H29 ネットワークプログラミング　レポート

曽根 森之介

2018/02/27

1　本レポートについて

本授業で作製したロボットプログラムを以下に示す.

改良点

・GUI の実装

・login, logout コマンドの実装

・燃料座標の読取り，移動

ソースコード 1　Robot.java

```
1  //「海ゲーム」GUI クライアントプログラム UmiClient.java
2  // このプログラムは,海ゲームのクライアントプログラムです
3  // 使い方java UmiClient
4  // 起動してlogin ボタンを押すと,接続先サーバの名前や利用者の名前を問い合わせてきます
5  // サーバ名と利用者名を入力してください
6  // 続いてOK ボタンを押すと,ポート番号 10000 番でサーバと接続します
7  //
8  // プログラムを停止するにはlogout ボタンを押してください
9
10 // ライブラリの利用
11 import java.awt.*;// グラフィックス
12 import java.awt.event.*;// イベント関連
13 import java.net.*;// ネットワーク関連
14 import java.io.*;
15 import java.util.*;
16
17 // UmiClient_robot クラス
18 // UmiClient_robot プログラムの中心となるクラスです
19 public class UmiClient_robot implements Runnable {
20         Frame f;// クライアント情報表示用ウィンドウ
21         Panel p;// 上下左右の移動ボタンと海の状態を表示するパネル
22         Canvas c;// 海の状態を表示するキャンバス
23
24         int a, b, x, y = 0;
25         int sleeptime = 5 ;
26
27         // コンストラクタ
28         // GUI 画面の初期配置を行います
29         public UmiClient_robot ()
30         {
31                 Button b;
```

```
32              f = new Frame();//クライアント情報ウィンドウ全体の表示
33              p = new Panel();//海表示部分と操作ボタンの表示
34              p.setLayout(new BorderLayout());
35
36              // up ボタンの作成
37              b = new Button("up");
38              b.addActionListener(new ActionListener(){
39                      // up ボタンが押されたら up コマンドを送信します
40                      public void actionPerformed(ActionEvent e){
41                              sendCommand("up");
42                      }
43              });
44              p.add(b, BorderLayout.NORTH);
45
46              // left ボタンの作成
47              b = new Button("left");
48              b.addActionListener(new ActionListener(){
49                      // left ボタンが押されたら left コマンドを送信します
50                      public void actionPerformed(ActionEvent e){
51                              sendCommand("left");
52                      }
53              });
54              p.add(b, BorderLayout.WEST);
55
56              // right ボタンの作成
57              b = new Button("right");
58              b.addActionListener(new ActionListener(){
59                      // right ボタンが押されたら right コマンドを送信します
60                      public void actionPerformed(ActionEvent e){
61                              sendCommand("right");
62                      }
63              });
64              p.add(b, BorderLayout.EAST);
65
66              // down ボタンの作成
67              b = new Button("down");
68              b.addActionListener(new ActionListener(){
69                      // down ボタンが押されたら down コマンドを送信します
70                      public void actionPerformed(ActionEvent e){
71                              sendCommand("down");
72                      }
73              });
74              p.add(b, BorderLayout.SOUTH);
75
76              // 海上の様子を表示するCanvas を作成します
77              c = new Canvas();
78              c.setSize(256,256);// 大きさの設定
79              // フレームに必要な部品を取り付けます
80              p.add(c);
81              f.add(p);
82
83              // フレームf に login ボタンを取り付けます
84              b = new Button("login");
85              b.addActionListener(new ActionListener(){
86                      public void actionPerformed(ActionEvent e){
87                              // login ボタンが押された場合の処理
88                              // サーバがセットされていなければlogin 処理を行います
```

```
89                                      if(server == null) login();
90                              }
91                      });
92                      f.add(b,BorderLayout.NORTH);
93
94                      // フレームfに logout ボタンを取り付けます
95                      b = new Button("logout");
96                      b.addActionListener(new ActionListener(){
97                              public void actionPerformed(ActionEvent e){
98                                      logout();
99                              }
100                     });
101                     f.add(b,BorderLayout.SOUTH);
102
103                     // フレームfを表示します
104                     f.setSize(335,345);
105 //  f.show();
106                     f.setVisible(true);
107             }
108
109             // run メソッド
110             // 500ミリ秒ごとに画面を更新します
111             public void run(){
112                     while (true){
113                             try {
114                                     Thread.sleep(100);
115                             }catch(Exception e){
116                             }
117                             // repain メソッドを用いて,サーバ上の情報を画面に出力します
118                             repaint();
119                     }
120             }
121
122             // login 処理関連のオブジェクト
123             int sx = 100;
124             int sy = 100;
125             TextField host, tf_name;
126             Dialog d;
127
128             // login メソッド
129             // login ウィンドウを表示し,必要な情報を得ます
130             // 実際のlogin 処理は,realLogin メソッドで行います
131             void login(){
132                     // ウィンドウの表示とデータの入力
133                     d = new Dialog(f, true);
134                     host = new TextField(10) ;
