

# 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

<b>1</b>	<b>ソースコードの解説</b>	<b>2</b>
1.1	Corners.java . . . . .	2
1.2	Crazy.java . . . . .	6
1.2.1	ソースコード . . . . .	6
1.2.2	考察 . . . . .	8
1.2.3	メリット . . . . .	9
1.2.4	デメリット . . . . .	9
1.3	Fire.java . . . . .	10
1.3.1	ソースコード . . . . .	10
1.3.2	考察 . . . . .	12
1.3.3	メリット . . . . .	12
1.3.4	デメリット . . . . .	12
1.4	Interactive.java . . . . .	13
1.4.1	ソースコード . . . . .	13
1.4.2	考察 . . . . .	17
1.4.3	メリット . . . . .	17
1.4.4	デメリット . . . . .	17
1.5	Interactive_v2.java . . . . .	17
1.5.1	ソースコード . . . . .	17
1.5.2	考察 . . . . .	23
1.5.3	メリット . . . . .	23
1.5.4	デメリット . . . . .	23
1.6	MyFirstJuniorRobot.java . . . . .	23
1.6.1	ソースコード . . . . .	23
1.6.2	考察 . . . . .	25
1.6.3	メリット . . . . .	25
1.6.4	デメリット . . . . .	25
1.7	PaintingRobot.java . . . . .	25
1.7.1	ソースコード . . . . .	25
1.7.2	考察 . . . . .	27
1.7.3	メリット . . . . .	27
1.7.4	デメリット . . . . .	27
1.8	RamFire.java . . . . .	27
1.8.1	ソースコード . . . . .	27
1.8.2	考察 . . . . .	29
1.8.3	メリット . . . . .	29
1.8.4	デメリット . . . . .	29
1.9	SittingDuck.java . . . . .	29

1.9.1	ソースコード	29
1.9.2	考察	31
1.9.3	メリット	31
1.9.4	デメリット	31
1.10	SpinBot.java	32
1.10.1	ソースコード	32
1.10.2	考察	33
1.10.3	メリット	33
1.10.4	デメリット	33
1.11	Target.java	33
1.11.1	ソースコード	33
1.11.2	考察	35
1.11.3	メリット	35
1.11.4	デメリット	35
1.12	TrackFire.java	35
1.12.1	ソースコード	35
1.12.2	考察	36
1.12.3	メリット	36
1.12.4	デメリット	37
1.13	Tracker.java	37
1.13.1	ソースコード	37
1.13.2	考察	40
1.13.3	メリット	40
1.13.4	デメリット	40
1.14	VelociRobot.java	40
1.14.1	ソースコード	40
1.14.2	考察	41
1.14.3	メリット	41
1.14.4	デメリット	41
1.15	Walls.java	41
1.15.1	ソースコード	41
1.15.2	考察	43
1.15.3	メリット	43
1.15.4	デメリット	43

