ProgrammingII
Report #1

所 属 :琉球大学 工学部 情報工学科
学籍番号:155730B
氏 名 :清水 隆博
提 出 日:2015 年 12 月 21 日

目 次

1 サンプルプログラムの解説

1.1 Corners.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   * 所謂コピーライトの表記。特に意味は無い
8   */
9  package sample;
10
11
12 import robocode.DeathEvent;
13 import robocode.Robot;
14 import robocode.ScannedRobotEvent;
15 import static robocode.util.Utils.normalRelativeAngleDegrees;
16
17 //主要メソッドを import する
18
19 import java.awt.*;
20
21 /**
22  * Corners - a sample robot by Mathew Nelson.
23  * <p/>
24  * This robot moves to a corner, then swings the gun back and forth.
25  *このロボットは角へ移動し , 主砲を前後あちらこちらに動かす
26  *
27  * If it dies, it tries a new corner in the next round.
28  *もしこのロボット自身が撃沈した場合 , 新しい角から次のラウンドを始める
29  * @author Mathew A. Nelson (original)
30  * @author Flemming N. Larsen (contributor)
31  */
32
33 public class Corners extends Robot {
34     int others; // Number of other robots in the game
35     //このゲームで他のロボットの数を把握する
36     static int corner = 0; // static な int 型変数 corner を宣言し 0 を代入
37     // static は round 間で保存される
38     boolean stopWhenSeeRobot = false; //
           boolean 型変数 stopWhenSeeRobot を宣言し, false を代入
39     // この一行がどう活用されるかはgo Corner()を参照
40
41     /**
42      * run: Corners' main run function.
43      * run メソッド: Corner の main となる run メソッド
44      */
45
46     public void run() {
47         // 機体の色を設定
48         setBodyColor(Color.red);
```

```
49          setGunColor(Color.black);
50          setRadarColor(Color.yellow);
51          setBulletColor(Color.green);
52          setScanColor(Color.green);
53
54          // robocode.Robot のメソッド，getOthers を用いて残っている敵の数を other に渡す
55          others = getOthers();
56
57          // 角へ移動する
58          goCorner();
59
60          // 大砲の回転速度を 3に初期化
61          int gunIncrement = 3;
62          //ここでは int 型変数 gunIncrement を宣言し，そこに 3 を代入している
63
64          // 大砲を行ったり来たり回転させる
65          while (true) {
66              for (int i = 0; i < 30; i++) { //カウンタ変数 i が 30 になるまでループ
67                  turnGunLeft(gunIncrement);//先ほど宣言した
                        gunIncrement を turnGunLeft に渡す。
68              //此処では速さ 3
69              }
70              gunIncrement *= -1; //gunIncrement の値に-1をかける。これで逆向きに回転す
                    る。
71          }
72      }
73
74      /**
75       * goCorner: これはとても効率の悪い角へ向かうメソッドである。もっと良くしてください
76       */
77      public void goCorner() {
78          // 回転している時は自機は止まらない
79          stopWhenSeeRobot = false;
80          // 回転時に正面を向いた壁の右側へと移動する
81          turnRight(normalRelativeAngleDegrees(corner - getHeading()));
82
83          /*Robot クラスの turnRight メソッド(機体を右回転する)を使用
84           * turnRIght に渡す値を正規化するために
85           * normalRelativeAngleDegrees メソッドを通して
86           * 現在機体が向いている座標をgetHeading メソッド取得し
87           *corner からこれを引いた分の角度回転する
88           *turnRight メソッドは処理が終了するまで戻らないので回転中は自機は何もしない*/
89
90          //stopWhenSeeRobot 変数をtrue にする
91
92          stopWhenSeeRobot = true;
93
94          //5000pixel 分直進する。先ほど回転したので壁に当たる.
95          //Robot 内の ahead メソッドは壁に衝突すると動作を完了するので
96          //衝突後 turnLeft の動きに移行する
97
98          ahead(5000);
99          // 角に正面を向く
100         turnLeft(90);
```

```
101         // 角へと移動する
102         ahead(5000);
103         // 左に 90度移動して動作完了
104         turnGunLeft(90);
105     }
106 
107     /**
108      * onScannedRobot メソッド: 静止して砲撃
109      */
110     public void onScannedRobot(ScannedRobotEvent e) {
111         // 止まる動作を含むか, 止まらずに撃つかの判断
112         if (stopWhenSeeRobot) {
113             //stopWhenSeeRobot が true だった時。すなわち goCorner()で回転が終了した際
                    は
114             // stop メソッドを用いて動作を全て終了
115             stop();
116             // 以下に定義されているsmartFire メソッドを呼び出し，引数として e,
                    getDistance を渡す
117             //getDistance では自機と相手の戦車との距離を計測する
118             smartFire(e.getDistance());
119             // scan()メソッドで他のロボットを確認
120             // NOTE: もし,scan を呼び出している際に robot が見つかったならば
121             // 再びこのメソッド内の先頭に回帰する
122             scan();
123             // ここでrobot が検出されなければ
124             //resume メソッドを用いて run メソッドに回帰する
125             resume();
126         } else { //stopWhenSeeRobot が false だった場合。
127             //すなわち goCorner で回転し始める前は smartFire のみ
128             smartFire(e.getDistance());
129         }
130     }
131 
132     /**
133      * smartFire: カスタムされた
                fire メソッド。firepower と距離によって動きが決定される。
134      *
135      * @param(パラメーター) robotDistance the distance to the robot to fire at
136      * robotDistance には onScannedRobot で読み込んだ getDistance の値が代入
137      */
138     public void smartFire(double robotDistance) {
139         if (robotDistance > 200 || getEnergy() < 15) {//
                robot との距離が 200pixel 以上または，自機のエネルギーが 15 未満の場合
140             fire(1);  //威力 1でfire
141         } else if (robotDistance > 50) { //距離が 50pixel 以上ならば
142             fire(2);   //威力 2でfire
143         } else {
144             fire(3); //それ以外は威力 3でfire
145         }
146     }
147 
148     /**
149      * onDeath メソッドのオーバーライド: 全て撃沈した場合。次のゲームでは別コーナーからの
                開始を決定する。
```

