設定したので（提出締切は同じ），受講生はこの期間中，本格的には本課題に取り組んでいないようであった．課題に関する質問や自習室での学生間の情報交換状況から推測して，期間の最初のうちに一部の受講生が着手し，しばらく沈静化した後，締切 2 週間前くらいから多くの受講生が着手し始めた模様であった[*5].

課題提出者は 137 名であった．履修登録者は 169 名であるので，32 名が提出しなかったことになるが，うち 21 名は平常時の授業にもほぼ出席していなかった．残りの 11 名の多くは筆記試験では極めて成績が悪いということはなく，未提出の原因は Java 言語自体の理解不足によるものではないと推測された．なお，課題提出者のうち 6 名のプログラムは問題のあるものであった（デッドコピー，去年度に実施した課題内容のプログラム，未着手のもの）.

提出プログラムで採られた戦略を 4.2 節で説明した分類に分けると，無思考型は 17 本，単純迎撃型は 27 本，相性考慮型は 62 本，履歴活用型は 6 本，盤面状態予想型は 15 本，それ以外は 4 本であった.

提出プログラムを使用して 1 次予選〜3 次予選のリーグ戦，および決勝トーナメント戦からなる大会を実施した．この大会の開催については課題説明時に受講生に予告した．提出プログラムのうち，前述の問題のある 6 本以外では，4 本がコンパイル不能であったが，うち 2 本はファイル名とクラス名の不一致（後からファイル名を変更したためと見られる）および全角スペースの混入の軽微なミスによるものであった．試合結果は適宜 Web ページ上で公開した.

1 次予選では主に不具合のあるプログラムが淘汰された．10 本は NullPointerException を高い率で発生し，うち 6 本は，4.1 で述べたように，null が返値となる状況に対処していないことによるものであった．例外が発生した場合はそのゲームは不戦敗となるので，自動的に負けが多くなり圧倒的に不利となる．2 次予選ではアルゴリズム上のミスが含まれているものや，作戦的に良くないものが主に敗退した．3 次予選に残ったロボのほとんどは相性考慮型以上の高度な戦略を採ったものであった．基本的な戦略は同じでも配慮するミサイルの種類や不確定要素への対処方法の差で優劣がついていた．決勝トーナメントに進出した 18 本のうち 9 本は盤面状態予想型であり，そのうちの 7 本が準々決勝まで勝ち残った.

提出プログラムは，プログラミングスキル（アルゴリズム，オブジェクト指向），可読性，アイデア（思考ルーチンの特徴，強さ），オリジナリティの観点で評価した（これらの評価項目に関しても課題内容説明時に受講生に告知した）．表 1 は各戦略ごとの評価点の平均，標準偏差，最小・最大値，実装されていたメソッド数の平均である．評点は概ね 100 点満点としているが加点法による採点であるため 100 点を超えている場合もある.

アンケート調査については，本課題は成績に反映されるため回答にバイアスがかかると予想し，受講生全体に対しては行わなかったが，課題提出期日から約 1 ヶ月後に 8 名の有志による回答を得た．本課題に費やした時間については平均 17.9 時間であった．5 名は提出までにテストプレ

[*5] 締切 2 週間前から本学の定期試験期間が始まっていたことも大きく影響していると思われる.

**表 1** 各戦略ごとの評点の統計．無:無思考型，単:単純迎撃型, 相:相性考慮型, 履:履歴活用型, 盤:盤面状態予想型, 他:その他

**Table 1** Statistics of scores.

| | 本数 | 平均 | 標準偏差 | 最小値 | 最大値 | 平均メソッド数 |
|---|---|---|---|---|---|---|
| 無 | 17 | 53.6 | 11.2 | 38 | 75 | 2.67 |
| 単 | 27 | 53.0 | 12.3 | 30 | 79 | 2.04 |
| 相 | 62 | 59.5 | 12.3 | 33 | 82 | 2.60 |
| 履 | 6 | 78.5 | 6.3 | 71 | 86 | 5.50 |
| 盤 | 15 | 99.2 | 10.1 | 72 | 112 | 9.00 |
| 他 | 4 | 69.3 | 22.1 | 50 | 100 | 3.25 |

イを数十回以上行っており，残りの 3 名についても数回以上は行っていた．作成したロボの強さの自己評価については 6 名は普通以下と判断しており，プログラム自体の出来にもまだ満足し切れていないと回答した．ゲーム内容については 7 名が楽しめたと回答した．また，5 名が本課題は Java 言語の理解に役立ったと回答した．なお，1 名はゲーム内容は楽しめず Java 言語の習得に役に立たなかったと回答したが，プログラムの評点は高かった．