## 2 あとがき

43

# 1 ソースコードの解説

## 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; //
39      // boolean 型変数 stopWhenSeeRobot を宣言し、false を代入
40      // この一行がどう活用されるかは go Corner() を参照
41
42      /**
43       * run: Corners' main run function.
44       * run メソッド: Corner の main となる run メソッド
45       */
46
47      public void run() {
48          // 機体の色を設定
49          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
68                 gunIncrement を turnGunLeft に渡す。
69             //此处では速さ 3
70             }
71             gunIncrement *= -1; //gunIncrement の値に-1をかける。これで逆向きに回転す
72             る。
73         }
74     }
75
76     /**
77     * 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
78     */
79     public void goCorner() {
80         // 回転している時は自機は止まらない
81         stopWhenSeeRobot = false;
82         // 回転時に正面を向いた壁の右側へと移動する
83         turnRight(normalRelativeAngleDegrees(corner - getHeading()));
84
85         /*Robot クラスの turnRight メソッド(機体を右回転する)を使用
86         * turnRight に渡す値を正規化するために
87         * normalRelativeAngleDegrees メソッドを通して
88         * 現在機体が向いている座標をgetHeading メソッド取得し
89         *corner からこれを引いた分の角度回転する
90         *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*/
91
92         //stopWhenSeeRobot 変数をtrueにする
93
94         stopWhenSeeRobot = true;
95
96         //5000pixel 分直進する。先ほど回転したので壁に当たる。
97         //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
98         //衝突後 turnLeft の動きに移行する
99
100        ahead(5000);
101        // 角に正面を向く
102        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
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,
117             // getDistance を渡す
118             //getDistance では自機と相手の戦車との距離を計測する
119             smartFire(e.getDistance());
120             // scan()メソッドで他のロボットを確認
121             // NOTE: もし,scan を呼び出している際に robot が見つかったならば
122             // 再びこのメソッド内の先頭に回帰する
123             scan();
124             // ここでrobot が検出されなければ
125             //resume メソッドを用いて run メソッドに回帰する
126             resume();
127         } else { //stopWhenSeeRobot が false だった場合。
128             //すなわち goCorner で回転し始める前は smartFire のみ
129             smartFire(e.getDistance());
130         }
131     }
132
133     /**
134     * smartFire: カスタムされた
135     * fire メソッド。firepower と距離によって動きが決定される。
136     *
137     * @param(パラメーター) robotDistance the distance to the robot to fire at
138     * robotDistance には onScannedRobot で読み込んだ getDistance の値が代入
139     */
140     public void smartFire(double robotDistance) {
141         if (robotDistance > 200 || getEnergy() < 15) {
142             //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
143             fire(1); //威力 1でfire
144         } else if (robotDistance > 50) { //距離が 50pixel 以上ならば
145             fire(2); //威力 2でfire
146         } else {
147             fire(3); //それ以外は威力 3でfire
148         }
149     }
150
151     /**
152     * 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
153     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others 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 (
174             //robocode では画面の上方向が0度)
175             if (corner == 270) { //corner の値が270 だった場合
176                 corner = -90; //逆に corner の値を-90にする。(360度を越してしまう為)
177             }
178             out.println("I died and did poorly... switching corner to " +
179                 corner);
180
181             //そしてこのコメントと corner の値を print
182         } else {
183             out.println("I died but did well. I will still use corner " +
184                 corner);
185             //死んだ敵機が
186             //75%以上の場合は, このコメントのみ出力し, corner の場所に変更しない
187         }
188     }
189 }

```

## 1.2 Crazy.java

### 1.2.1 ソースコード

```
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      //AdvancedRobot 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.(
49                  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(
112             run メソッド内で true にされている)
113             setBack(40000); //40000pixel 戻るように set を行う(直ちに処理は行われない)
114             movingForward = false; //movingForward を false に set する
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) { //
127         ScannedRobotEvent から値を受け取った時
128         fire(1); //威力 1で砲撃
129     }
130
131     /**
132     * onHitRobot: 戻る
133     */
134     public void onHitRobot(HitRobotEvent e) { //敵機に衝突した場合
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

### 1.3.1 ソースコード

```
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
35                      // int 型変数 dist を宣言し、50 を代入
36
37      /**
38       * run: Fire のメインとなる run メソッド
39       */
40      public void run() {
41          //色の設定
42          setBodyColor(Color.orange);
43          setGunColor(Color.orange);
44          setRadarColor(Color.red);
45          setScanColor(Color.red);
46          setBulletColor(Color.red);
47
48          //ゆっくりと砲身を右回転し続ける
49
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) { //
80     onHitByBullet のコンストラクタ
81     turnRight(normalRelativeAngleDegrees(90 - (getHeading() - e.
82         getHeading())));
83
84     /**
85      * getHeading メソッドから取得した自機の向きの角度と
86      * 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
87      れ取得。
88      * 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
89      turnRight メソッドに渡す
90      */
91
92     ahead(dist); // 先に要していたdist の値分前進する
93     dist *= -1; //dist の値に(-1)を乗算する
94     scan(); //もう一度 scan を呼び, onScannedRobot を呼び出しやすくする
95 }
96
97 /**
98  * onHitRobot: 敵機に衝突した場合,砲撃して逃走
99  */
100 public void onHitRobot(HitRobotEvent e) { //onHitRobot のオーバーライド
101     double turnGunAmt = normalRelativeAngleDegrees(e.getBearing() +
102         getHeading() - getGunHeading());