```
150      */
151     public void onDeath(DeathEvent e) {
152         /**
153          *敵機が 0になることはない (勝利時に呼び出されるメソッドはonWin であるので)
154          *しかしsample のコメントによれば，転ばぬ先の杖として一応定義されている。
155          *そのまま特に返り値を返さず
156              *return する
157         */
158         if (others == 0) {
159             return;
160         }
161 
162         /**
163         * (もし自機が死んだ際, 敵機が 75%以下ならば角を変更する)
164         * と書かれているが, 実際の計算を見てみると死んだ敵機が 75%以下の時という計算になっている (コメントミス？)
165         */
166         if ((others - getOthers()) / (double) others < .75) {
167             /**
168              *最初に保存しておいた
169                  others から現在生存している機体数を getOthers メソッドで取得
170              *その差分をothes で割り，75%以下だったら
171              */
172             corner += 90;
173             //corner の値に+90する。つまり時計回りに目標の角を変更する (robocode では画面の上方向が 0 度)
174             if (corner == 270) { //corner の値が 270 だった場合
175                 corner = -90; //逆に corner の値を-90にする。(360度を越してしまう為)
176             }
177             out.println("I died and did poorly... switching corner to " +
178                 corner);
179 
180              //そしてこのコメントと corner の値を print
181         } else {
182             out.println("I died but did well. I will still use corner " +
183                 corner);
184             //死んだ敵機が
185                 75%以上の場合は, このコメントのみ出力し , corner の場所は変更しない
186         }
187     }
188 }
```

1.2 Crazy.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   *所謂コピーライト
8   */
9  package sample;
10
11 //Crazy.java が収納されている package 名
12
13 import robocode.*;
14
15 import java.awt.*;
16
17 //Crazy.java で使用するために import してきたクラス
18
19 /**
20  * Crazy - a sample robot by Mathew Nelson.
21  * <p/>
22  * このロボットはクレイジーなパターンであちらこちらに動く
23  *
24  * @author Mathew A. Nelson (original)
25  * @author Flemming N. Larsen (contributor)
26  * crazy の作者についてのcomment
27  */
28
29 public class Crazy extends AdvancedRobot {
30  //AdcancedRobot を継承し Crazy クラスとして実装
31
32     boolean movingForward; //boolean 型変数 movindForward を宣言
33
34     /**
35      * run メソッド : Crazy のメインとなる run メソッド(関数)
36      */
37
38     public void run() {
39         // 機体の色の設定
40         setBodyColor(new Color(0, 200, 0));
41         setGunColor(new Color(0, 150, 50));
42         setRadarColor(new Color(0, 100, 100));
43         setBulletColor(new Color(255, 255, 100));
44         setScanColor(new Color(255, 200, 200));
45
46         //無限ループ
47         while (true) {
48             // ゲーム中 40000pixel 移動する.(
                   sample プログラムなので長い数字なら何でも良い)
49             setAhead(40000);
50             /**
51               *AdvancedRobot クラスのメソッド setAhead に 4000pixel を渡している
```

```
52              *set 系のメソッドは宣言してもすぐには実行されずキャッシュに保存される
53              *このメソッドが実際に動くのはexecute()メソッドを呼び出すか,実行動作を行う
                    時である。
54              */
55
56          movingForward = true;
57
58          //変数 movingForward を true にする
59
60          //setTurnRight メソッドに 90 を入れる。(数値は角度として渡される)
61          //setTurnRight メソッドは機体を右に回転させるメソッドである
62          //これも setAhead メソッドと同様に実行動作が行われるまで待機される。
63
64          setTurnRight(90);
65
66          waitFor(new TurnCompleteCondition(this));
67
68          /**
69            *waitFor メソッドは executes メソッドの様に即座に実行されるメソッドであり
70            *このメソッドを読み込んだ瞬間setAhead,setTurnRight が実行される
71            *waitFor メソッドは条件設定した TurnCompleteCondition クラスが
72            *完了した事を読み取るまで戻されない
73            *
74            *TurnCompleteCondition は回転が完了したかどうかを判断するクラスなので
75            *setTurnRight(90)が完了するまで保持される。
76                */
77
78          setTurnLeft(180);
79
80          //setTurnLeft メソッドに数値 180 を渡す。
81          //動きは setTurnRight メソッドの逆で左回転する
82
83
84          waitFor(new TurnCompleteCondition(this));
85
86          //先ほどと同様に waitFor メソッドを読み込んだ瞬間に左回転が開始される
87
88          setTurnRight(180);
89
90          //setTurnRight メソッドに 180 の数値を渡す
91
92          waitFor(new TurnCompleteCondition(this));
93          // 同様にwaitFor メソッドで確認を取る。ここまでがループされる
94      }
95  }
96
97  /**
98   * onHitWall: 壁と衝突した際に使用されるメソッド
99   */
100 public void onHitWall(HitWallEvent e) { //イベント処理より壁に当たったことを受
        け取った場合
101     //跳ね返るような動きをする
102     reverseDirection();
103         //下に定義されている reverseDirection メソッドを呼び出す
```

```
104
105     }
106
107     /**
108      * reverseDirection: 何かに衝突した際に呼び出され,前進か交代をset するメソッド
109      */
110     public void reverseDirection() {
111         if (movingForward) { //boolean 型変数 movingForward が true だった場合(
              run メソッド内で true にされている)
112             setBack(40000); //40000pixel 戻るように set を行う(直ちに処理は行われない)
113             movingForward = false; //movingForward を false に set する
114
115
116         } else {        //movingForward が false だった場合
117
118             setAhead(40000); //40000pixel 直進するように set
119             movingForward = true; //movingForward を true にする
120         }
121     }
122
123     /**
124      * onScannedRobot メソッド:ロボットをscan した場合, 砲撃する
125      */
126     public void onScannedRobot(ScannedRobotEvent e) { //
          ScannnedRobotEvent から値を受け取った時
127         fire(1); //威力 1で砲撃
128     }
129
130     /**
131      * onHitRobot: 戻る
132      */
133     public void onHitRobot(HitRobotEvent e) { //敵機に衝突した場合
134
135
136         if (e.isMyFault()) { //自分から当たった場合
137             reverseDirection(); //reverseDirection を呼び出す
138         }
139     }
140 }
```

1.3 Fire.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   * 所謂コピーライト
8   */
9  package sample;
10
11
12 import robocode.HitByBulletEvent;
13 import robocode.HitRobotEvent;
14 import robocode.Robot;
15 import robocode.ScannedRobotEvent;
16 import static robocode.util.Utils.normalRelativeAngleDegrees;
17
18 import java.awt.*;
19
20 //Fire.java で使用するためのクラスを import
21
22 /**
23  * Fire - a sample robot by Mathew Nelson, and maintained.
24  * <p/>
25  * Fire は銃身を回転させながら，砲撃して移動する．
26  *
27  * @author Mathew A. Nelson (original)
28  * @author Flemming N. Larsen (contributor)
29  * 作者の説明とちょっとした概要
30  */
31
32
33 public class Fire extends Robot {
34     int dist = 50; //被弾した際に動く距離として使用するために
           int 型変数 dirst を宣言し，50 を代入
35
36     /**
37      * run: Fire のメインとなる run メソッド
38      */
39     public void run() {
40         //色の設定
41         setBodyColor(Color.orange);
42         setGunColor(Color.orange);
43         setRadarColor(Color.red);
44         setScanColor(Color.red);
45         setBulletColor(Color.red);
46
47         //ゆっくりと砲身を右回転し続ける
48
49         while (true) {
50             turnGunRight(5); //5度右回転
51         }
```

```
52      }
53
54      /**
55       * onScannedRobot: 敵機を検知したら砲撃
56       */
57      public void onScannedRobot(ScannedRobotEvent e) {
58          //敵機が 50pixel 以下の距離，かつ自機のエネルギーが 50 以上の場合
59          // 最高威力で砲撃!
60
61          if (e.getDistance() < 50 && getEnergy() > 50) {
62
63              //e.getDistance()でscan した敵機との距離を図り
64              //getEnergy で自機のエネルギーを数値化する
65
66              fire(3);   //最大出力で砲撃
67
68          } // それ以外の場合は威力 1で砲撃.
69          else {
70              fire(1);
71          }
72          // もう一度scan することで連続攻撃を実装
73          scan();
74      }
75
76      /**
77       * onHitByBullet: 弾丸に対して垂直に移動し,少し動く
78       */
79      public void onHitByBullet(HitByBulletEvent e) { //
            onHitByBullet のコンストラクタ
80          turnRight(normalRelativeAngleDegrees(90 - (getHeading() - e.
                getHeading())));
81
82      /**
83           * getHeading メソッドから取得した自機の向きの角度と
84       * HitByBulletEvent のメソッド"getHeading"で命中した時点での弾丸の進行方向をそれぞ
             れ取得。
85       * 自機の向きから弾丸の向きを引いた物を, さらに 90度から引いて, この差を
             turnRight メソッドに渡す
86       */
87
88          ahead(dist); // 先に要していたdist の値分前進する
89          dist *= -1; //dist の値に(-1)を乗算する
90          scan(); //もう一度 scan を呼び,onScannnedRobot を呼び出しやすくする
91      }
92
93      /**
94       * onHitRobot: 敵機に衝突した場合,砲撃して逃走
95       */
96      public void onHitRobot(HitRobotEvent e) { //onHitRobot のオーバーライド
97          double turnGunAmt = normalRelativeAngleDegrees(e.getBearing() +
                getHeading() - getGunHeading());
98
99      /**
100      * HitRobotEvent のメソッド getBearing を用いて衝突したロボットとの相対角度を取得
```

```
101         * その角度と自機が向いている角度を足し合わせる
102         *つまり，敵機の角度を算出し，そこから現在自機の主砲が向いている角度を引く。
103         *そこで出た角度をturnGunAmt に保存
104         */
105
106             turnGunRight(turnGunAmt); //先の turnGunAmt の分のみ主砲を右回転
107             fire(3);//威力 3で砲撃
108         }
109     }
```

1.4 Interactive.java

```
/**
 * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * http://robocode.sourceforge.net/license/epl-v10.html
 * 所謂コピーライト
 */

package sample;


import robocode.AdvancedRobot;
import static robocode.util.Utils.normalAbsoluteAngle;
import static robocode.util.Utils.normalRelativeAngle;

import java.awt.*;
import java.awt.event.KeyEvent;
import static java.awt.event.KeyEvent.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseWheelEvent;

//Interactive を構成する上で必要なクラスを import する

/**
 * Interactive - a sample robot by Flemming N. Larsen.
 * <p/>
 * このロボットはマウスとKeyboard のみでコントロールを行う
 * <p/>
 * キーボード入力:
 * - W 又は 上矢印:  前進する
 * - S 又は 下矢印:  後進する
 * - A 又は 右矢印:  右回転
 * - D 又は 左矢印:  左回転
 * マウス入力:
 * - 移動:    動きを追って , 銃が移動
 * - 上にホイール:   前進
 * - 下ホイール:     後進
 * - ボタン 1:   砲撃をこの大きさで行う = 1
 * - ボタン 2:   砲撃をこの大きさで行う = 2
 * - ボタン 3:   砲撃をこの大きさで行う = 3
 * <p/>
 * 砲撃の威力に応じて弾丸の色を変更:
 * - Power = 1: 黄色
 * - Power = 2: オレンジ
 * - Power = 3: 赤
 * <p/>
 * Note that the robot will continue firing as long as the mouse button is
 * pressed down.
 * <p/>
 * By enabling the "Paint" button on the robot console window for this robot,
 * a cross hair will be painted for the robots current aim (controlled by the
```

