スパイダソリティア　プログラム解説

presented by adp

2006 年 10 月 14 日

目次

1　設計の過程

1.1　クラス図

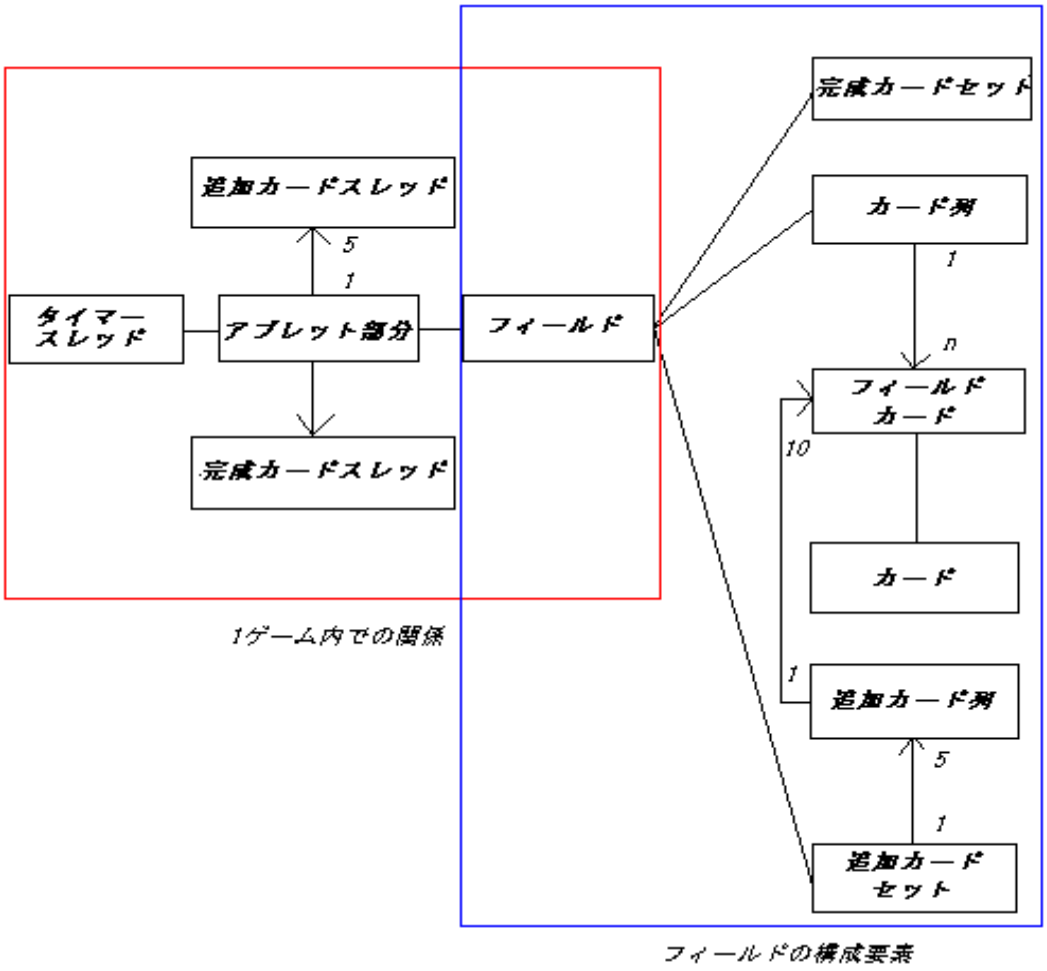

図 1　このプログラムのクラス図

1.2 クラス図の解説

1.2.1 アプレット部分

画面への出力、音の出力、スレッドの開始などプログラムの出力部分を記述する。このクラスがアプレットの実行単位である。この部分は SpiderSolitia.java に記述することにする。

1.2.2 タイマースレッド

ゲームをプレイ中、時間を測定する際に、必要な処理を記述する。この部分は TimerCount.java に記述することにする。

1.2.3 追加カードスレッド

このゲームでは、カードを各カード列に追加する際に、アニメーションをさせる。そのための処理を記述する。この部分は AddAnimation.java に記述することにする。

1.2.4 完成カードスレッド

このゲームでは、13〜1 まで揃えて完成カードセットに収納する際にアニメーションをさせる。そのための処理を記述する。この部分は CompleteAnimation.java に記述することにする。

1.2.5 フィールド

実際にゲームの処理を行う部分。そのための処理を Field.java に記述することにする。

1.2.6 完成カードセット

13〜1 まで揃えたカードセットの情報を処理・記憶するための部分。そのための処理を CompleteSet.java に記述する。

1.2.7 カード列

このゲームで使用するカード列 (１０列）を表す。CardRow.java で処理を記述することにする。

1.2.8 フィールドカード

このゲームで使用するカードに座標位置や表裏の判定を加え、使いやすくしたフィールドカードを表す。FieldCard.java で処理を記述することにする。

1.2.9 カード

このゲームで使用するカードを表す。Card.java で処理を記述することにする。

1.2.10 追加カード列

クリックした際、前述したカード列に追加するカードの列を表す。AddCardRow.java で処理を記述することにする。

1.2.11 追加カードセット

追加カード列があといくつあるかなどの情報を管理する追加カードセットを表す。AddCardSet.java で処理を記述することにする。

2 使用したクラスライブラリ

- java.applet.*;
  - java.applet.Applet;
  - java.applet.AudioClip;
- java.awt.*;
  - java.awt.Color;
  - java.awt.Dimension;
  - java.awt.Font;
  - java.awt.Graphics;
  - java.awt.Image;
  - java.awt.MediaTracker;
- java.awt.event.*;
  - java.awt.event.MouseEvent;
  - java.awt.event.MouseListener;
  - java.awt.event.MouseMotionListener;