```

```
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

### 1.4.1 ソースコード

```
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 static robocode.util.Utils.normalAbsoluteAngle;
15 import static robocode.util.Utils.normalRelativeAngle;
16
17 import java.awt.*;
18 import java.awt.event.KeyEvent;
19 import static java.awt.event.KeyEvent.*;
20 import java.awt.event.MouseEvent;
21 import java.awt.event.MouseWheelEvent;
22
23 //Interactive を構成する上で必要なクラスを import する
24
25 /**
26  * Interactive - a sample robot by Flemming N. Larsen.
27  * <p/>
28  * このロボットはマウスとKeyboardのみでコントロールを行う
29  * <p/>
30  * キーボード入力:
31  * - W 又は 上矢印: 前進する
32  * - S 又は 下矢印: 後進する
33  * - A 又は 右矢印: 右回転
34  * - D 又は 左矢印: 左回転
35  * マウス入力:
36  * - 移動: 動きを追って, 銃が移動
37  * - 上にホイール: 前進
38  * - 下ホイール: 後進
39  * - ボタン 1: 砲撃をこの大きさで行う = 1
40  * - ボタン 2: 砲撃をこの大きさで行う = 2
41  * - ボタン 3: 砲撃をこの大きさで行う = 3
42  * <p/>
43  * 砲撃の威力に応じて弾丸の色を変更:
44  * - Power = 1: 黄色
45  * - Power = 2: オレンジ
46  * - Power = 3: 赤
47  * <p/>
48  * Note that the robot will continue firing as long as the mouse button is
49  * pressed down.
50  * <p/>
```

```

51  * By enabling the "Paint" button on the robot console window for this robot,
52  * 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  $\theta$  (シータ)を返す
107
108 setTurnGunRightRadians(normalRelativeAngle(angle -
    getGunHeadingRadians()));
109 //angle から現在の自機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) { //keyPressed 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 moveDirection に 1 を代入
131                 moveAmount = Double.POSITIVE_INFINITY; //変数
                    moveAmount に double 型の POSITIVE_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 }

```

#### 1.4.2 考察

#### 1.4.3 メリット

1. hoge

#### 1.4.4 デメリット

1. hoge

### 1.5 Interactive\_v2.java

#### 1.5.1 ソースコード

```

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.Utills;
13  import static robocode.util.Utills.normalAbsoluteAngle;
14  import static robocode.util.Utills.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
28   *   Flemming N. Larsen
29   *   to use absolute movements (up, right, down, left) by Tuan Anh
30   *   Nguyen.
31   * <p/>
32   * このロボットはマウスとKeyboard のみでコントロールを行う
33   * <p/>
34   * キーボード入力:
35   * - W 又は 上矢印: 前進する
36   * - S 又は 下矢印: 後進する
37   * - A 又は 右矢印: 右回転
38   * - D 又は 左矢印: 左回転
39   * マウス入力:
40   * - 移動: 動きを追って, 銃が移動
41   * - 上にホイール: 前進
42   * - 下ホイール: 後進
43   * - ボタン 1: 砲撃をこの大きさで行う = 1
44   * - ボタン 2: 砲撃をこの大きさで行う = 2
45   * - ボタン 3: 砲撃をこの大きさで行う = 3
46   * <p/>
47   * 砲撃の威力に応じて弾丸の色を変更:
48   * - Power = 1: 黄色
49   * - Power = 2: オレンジ
50   * - Power = 3: 赤
51   * Note that the robot will continue firing as long as the mouse button is
52   * pressed down.
53   * <p/>
54   * By enabling the "Paint" button on the robot console window for this robot,

```