```
53  * mouse).
54  *
55  * @author Flemming N. Larsen (original)
56  *
57  * @version 1.2
58  *
59  * @since 1.3.4
60  */
61 public class Interactive extends AdvancedRobot { //
        AdvancedRobot を継承した Interactive メソッド
62
63     //移動指示       : 1 = 前進        0 = そのまま          -1 = 後退
64     int moveDirection; //int 型変数moveDirection を宣言
65
66     //回転指示  :       1= 右回転       0 = 回転しない  -1 = 左回転
67     int turnDirection;
68
69     //移動時進んだ分の距離 (ピクセル)の合計
70     double moveAmount;
71
72     //目標の (x,y)座標と連携
73     int aimX, aimY;
74
75     //火力。          もし 0なら砲撃しない
76     int firePower;
77
78     //robot を実行するには run メソッドを実行しなければならない
79     public void run() {
80
81         //ロボットの機体の色をコントロールする
82         //ボディを黒色，砲身を白色，レーダーを赤色
83         setColors(Color.BLACK, Color.WHITE, Color.RED);
84
85         // 永遠ループ
86         for (;;) {
87             //ロボットを前進，後退，もしくは止まることができるように設定を行う
88             //移動方向とマウス移動時におけるピクセルの合計の乗算で前進を設定する
89             setAhead(moveAmount * moveDirection);
90
91             //残り移動距離が 0ピクセルに近づくいくようにデクリメント
92             //もしマウスホイールが止まれば自動でロボットが停止
93             //回転も終了
94             moveAmount = Math.max(0, moveAmount - 1);
95             //math クラスの max メソッドで 0 か moveAmount-1の多い方の値を
                   moveAmount に渡す
96
97             //右回転，及び左回転 (共に最大スピード)もしくは
98             //回転方向の操作で回転を止めるかを決定
99             setTurnRight(45 * turnDirection); // degrees
100            //45にturnDirection の値を乗算(
                   turnDirection は± 1 か 0 なので 45 をかけ，45 度回転)
101            //set メソッドなので実行動作が行われるまで待機
102
103            //マウスの (x,y)座標と現在の自機の (x,y)座標を持って弾丸の目標点を計算
```

```
104
105             double angle = normalAbsoluteAngle(Math.atan2(aimX - getX(), aimY
                    - getY()));
106             //double 型変数 angle に math クラスの atan2 を用いて角度θ(シータ)を返す
107
108             setTurnGunRightRadians(normalRelativeAngle(angle -
                    getGunHeadingRadians()));
109             //angle から現在の大砲の向きの絶対角度を引いた値分右回転
110             //これも set メソッドなので実行動作が行われるまで待機
111
112             //fire パワーが 0 ではない限り設定された分の砲撃を行う
113             if (firePower > 0) {
114                 setFire(firePower);
115             }
116
117             //execute()メソッドを用いて今まで待機していた全てのset メソッドを開始
118             execute();
119
120             //次のループに突入
121         }
122     }
123
124     //キーボードから何かが入力されればこのメソッドが呼び出される
125     public void onKeyPressed(KeyEvent e) { //keyPresssed のオーバーライド
126         switch (e.getKeyCode()) { //押されたキーによってスイッチ分岐
127         case VK_UP://上矢印
128         case VK_W: //w キー
129             //キーが入力され続ければ永続的に前進を行う
130             moveDirection = 1; //変数 moveDirction に 1 を代入
131             moveAmount = Double.POSITIVE_INFINITY; //変数
                    moveAmout に double 型の PSITIVE_INFINITY を収納
132             break;
133
134         case VK_DOWN://下矢印
135         case VK_S://s キー
136             //キーが入力され続ければ永続的に後進を行う
137             moveDirection = -1;//変数 moveDirection に-1を代入
138             moveAmount = Double.POSITIVE_INFINITY; //
                    moveAmount に POSITIVE_INFINITY を収納
139             break;
140
141         case VK_RIGHT://右矢印
142         case VK_D://d キー
143             // 右方向に回転
144             turnDirection = 1;
145             break;
146
147         case VK_LEFT:
148         case VK_A:
149             // 左方向に回転
150             turnDirection = -1;
151             break;
152         }
153     }
```

```
154
155     //キーボードから手を話したらこのメソッドが呼び出される
156     public void onKeyReleased(KeyEvent e) { //keyReleased メソッドのオーバーライド
157         switch (e.getKeyCode()) { //e.getKeyCode メソッド(押されていたキー)によって
                分岐
158         case VK_UP:
159         case VK_W:
160         case VK_DOWN:
161         case VK_S:
162             //上下矢印キー，W，S キーの場合
163             //先ほどまで進んでいた方向を向いたま停止
164             moveDirection = 0;
165             moveAmount = 0; //moveDirection 及び movrAmount に 0 を代入して break
166             break;
167
168         case VK_RIGHT:
169         case VK_D:
170         case VK_LEFT:
171         case VK_A:
172             //左右矢印キー，D,A キーの場合
173             //回転を停止
174             turnDirection = 0;
175             break;
176         }
177     }
178
179     //マウスホイールが回転し始めたらこのメソッドが呼び出される
180     public void onMouseWheelMoved(MouseWheelEvent e) { //
            MouseWheelEvet クラスからイベントを読みこむ
181
182         //変数 moveDirection にマウスホイールを回転させたクリック数を返す
183         //この際マウスホイールが上側に回転した場合は負の値が返される。下側に回転した場合
                は正の値
184
185         moveDirection = (e.getWheelRotation() < 0) ? 1 : -1;
186
187         //moAmount に getWheelRotation の値に 5 をかけた値を渡す(ここで 5は適当に決定され
                た。大きいほど早く回転する)
188
189         moveAmount += Math.abs(e.getWheelRotation()) * 5;
190     }
191
192     //マウスが移動した場合にこのメソッドが呼び出される
193
194     public void onMouseMoved(MouseEvent e) {
195         // 目標の設定はマウスポインタの (x,y)座標と同期
196         aimX = e.getX();
197         aimY = e.getY();
198     }
199
200     //マウスのボタンが押された際このメソッドが呼び出される
201     public void onMousePressed(MouseEvent e) {
202         if (e.getButton() == MouseEvent.BUTTON3) {
203             //ボタン 3:威力 3の砲撃:色は赤色
```