135                     tf_name = new TextField(10) ;
136                     d.setLayout(new GridLayout(3,2));
137                     d.add(new Label("host:"));
138                     d.add(host);
139                     d.add(new Label("name:"));
140                     d.add(tf_name);
141                     Button b = new Button("OK");
142                     b.addActionListener(new ActionListener(){
143                             // 入力が完了したら,readlLogin メソッドを使ってサーバに login します
144                             public void actionPerformed(ActionEvent e){
145                                     realLogin(host.getText(), tf_name.getText());
```

```
146                                     d.dispose();
147                             }
148                     });
149                     d.add(b);
150                     d.setResizable(true);
151                     d.setSize(200, 150);
152 //  d.show();
153                     d.setVisible(true);
154                     (new Thread(this)).start();
155             }
156
157             // realLogin 関連のオブジェクト
158             Socket server;// ゲームサーバとの接続ソケット
159             int port = 10000;// 接続ポート
160             BufferedReader in;// 入力ストリーム
161             PrintWriter out;// 出力ストリーム
162             String name;// ゲーム参加者の名前
163
164             // realLogin メソッド
165             // サーバへのlogin 処理を行います
166             void realLogin(String host, String name){
167                     try {
168                             // サーバとの接続
169                             this.name = name;
170                             server = new Socket(host, port);
171                             in = new BufferedReader(new InputStreamReader(
172                               server.getInputStream()));
173                             out = new PrintWriter(server.getOutputStream());
174
175                             // login コマンドの送付
176                             out.println("login␣" + name);
177                             out.flush();
178                             repaint();
179                     }catch(Exception e){
180                             e.printStackTrace();
181                             System.exit(1);
182                     }
183             }
184
185             // logout メソッド
186             // logout 処理を行います
187             void logout(){
188                     try {
189                             // logout コマンドの送付
190                             out.println("logout");
191                             out.flush();
192                             server.close();
193                     }catch (Exception e){
194                             ;
195                     }
196                     System.exit(0);
197             }
198
199             // repaint メソッド
200             // サーバからゲームの情報を得て,クライアントの画面を描き直します
201             void repaint(){
202                     // サーバにstat コマンドを送付し,盤面の様子などの情報を得ます
```

```
203                 out.println("stat");
204                 out.flush();
205
206                 try {
207                         String line = in.readLine();// サーバからの入力の読み込み
208                         Graphics g = c.getGraphics();// Canvas c に海の様子を描きます
209                         Font font = new Font(null, Font.PLAIN, 8);
210
211                         // 海の描画 (単なる青い四角形です)
212                         g.setColor(Color.blue);
213                         g.fillRect(0, 0, 256, 256);
214
215                         //ship_info から始まる船の情報の先頭行を探します
216                         while (!"ship_info".equalsIgnoreCase(line))
217                                 line = in.readLine();
218
219                         // 船の情報ship_info の表示
220                         // ship_info はピリオドのみの行で終了です
221                         line = in.readLine();
222                         while (!".".equals(line)){
223                                 StringTokenizer st = new StringTokenizer(line);
224                                 // 名前を読み取ります
225                                 String obj_name = st.nextToken().trim();
226
227                                 // 自分の船は赤 (red)で表示し,他人の船は緑 (green)で表示します
228                                 if (obj_name.equals(name))//自分の船
229                                         g.setColor(Color.red);
230                                 else // 他人の船
231                                         g.setColor(Color.green);
232
233                                 // 船の位置座標を読み取ります
234                                 a = Integer.parseInt(st.nextToken()) ;
235                                 b = Integer.parseInt(st.nextToken()) ;
236
237                                 // 船を表示します
238                                 g.fillOval(a - 10, 256 - b - 10, 20, 20);
239 //                              // 得点を船の右下に表示します
240 //                              g.drawString(st.nextToken(),a+10,256-b+10) ;
241                                 // 名前を船の右上に表示します
242                                 g.drawString(obj_name,a+10,256-b-10) ;
243
244                                 // 次の 1 行を読み取ります