```

53  * a cross hair will be painted for the robots current aim (controlled by the
54  * mouse).
55  *
56  * @author Flemming N. Larsen (original)
57  * @author Tuan Anh Nguyen (contributor)
58  *
59  * @version 1.0
60  *
61  * @since 1.7.2.2
62  */
63  public class Interactive_v2 extends AdvancedRobot {
64
65      //エイムの (x,y)座標を宣言
66      int aimX, aimY;
67
68      //fire パワーを宣言。firePower=0は撃たないことを意味する。
69      int firePower;
70
71      //スクリーン上での方向を列挙型 enum で Direction の中に宣言
72      private enum Direction {
73          UP,
74          DOWN,
75          LEFT,
76          RIGHT
77      }
78
79      //現在の移動方向
80      //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
81      private final Set<Direction> directions = new HashSet<Direction>();
82
83      //robocode 上でのメインメソッドの様な run メソッド
84      public void run() {
85
86          //色の設定
87          //機体:黒,銃身:白,レーザー:赤色
88          setColors(Color.BLACK, Color.WHITE, Color.RED);
89
90          //永遠ループ
91          for (;;) {
92              //機体が動こうとするとき distanceToMove と距離は同じとみなす
93              //setAhead メソッドを用いて distanceToMove の値の分前進をさせる待機命令
94
95              setAhead(distanceToMove());
96
97              //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
98              setTurnRight(angleToTurnInDegrees());
99
100             // 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
101
102             double angle = normalAbsoluteAngle(Math.atan2(aimX - getX(), aimY
103                 - getY()));
104             //double 型変数 angle に aimX,Y から現在のロボットの x,
105             y 座標を引いた絶対角度を渡す

```

```

105         setTurnGunRightRadians(normalRelativeAngle(angle -
106             getGunHeadingRadians()));
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); //
127                     Direction で定義した UP を directions に追加
128                 break;
129
130             case VK_DOWN:
131             case VK_S: //下矢印キー, S キーの場合
132                 directions.add(Direction.DOWN); //
133                     Direction で定義した DOWN を directions に追加
134                 break;
135
136             case VK_RIGHT:
137             case VK_D: //右矢印キー, D キーの場合
138                 directions.add(Direction.RIGHT); //
139                     Direction の RIGHT を directions に追加
140                 break;
141
142             case VK_LEFT:
143             case VK_A: //左キー, A キーの場合
144                 directions.add(Direction.LEFT); //
145                     Direction の LEFT を directions に追加
146                 break;
147         }
148     }
149
150     // キーボードから指を離した際に呼び出されるメソッド
151
152     public void onKeyReleased(KeyEvent e) {
153         switch (e.getKeyCode()) { //押されていたキーで分岐
154             case VK_UP:
155             case VK_W:
156                 directions.remove(Direction.UP); //上キー,
157                     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 }

```

### 1.5.2 考察

### 1.5.3 メリット

1. hoge

### 1.5.4 デメリット

1. hoge

## 1.6 MyFirstJuniorRobot.java

### 1.6.1 ソースコード

```

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、ロボットが見つければ周り、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 perpendicular 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  }

```

### 1.6.2 考察

### 1.6.3 メリット

1. hoge

### 1.6.4 デメリット

1. hoge

## 1.7 PaintingRobot.java

### 1.7.1 ソースコード

```
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 SGSSample)
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 }

```

## 1.7.2 考察

## 1.7.3 メリット

1. hoge

## 1.7.4 デメリット

1. hoge

# 1.8 RamFire.java

## 1.8.1 ソースコード

```

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); //
45                  turnDirection の値に 5 をかけた分だけ右回転
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 }

```

## 1.8.2 考察

## 1.8.3 メリット

1. hoge

## 1.8.4 デメリット

1. hoge

# 1.9 SittingDuck.java

## 1.9.1 ソースコード

```

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 {
48             BufferedReader reader = null;
49             try {
50                 // Read file "count.dat" which contains 2 lines, a round count,
                    and a battle count
51                 reader = new BufferedReader(new FileReader(getDataFile("count.
                        dat"))));
52
53                 // Try to get the counts
54                 roundCount = Integer.parseInt(reader.readLine());
55                 battleCount = Integer.parseInt(reader.readLine());
56
57             } finally {
58                 if (reader != null) {
59                     reader.close();
60                 }
61             }
62         } catch (IOException e) {
63             // Something went wrong reading the file, reset to 0.
64             roundCount = 0;
65             battleCount = 0;
66         } catch (NumberFormatException e) {
67             // Something went wrong converting to ints, reset to 0

```