```
204             firePower = 3;
205             setBulletColor(Color.RED);
206         } else if (e.getButton() == MouseEvent.BUTTON2) {
207             //ボタン 2:威力 2の砲撃:色はオレンジ
208             firePower = 2;
209             setBulletColor(Color.ORANGE);
210         } else {
211             //ボタン 1, もしくは定義されていないボタン
212             //威力 1の砲撃:色は黄色
213             firePower = 1;
214             setBulletColor(Color.YELLOW);
215         }
216     }
217 
218     //マウスボタンから手が離れた場合呼び出されるメソッド
219     public void onMouseReleased(MouseEvent e) {
220         // 0の威力で砲撃 (撃たない)
221         firePower = 0;
222     }
223 
224     //paint ボタンを押すことでエイムを表示
225     //ペイントボタンはバトルフィールド上でクリックすることができる
226 
227     public void onPaint(Graphics2D g) {
228         //現在のエイム範囲を赤色のサークルで表示する
229 
230         g.setColor(Color.RED);
231         g.drawOval(aimX - 15, aimY - 15, 30, 30);
232         g.drawLine(aimX, aimY - 4, aimX, aimY + 4);
233         g.drawLine(aimX - 4, aimY, aimX + 4, aimY);
234     }
235 }
```

1.5 Interactive_v2.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   */
8  package sample;
9
10
11 import robocode.AdvancedRobot;
12 import robocode.util.Utils;
13 import static robocode.util.Utils.normalAbsoluteAngle;
14 import static robocode.util.Utils.normalRelativeAngle;
15
16 import java.awt.*;
17 import java.awt.event.KeyEvent;
18 import static java.awt.event.KeyEvent.*;
19 import java.awt.event.MouseEvent;
20 import java.awt.event.MouseWheelEvent;
21 import java.util.HashSet;
22 import java.util.Set;
23
24 //Interactive_v2 上で使用するメソッドの為に必要なクラスを import する
25
26 /**
27  * Interactive_v2 - a modified version of the sample robot Interactive by
        Flemming N. Larsen
28  *                 to use absolute movements (up, right, down, left) by Tuan Anh
        Nguyen.
29  * <p/>
30  * このロボットはマウスとKeyboard のみでコントロールを行う
31  * <p/>
32  * キーボード入力:
33  * - W 又は 上矢印:  前進する
34  * - S 又は 下矢印:  後進する
35  * - A 又は 右矢印:  右回転
36  * - D 又は 左矢印:  左回転
37  * マウス入力:
38  * - 移動:    動きを追って , 銃が移動
39  * - 上にホイール:   前進
40  * - 下ホイール:     後進
41  * - ボタン 1:   砲撃をこの大きさで行う = 1
42  * - ボタン 2:   砲撃をこの大きさで行う = 2
43  * - ボタン 3:   砲撃をこの大きさで行う = 3
44  * <p/>
45  * 砲撃の威力に応じて弾丸の色を変更:
46  * - Power = 1: 黄色
47  * - Power = 2: オレンジ
48  * - Power = 3: 赤
49  * Note that the robot will continue firing as long as the mouse button is
50  * pressed down.
```

```
 * <p/>
 * By enabling the "Paint" button on the robot console window for this robot,
 * a cross hair will be painted for the robots current aim (controlled by the
 * mouse).
 *
 * @author Flemming N. Larsen (original)
 * @author Tuan Anh Nguyen (contributor)
 *
 * @version 1.0
 *
 * @since 1.7.2.2
 */
public class Interactive_v2 extends AdvancedRobot {

    //エイムの(x,y)座標を宣言
    int aimX, aimY;

    //fireパワーを宣言。firePower=0は撃たないことを意味する。
    int firePower;

    //スクリーン上での方向を列挙型enumでDirectionの中に宣言
    private enum Direction {
        UP,
        DOWN,
        LEFT,
        RIGHT
    }

    //現在の移動方向
    //final修飾子を使った為，以下set<Direction> directionは変更することが出来ない
    private final Set<Direction> directions = new HashSet<Direction>();

    //robocode上でのメインメソッドの様なrunメソッド
    public void run() {

        //色の設定
        //機体:黒,銃身:白,レーザー:赤色
        setColors(Color.BLACK, Color.WHITE, Color.RED);

        //永遠ループ
        for (;;) {
            //機体が動こうとするときdistanceToMoveと距離は同じとみなす
            //setAheadメソッドを用いてdistanceToMoveの値の分前進をさせる待機命令

            setAhead(distanceToMove());

            //ボディーを回転させることによって正しい方向を示させる
            setTurnRight(angleToTurnInDegrees());

            // 銃身とエイムの(x,y)座標をマウスポインタによって決定させる

            double angle = normalAbsoluteAngle(Math.atan2(aimX - getX(), aimY
                - getY()));
```

```
103             //double 型変数 angle に aimX,Y から現在のロボットの x,
                    y 座標を引いた絶対角度を渡す
104
105             setTurnGunRightRadians(normalRelativeAngle(angle -
                    getGunHeadingRadians()));
106
107             //ラジアン角度で angle から現在主砲が向いている角度を引いた分だけ主砲を右回転
108
109             //砲撃は下記で設定した特殊な砲撃を行う。尚 0は撃たないことを意味する
110             if (firePower > 0) {
111                 setFire(firePower);
112             }
113
114             //execute()メソッドを用いて全ての待機された命令を開始する
115             execute();
116
117             // 次のループを再開させる
118         }
119     }
120
121     // キーボード入力がされた時に呼び出されるメソッド
122     public void onKeyPressed(KeyEvent e) {
123         switch (e.getKeyCode()) { //押されたキーによって分岐
124         case VK_UP:
125         case VK_W: //上矢印キー, W キーの場合
126             directions.add(Direction.UP); //
                    Direction で定義した UP を directions に追加
127             break;
128
129         case VK_DOWN:
130         case VK_S: //下矢印キー, S キーの場合
131             directions.add(Direction.DOWN); //
                    Direction で定義した DOWN を directions に追加
132             break;
133
134         case VK_RIGHT:
135         case VK_D: //右矢印キー, D キーの場合
136             directions.add(Direction.RIGHT); //
                    Direction の RIGHT を directions に追加
137             break;
138
139         case VK_LEFT:
140         case VK_A://左キー, A キーの場合
141             directions.add(Direction.LEFT); //
                    Direction の LEFT を directions に追加
142             break;
143         }
144     }
145
146     // キーボードから指を離した際に呼び出されるメソッド
147
148     public void onKeyReleased(KeyEvent e) {
149         switch (e.getKeyCode()) { //押されていたキーで分岐
150         case VK_UP:
```

```
151             case VK_W:
152                 directions.remove(Direction.UP);//上キー,
                        W キーの場合 directions から UP の要素をリムーブする
153                 break;
154
155             case VK_DOWN:
156             case VK_S:
157                 directions.remove(Direction.DOWN); //下キー,
                        S キーの場合, directions から DOWN の要素をリムーブする
158                 break;
159
160             case VK_RIGHT:
161             case VK_D:
162                 directions.remove(Direction.RIGHT);//右キー,
                        D キーの場合, directions から RIGHT の要素をリムーブする
163                 break;
164
165             case VK_LEFT:
166             case VK_A:
167                 directions.remove(Direction.LEFT);//左キー,
                        A キーの場合, directions から LEFT の要素をリムーブする
168                 break;
169             }
170         }
171
172         //マウスのホイールが動いた時に呼び出されるメソッド
173         public void onMouseWheelMoved(MouseWheelEvent e) {//特に意味は無い
174         }
175
176         //マウスが動いた際に呼び出されるメソッド
177         public void onMouseMoved(MouseEvent e) {
178             //エイムの値を現在マウスが指している値にする
179             aimX = e.getX();
180             aimY = e.getY();
181         }
182
183         //マウスのボタンが押された際に呼び出されるメソッド
184         public void onMousePressed(MouseEvent e) {
185             if (e.getButton() == MouseEvent.BUTTON3) {
186                 //3ボタン : 威力 3の砲撃 色は赤色
187                 firePower = 3;
188                 setBulletColor(Color.RED);
189             } else if (e.getButton() == MouseEvent.BUTTON2) {
190                  //2ボタン : 威力 2の砲撃 色はオレンジ
191                 firePower = 2;
192                 setBulletColor(Color.ORANGE);
193             } else {
194                 // 1ボタンかそれ以外のボタンが押された時
195                 //威力 1の砲撃 色は黄色
196                 firePower = 1;
197                 setBulletColor(Color.YELLOW);
198             }
199         }
200
201         //マウスのボタンから指が離された時に呼び出されるメソッド
```

```
202     public void onMouseReleased(MouseEvent e) {
203         // Fire power = 0は撃たないことを意味する。
204         firePower = 0;
205     }
206 
207     //robot コンソールからこの機体はペイントボタンを押す事ができる。
208     //押すとペイント機能 (エイム表示)をすることが可能
209 
210     public void onPaint(Graphics2D g) {
211 
212         //エイム範囲を赤色のサークルとして表示
213 
214         g.setColor(Color.RED);
215         g.drawOval(aimX - 15, aimY - 15, 30, 30);
216         g.drawLine(aimX, aimY - 4, aimX, aimY + 4);
217         g.drawLine(aimX - 4, aimY, aimX + 4, aimY);
218     }
219 
220     // アングルを戻した時発生する微小区間を決定し,その方向にロボットを剥ける 。
221     private double angleToTurnInDegrees() {
222         if (directions.isEmpty()) {
223             return 0;  //direction が空だった場合 0 を返す
224         }
225         return Utils.normalRelativeAngleDegrees(desiredDirection() -
                getHeading());
226             //disiredDirection()メソッドを呼び出し,そこから帰ってきた値と
                    robot の現在向いている角度を引いた者を返す
227     }
228 
229     // 移動距離を返すメソッド
230     private double distanceToMove() {
231         //既に directions の値が空になってしまっていた場合, 0 を返す
232         if (directions.isEmpty()) {
233             return 0;
234         }
235         //もし angleToTurnInDegrees が 45 より大きければ 5 ピクセルのみ動かす
236         if (Math.abs(angleToTurnInDegrees()) > 45) {
237             return 5;
238         }
239         //それ以外ならフルスピードで移動する
240         return Double.POSITIVE_INFINITY;
241     }
242 
243     //2つ以上の移動キーが入力された場合,斜め移動を行う
244 
245     private double desiredDirection() {
246         if (directions.contains(Direction.UP)) { //もし UP が収納されていて
247             if (directions.contains(Direction.RIGHT)) {//右移動が支持されたら
248                 return 45; //45を返す
249             }
250             if (directions.contains(Direction.LEFT)) { //左移動が指示されたら
251                 return 315; //315が返される
252             }
253             return 0;//ただ UP だけなら 0 を返す
```

```
254         }
255         if (directions.contains(Direction.DOWN)) { //もし DOWN が収納されていて
256             if (directions.contains(Direction.RIGHT)) { //右移動が指示されたら
257                 return 135; //135を返す
258             }
259             if (directions.contains(Direction.LEFT)) { //左移動ならば
260                 return 225; //225を返す
261             }
262             return 180;    //それ以外は 180を返す
263         }
264         if (directions.contains(Direction.RIGHT)) {
265             return 90; //右移動のみだったら 90度を返す
266         }
267         if (directions.contains(Direction.LEFT)) { //左移動のみなら 270を返す
268             return 270;
269         }
270         return 0;  //それ以外なら 0を返す
271     }
272 }
```

1.6 MyFirstJuniorRobot.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   * 所謂コピーライト
8   */
9  package sample;
10
11
12 import robocode.JuniorRobot;
13
14 //MyFirstRobot を構成する上で使用するメソッドが含まれているクラスを import する
15
16 /**
17  * MyFirstJuniorRobot - a sample robot by Flemming N. Larsen
18  * <p/>
19  * シーソーの様な動きをとる。もし敵機を検知出来なければ砲身を振り回す
20  * もし，ロボットが見つかれば周り，そして砲撃を行う
21  *
22  * @author Flemming N. Larsen (original)
23  */
24 public class MyFirstJuniorRobot extends JuniorRobot { //JuniorRobot を継承
25
26     /**
27      * MyFirstJuniorRobot's run メソッド - シーソーの様な動きがデフォルト
28      */
29     public void run() {
30         //色の設定
31         //機体:緑 主砲:黒 レーザー:青
32         setColors(green, black, blue);
33
34         //永遠にシーソー
35         while (true) {
36             ahead(100); //100ピクセル前進
37             turnGunRight(360); // 砲身を右に 360度回転
38             back(100); // 100ピクセル後進
39             turnGunRight(360); // 砲身を左に 360度回転
40         }
41     }
42
43     /**
44      *もし敵機を見つけた場合，そちらを向いて砲撃
45      */
46     public void onScannedRobot() {
47         //scan した敵機の方に砲身を向ける
48         turnGunTo(scannedAngle);
49
50         //威力 1で砲撃
51         fire(1);
52     }
```