## 6. 考察

まず，2.2 節で挙げた課題の要件がどのように満たされたかを考察する．

要件 1. については，提出されたプログラムのほとんどが想定した最低限のレベルをクリアしていた一方で内容は画一的ではなく，理解度を測ることができた．また，アンケート回答者のうち半数超が本課題が Java 言語の理解に役立ったと答えたが，本課題に無関係に理解できていた受講生も存在していたことを含めると，理解の深化に役立ったと言える．

要件 2. については，NullPointerException を頻発した 10 本はほとんどテストを行っていなかったと見られるが，大半は最低限のテスト作業を行ったと言える．またアンケートから，課題に費やした時間の平均が 17.9 時間と最低限のプログラムに必要な時間よりかなり長かったこと，テストプレイを数十回以上行っている受講生が半数以上いたこと，および，提出プログラム中のコメントに記入されていたアイデアの発展履歴から，多くの受講生は試行錯誤してプログラムを完成させたことが伺われる．

要件 3. については，アンケートでは大半の回答者がゲーム内容を楽しめたと答えた．また，大会の経過を授業用 Twitter アカウントで告知したが，Twitter 上でのリツイートやリプライ等の反応から，多くの受講生は自分のロボの戦績に興味があることが伺われた．更に，上述のように試行錯誤してプログラムを完成させた一方で，アンケートでは多くの回答者がプログラムの出来には満足しておらずより良くできると考えていた．これらのことから，本課題内容によりモチベーションの向上と維持ができたと言える．

要件 4. については，表 1 に示した戦略ごとの評点及び平均メソッド数の分布から，理解度に応じて高度なプログラムが作成できるようになっていたと言える．また，総じて強いプログラムほどプログラムの書き方が洗練されていたりコメントが充実していて評点が高い傾向にあった．

要件 5. については，基本的な戦略は同じでも詳細は異なっている提出プログラムが多かったこと，ソースコードレベルで完全に一致していたプログラムは 2 本，部分的に一致していたプログラムは 8 本であったことから，課題内容は十分に自由度が高かったと言える．

ところで，以上の要件を満たすにはゲーム内容も大きく関係している．つまり，適度に複雑で面白い必要があるし，思考ルーチンのアルゴリズム（戦略）をある程度想定して設計する必要もある．

ロボバトラー 2 では不確定要素を導入することで，簡単なルールで複雑なゲームとすることができた．昨年度に実施したロボバトラーでは確定完全情報ゲームとしたが，取り得る行動パタンを増やすために使用できるメソッドを増やす必要があった (23 種類)．一方，ロボバトラー 2 では 5 種類のみのメソッドで適度に複雑にすることができた．ただし，不確定要素が大きすぎると思考ルーチンの優劣が勝敗に影響しなくなりゲームとして成立しなくなるので，適度なバランスが重要となるであろう．

ゲームの設計については，ゲームデザインの経験はさほど必要ないと考えている．著者は過去にボードゲームやコンピュータゲームを 2，3 設計したことがある程度であった．今回の場合，次のような手順でルールを考案した：まず，ゲームのモチーフとして陣取りを考えた．課題の設計として，自分が作成しているクラスとは別のクラスオブジェクトを操作する内容を盛り込みたかったので，与えられたミサイルオブジェクトを使って自陣を獲得する方式にすることにした．ここで，手番で採り得る手のバリエーションを増やすことと，継承とオーバーライドの実例としてミサイルの種類をいくつか用意することにした．しかし，プレイヤーが撃つミサイルの種類を自由に選択でき，しかも相手の撃ったミサイルに関する情報を完全に知ることができるとすると，お互いに陣地を取り返すだけになってしまってゲームとして成立しない．そこで，相手の撃ったミサイルの目標座標には誤差が入る，ミサイルの種類はランダムに与えられて自分では決められない，といった不確実性を導入することにした．また，盤面はミサイルの爆撃範囲が適度に重なる広さとすることで，効率よく敵陣を奪い返す方法を考えるゲームとした．一方で，運の要素があまり強くならないように，1 ゲームで与えられるミサイルの種類と数は双方で同じとなるようにし，また連戦の場合は同じ順番のミサイルセットのまま先手と後手を入れ替えて対戦させるようにした．その後，思考テストおよび試作プログラムでのテストを行って，ルールの詳細を決定した．