```

68         roundCount = 0;
69         battleCount = 0;
70     }
71
72     // Increment the # of rounds
73     roundCount++;
74
75     // If we haven't incremented # of battles already,
76     // Note: Because robots are only instantiated once per battle, member
77     //       variables remain valid throughout it.
78     if (!incrementedBattles) {
79         // Increment # of battles
80         battleCount++;
81         incrementedBattles = true;
82     }
83
84     PrintStream w = null;
85     try {
86         w = new PrintStream(new RobocodeFileOutputStream(getDataFile("
87             count.dat")));
88
89         w.println(roundCount);
90         w.println(battleCount);
91
92         // PrintStreams don't throw IOExceptions during prints, they simply
93         // set a flag.... so check it here.
94         if (w.checkError()) {
95             out.println("I could not write the count!");
96         }
97     } catch (IOException e) {
98         out.println("IOException trying to write: ");
99         e.printStackTrace(out);
100     } finally {
101         if (w != null) {
102             w.close();
103         }
104     }
105
106     out.println("I have been a sitting duck for " + roundCount + "
107         rounds, in " + battleCount + " battles.");
108 }

```

### 1.9.2 考察

### 1.9.3 メリット

1. hoge

### 1.9.4 デメリット

1. hoge



## 1.10 SpinBot.java

### 1.10.1 ソースコード

```
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.HitRobotEvent;
13  import robocode.ScannedRobotEvent;
14
15  import java.awt.*;
16
17
18  /**
19   * SpinBot - a sample robot by Mathew Nelson.
20   * <p/>
21   * Moves in a circle, firing hard when an enemy is detected.
22   *
23   * @author Mathew A. Nelson (original)
24   * @author Flemming N. Larsen (contributor)
25   */
26  public class SpinBot extends AdvancedRobot {
27
28      /**
29       * SpinBot's run method - Circle
30       */
31      public void run() {
32          // Set colors
33          setBodyColor(Color.blue);
34          setGunColor(Color.blue);
35          setRadarColor(Color.black);
36          setScanColor(Color.yellow);
37
38          // Loop forever
39          while (true) {
40              // Tell the game that when we take move,
41              // we'll also want to turn right... a lot.
42              setTurnRight(10000);
43              // Limit our speed to 5
44              setMaxVelocity(5);
45              // Start moving (and turning)
46              ahead(10000);
47              // Repeat.
48          }
49      }
50  }
```

```

51  /**
52   * onScannedRobot: Fire hard!
53   */
54  public void onScannedRobot(ScannedRobotEvent e) {
55      fire(3);
56  }
57
58  /**
59   * onHitRobot: If it's our fault, we'll stop turning and moving,
60   * so we need to turn again to keep spinning.
61   */
62  public void onHitRobot(HitRobotEvent e) {
63      if (e.getBearing() > -10 && e.getBearing() < 10) {
64          fire(3);
65      }
66      if (e.isMyFault()) {
67          turnRight(10);
68      }
69  }
70 }

```

### 1.10.2 考察

### 1.10.3 メリット

1. hoge

### 1.10.4 デメリット

1. hoge

## 1.11 Target.java

### 1.11.1 ソースコード

```

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.Condition;
13  import robocode.CustomEvent;

```