3 ソースプログラム

3.1 SpiderSolitia.java

SpiderSolitia.java のソースその 1

```
001   import java.applet.Applet;
002   import java.applet.AudioClip;
003   import java.awt.Color;
004   import java.awt.Dimension;
005   import java.awt.Font;
006   import java.awt.Graphics;
007   import java.awt.Image;
008   import java.awt.MediaTracker;
009   import java.awt.event.MouseEvent;
010   import java.awt.event.MouseListener;
011   import java.awt.event.MouseMotionListener;
012   public class SpiderSolitia extends Applet
013         implements MouseListener,MouseMotionListener,Runnable{
014     private Dimension d;
015     private Image offscreen;
016     private Graphics grf;
017     private MediaTracker mt;
018     private Field field;
019     private Color bgcolor;
020     private Color areacolor;
021     private Color mojicolor;
022     private Image [] card = new Image[13];
023     private Image uracard;
024     private Image startbutton;
025     private Image restartbutton;
026     private AudioClip au1;
027     private Thread thread = null;
028     private Thread thread2 = null;
029     private Thread thread3 = null;
030     private Thread thread4 = null;
031     private boolean start;
032     private boolean push;
033     private int mouse_x;
034     private int mouse_y;
```

SpiderSolitia.java のソースその 2

```
035     private int xx[] = new int[10];
036     public SpiderSolitia(){
037       super();
038       addMouseListener(this);
039       addMouseMotionListener(this);
040       start = false;
041       push = false;
042     }
043     public void init(){
044       d = getSize();
045       offscreen = createImage(d.width,d.height);
046       grf = offscreen.getGraphics();
047       field = new Field(this);
048       mt = new MediaTracker(this);
049       for(int i = 1;i < 14;i++){
050         if(i >= 10){
051           card[i - 1] = getImage(getCodeBase(),i+".png");
052           mt.addImage(card[i - 1],0);
053         }else{
054           card[i - 1] = getImage(getCodeBase(),"0"+i+".png");
055           mt.addImage(card[i - 1],0);
056         }
057       }
058       uracard = getImage(getCodeBase(),"ura.png");
059       mt.addImage(uracard,0);
060       startbutton = getImage(getCodeBase(),"start_button.png");
061       mt.addImage(startbutton,0);
062       restartbutton = getImage(getCodeBase(),"restart_button.png");
063       mt.addImage(restartbutton,0);
064       au1 = getAudioClip(getDocumentBase(),"shu.wav");
065       bgcolor = new Color(208,255,208);
066       areacolor = new Color(55,255,55);
067       mojicolor = new Color(0,0,0);
068       for(int i = 0;i < 10;i++){
069         xx[i] = 15 + 55 * i;
070       }
071     }
072     public void paint(Graphics g){
073       update(g);
```

SpiderSolitia.java のソースその 3

```
074    }
075     public void update(Graphics g){
076       if(!mt.checkID(0)){
077         g.clearRect(0,0,d.width,d.height);
078         g.setColor(Color.black);
079         g.drawString("Please wait....",d.width / 4,d.height / 2);
080         return;
081       }
082       grf.setColor(bgcolor);
083       grf.fillRect(0,0,d.width,d.height);
084       grf.setColor(mojicolor);
085       grf.setFont(new Font("Monospaced",Font.BOLD,24));
086       grf.drawString("スパイダソリティア",180,30);
087       grf.setFont(new Font("Monospaced",Font.BOLD,14));
088       grf.drawString("タイム（秒）"+field.getTime(),430,40);
089       for(int i = 0;i < 10;i++){
090         grf.setColor(Color.RED);
091         grf.fillRect(xx[i],50,50,70);
092         grf.setColor(bgcolor);
093         grf.fillRect(xx[i]+2,52,46,66);
094         if(start){
095           CardRow cr = field.getCardRow(i);
096           for(int j = 0;j < cr.getSize();j++){
097             FieldCard fc = cr.getCard(j);
098             int x = fc.getIndex_x();
099             int y = fc.getIndex_y();
100            boolean front = fc.getFrontstate();
101            int su = fc.getCard().getNumber() - 1;
102            if(front){
103              grf.drawImage(card[su],x,y,this);
104            }else{
105              grf.drawImage(uracard,x,y,this);
106            }
107          }
108        }
109       }
110       if(push){
111         FieldCard [] fc = field.getMoveFieldCard();
112         int fc_cnt = field.getMoveFieldCardLength();
```

SpiderSolitia.java のソースその 4

```
113         for(int i = 0;i < fc_cnt;i++){
114           int x = fc[i].getIndex_x();
115           int y = fc[i].getIndex_y();
116           boolean front = fc[i].getFrontstate();
117           int su = fc[i].getCard().getNumber() - 1;
118           if(front){
119             grf.drawImage(card[su],x,y,this);
120           }else{
121             grf.drawImage(uracard,x,y,this);
122           }
123         }
124       }
125       if(!start){
126         grf.drawImage(startbutton,401,515,this);
127       }else{
128         grf.drawImage(restartbutton,401,515,this);
129       }
130       int compsu = field.getCs().getCompleteCount();
131       for(int i = 0;i < compsu;i++){
132         grf.drawImage(card[12],15+15*i,485,this);
133       }
134       if(compsu == 8){
135         grf.setColor(Color.RED);
136         grf.setFont(new Font("Monospaced",Font.BOLD,18));
137         grf.drawString("GAME  CLEAR",190,300);
138       }
139       grf.setColor(areacolor);
140       grf.fillRect(227,485,120,80);
141       grf.setColor(mojicolor);
142       grf.setFont(new Font("Monospaced",Font.BOLD,14));
143       grf.drawString("得点 : "+field.getTokuten(),227,520);
144       grf.drawString("回数 : "+field.getKaisu(),227,540);
145       AddCardSet acs = field.getAcs();
146       for(int i = 0;i < 5;i++){
147         if(!acs.getAddCardRow(i).getEnd()){
148           grf.drawImage(uracard,510,475+5*i,this);
149         }
150       }
151       g.drawImage(offscreen,0,0,this);
```