```
53
54      /**
55       *弾があたった場合，弾に垂直に成るように回転
56       *そうすればシーソーの動きで弾丸を避けれるかもしれない
57       */
58      public void onHitByBullet() {
59          // Move ahead 100 and in the same time turn left papendicular to the
                bullet
60
61          turnAheadLeft(100, 90 - hitByBulletBearing);
62
63          //JuniorRobot 内のメソッド turnAheadLeft を用いて 100 ピクセル前進しながら，
64          //90度からhitByBulletBearing で弾丸から機体の角度を引いた値左回転する
65      }
66  }
```

1.7 MyFirstRobot.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   * 所謂コピーライト
8   */
9  package sample;
10
11
12 import robocode.HitByBulletEvent;
13 import robocode.Robot;
14 import robocode.ScannedRobotEvent;
15
16 //MyFirstRobot を構成する上で使用するメソッドが収納されているクラスを import
17
18 /**
19  * MyFirstRobot - a sample robot by Mathew Nelson.
20  * <p/>
21  * シーソーの様な動きを行う。各メソッドの最後に砲身を回転させる
22  *
23  * @author Mathew A. Nelson (original)
24  */
25 public class MyFirstRobot extends Robot { //Robot から継承
26
27     /**
28      * MyFirstRobot's run メソッド - シーソーみたいな動き
29      */
30     public void run() {
31
32         while (true) {
33             ahead(100); // 100ピクセル前進
34             turnGunRight(360); // 砲身を右に 360度回転
35             back(100); // 100ピクセル後進
36             turnGunRight(360); // 砲身を 360度右回転
37         }
38     }
39
40     /**
41      *敵機を見つけた場合攻撃
42      */
43     public void onScannedRobot(ScannedRobotEvent e) { //
           onScannnedRobot をオーバーライド
44         fire(1); //威力 1で攻撃
45     }
46
47     /**
48      *もし被弾した場合, 弾丸と直角に回転
49      *おそらくこのシーソーの動きをすれば永久的に避けることができるだろう
50      */
51     public void onHitByBullet(HitByBulletEvent e) {
```

```
52          turnLeft(90 - e.getBearing());//被弾時の位置を 90度から引いて,左回転
53      }
54  }
```

1.8 PaintingRobot.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   */
8  package sample;
9
10
11 import robocode.HitByBulletEvent;
12 import robocode.Robot;
13 import robocode.ScannedRobotEvent;
14
15 import java.awt.*;
16
17
18 /**
19  * Painting Robot は onPaint()メソッドと,getGraphics()メソッドのデモンストレーション
20  * Also demonstrate feature of debugging properties on RobotDialog
21  * <p/>
22  * 基本的にはシーソーのような動きを行い,各メソッドの最後に回転する
23  *painting を使用した際, 自機の周囲に円を作る
24  *
25  * @author Stefan Westen (original SGSample)
26  * @author Pavel Savara (contributor)
27  * 所謂コピーライト
28  */
29
30 public class PaintingRobot extends Robot { //Robotclass から継承
31
32     /**
33      * PaintingRobot's run method - シーソー動作
34      */
35     public void run() {
36         while (true) {//永遠ループ
37             ahead(100); //100ピクセル前進
38             turnGunRight(360); //360度砲身を右回転
39             back(100); //100ピクセル後退
40             turnGunRight(360); //360度砲身を右回転
41         }
42     }
43
44     /**
45      *敵機を発見した際, 砲撃
46      */
47     public void onScannedRobot(ScannedRobotEvent e) {
48         //ロボットダイアログのデバッグプロパティの機能のデモンストレーション
49         setDebugProperty("lastScannedRobot", e.getName() + " at " + e.
               getBearing() + " degrees at time " + getTime());
50         /**
51           *setDebugProperty は String を 2 つ受け取る
```

```
52            *2つめのstring には e.getName メソッドで scan したロボット名
53            *getBearing で相対速度，getTime でゲーム時間をそれぞれ文字列として渡す
54            */
55          fire(1);//威力 1で砲撃
56      }
57
58      /**
59       * 被弾時には，弾丸に垂直になるように動く
60       * 上手く行けば永遠に弾丸からシーソーの動きで避けることができるかもしれない
61       *追加で，被弾したらオレンジの円を出力する
62       */
63      public void onHitByBullet(HitByBulletEvent e) {
64          //ロボットダイアログ上のデバッグプロパティの特徴となる機能を示す
65          setDebugProperty("lastHitBy", e.getName() + " with power of bullet "
                + e.getPower() + " at time " + getTime());
66
67            /**
68                     *setDebugProperty は String を 2 つ受け取る
69                     *2つめのstring には e.getName メソッドで scan したロボット名
70                     *getPower で被弾した弾丸のパワーを
71              *getTime でゲーム時間をそれぞれ文字列として渡す
72                     */
73
74
75          setDebugProperty("lastScannedRobot", null);
76
77          //デバッグプロパティを削除
78
79          //先頭表示 (バトル・ビュー)をペイントすることでデバックを行う
80          Graphics2D g = getGraphics();
81
82          g.setColor(Color.orange);
83          g.drawOval((int) (getX() - 55), (int) (getY() - 55), 110, 110);
84          g.drawOval((int) (getX() - 56), (int) (getY() - 56), 112, 112);
85          g.drawOval((int) (getX() - 59), (int) (getY() - 59), 118, 118);
86          g.drawOval((int) (getX() - 60), (int) (getY() - 60), 120, 120);
87
88          //90度から敵機との相対角度を引いた分左回転
89          turnLeft(90 - e.getBearing());
90      }
91
92      /**
93        * paintingrobot の周辺に赤い円を作る
94        */
95      public void onPaint(Graphics2D g) {
96          g.setColor(Color.red);
97          g.drawOval((int) (getX() - 50), (int) (getY() - 50), 100, 100);
98          g.setColor(new Color(0, 0xFF, 0, 30));
99          g.fillOval((int) (getX() - 60), (int) (getY() - 60), 120, 120);
100     }
101 }
```

1.9 RamFire.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   * 所謂コピーライト
8   */
9
10 package sample;
11
12
13 import robocode.HitRobotEvent;
14 import robocode.Robot;
15 import robocode.ScannedRobotEvent;
16
17 import java.awt.*;
18
19  //RamFire.java を構築する上で必要なクラスを import
20
21 /**
22  * RamFire - a sample robot by Mathew Nelson.
23  * <p/>
24  * ドライバーは他のロボットの衝突をしようとする
25  * もし衝突したら砲撃を行う
26  *
27  * @author Mathew A. Nelson (original)
28  * @author Flemming N. Larsen (contributor)
29  */
30 public class RamFire extends Robot { //Robot クラスから継承
31     int turnDirection = 1; //時計回りか反時計回り
32                 //int 型変数 turnDirection を宣言し 1 を代入
33
34     /**
35      * run:そのあたりを回転してロボットを見つける
36      */
37     public void run() {
38         //色の設定 機体:黄緑 砲身:グレー レーダー:ダークグレー
39         setBodyColor(Color.lightGray);
40         setGunColor(Color.gray);
41         setRadarColor(Color.darkGray);
42
43         while (true) { //永遠ループ
44             turnRight(5 * turnDirection); //
                   turnDirection の値に 5 をかけた分だけ右回転
45         }
46     }
47
48     /**
49      * onScannedRobot: 他のロボットを発見したら突撃
50      */
51     public void onScannedRobot(ScannedRobotEvent e) {
```

```
52
53          if (e.getBearing() >= 0) { //敵機との相対角度が0度以上なら
54              turnDirection = 1; //turnDirectionに1を代入
55          } else {          //もし0度未満であるならば
56              turnDirection = -1;//-1を代入
57          }
58
59          turnRight(e.getBearing()); //敵機の方向に右回転
60          ahead(e.getDistance() + 5); //敵機との相対距離+5ピクセル前進
61          scan(); // scanする事でもう一度前進する可能性
62      }
63
64      /**
65       * onHitRobotメソッド: 敵機と向かい合って,最大火力で砲撃し,再び体当たり
66       */
67      public void onHitRobot(HitRobotEvent e) {
68          if (e.getBearing() >= 0) { //敵機との相対角度が0度以上なら
69              turnDirection = 1;//turnDirectionに1を代入
70          } else {        //もし角度が0度未満なら
71              turnDirection = -1; //-1をturnDirectionに代入
72          }
73          turnRight(e.getBearing()); //相対角度の分だけ右回転
74
75          //砲撃ではなく体当たりで敵機を撃破すればボーナスポイントが手に入るので
76          //このロボットでは体当たりで止めを指す
77          if (e.getEnergy() > 16) { //もし，敵機のエナジーが16以上なら
78              fire(3);   //威力3で砲撃
79          } else if (e.getEnergy() > 10) { //もし，10以上なら
80              fire(2);   //威力3で砲撃
81          } else if (e.getEnergy() > 4) { //もし，4以上なら
82              fire(1);   //威力1で砲撃
83          } else if (e.getEnergy() > 2) { //もし，2以上なら
84              fire(.5); //0.5で砲撃
85          } else if (e.getEnergy() > .4) { //もし，0.4以上なら
86              fire(.1); //威力0.1で砲撃
87          }
88          ahead(40); //そして突撃
89      }
90  }
```