```

14
15 import java.awt.*;
16
17
18 /**
19  * Target - a sample robot by Mathew Nelson.
20  * <p/>
21  * Sits still. Moves every time energy drops by 20.
22  * This Robot demonstrates custom events.
23  *
24  * @author Mathew A. Nelson (original)
25  * @author Flemming N. Larsen (contributor)
26  */
27 public class Target extends AdvancedRobot {
28
29     int trigger; // Keeps track of when to move
30
31     /**
32      * TrackFire's run method
33      */
34     public void run() {
35         // Set colors
36         setBodyColor(Color.white);
37         setGunColor(Color.white);
38         setRadarColor(Color.white);
39
40         // Initially, we'll move when life hits 80
41         trigger = 80;
42         // Add a custom event named "trigger hit",
43         addCustomEvent(new Condition("triggerhit") {
44             public boolean test() {
45                 return (getEnergy() <= trigger);
46             }
47         });
48     }
49
50     /**
51      * onCustomEvent handler
52      */
53     public void onCustomEvent(CustomEvent e) {
54         // If our custom event "triggerhit" went off,
55         if (e.getCondition().getName().equals("triggerhit")) {
56             // Adjust the trigger value, or
57             // else the event will fire again and again and again...
58             trigger -= 20;
59             out.println("Ouch, down to " + (int) (getEnergy() + .5) + " energy
60                 .");
61             // move around a bit.
62             turnLeft(65);
63             ahead(100);
64         }
65     }

```

### 1.11.2 考察

### 1.11.3 メリット

1. hoge

### 1.11.4 デメリット

1. hoge

## 1.12 TrackFire.java

### 1.12.1 ソースコード

```
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.Robot;
12  import robocode.ScannedRobotEvent;
13  import robocode.WinEvent;
14  import static robocode.util.Utils.normalRelativeAngleDegrees;
15
16  import java.awt.*;
17
18
19  /**
20   * TrackFire - a sample robot by Mathew Nelson.
21   * <p/>
22   * Sits still. Tracks and fires at the nearest robot it sees.
23   *
24   * @author Mathew A. Nelson (original)
25   * @author Flemming N. Larsen (contributor)
26   */
27  public class TrackFire extends Robot {
28
29      /**
30       * TrackFire's run method
31       */
32      public void run() {
33          // Set colors
34          setBodyColor(Color.pink);
35          setGunColor(Color.pink);
36          setRadarColor(Color.pink);
37          setScanColor(Color.pink);
```

```

38         setBulletColor(Color.pink);
39
40         // Loop forever
41         while (true) {
42             turnGunRight(10); // Scans automatically
43         }
44     }
45
46     /**
47      * onScannedRobot: We have a target. Go get it.
48      */
49     public void onScannedRobot(ScannedRobotEvent e) {
50         // Calculate exact location of the robot
51         double absoluteBearing = getHeading() + e.getBearing();
52         double bearingFromGun = normalRelativeAngleDegrees(absoluteBearing -
            getGunHeading());
53
54         // If it's close enough, fire!
55         if (Math.abs(bearingFromGun) <= 3) {
56             turnGunRight(bearingFromGun);
57             // We check gun heat here, because calling fire()
58             // uses a turn, which could cause us to lose track
59             // of the other robot.
60             if (getGunHeat() == 0) {
61                 fire(Math.min(3 - Math.abs(bearingFromGun), getEnergy() -
                    .1));
62             }
63         } // otherwise just set the gun to turn.
64         // Note: This will have no effect until we call scan()
65         else {
66             turnGunRight(bearingFromGun);
67         }
68         // Generates another scan event if we see a robot.
69         // We only need to call this if the gun (and therefore radar)
70         // are not turning. Otherwise, scan is called automatically.
71         if (bearingFromGun == 0) {
72             scan();
73         }
74     }
75
76     public void onWin(WinEvent e) {
77         // Victory dance
78         turnRight(36000);
79     }
80 }

```

### 1.12.2 考察

### 1.12.3 メリット

1. hoge

#### 1.12.4 デメリット

1. hoge

### 1.13 Tracker.java

#### 1.13.1 ソースコード

```
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.HitRobotEvent;
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
20  /**
21   * Tracker - a sample robot by Mathew Nelson.
22   * <p/>
23   * Locks onto a robot, moves close, fires when close.
24   *
25   * @author Mathew A. Nelson (original)
26   * @author Flemming N. Larsen (contributor)
27   */
28  public class Tracker extends Robot {
29      int count = 0; // Keeps track of how long we've
30      // been searching for our target
31      double gunTurnAmt; // How much to turn our gun when searching
32      String trackName; // Name of the robot we're currently tracking
33
34      /**
35       * run: Tracker's main run function
36       */
37      public void run() {
38          // Set colors
39          setBodyColor(new Color(128, 128, 50));
40          setGunColor(new Color(50, 50, 20));
41          setRadarColor(new Color(200, 200, 70));
42          setScanColor(Color.white);
43          setBulletColor(Color.blue);
44      }
```