この際，テストプレイが重要となる．実際にプレイしてみることでゲームの難易度や面白さを確認して調整する必要がある．また，想定外のルールの不備が見つかることもある．今回の場合，最初に設計したルールでは後手が有利であることが判明したので，1 ゲームあたりの使用ミサイル数を増やす調整を行った．ゲームごとに先手と後手を入れ替えることもあわせて，先手後手の有利不利はほぼ解消できた．理想的には，受講生と同レベルのプログラミングスキルを持つテストプレイヤーにより長期間テストを行うのが良いが，どの程度行えばよいかが問題になる．ゲームを作ること自体が目的ではないので，致命的な欠陥がなければ，適度に調整できていれば十分である．

なお，ゲーム設計に関しては，ルールのパタンをストックすることで設計の負担を軽減できると考えている．つまり，ゼロからルールを設計するのではなく，ストックされたパタンを組み合わせてアレンジする方式である．ロボバトラー 2 の場合，交互にミサイルを撃って陣地を獲得する，ミサイルの順番はランダムに指定される，相手の状態の観測結果に誤差を含ませる，などがパタン化できる．例えば交互にミサイルを撃つパタンを使って，フィールド上に予め配置した基地を爆撃する別のゲームが作れる．

あるいは，一部のルール変更でもゲームの性格を大きく変えることもできる．例えばロボバトラー 2 においてミサイルの順番を指定されたものでなくロボが自由に選択できる代わりに，相手の撃ったミサイルの種類が分からないとすると，ゲームの主題は，相手の撃ったミサイルの種類をどのように推測するか，となる．この場合，相手の撃ったミサイルの種類を推測する手がかりを得る方法を導入することになるであろう．更に，不確実性を適量に調整するために，相手の撃った座標は確実に分かるようにするなどの変更も必要になるであろう．

## 7. まとめ

オブジェクト指向プログラミングの導入として Java 言語の基礎を扱う授業におけるプログラム作成課題として，プログラミングゲーム形式の課題を実施した．課題の要件として，オブジェクト指向プログラミングの理解を測る・深める，デバッグ・テスト作業を含めたソフトウェア開発を経験させる，モチベーションを維持させる，幅広い理解度に対応可能とする，自由度を高くする，の 5 項目を設定した．これらの要件を満たすために，学習者は与えられた親クラスを継承して，与えられたメソッドやオブジェクトを利用して思考ルーチンを実装した自分のクラスを作成し，作成したプログラム同士で対戦を行う内容のゲームを課題とした．実施した結果，設定した要件を満たしていたことを確認した．

今年度も同様の手法での課題を実施する予定である．その際ロボバトラー 2 では扱わなかったプログラミングの要素，例えばコンストラクタを盛り込んだルールを設計したい．また，新規課題（ルール）作成の簡便化を目指して，ルールパタンのストックを進めていきたい．

**謝辞**　本授業を共同で運営した玉田春昭氏，および TA 諸氏に感謝いたします．

### 参考文献

[1] Sarah Esper, Stephen Foster and William Griswold. CodeSpells: Embodying the Metaphor of Wizardry through Programming. ITICSE 2013. (to be presented)

[2] 尾崎浩和, 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 : 五五ゲームの実行環境と大会運営方法. 電子情報通信学会技術研究報告, ET, 教育工学 106(35), 55–60, 2006.

[3] Robocode http://robocode.sourceforge.net (2013 年 7 月 8 日確認)

## 付　　録
### ロボバトラー 2 ルール

## A.1 概要

（4.1 節の第 2 段落と同じ内容なので省略）

## A.2 ロボとバトルフィールド

ロボバトラー 2 はバトルフィールドと呼ぶ縦横 9 マスの上で行われます．ロボはミサイルを撃つだけなので，バトルフィールド上には表示されません．お互いがミサイルを撃って陣地となったマス目は赤・青で塗りつぶされます（赤が第 1 プレイヤー，青が第 2 プレイヤー）

## A.3 バトル（戦闘）

### A.3.1 バトル（戦闘）

双方のロボがミサイルを撃ち尽くして勝敗が決するまでをバトルと呼びます．

### A.3.2 バトルの初期状態

バトル開始時はバトルフィールド上のすべてのマス目はどちらの陣地でもありません（表示は白）．各ロボはそれぞれ，Line 型 4 発，Plus 型 4 発，X 型 4 発，BigPlus 型 2 発，BigX 型 2 発の計 16 発のミサイルを，ランダムな順番で準備します．自分の番に，この順番でミサイルが 1 発与えられます．