1.10 SittingDuck.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   * 所謂コピーライト
8   */
9  package sample;
10
11 //SittingDuck が収録されている package 名
12
13 import robocode.AdvancedRobot;
14 import robocode.RobocodeFileOutputStream;
15
16 import java.awt.*;
17 import java.io.BufferedReader;
18 import java.io.FileReader;
19 import java.io.IOException;
20 import java.io.PrintStream;
21
22 //構成するために必要なメソッドが含まれているクラスを import する
23
24 /**
25  * SittingDuck - a sample robot by Mathew Nelson.
26  * <p/>
27  * とくに何もしないで静止している。このロボットは粘り強さを示したデモンストレーション機で
        ある。
28  *
29  * @author Mathew A. Nelson (original)
30  * @author Flemming N. Larsen (contributor)
31  * @author Andrew Magargle (contributor)
32  */
33
34 public class SittingDuck extends AdvancedRobot { //AdvancedRobot を継承
35     static boolean incrementedBattles = false; //
          static な boolean 型変数 incrementedBattles を宣言し，false を代入
36
37     public void run() {
38
39         setBodyColor(Color.yellow);
40         setGunColor(Color.yellow);
41                 //機体の色設定。機体と砲身が黄色
42
43         int roundCount, battleCount;
44
45             //int 型変数，roundCount，battleCount をそれぞれ宣言
46
47         try { //try 処理開始。例外処理が発生したら catch される
48             BufferedReader reader = null; //BufferedReader reader に null を入れる
                  (例外処理)
49             try {
```

```
50                  //  "count.dat"count.dat ファイル(ラウンドcount とバトル count を含む)
51                  //を読み出すことを試行
52
53                  reader = new BufferedReader(new FileReader(getDataFile("count.
                        dat")));
54
55                  // count の値を受け取ることを試みる
56                  roundCount = Integer.parseInt(reader.readLine());
57                  battleCount = Integer.parseInt(reader.readLine());
58
59              } finally { //finally 内部はいつでも処理される
60                  if (reader != null) {
61                      reader.close(); //もし
                            reader が null でないなら, reader.close()メソッドを呼び出す
62                  }
63              }
64          } catch (IOException e) {
65              //ファイルの読み込みができなかった場合,両者のcount を 0 にする
66              roundCount = 0;
67              battleCount = 0;
68          } catch (NumberFormatException e) {
69              // 同じくファイルの読み込みができなかった場合,両者のcount を 0 にする
70              roundCount = 0;
71              battleCount = 0;
72          }
73
74          // ラウンド終了時にインクリメント
75          roundCount++;
76
77          //もし先ほどインクリメントするのを忘れていたら
78          //メンバ変数は別に今までどおり有効となっている
79          if (!incrementedBattles) {
80              // バトルに付き#の値をインクリメント
81              battleCount++;
82              incrementedBattles = true;
83          }
84
85          PrintStream w = null;
86          try {
87              w = new PrintStream(new RobocodeFileOutputStream(getDataFile("
                        count.dat")));
88
89              w.println(roundCount);
90              w.println(battleCount);
91
92              //
                        PrintStreams はエラー処理を出さない。フラグ設定のみ行うので詳しくはここ参照
93              if (w.checkError()) {
94                  out.println("I could not write the count!");
95              }
96          } catch (IOException e) {
97              out.println("IOException trying to write: ");
98              e.printStackTrace(out);
```

```
 99          } finally {
100              if (w != null) {
101                  w.close();
102              }
103          }
104          out.println("I have been a sitting duck for " + roundCount + "
                 rounds, in " + battleCount + " battles.");
105      }
106  }
```

1.11 SpinBot.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   * 所謂コピーライト
8   */
9
10 package sample;
11
12
13 import robocode.AdvancedRobot;
14 import robocode.HitRobotEvent;
15 import robocode.ScannedRobotEvent;
16
17 import java.awt.*;
18
19 //SpinBot を構成するために import する
20
21 /**
22  * SpinBot - a sample robot by Mathew Nelson.
23  * <p/>
24  * 円運動を行い,敵機を見つけ時に砲撃する
25  *
26  * @author Mathew A. Nelson (original)
27  * @author Flemming N. Larsen (contributor)
28  */
29 public class SpinBot extends AdvancedRobot { //AdvancedRobot を継承
30
31     /**
32      * SpinBot's run メソッド - 円運動
33      */
34     public void run() {
35         //色の設定
36         //機体:青,砲身;青,レーダー,黒, ;scan:黄色
37         setBodyColor(Color.blue);
38         setGunColor(Color.blue);
39         setRadarColor(Color.black);
40         setScanColor(Color.yellow);
41
42         //永久ループ
43         while (true) {
44             //ゲーム開始時に自由に動けるかどうかを問い合わせる
45             // このロボットは右回転をよくする予定
46             setTurnRight(10000);
47             //setTurnRight メソッドを用いて 10000 度右回転させるように待機
48
49             // ロボットのSPEED を 5 に制限
50
51             setMaxVelocity(5);
52
```

```
53              /**
54                    *前進を始める。
55                *先ほど右回転を待機させていたので,前進と同時に
56                *右回転が始まる
57                */
58              ahead(10000);
59              // 繰り返し.
60          }
61      }
62
63      /**
64       * onScannedRobot 時のメソッド:高火力での砲撃!
65       */
66      public void onScannedRobot(ScannedRobotEvent e) {
67          fire(3);//威力 3での砲撃
68      }
69
70      /**
71       * onHitRobot: もしダメなら回転と前進がストップする
72       * そうなった場合,回転を維持しながら回って脱出
73       */
74      public void onHitRobot(HitRobotEvent e) { //敵機にぶつかった場合のイベント
75          if (e.getBearing() > -10 && e.getBearing() < 10) { //相対角度
                が-10度より上で, 10度未満であるならば
76              fire(3);   //威力 3で砲撃
77          }
78          if (e.isMyFault()) { //もし自分から突撃していれば
79              turnRight(10); //10度右回転
80          }
81      }
82  }
```

1.12 Target.java

```
/**
 * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * http://robocode.sourceforge.net/license/epl-v10.html
 * 所謂コピーライト
 */
package sample;


import robocode.AdvancedRobot;
import robocode.Condition;
import robocode.CustomEvent;

import java.awt.*;

//Target.java を構成する上で必要なクラスを import

/**
 * Target - a sample robot by Mathew Nelson.
 * <p/>
 * じっと座っている。エネルギーを 20失った場合動く
 * このロボットはカスタムイベントのデモンストレーションである
 *
 * @author Mathew A. Nelson (original)
 * @author Flemming N. Larsen (contributor)
 */
public class Target extends AdvancedRobot { //AdvancedRobot から拡張

    int trigger; //int 型変数を宣言

    /**
     * TrackFire's run メソッド
     */
    public void run() {
        //色の設定
        //機体:白,砲身:白,レーダー:白
        setBodyColor(Color.white);
        setGunColor(Color.white);
        setRadarColor(Color.white);

        //最初，自機が動く際はエネルギーが 80の時
        trigger = 80;
        //"trigger hit "という名前のカスタムイベントを生成
        addCustomEvent(new Condition("triggerhit") {
            public boolean test() {
                return (getEnergy() <= trigger);
        //エネルギーを確認し，trigger の値以下なら true を返す
            }
        });
    }
```

```
53
54      /**
55       * カスタムイベント発生時に呼び出されるメソッド
56       */
57      public void onCustomEvent(CustomEvent e) {
58          //このカスタムイベントは trigger hit と呼ばれる
59          if (e.getCondition().getName().equals("triggerhit")) {
60              //もし発生したコンディションが triggerhit と呼ばれるものなら
61              // trigger の値を調整するか
62              // 砲撃イベントを何度も行う
63              trigger -= 20;
64              //trigger の値から-20引く。
65
66              out.println("Ouch, down to " + (int) (getEnergy() + .5) + " energy
                    .");
67              // getEnergy で入手した数値+0.5を現在のエネルギーとして出力
68              turnLeft(65); //左 65度回転
69              ahead(100); //100ピクセル直進
70          }
71      }
72  }
```