```

45 // Prepare gun
46 trackName = null; // Initialize to not tracking anyone
47 setAdjustGunForRobotTurn(true); // Keep the gun still when we turn
48 gunTurnAmt = 10; // Initialize gunTurn to 10
49
50 // Loop forever
51 while (true) {
52     // turn the Gun (looks for enemy)
53     turnGunRight(gunTurnAmt);
54     // Keep track of how long we've been looking
55     count++;
56     // If we've haven't seen our target for 2 turns, look left
57     if (count > 2) {
58         gunTurnAmt = -10;
59     }
60     // If we still haven't seen our target for 5 turns, look right
61     if (count > 5) {
62         gunTurnAmt = 10;
63     }
64     // If we *still* haven't seen our target after 10 turns, find another
        target
65     if (count > 11) {
66         trackName = null;
67     }
68 }
69 }
70
71 /**
72  * onScannedRobot: Here's the good stuff
73  */
74 public void onScannedRobot(ScannedRobotEvent e) {
75
76     // If we have a target, and this isn't it, return immediately
77     // so we can get more ScannedRobotEvents.
78     if (trackName != null && !e.getName().equals(trackName)) {
79         return;
80     }
81
82     // If we don't have a target, well, now we do!
83     if (trackName == null) {
84         trackName = e.getName();
85         out.println("Tracking " + trackName);
86     }
87     // This is our target. Reset count (see the run method)
88     count = 0;
89     // If our target is too far away, turn and move toward it.
90     if (e.getDistance() > 150) {
91         gunTurnAmt = normalRelativeAngleDegrees(e.getBearing() + (
            getHeading() - getRadarHeading()));
92
93         turnGunRight(gunTurnAmt); // Try changing these to setTurnGunRight,
94         turnRight(e.getBearing()); // and see how much Tracker improves...
95         // (you'll have to make Tracker an AdvancedRobot)
96         ahead(e.getDistance() - 140);

```

```

97         return;
98     }
99
100    // Our target is close.
101    gunTurnAmt = normalRelativeAngleDegrees(e.getBearing() + (getHeading
        () - getRadarHeading()));
102    turnGunRight(gunTurnAmt);
103    fire(3);
104
105    // Our target is too close! Back up.
106    if (e.getDistance() < 100) {
107        if (e.getBearing() > -90 && e.getBearing() <= 90) {
108            back(40);
109        } else {
110            ahead(40);
111        }
112    }
113    scan();
114 }
115
116 /**
117  * onHitRobot: Set him as our new target
118  */
119 public void onHitRobot(HitRobotEvent e) {
120     // Only print if he's not already our target.
121     if (trackName != null && !trackName.equals(e.getName())) {
122         out.println("Tracking " + e.getName() + " due to collision");
123     }
124     // Set the target
125     trackName = e.getName();
126     // Back up a bit.
127     // Note: We won't get scan events while we're doing this!
128     // An AdvancedRobot might use setBack(); execute();
129     gunTurnAmt = normalRelativeAngleDegrees(e.getBearing() + (getHeading
        () - getRadarHeading()));
130     turnGunRight(gunTurnAmt);
131     fire(3);
132     back(50);
133 }
134
135 /**
136  * onWin: Do a victory dance
137  */
138 public void onWin(WinEvent e) {
139     for (int i = 0; i < 50; i++) {
140         turnRight(30);
141         turnLeft(30);
142     }
143 }
144 }

```



### 1.13.2 考察

### 1.13.3 メリット

1. hoge

### 1.13.4 デメリット

1. hoge

## 1.14 VelociRobot.java

### 1.14.1 ソースコード