SpiderSolitia.java のソースその 5

```
152    }
153    public void start(){
154      if(thread == null){
155        thread = new Thread(this);
156        thread.start();
157      }
158    }
159    public void stop(){
160      if(thread != null){
161        thread.stop();
162        thread = null;
163      }
164    }
165    public void run(){
166      try{
167        mt.waitForID(0);
168      }catch(InterruptedException e){
169        return;
170      }
171      repaint();
172    }
173    public void mouseClicked(MouseEvent arg0) {
174      int ix = arg0.getX();
175      int iy = arg0.getY();
176      if(start){
177        if(thread3 == null && thread4 == null){
178          field.addCardSet(ix,iy);
179        }
180      }
181      if(ix >= 401 && ix < 456 && iy >= 516 && iy < 541){
182        start = true;
183        init();
184        timeThreadStart();
185        repaint();
186      }
187      repaint();
188    }
189    public void timeThreadStart() {
190      TimerCount tc = new TimerCount(this,field);
```

SpiderSolitia.java のソースその 6

```
191       thread2 = new Thread(tc);
192       thread2.start();
193     }
194     public void addThreadStart(FieldCard [] fc){
195       AddAnimation aa = new AddAnimation(this,field,fc);
196       thread3 = new Thread(aa);
197       thread3.start();
198     }
199     public void addThreadStop(){
200       thread3 = null;
201     }
202     public void completeThreadStart(int x,FieldCard [] fc){
203       CompleteAnimation ca = new CompleteAnimation(this,field,fc,x);
204       thread4 = new Thread(ca);
205       thread4.start();
206     }
207     public void completeThreadStop(){
208       thread4 = null;
209     }
210     public void mousePressed(MouseEvent arg0) {
211       if(start){
212         if(thread3 == null && thread4 == null){
213           int ix = arg0.getX();
214           int iy = arg0.getY();
215           field.movePrepare(ix,iy);
216         }
217       }
218     }
219     public int getXX(int n){
220       return xx[n];
221     }
222     public boolean getPush(){
223       return push;
224     }
225     public void setPush(boolean push){
226       this.push = push;
227     }
228     public void setMouse_x(int mouse_x){
229       this.mouse_x = mouse_x;
```

SpiderSolitia.java のソースその 7

```
230    }
231    public void setMouse_y(int mouse_y){
232      this.mouse_y = mouse_y;
233    }
234    public void mouseReleased(MouseEvent arg0) {
235      if(push){
236        int ix = arg0.getX();
237        int iy = arg0.getY();
238        boolean onsound = field.movemain(ix,iy);
239        if(onsound){
240          musicPlay();
241        }
242        repaint();
243      }
244    }
245    public void mouseEntered(MouseEvent arg0) {
246    }
247    public void mouseExited(MouseEvent arg0) {
248    }
249    public void mouseDragged(MouseEvent arg0) {
250      int ix = arg0.getX();
251      int iy = arg0.getY();
252      if(push){
253        field.moveIndex(ix,iy,mouse_x,mouse_y);
254        repaint();
255      }
256    }
257    public void mouseMoved(MouseEvent arg0) {
258    }
259    public void musicPlay(){
260      au1.play();
261    }
262  }
```

3.2 TimerCount.java

TimerCount.java のソース

```
001  public class TimerCount implements Runnable {
002    private SpiderSolitia st;
003    private Field field;
004    public TimerCount(SpiderSolitia st,Field field){
005      this.st = st;
006      this.field = field;
007    }
008    public void run() {
009      try{
010        while(!field.isComplete()){
011          Thread.sleep(1000);
012          field.addTime();
013          if(field.getTime() % 20 == 0){
014            field.setTokuten(-1);
015          }
016          st.repaint();
017        }
018      }catch(InterruptedException e){
019      }
020    }
021  }
```

3.3 AddAnimation.java

AddAnimation.java のソースその 1

```
001  public class AddAnimation implements Runnable{
002    private Field field;
003    private SpiderSolitia st;
004    private FieldCard [] fc;
005    private int[] dx;
006    private int[] dy;
007    public AddAnimation(SpiderSolitia st,Field field,FieldCard [] fc){
008      this.field = field;
009      this.st = st;
010      this.fc = fc;
011      dx = new int[10];
012      dy = new int[10];
013    }
014    public void run(){
015      int cnt = 0;
016      int dcnt = 1;
017      for(int i = 0;i < 10;i++){
018        dx[i] = 510 - st.getXX(i);
019        dy[i] = 475 - (50 + (field.getCardRow(i).getSize() * 12));
020      }
021      field.getCardRow(cnt).addCard(fc[cnt]);
022      st.musicPlay();
023      while(cnt <= 9){
024        try{
025          Thread.sleep(10);
026          int set_x = 510 - dx[cnt] * dcnt / 20;
027          int set_y = 475 - dy[cnt] * dcnt / 20;
028          fc[cnt].setFrontstate(true);
029          fc[cnt].setIndex_x(set_x);
030          fc[cnt].setIndex_y(set_y);
031          dcnt = dcnt + 1;
032          if(dcnt > 20){
033            cnt = cnt + 1;
034            if(cnt != 10){
```

AddAnimation.java のソースその 2

```
035             field.getCardRow(cnt).addCard(fc[cnt]);
036             st.musicPlay();
037           }
038            dcnt = 1;
039         }
040          st.repaint();
041       }catch(InterruptedException e){
042       }
043     }
044      st.addThreadStop();
045   }
046 }
```