1.13 TrackFire.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   * 所謂コピーライト
8   */
9  package sample;
10
11
12 import robocode.Robot;
13 import robocode.ScannedRobotEvent;
14 import robocode.WinEvent;
15 import static robocode.util.Utils.normalRelativeAngleDegrees;
16
17 import java.awt.*;
18
19 //TrackFire.java を構成する上で必要なクラスを import
20
21 /**
22  * TrackFire - a sample robot by Mathew Nelson.
23  * <p/>
24  * じっと座っている。周回し，砲撃するのを一番近いロボットが発見出来た時に行う
25  *
26  * @author Mathew A. Nelson (original)
27  * @author Flemming N. Larsen (contributor)
28  */
29 public class TrackFire extends Robot {
30
31     /**
32      * TrackFire's run メソッド
33      */
34     public void run() {
35         //色の設定 一通りピンク
36         setBodyColor(Color.pink);
37         setGunColor(Color.pink);
38         setRadarColor(Color.pink);
39         setScanColor(Color.pink);
40         setBulletColor(Color.pink);
41
42         //永遠ループ
43         while (true) {
44             turnGunRight(10); //砲身を右に 10度回転
45         }
46     }
47
48     /**
49      * onScannedRobot: 敵機を発見したら砲撃
50      */
51     public void onScannedRobot(ScannedRobotEvent e) {
52         //ロボットの位置を計算
```

```
53          double absoluteBearing = getHeading() + e.getBearing();
54          //double 型変数 absoluteBearing を宣言かつ
55          //現在の自機の角度と敵機との相対角度を足した値に設定
56
57          double bearingFromGun = normalRelativeAngleDegrees(absoluteBearing -
                getGunHeading());
58          //double 型変数 bearingFromGun を宣言かつ
59          //敵機の絶対角度から自機の主砲絶対角度を引いたものに設定
60
61          //充分に近ければ砲撃
62          if (Math.abs(bearingFromGun) <= 3) { //もし
                bearingFromGun の絶対値が 3 以下なら
63              turnGunRight(bearingFromGun);//その分右回転 (絶対値)
64              //fire メソッドを使いたいので自機の熱量を確認
65              //また，回転した時に他のロボットがいないか発見を行う
66
67              if (getGunHeat() == 0) { //もし，砲身の熱量が 0ならば
68                  fire(Math.min(3 - Math.abs(bearingFromGun), getEnergy() -
                        .1));
69                  //3からbearingFromGun の絶対値を引いた数か，getEnergy から 0.1を引いた
                        数
70                  //そのうちの小さい方の威力で砲撃
71              }
72          }
73
74          else { //それ以外は
75              turnGunRight(bearingFromGun); ///bearingFromGun の値分右回転
76          }
77
78          //基本的に scan は自動的に行われるのでこのプログラムは自発的には殆ど scan しない
79          // bearingFromGun が 0 の時だけ scan する
80          if (bearingFromGun == 0) {
81              scan();
82          }
83      }
84
85      public void onWin(WinEvent e) { //勝利時に呼び出されるメソッド
86          // 勝利の舞
87          turnRight(36000);//右に 36000度回転
88      }
89  }
```

1.14 Tracker.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   * 所謂コピーライト
8   */
9  package sample;
10
11
12 import robocode.HitRobotEvent;
13 import robocode.Robot;
14 import robocode.ScannedRobotEvent;
15 import robocode.WinEvent;
16 import static robocode.util.Utils.normalRelativeAngleDegrees;
17
18 import java.awt.*;
19
20 //Tracker を構成する上で必要なクラスを import
21
22 /**
23  * Tracker - a sample robot by Mathew Nelson.
24  * <p/>
25  * 上に行くロボットを見つけたら , 近づいて , 充分に近づければ砲撃
26  *
27  * @author Mathew A. Nelson (original)
28  * @author Flemming N. Larsen (contributor)
29  */
30 public class Tracker extends Robot {
31     int count = 0; //自機の周回を count すつための変数を宣言
32
33     double gunTurnAmt; //サーチ時にいくらか主砲を回転させる
34     String trackName; // 現在追跡している敵機の名前を保存
35
36     /**
37      * run: Tracker's のメインとなるrun メソッド(関数)
38      */
39     public void run() {
40         // 色の設定
41         setBodyColor(new Color(128, 128, 50));
42         setGunColor(new Color(50, 50, 20));
43         setRadarColor(new Color(200, 200, 70));
44         setScanColor(Color.white);
45         setBulletColor(Color.blue);
46
47         // 砲身の準備
48         trackName = null; // 現在はまだ何も追跡していないのでnull を入れる
49         setAdjustGunForRobotTurn(true); //回転中は主砲を停止
50         gunTurnAmt = 10; // 今現在はgunTurn を 10 に設定
51
52         // 無限ループ
```

```
53          while (true) {
54              // 敵機を探しながら主砲を回転
55              turnGunRight(gunTurnAmt);//gunTurnAmt 分
56              //どれくらい見たかを保存する為に count をインクリメント
57              count++;
58              //二周しても敵機が見つからなかったら左回転
59              if (count > 2) {
60                  gunTurnAmt = -10;//-10で左回転を意味する
61              }
62              //5count でも敵機を発見出来なかった場合，右回転に戻す
63              if (count > 5) {
64                  gunTurnAmt = 10;
65              }
66              //もし 10count しても敵機を発見出来なかった場合，別の機体を標的にする
67              if (count > 11) {
68                  trackName = null;
69              }
70          }
71      }
72
73      /**
74       * onScannedRobot: 敵機を見つけた時に呼び出されるメソッド
75       */
76      public void onScannedRobot(ScannedRobotEvent e) {
77
78          //もし，現在ターゲットが決まっていても他の敵機を見つけたら
79          // 更にスキャンイベントを入手することができる
80          if (trackName != null && !e.getName().equals(trackName)) {
81              //trackName が null でなく，getName が trackName の値と違う時
82              return; //終了
83          }
84
85          //もしターゲットがなくてもやってしまう
86          if (trackName == null) { //trackName が null だったら
87              trackName = e.getName(); //trackName に発見した敵機の名前を挿れて
88              out.println("Tracking " + trackName); //敵機の名前を出力
89          }
90          //ターゲットならメインメソッドを参照し，count を 0 にする
91          count = 0;
92          //もしターゲットが遠距離なら，ターゲットに対してターンして移動
93          if (e.getDistance() > 150) { //ターゲットとの距離が 150ピクセル以上なら
94              gunTurnAmt = normalRelativeAngleDegrees(e.getBearing() + (
                    getHeading() - getRadarHeading()));
95          //gunTurnAmt に(ターゲットとの相対角度)+(現在の角度-レーダーの角度)の角度を渡す
96
97              turnGunRight(gunTurnAmt); //gunTurnAmt の分だけ砲身右回転
98              turnRight(e.getBearing()); //turnRight に敵機との相対角度を渡す
99              ahead(e.getDistance() - 140); //敵機との距離から 140ピクセル引いた分前進
100             return;
101         }
102
103         //もし敵機と近づいていた場合
104         gunTurnAmt = normalRelativeAngleDegrees(e.getBearing() + (getHeading
                () - getRadarHeading()));
```

```
105         turnGunRight(gunTurnAmt);
106         fire(3);//威力3で砲撃を追加
107 
108         //とても近づいていた場合，バック
109         if (e.getDistance() < 100) { //距離が100以下なら
110             if (e.getBearing() > -90 && e.getBearing() <= 90) {//さらに絶対値
                    90以下なら
111                 back(40);//40ピクセルバック
112             } else {
113                 ahead(40);//それ以外なら40ピクセル前進
114             }
115         }
116         scan();//scan
117     }
118 
119     /**
120      * onHitRobot:被弾時には当ててきた方を新しいターゲットにする
121      */
122     public void onHitRobot(HitRobotEvent e) {
123         //このメソッドが呼び出された際にまだターゲットが決まっていない場合
124         if (trackName != null && !trackName.equals(e.getName())) {
125             out.println("Tracking " + e.getName() + " due to collision");
126         //コメントを出力
127 
128         }
129         // ターゲットをセット
130         trackName = e.getName();
131 
132         gunTurnAmt = normalRelativeAngleDegrees(e.getBearing() + (getHeading
                () - getRadarHeading()));
133         turnGunRight(gunTurnAmt);
134         fire(3);
135         back(50); //さきほどまでの処理に50ピクセルバックを加える
136     }
137 
138     /**
139      * onWin: 勝利の舞
140      */
141     public void onWin(WinEvent e) {
142         for (int i = 0; i < 50; i++) { //i が50になるまで
143             turnRight(30);//右30度回転
144             turnLeft(30);//左30度回転
145         }
146     }
147 }
```