```
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.HitWallEvent;
13  import robocode.RateControlRobot;
14  import robocode.ScannedRobotEvent;
15
16
17  /**
18   * This is a sample of a robot using the RateControlRobot class
19   *
20   * @author Joshua Galecki (original)
21   */
22  public class VelociRobot extends RateControlRobot {
23
24      int turnCounter;
25      public void run() {
26
27          turnCounter = 0;
28          setGunRotationRate(15);
29
30          while (true) {
31              if (turnCounter % 64 == 0) {
32                  // Straighten out, if we were hit by a bullet and are turning
33                  setTurnRate(0);
34                  // Go forward with a velocity of 4
35                  setVelocityRate(4);
36              }
37              if (turnCounter % 64 == 32) {
```

```

38         // Go backwards, faster
39         setVelocityRate(-6);
40     }
41     turnCounter++;
42     execute();
43 }
44 }
45
46 public void onScannedRobot(ScannedRobotEvent e) {
47     fire(1);
48 }
49
50 public void onHitByBullet(HitByBulletEvent e) {
51     // Turn to confuse the other robot
52     setTurnRate(5);
53 }
54
55 public void onHitWall(HitWallEvent e) {
56     // Move away from the wall
57     setVelocityRate(-1 * getVelocityRate());
58 }
59 }

```

#### 1.14.2 考察

#### 1.14.3 メリット

1. hoge

#### 1.14.4 デメリット

1. hoge

### 1.15 Walls.java

#### 1.15.1 ソースコード

```

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.HitRobotEvent;

```

```

12 import robocode.Robot;
13 import robocode.ScannedRobotEvent;
14
15 import java.awt.*;
16
17
18 /**
19  * Walls - a sample robot by Mathew Nelson, and maintained by Flemming N. Larsen
20  * <p/>
21  * Moves around the outer edge with the gun facing in.
22  *
23  * @author Mathew A. Nelson (original)
24  * @author Flemming N. Larsen (contributor)
25  */
26 public class Walls extends Robot {
27
28     boolean peek; // Don't turn if there's a robot there
29     double moveAmount; // How much to move
30
31     /**
32      * run: Move around the walls
33      */
34     public void run() {
35         // Set colors
36         setBodyColor(Color.black);
37         setGunColor(Color.black);
38         setRadarColor(Color.orange);
39         setBulletColor(Color.cyan);
40         setScanColor(Color.cyan);
41
42         // Initialize moveAmount to the maximum possible for this battlefield.
43         moveAmount = Math.max(getBattleFieldWidth(), getBattleFieldHeight
44             ());
45         // Initialize peek to false
46         peek = false;
47
48         // turnLeft to face a wall.
49         // getHeading() % 90 means the remainder of
50         // getHeading() divided by 90.
51         turnLeft(getHeading() % 90);
52         ahead(moveAmount);
53         // Turn the gun to turn right 90 degrees.
54         peek = true;
55         turnGunRight(90);
56         turnRight(90);
57
58         while (true) {
59             // Look before we turn when ahead() completes.
60             peek = true;
61             // Move up the wall
62             ahead(moveAmount);
63             // Don't look now
64             peek = false;
65             // Turn to the next wall

```

```

65         turnRight(90);
66     }
67 }
68
69 /**
70  * onHitRobot: Move away a bit.
71  */
72 public void onHitRobot(HitRobotEvent e) {
73     // If he's in front of us, set back up a bit.
74     if (e.getBearing() > -90 && e.getBearing() < 90) {
75         back(100);
76     } // else he's in back of us, so set ahead a bit.
77     else {
78         ahead(100);
79     }
80 }
81
82 /**
83  * onScannedRobot: Fire!
84  */
85 public void onScannedRobot(ScannedRobotEvent e) {
86     fire(2);
87     // Note that scan is called automatically when the robot is moving.
88     // By calling it manually here, we make sure we generate another scan
89     // event if there's a robot on the next
90     // wall, so that we do not start moving up it until it's gone.
91     if (peek) {
92         scan();
93     }
94 }

```

### 1.15.2 考察

#### 1.15.3 メリット

1. hoge

#### 1.15.4 デメリット

1. hoge

## 2 あとがき

う え い

## 参考文献

[1] Java 言語プログラミングレッスン

[2] java の教科書