3.4 CompleteAnimation.java

CompleteAnimation.java のソースその 1

```
001  public class CompleteAnimation implements Runnable{
002    private Field field;
003    private SpiderSolitia st;
004    private FieldCard [] fc;
005    private int[] dx;
006    private int[] dy;
007    private int x;
008    public CompleteAnimation(SpiderSolitia st,Field field,FieldCard [] fc,int x){
009      this.field = field;
010      this.st = st;
011      this.fc = fc;
012      this.x = x;
013      dx = new int[13];
014      dy = new int[13];
015    }
016    public void run(){
017      int cnt = 0;
018      int dcnt = 1;
019      for(int i = 0;i < 13;i++){
020        dx[i] = (15 + 15 * field.getCs().getCompleteCount()) - fc[i].getIndex_x();
021        dy[i] = 485 - fc[i].getIndex_y();
022      }
023      while(cnt <= 12){
024        try{
025          Thread.sleep(10);
026          int set_x = fc[cnt].getIndex_x() + dx[cnt] * dcnt / 20;
027          int set_y = fc[cnt].getIndex_y() + dy[cnt] * dcnt / 20;
028          fc[cnt].setFrontstate(true);
029          fc[cnt].setIndex_x(set_x);
030          fc[cnt].setIndex_y(set_y);
031          dcnt = dcnt + 1;
032          if(dcnt > 20){
033            field.getCardRow(x).removeCard();
034            if(cnt != 12){
```

CompleteAnimation.java のソースその 2

```
035             st.musicPlay();
036           }
037           cnt = cnt + 1;
038           dcnt = 1;
039         }
040         st.repaint();
041       }catch(InterruptedException e){
042       }
043     }
044     field.frontCheck(field.getCardRow(x));
045     st.completeThreadStop();
046     st.repaint();
047   }
048 }
```

3.5 Field.java

Field.java のソースその 1

```
001  public class Field {
002    private SpiderSolitia st;
003    private int tokuten;
004    private int kaisu;
005    private int time;
006    private CardRow [] cr;
007    private AddCardSet acs;
008    private Card [] card;
009    private int [] check = new int[104];
010    private CompleteSet cs;
011    private FieldCard [] fc;
012    private int [][] zahyou;
013    private int fc_cnt;
014    public Field(SpiderSolitia st){
015      this.st = st;
016      cr = new CardRow[10];
017      for(int i = 0;i < 10;i++){
018        cr[i] = new CardRow();
019      }
020      acs = new AddCardSet();
021      cs = new CompleteSet();
022      tokuten = 500;
023      kaisu = 0;
024      time = 0;
025      fc = new FieldCard[13];
026      zahyou = new int[13][2];
027      makeCard();
028      deliveryCard();
029    }
030    public void setTokuten(int ten){
031      tokuten = tokuten + ten;
032    }
033    public int getTokuten(){
034      return tokuten;
```

Field.java のソースその 2

```
035    }
036    public void addKaisu(){
037      kaisu = kaisu + 1;
038    }
039    public int getKaisu(){
040      return kaisu;
041    }
042    public void addTime(){
043      time = time + 1;
044    }
045    public int getTime(){
046      return time;
047    }
048    public CardRow getCardRow(int i){
049      return cr[i];
050    }
051    public AddCardSet getAcs(){
052      return acs;
053    }
054    public CompleteSet getCs(){
055      return cs;
056    }
057    public void makeCard(){
058      card = new Card[104];
059      for(int i = 0;i < 104;i++){
060        int su = i / 8 + 1;
061        card[i] = new Card(su,1,1);
062        check[i] = 0;
063      }
064    }
065    public void deliveryCard(){
066      for(int i = 0;i < 10;i++){
067        int su = 6;
068        if(i > 3){
069          su = 5;
070        }
071        for(int j = 0;j < su;j++){
072          int index = (int)(Math.random() * 104);
073          while(check[index] != 0){
```

Field.java のソースその 3

```
074           index = (int)(Math.random() * 104);
075         }
076         boolean front = false;
077         if(j == su - 1){
078           front = true;
079         }
080         cr[i].addCard(new FieldCard(card[index],15+i*55,50+j*12,front));
081       }
082     }
083     for(int i = 0;i < 5;i++){
084       for(int j = 0;j < 10;j++){
085         int index = (int)(Math.random() * 104);
086         while(check[index] != 0){
087           index = (int)(Math.random() * 104);
088         }
089         AddCardRow ar = acs.getAddCardRow(i);
090         ar.setCard(j,new FieldCard(card[index],510,475+5*i,false));
091       }
092     }
093   }
094   public int getAddlastcardsu() {
095     int count = 0;
096     for(int i = 0;i < 5;i++){
097       if(!acs.getAddCardRow(i).getEnd()){
098         count++;
099       }
100       return count;
101     }
102   }
103   public boolean isComplete(){
104     if(cs.getCompleteCount() == 8){
105       return true;
106     }
107     return false;
108   }
109   public void movePrepare(int ix,int iy){
110     fc_cnt = 0;
111     int x = getRow(ix,iy);
112     if(x != 99){
```

Field.java のソースその 4

```
113         int size = cr[x].getSize();
114         int y = (iy - 50) / 12;
115         if(y >= size - 1){
116           if(iy < 120+12*(size - 1)){
117             y = size - 1;
118           }else{
119             y = size;
120           }
121         }
122         if(y < size){
123           if(cr[x].getCard(y).getFrontstate()){
124             int baseNumber = cr[x].getCard(y).getCard().getNumber();
125             boolean isMove = true;
126             int yy = y + 1;
127             for(int i = yy;i < size;i++){
128               int number = cr[x].getCard(i).getCard().getNumber();
129               if(baseNumber == number + 1){
130                 baseNumber = number;
131               }else{
132                 isMove = false;
133               }
134             }
135             if(isMove){
136               st.setPush(true);
137               for(int i = y;i < size;i++){
138                 fc[fc_cnt] = cr[x].getCard(i);
139                 zahyou[fc_cnt][0] = x;
140                 zahyou[fc_cnt][1] = i;
141                 fc_cnt++;
142               }
143               st.setMouse_x(ix);
144               st.setMouse_y(iy);
145             }
146           }
147         }
148       }
149     }
150     private int getRow(int ix,int iy){
151       for(int i = 0;i < 10;i++){
```