245                                 line = in.readLine();
246                         }
247
248                         // energy_info から始まる,燃料タンクの情報を待ち受けます
249                         while (!"energy_info".equalsIgnoreCase(line))
250                                 line = in.readLine();
251
252                         // 燃料タンクの情報energy_info の表示
253                         // energy_info はピリオドのみの行で終了です
254                         line = in.readLine();
255                         while (!".".equals(line)){
256                                 StringTokenizer st = new StringTokenizer(line);
257
258                                 Thread.sleep(sleeptime*10);
259                                 // 燃料タンクの位置座標を読み取ります
```

```
260                     x = Integer.parseInt(st.nextToken()) ;
261                     y = Integer.parseInt(st.nextToken()) ;
262                     int ene = Integer.parseInt(st.nextToken()) ;
263
264                     int da = Math.abs(x-a);
265                     int db = Math.abs(y-b);
266
267                     if(da<=5 && y>b && db>5) {
268                             Thread.sleep(sleeptime*100);
269                             out.println("up");
270                             out.println("up");
271                             out.flush();
272                     }
273
274                     if(da<=5 && y<b && db>5) {
275                             Thread.sleep(sleeptime*100);
276                             out.println("down");
277                             out.println("down");
278                             out.flush();
279                     }
280
281                     if(db<=5 && x>a && da>5) {
282                             Thread.sleep(sleeptime*100);
283                             out.println("right");
284                             out.println("right");
285                             out.flush();
286                     }
287
288                     if(db<=5 && x<a && da>5) {
289                             Thread.sleep(sleeptime*100);
290                             out.println("left");
291                             out.println("left");
292                             out.flush();
293                     }
294
295
296                     // energy on the left diagonally bottom
297                     if(a>x && b>y) {
298                             Thread.sleep(sleeptime*100);
299                             out.println("left");
300                             out.println("down");
301                             out.flush();
302                     }
303
304                     // energy on the right diagonally bottom
305                     if(a<x && b>y) {
306                             Thread.sleep(sleeptime*100);
307                             out.println("right");
308                             out.println("down");
309                             out.flush();
310                     }
311
312                     // energy on the left diagonally top
313                     if(a>x && b<y) {
314                             Thread.sleep(sleeptime*100);
315                             out.println("left");
316                             out.println("up");
```

```
317                                     out.flush();;
318                             }
319
320                             // energy on the right diagonally top
321                             if(a<x && b<y) {
322                                     Thread.sleep(sleeptime*100);
323                                     out.println("right");
324                                     out.println("up");
325                                     out.flush();
326                             }
327
328                             // 燃料タンクは,白抜きの赤丸で示します
329                             g.setColor(Color.red);
330                             g.fillOval(x - 5, 256 - y - 5, 10, 10);
331                             g.setColor(Color.white);
332                             g.fillOval(x - 3, 256 - y - 3, 6, 6);
333                             g.setColor(Color.black);
334                             g.setFont(font);
335                             g.drawString(""+ene, x-3+2, 256-y+3);
336
337                             // 次の 1 行を読み取ります
338                             line = in.readLine();
339                     }
340             }catch (Exception e){
341             e.printStackTrace();
342             System.exit(1);
343             }
344     }
345
346     // sendCommand メソッド
347     // サーバへコマンドを送信します
348     void sendCommand(String s){
349             if ("up".equals(s)){
350                     out.println("up");
351             }else if ("down".equals(s)){
352                     out.println("down");
353             }else if ("left".equals(s)){
354                     out.println("left");
355             }else if ("right".equals(s)){
356                     out.println("right");
357             }
358             out.flush();
359     }
360
361     // main メソッド
362     // UmiClient を起動します
363     public static void main(String[] arg){
364             new UmiClient_robot();
365     }
366 }
```