### A.3.3 バトルの終了条件

バトルは双方のロボとも 16 発のミサイルを撃ち尽くしたら終了します．

## A.4 ミサイル

### A.4.1 ミサイルとは

ロボは自分の番に受け取ったミサイルを，目標地点を定

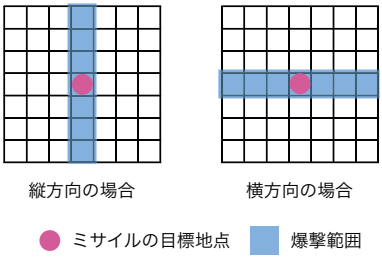
めて撃ちます．ミサイルは種類ごとに爆撃範囲が定められており，目標地点を中心に爆撃範囲のマス目が撃ったロボの陣地となります（元がどちらの陣地であったかは関係ありません）．

#### A.4.2　ミサイルの種類

ミサイルには，Line 型，Plus 型，X 型，BigPlus 型，BigX 型の 5 種類があります．

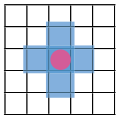
##### A.4.2.1　Line 型ミサイル

Line 型ミサイルは目標地点を中心に縦方向あるいは横方向に 7 マスを爆撃範囲とするミサイルです．縦方向か横方向かはミサイルを撃つ前に設定可能です．



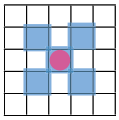

##### A.4.2.2　Plus 型ミサイル

Plus 型ミサイルは目標地点および，目標地点の縦横の隣の，計 5 マスを爆撃範囲とするミサイルです．



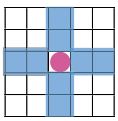
##### A.4.2.3　X 型ミサイル

X 型ミサイルは目標地点および，目標地点の斜め方向の隣の，計 5 マスを爆撃範囲とするミサイルです．



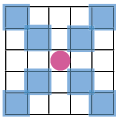
##### A.4.2.4　BigPlus 型ミサイル

BigPlus 型ミサイルは目標地点の縦横の隣 2 マスの，計 8 マスを爆撃範囲とするミサイルです（目標地点を含まないことに注意）．



##### A.4.2.5　BigX 型ミサイル

BigX 型ミサイルは目標地点から斜め方向 2 マスの，計 8 マスを爆撃範囲とするミサイルです（目標地点を含まないことに注意）．



### A.5　探索

#### A.5.1　探索

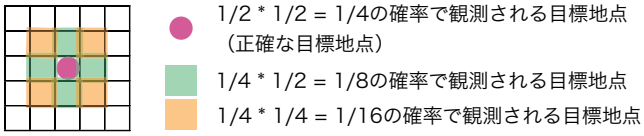
ロボは自分の番に，直前に相手が撃ったミサイルの情報を得ることができます．

#### A.5.2　直前に撃ったミサイルの種類

直前に相手が撃ったミサイルの種類を正確に知ることができます．

#### A.5.3　直前に撃ったミサイルの目標地点

直前に相手が撃ったミサイルの目標地点を，所定の正確さで知ることができます．縦および横方向それぞれに関して，1/4 の確率で誤差が含まれて両隣の座標となります．ただし，誤差を含んだ結果がバトルフィールド外になる場合は正しい座標となります．図示すると次図のようになります．




#### A.5.4　パワーサーチ

1 つのバトル中に 5 回まで，直前に相手が撃ったミサイルの目標地点を正確に知ることができます．パワーサーチ回数が残っていない場合は 5.3 のように誤差を含む探索が行われます．

#### A.5.5　探索の注意点

同じ手番では，何回探索を行っても同じ結果となります．ただしパワーサーチは誤差を含まないので，通常の探索とは別の結果を返すことがあります．パワーサーチは同じ手番で複数回行うと，行った回数だけ残り回数が減ってしまうことに注意してください．

### A.6　総合勝利

#### A.6.1　勝利条件

1 試合は 7 本勝負で，勝ちバトル数の多い方が総合勝利です．

#### A.6.2　先攻・後攻

バトルの先攻・後攻はバトルごとに交互に入れ替わります．ただし，最終バトルの先攻は抽選で決定します．