Field.java のソースその 5

```
152         if(ix >= st.getXX(i) && ix < st.getXX(i) + 50){
153           return i;
154         }
155       }
156       return 99;
157     }
158     public boolean movemain(int ix,int iy){
159       boolean isAble = false;
160       int x = getRow(ix,iy);
161       if(x != 99){
162         int maisu = cr[x].getSize();
163         int from = 50 + (maisu - 1) * 12;
164         int to = from + 70;
165         if(fc[0].getIndex_y() >= from) && fc[0].getIndex_y() < to){
166           int bnumber = -1;
167           if(maisu > 0){
168             bnumber = cr[x].getCard(maisu - 1).getCard().getNumber();
169           }
170           int number = fc[0].getCard().getNumber();
171           if(maisu == 0 || bnumber == number + 1){
172             isAble = true;
173             for(int i = 0;i < fc_cnt;i++){
174               maisu = cr[x].getSize();
175               fc[i].setIndex_x(st.getXX(x));
176               fc[i].setIndex_y(50 + maisu * 12);
177               cr[x].addCard(fc[i]);
178               cr[zahyou[i][0]].removeCard();
179             }
180             frontCheck(cr[zahyou[0][0]]);
181             setTokuten(-1);
182             addKaisu();
183             if(completeCheck(cr[x])){
184               FieldCard [] fcard = movecardSet(x);
185               st.completeThreadStart(x,fcard);
186               cs.addCompleteCount();
187               setTokuten(100);
188               return true;
189             }
190           }else{
```

Field.java のソースその 6

```
191           setReverse();
192         }
193       }else{
194         setReverse();
195       }
196     }else{
197       setReverse();
198     }
199     st.setPush(false);
200     return isAble;
201   }
202   public void setReverse(){
203     for(int i = 0;i < fc_cnt;i++){
204       cr[zahyou[0][0]].removeCard();
205     }
206     for(int i = 0;i < fc_cnt;i++){
207       int maisu = cr[zahyou[0][0]].getSize();
208       fc[i].setIndex_x(st.getXX(zahyou[0][0]));
209       fc[i].setIndex_y(50 + maisu * 12);
210       cr[zahyou[0][0]].addCard(fc[i]);
211     }
212   }
213   public FieldCard [] movecardSet(int x){
214     int maisu = cr[x].getSize();
215     FieldCard [] fcard = new FieldCard[13];
216     for(int i = 0;i < 13;i++){
217       fcard[i] = cr[x].getCard(maisu - (i + 1));
218     }
219     return fcard;
220   }
221   public void moveIndex(int ix,int iy,int mouse_x,int mouse_y){
222     for(int i = 0;i < fc_cnt;i++){
223       int idx = fc[i].getIndex_x();
224       int idy = fc[i].getIndex_y();
225       fc[i].setIndex_x(idx+ix-mouse_x);
226       fc[i].setIndex_y(idy+iy-mouse_y);
227     }
228     st.setMouse_x(ix);
229     st.setMouse_y(iy);
```

Field.java のソースその 7

```
230    }
231    public void addCardSet(int ix,int iy) {
232      for(int i = 0;i < 10;i++){
233        if(cr[i].getSize() == 0){
234          return;
235        }
236      }
237      int add_su = getAddlastcardsu();
238      int yf = 475 + 5 * add_su;
239      int yt = 545 + 5 * add_su;
240      if(add_su >0 && ix >= 510 && ix < 560 && iy >= yf && iy < yt){
241        FieldCard [] fc = acs.getAddCardRow(add_su - 1).getCard();
242        st.addThreadStart(fc);
243        acs.getAddCardRow(add_su - 1).setEnd(true);
244      }
245    }
246    public FieldCard[] getMoveFieldCard() {
247      return fc;
248    }
249    public int getMoveFieldCardLength(){
250      return fc_cnt;
251    }
252    public boolean completeCheck(CardRow cr){
253      boolean flg = false;
254      for(int i = 0;i < cr.getSize();i++){
255        if(cr.getCard(i).getFrontstate()){
256          if(cr.getCard(i).getCard().getNumber() == 13){
257            if(completesubCheck(cr,i)){
258              return true;
259            }
260          }
261        }
262      }
263      return false;
264    }
265    public boolean completesubCheck(CardRow cr,int i){
266      int base_number = 12;
267      for(int j = i + 1;j < cr.getSize();j++){
268        if(base_number == cr.getCard(j).getCard().getNumber()){
```

Field.java のソースその 8

```
269          base_number--;
270          if(base_number == 0){
271            return true;
272          }
273        }else{
274          return false;
275        }
276      }
277      return false;
278    }
279    public void frontCheck(CardRow cr){
280      int size = cr.getSize();
281      if(size > 0){
282        if(!cr.getCard(size-1).getFrontstate()){
283          cr.getCard(size-1).setFrontstate(true);
284        }
285      }
286    }
287  }
```

3.6 CompleteSet.java

CompleteSet.java のソース

```
001  public class CompleteSet {
002    private int clearCount;
003    private int completeCount;
004    public CompleteSet(){
005      clearCount = 8;
006      completeCount = 0;
007    }
008    public int getClearCount(){
009      return clearCount;
010    }
011    public int getCompleteCount(){
012      return completeCount;
013    }
014    public void addCompleteCount(){
015      completeCount++;
016    }
017  }
```

3.7 CardRow.java

CardRow.java のソース

```
001  import java.util.ArrayList;
002  public class CardRow {
003    private ArrayList array;
004    public CardRow(){
005      array = new ArrayList();
006    }
007    public void addCard(FieldCard fc){
008      array.add(fc);
009    }
010    public void removeCard(){
011      array.remove(array.size() - 1);
012    }
013    public int getSize(){
014      return array.size();
015    }
016    public FieldCard getCard(int n){
017      return (FieldCard)(array.get(n));
018    }
019  }
```