1.15 VelociRobot.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   * 所謂コピーライト
8   */
9  package sample;
10
11
12 import robocode.HitByBulletEvent;
13 import robocode.HitWallEvent;
14 import robocode.RateControlRobot;
15 import robocode.ScannedRobotEvent;
16
17 //必要なクラスを import
18
19 /**
20  *Rate control robot クラスのサンプル
21  *
22  * @author Joshua Galecki (original)
23  */
24 public class VelociRobot extends RateControlRobot { //RateControlRobot の継承
25
26     int turnCounter; //int 型変数を宣言
27     public void run() { //run メソッド
28
29         turnCounter = 0;//turnCounter を初期化し，右に 15 回転
30         setGunRotationRate(15);
31
32         while (true) { //無限ループ
33             if (turnCounter % 64 == 0) { //turnCounter を 64 で割った余りが 0 ならば
34                 // 被弾したらターン後まっすぐに直す
35                 setTurnRate(0);
36                 //速度 4で前進 (set メソッドなのですぐには実行されない)
37                 setVelocityRate(4);
38             }
39             if (turnCounter % 64 == 32) { //turnCounter を 64 で割った余りが 32 なら
40                 // 素早く後退する
41                 setVelocityRate(-6);
42             }
43             turnCounter++; //turnCounter をインクリメント
44             execute(); //全ての待機動作を実行
45         }
46     }
47
48     public void onScannedRobot(ScannedRobotEvent e) {
49         fire(1);//敵機を発見したら威力 1で砲撃
50     }
51
52     public void onHitByBullet(HitByBulletEvent e) {
```

```
53          //ほかのロボットと混乱させるようにターンを行う
54          setTurnRate(5);//5ピクセルでターンをする様にセット
55      }
56
57      public void onHitWall(HitWallEvent e) {
58          //壁から離れる
59          setVelocityRate(-1 * getVelocityRate());
60          //速度を getVelocityRate に-1をかけたものにして前進
61      }
62  }
```

1.16 Walls.java

```
1  /**
2   * Copyright (c) 2001-2014 Mathew A. Nelson and Robocode contributors
3   * All rights reserved. This program and the accompanying materials
4   * are made available under the terms of the Eclipse Public License v1.0
5   * which accompanies this distribution, and is available at
6   * http://robocode.sourceforge.net/license/epl-v10.html
7   * 所謂コピーライト
8   */
9  package sample;
10
11
12 import robocode.HitRobotEvent;
13 import robocode.Robot;
14 import robocode.ScannedRobotEvent;
15
16 import java.awt.*;
17
18 //必要なクラスを import
19
20 /**
21  * Walls - a sample robot by Mathew Nelson, and maintained by Flemming N. Larsen
22  * <p/>
23  *外壁にそって動き , 顔面に砲撃
24  *
25  * @author Mathew A. Nelson (original)
26  * @author Flemming N. Larsen (contributor)
27  */
28 public class Walls extends Robot { //Robot の継承
29
30     boolean peek; // もしロボットがいたらそこでターンをしない
31     double moveAmount; // どれくらい移動するかを決定
32
33     /**
34      * run メソッド
35      */
36     public void run() {
37         //色の設定
38         setBodyColor(Color.black);
39         setGunColor(Color.black);
40         setRadarColor(Color.orange);
41         setBulletColor(Color.cyan);
42         setScanColor(Color.cyan);
43
44         //moveAmount をバトルフィールド上での最大値に変更
45         moveAmount = Math.max(getBattleFieldWidth(), getBattleFieldHeight
               ());
46         //math クラスの max メソッドを用いて大きい方の値を代入
47
48         //peek を false に設定
49         peek = false;
50
51         //壁に向かって左折
```

```
52          // getHeading()の値を90で割った余りの角度分左回転
53          turnLeft(getHeading() % 90);
54          ahead(moveAmount); //moveAmount の分直進
55          // 主砲を90度右回転する
56          peek = true;  //peek を true に設定
57          turnGunRight(90); //主砲をまず右に90度回転
58          turnRight(90); //その後機体を右に90度回転
59
60          while (true) {//永遠ループ
61              //この辺りの処理は ahead が完了するまでには終了している
62              peek = true;
63              // 壁に向かう
64              ahead(moveAmount);
65
66              peek = false;
67              // peek を false に設定
68              turnRight(90);
69              //機体を右に90度回転
70          }
71      }
72
73      /**
74       * onHitRobot: 衝突したら少し逃げる.
75       */
76      public void onHitRobot(HitRobotEvent e) {
77          //正面にいたら後退をする
78          if (e.getBearing() > -90 && e.getBearing() < 90) { //敵機が絶対値
                90度以下なら
79              back(100);//100ピクセル後進
80          } //それ以外は逆に進む
81          else {
82              ahead(100);//100ピクセル前進
83          }
84      }
85
86      /**
87       * onScannedRobot: 砲撃!
88       */
89      public void onScannedRobot(ScannedRobotEvent e) {
90          fire(2);//威力2で砲撃
91
92          if (peek) { //peek の値が true なら scan を行う
93              scan();
94          }
95      }
96  }
```

2　各ロボットの対戦

2.1　乱戦

講義の決戦が乱闘のようなので乱闘ルールで計測した

今回の対戦ルールは下記の様に行った

1. 使用バトルフィールドの大きさは 800 × 600
2. ラウンド数は 10rounds
3. Sectry Border Size はデフォルト値 100 で計測した
4. Interactive 系統 2 機は使用者の特性が如実に現れるので今回は不参加の処置を取った。

2.1.1　結果

1. Walls
2. SpinBot
3. Tracker
4. TrackFire
5. Fire
6. RamFire
7. Crazy
8. VelociRobot
9. MyFirstRobot
10. PaintingRobot
11. MyFirstJuniorRobot
12. Corners
13. Target
14. SittingDuck

2.2 考察

2.3 全体考察

やはり Walls が乱戦によっても最強の結果となった。次いで SpinBot なので，これらを対策を睨みつつ，Walls ベースで自作 robot を作成すればまずまずの結果は得られるとは思う。Walls の弱点となるような，サーチにかからない，及び，壁から離れた場所・背後からの狙撃に対応できる robot 開発が求められると思う。Crazy が比較的中頃の順位に落ち着いているので，変な動きをして流れ弾を喰らうよりも，一定の安定するパターン処理をした方がいいと考えた。

2.4 個別考察

2.4.1 Corners

角から角へ移動するが，全体的に乱戦で角が開いている状況があまりなく，タイマンなら効果を発揮するが，あまりメリットがないと見れる

2.4.2 Crazy

無造作過ぎて壁にぶつかるため自滅することが多い。しかし，流れ弾を回避することが可能であるため，多少は生き延びることが多い。

2.4.3 Fire

タイムロスが多く，集中攻撃されるとすぐに終わる。乱戦時は敵が多いので，中盤までは強いが数がまばらになると弱体化する

2.4.4 Interactive

タイマンで自分で操作した所，慣れてないのであまり出来ない

2.4.5 Interactive_v2

同上。　こういうゲームに慣れてる人なら出来るのかもしれない。多少は使いやすい

2.4.6 MyFirstJuniorRobot

反復運動を行うのでタイマンでは強いと思うが数が多い乱戦の序盤で，そこが仇となって撃沈される事が多い。ただし終盤まで生き残れば walls すら倒せるレベルで強い。

2.4.7 MYFirstRobot

動きが限りなく Simple 故，後半まで生き延びる事がたまにある。ただし攻撃手段や移動手段にレパートリーが無いので詰められない

2.4.8 PaintingRobot

MyFirstJuniorRobot の下位互換の印象。手数が少なくなってしまった。

2.4.9 RamFire

タックルが専門なので戦闘では撃沈しやすいが，ボーナスポイントを稼ぎやすい。ゲームシステム的な面で強い

2.4.10 SittingDuck

何もしない (sample ロボットなので)。まず勝てない

2.4.11 SpinBot

常に回転しているため弾丸を避けることが上手く終始強い。しかしタイマンになると弾丸の命中率が良くなく，最後は walls に倒される

2.4.12 Target

移動するメソッドの sample なのでまず勝てない

2.4.13 TrackFire

動くことが無いので乱闘には不向きかとは思ったが，以外に生き残った。敵が多い分狙いがずれても流れ弾が当たる事が多く，弾を食らってもその分当てて回復する。しかし後半になるにつれてエネルギーの消費が激しく撃沈される。

2.4.14 Tracker

敵機に接近してから集中攻撃をしかける為短期決戦の様に強い。後半まで基本的に生き残る robot の 1 機。ただし，引いて近づく動作があるので spinrobot と wall が対当する後半の局面では弱い

2.4.15 VelociRobot

撃沈される時は相当早く撃沈する。動きがややゆっくりなのと MyFirstJuniorRobot などに機体とくっつかれると格好の的に成ることが多い。タイマンなら強いとは思うが，序盤でかなり撃沈されるのが傷。

2.4.16 Walls

最強クラスの 1 機であるが，corners や TrackFire の流れ弾，的にされるケースも多く，最終決戦まで生存していない時がまれにあった。もう少し回避機能を強化できれば最強となる可能性が大。

3　あとがき

タイマン戦闘のデータも取れば良かったかもしれないが，乱戦の状況を見て状況判断をして動きを変更する robot の方が汎用性が高いと感じた。java のオーバーライドや継承など，様々な要素がソースコードに含まれていたので読むのに結構苦労した。次回の Report もしっかりと取り組みたい。それと今回から TeX を利用して Report を作成した。こちらももっと勉強したい。

参考文献

[1] 本格学習 Java 入門 [改訂新版]

[2] Java 言語プログラミングレッスン