3.8 FieldCard.java

FieldCard.java のソース

```
001  public class FieldCard {
002    private Card card;
003    private int index_x;
004    private int index_y;
005    private boolean frontstate;
006    public FieldCard(Card card,int index_x,int index_y,boolean frontstate){
007      this.card = card;
008      this.index_x = index_x;
009      this.index_y = index_y;
010      this.frontstate = frontstate;
011    }
012    public Card getCard(){
013      return card;
014    }
015    public void setCard(Card card){
016      this.card = card;
017    }
018    public int getIndex_x(){
019      return index_x;
020    }
021    public void setIndex_x(int index_x){
022      this.index_x = index_x;
023    }
024    public int getIndex_y(){
025      return index_y;
026    }
027    public void setIndex_y(int index_y){
028      this.index_y = index_y;
029    }
030    public boolean getFrontstate(){
031      return frontstate;
032    }
033    public void setFrontstate(boolean frontstate){
034      this.frontstate = frontstate;
035    }
036  }
```

3.9 Card.java

Card.java のソース

```
001  public class Card {
002    private int number;
003    private int mark;
004    private int color;
005    public Card(int number,int mark,int color){
006      this.number = number;
007      this.mark = mark;
008      this.color = color;
009    }
010    public int getNumber(){
011      return number;
012    }
013    public int getMark(){
014      return mark;
015    }
016    public int getColor(){
017      return color;
018    }
019  }
```

3.10 AddCardRow.java

AddCardRow.java のソース

```
001  public class AddCardRow {
002    private FieldCard[] card;
003    private boolean end;
004    public AddCardRow(){
005      card = new FieldCard[10];
006      end = false;
007    }
008    public void setCard(int r,FieldCard ca){
009      card[r] = ca;
010    }
011    public FieldCard [] getCard(){
012      return card;
013    }
014    public boolean getEnd(){
015      return end;
016    }
017    public void setEnd(boolean end){
018      this.end = end;
019    }
020  }
```

3.11 AddCardSet.java

AddCardSet.java のソース

```
001  public class AddCardSet {
002    private AddCardRow acr[];
003    public AddCardSet(){
004      acr = new AddCardRow[5];
005      for(int i = 0;i < 5;i++){
006        acr[i] = new AddCardRow();
007      }
008    }
009    public void setCard(int c,int r,FieldCard ca){
010      acr[c].setCard(r,ca);
011    }
012    public AddCardRow getAddCardRow(int n){
013      return acr[n];
014    }
015  }
```

4 ソースプログラムの解説

4.1 SpiderSolitia.java

4.1.1 フィールド

- d・・・画面変数 (Dimension)
- offscreen・・・ダブルバッファリング用のイメージ変数 (Image)
- grf・・・ダブルバッファリング用のグラフィック変数 (Graphics)
- mt・・・画像読み込み用の MediaTracker 変数 (MediaTracker)
- field・・・ゲーム用のフィールド変数 (Field)
- bgcolor・・・背景描画用のカラー変数 (Color)
- areacolor・・・スコア・移動回数エリア描画用のカラー変数 (Color)
- mojicolor・・・文字描画用カラー変数 (Color)
- card[ ]・・・カード画像（表）格納用のイメージ変数 (Image[13])
- uracard・・・カード画像（裏）格納用のイメージ変数 (Image)
- startbutton・・・スタートボタン画像格納用のイメージ変数 (Image)
- restartbutton・・・リスタートボタン画像格納用のイメージ変数 (Image)
- au1・・・カード移動時の音格納用のオーディオクリップ変数 (AudioClip)
- thread・・・画像読み込み時に使用するスレッド変数 (Thread)
- thread2・・・タイマー動作時に使用するスレッド変数 (Thread)
- thread3・・・カード追加する時のアニメーション動作時に使用するスレッド変数 (Thread)
- thread4・・・13 から 1 までそろったときに完成カード列に移動させるアニメーション時に使用するスレッド変数 (Thread)
- start・・・ゲームがスタートしているかを示す true/false 変数 (boolean)
- push・・・マウスが押されているかを示す true/false 変数 (boolean)
- mouse_x・・・マウスの x 座標を格納する変数 (int)
- mouse_y・・・マウスの y 座標を格納する変数 (int)
- xx[ ]・・・カード列の左上端の x 座標を記憶しておく変数 (int[10])

4.1.2 メソッド

- init()
  ゲームをスタートするときに最初に行う処理メソッド
- paint(Graphics)
  仮の描画処理メソッド
- updata(Graphics)
  実際の描画処理メソッド
- start()
  スレッドをスタートさせるための処理メソッド
- stop()

スレッドをストップさせるための処理メソッド

- run()
  画像の読み込みを行うためのスレッドの処理メソッド
- mouseClicked(MouseEvent)
  マウスがクリックされたときの処理メソッド
- mousePressed(MouseEvent)
  マウスボタンが押されたときの処理メソッド
- mouseReleased(MouseEvent)
  マウスボタンが離されたときの処理メソッド
- mouseMoved(MouseEvent)
  マウスカーソルが動いたときの処理メソッド
- mouseDragged(MouseEvent)
  マウスをドラッグしたときの処理メソッド
- mouseExited(MouseEvent)
  マウスカーソルが画面外に出たときの処理メソッド
- mouseEntered(MouseEvent)
  マウスカーソルが画面内に入ってきたときの処理メソッド
- timeThreadStart()
  タイマー動作時に動作するスレッドをスタートさせる処理メソッド
- addThreadStart(FieldCard [ ])
  カード追加時のアニメーションを動作させるために使用するスレッドをスタートさせるメソッド
- addThreadStop()
  カード追加時のアニメーションを動作させるために使用するスレッドをストップさせるメソッド
- completeThreadStart(int,FieldCard [ ])
  完成エリアにカードを収納するアニメーションを動作させるために使用するスレッドをスタートさせるメソッド
- completeThreadStop()
  完成エリアにカードを収納するアニメーションを動作させるために使用するスレッドをストップさせるメソッド
- getPush()
  マウスがクリックされているか状態を返すメソッド
- setPush(boolean)
  マウスがクリックされているか状態をセットするメソッド
- musicPlay()
  音を流す処理を記述するメソッド
- setMouse_x(int)
  マウスの x 座標をセットするメソッド
- setMouse_y(int)
  マウスの y 座標をセットするメソッド
- getXX(int)

該当する列の左上の x 座標を取得するメソッド

4.1.3 コンストラクタ

- アプレット生成時にスーパークラスで行う処理を行う。
- マウスリスナーを使えるようにする。
- ゲームの動作状態を非動作状態にセットする。
- マウスボタンが押されているかという状態を非プッシュ状態にセットする。

4.2 TimerCount.java

4.2.1 フィールド

- st・・・SpiderSolitia のメソッドにアクセスするための変数
- field・・・Field のメソッドにアクセスするための変数

4.2.2 メソッド

- run()
  TimerCount の実際の動作部分を記述したもの。ゲーム終了までの時間を測定する。20 秒ごとに点数を１点減らす。そして画面を再描画する。

4.2.3 コンストラクタ

- 生成されたときに、現在使用している SpiderSolitia と Field のインスタンスをセットする。

4.3 AddAnimation.java

4.3.1 フィールド

- field・・・Field のメソッドにアクセスするための変数。
- st・・・SpiderSolitia のメソッドにアクセスするための変数。
- fc[ ]・・・追加するフィールドカードを格納するための配列。
- dx[ ]・・・アニメーションの際に座標位置を計算するために使用する x 座標の配列。
- dy[ ]・・・アニメーションの際に座標位置を計算するために使用する y 座標の配列。

4.3.2 メソッド

- run()
  カードが追加される際に見えるアニメーションを実現するメソッド。１０ミリ秒ごとに最終目的のカードの位置と最初のカードの位置の間の距離を２０等分した値の分だけ動かし、画面を再描画する。１枚が終わったら次の１枚といった感じで１０枚分移動させる。

4.3.3 コンストラクタ

- 引数として指定されたインスタンス (SpiderSolitia,Field,FieldCard) をセット。
- dx,dy 配列の領域を確保する。

4.4 CompleteAnimation.java

4.4.1 フィールド

- field・・・Field のメソッドにアクセスするための変数。
- st・・・SpiderSolitia のメソッドにアクセスするための変数。
- fc[ ]・・・追加するフィールドカードを格納するための配列。
- dx[ ]・・・アニメーションの際に座標位置を計算するために使用する x 座標の配列。
- dy[ ]・・・アニメーションの際に座標位置を計算するために使用する y 座標の配列。
- x・・・カードを削除する列番号を格納する変数。

4.4.2 メソッド

- run()
  カードが左下の完成カード列に移動するアニメーションを実現するメソッド。１０ミリ秒ごとに最終目的のカードの位置と最初のカードの位置の間の距離を２０等分した値の分だけ動かし、画面を再描画する。１枚が終わったら次の１枚といった感じで１３枚分移動させる。

4.4.3 コンストラクタ

- 引数として指定されたインスタンス (SpiderSolitia,Field,FieldCard) をセット。
- 引数として指定された列番号 (x) をフィールド x にセット。
- dx,dy 配列の領域を確保する。

4.5 Field.java

4.5.1 フィールド

- st・・・SpiderSolitia のメソッドにアクセスするための変数。
- tokuten・・・ゲームの得点を表わす変数
- kaisu・・・ゲームの移動回数を表わす変数
- time・・・ゲームの経過時間を表す変数
- cr[]・・・ゲームで使用する CardRow のインスタンスを格納する配列
- acs・・・ゲームで使用する AddCardSet のインスタンスを格納する変数
- card[]・・・ゲームで使用するカードを作成し、格納するためのカード配列
- check[]・・・card[] と対になっていて、該当するカードが既にカード列のカードとして使われたかどうかを格納する配列
- cs・・・ゲームで使用する CompleteSet のインスタンスを格納する変数。

- fc[]・・・ゲームで一時的に使用する FieldCard のインスタンスを格納する配列
- zahyou[][]・・・カードを移動させる際に、移動させるカードの列番号とカード番号を格納するために使う配列
- fc_cnt・・・移動させるカードの枚数を記憶するための変数。

4.5.2 メソッド

- addKaisu()
  移動回数を加算するメソッド。
- getKaisu()
  移動回数を取得する（返す）メソッド。
- addTime()
  ゲームの経過時間を加算するメソッド。
- getTime()
  ゲームの経過時間を取得する（返す）メソッド。
- getCardRow(int)
  指定された列番号の CardRow インスタンスを取得（返す）メソッド。
- getAcs()
  ゲームで使用する AddCardSet インスタンスを取得（返す）メソッド。
- getCs()
  ゲームで使用する CompleteSet インスタンスを取得（返す）メソッド。
- makeCard()
  このゲームで使用する Card インスタンスを生成し、card 配列に格納するメソッド。
- deliveryCard()
  makeCard() で生成したカードを各カード列と追加カード列に振り分けるメソッド。
- getAddlastcardsu()
  ゲームで残りの追加カード列の数を検索・取得するメソッド。
- isComplete()
  ゲームが終了したかを判定するメソッド。
- movePrepare(int,int)
  カードを動かすための準備をするメソッド。
- getRow(int,int)
  指定された座標によって、処理すべきカード列の列番号を返すメソッド。
- movemain(int,int)
  実際にカードを動かす処理を行うメソッド。
- setReverse()
  カードが動かせなかったとき、動かしたカードの座標を動かす前の値に戻すメソッド。
- movecardSet(int)
  引数により指定された列番号の列の完成したカード列 (13〜1) を配列に格納し、返すメソッド。
- moveIndex(int,int,int,int)

引数により指定された情報をもとに、現在動かしているカードの座標を適切な値にセットするメソッド。

- addCardSet(int,int)
  追加カード列をクリックしたとき、カードを各カード列に配布する処理を行うメソッド。
- getMoveFieldCard()
  現在、ドラッグして動かしている FieldCard が格納されている配列を返すメソッド。
- getMoveFieldCardLength()
  現在、上記の配列に格納されている FieldCard の枚数を返すメソッド。
- completeCheck(CardRow)
  引数により指定されたカード列の中で、完成カード列 (13〜1 までそろった列) が存在するかを返すメソッド。
- completesubCheck(CardRow)
  上記のメソッドの詳細処理を行っているメソッド。
- frontCheck(CardRow)
  移動させたなどして、引数により指定されたカード列内に表になっているカードが１枚もないときに、一番手前にあるカードを表にするメソッド。

4.5.3 コンストラクタ

- 引数として指定された SpiderSolitia のインスタンスをセットする。
- カード列配列の領域を確保する（１０個）
- 上記で確保した領域について、それぞれ CardRow インスタンスを生成し、格納する。
- AddCardSet インスタンスを生成し、セットする。
- CompleteSet インスタンスを生成し、セットする。
- 得点を初期値 (500) にセットする。
- 回数を初期値 (0) にセットする。
- 経過時間を初期値 (0) にセットする。
- FieldCard 配列の領域を確保する (１３個)
- zahyou 配列の領域を確保する (13 個,2 個)
- Card インスタンスを生成し、card 配列に格納する処理を実行する。
- 各カード列と追加カード列に振り分ける処理を実行する。

4.6 CompleteSet.java

4.6.1 フィールド

- clearCount・・・クリアするための完成カードセットの数を格納する。
- completeCount・・現在完成しているカードセットの数を記憶する。

4.6.2 メソッド

- getClearCount()
  クリアするための完成カードセットの数を返すメソッド。

- getCompleteCount()
  現在完成しているカードセットの数を返すメソッド。
- addCompleteCount()
  クリアするための完成カードセットの数を加算する。

4.6.3 コンストラクタ

- クリアするための完成カードセットの数の値をセットする。
- 現在完成しているカードセットの数を初期化 (0 に) する。

4.7 CardRow.java

4.7.1 フィールド

- array・・・カード列に存在するフィールドカードを格納する配列。

4.7.2 メソッド

- addCard(FieldCard)
  引数で指定されたフィールドカードをカード列に追加する。
- removeCard()
  カード列に存在するカード列で一番上のカードを削除する。
- getSize()
  カード列に存在するカードの枚数を返す。
- getCard(int)
  引数で指定した番号のカードを返す。

4.7.3 コンストラクタ

- 新しいカード列 (ArrayList) を生成する。

4.8 FieldCard.java

4.8.1 フィールド

- card・・・カードの数字やマークを記憶する Card クラスのインスタンスを記憶する。
- index_x・・・表示するカードの x 座標を記憶する変数。
- index_y・・・表示するカードの y 座標を記憶する変数。
- frontstate・・・カードが表か裏かを判定する変数。

4.8.2 メソッド

- getCard()
  自身が記憶している Card クラスのインスタンスを返すメソッド。

- setCard(Card)
  引数として与えられた Card クラスのインスタンスを自身の Card クラスのインスタンスとしてセットするメソッド。
- getIndex_x()
  表示するカードの x 座標を返すメソッド。
- setIndex_x(int)
  引数として指定された x 座標を表示するカードの x 座標としてセットするメソッド。
- getIndex_y()
  表示するカードの y 座標を返すメソッド。
- setIndex_y(int)
  引数として指定された y 座標を表示するカードの y 座標としてセットするメソッド。
- getFrontState()
  カードが表か裏かを判定する変数の値を返すメソッド。
- setFrontState(boolean)
  引数として指定された値をカードが表か裏かを判定する変数の値としてセットするメソッド。

4.8.3 コンストラクタ

- 引数として指定された Card インスタンスをセットする。
- 引数として指定された x 座標、y 座標をセットする。
- 引数として指定された表裏の判定の値をセットする。

4.9 Card.java

4.9.1 フィールド

- number・・・カードの数字を記憶する変数。
- mark・・・カードのマーク (ダイヤ・ハート・クローバー・スペード) を記憶する変数。
- color・・・カードの色 (ダイヤ・ハート=赤、クローバー・スペード=黒) を記憶する変数

4.9.2 メソッド

- getNumber()
  カードの数字を返すメソッド。
- getMark()
  カードのマークを返すメソッド。
- getColor()
  カードの色を返すメソッド。

4.9.3 コンストラクタ

- 引数として指定された数字をカードの数字としてセットする。

- 引数として指定されたマークをカードのマークとしてセットする。
- 引数として指定された色をカードの色としてセットする。

4.10 AddCardRow.java

4.10.1 フィールド

- card[]・・・追加するカード列の 1 セットの中にある 10 枚のフィールドカードを記憶する配列。
- end・・・すでに追加された（クリックされた）かを記憶する変数。

4.10.2 メソッド

- setCard(int,FieldCard)
  フィールドカードを記憶する配列の引数で指定された場所に引数で指定された FieldCard を格納するメソッド。
- getCard()
  フィールドカードを記憶する配列を返すメソッド。
- getEnd()
  すでに追加されたかを記憶する変数の値を返すメソッド。
- setEnd(boolean)
  引数として指定された値をすでに追加されたかを記憶する変数の値としてセットするメソッド。

4.10.3 コンストラクタ

- 配列の領域を 10 個確保する。
- 最初はまだ追加されていない状態なので、すでに追加されたかを記憶する変数の値を false にセットする。

4.11 AddCardSet.java

4.11.1 フィールド

- acr[]・・・追加カードセットに存在する追加カード列（5 個）を記憶する配列。

4.11.2 メソッド

- setCard(int,int,FieldCard)
  第 1 引数として指定された番地に格納された追加カード列の第 2 引数として指定された番地の場所にフィールドカードを格納するメソッド。
- getAddCardRow(int)
  引数として指定された番地に格納された追加カード列を返すメソッド。

4.11.3 コンストラクタ

- 配列の領域を 5 個確保する。
- 確保したそれぞれの配列の要素について、AddCardRow のインスタンスを生成し、格納する。