ソフトウェア第三
— Java アプレット，迷路課題 —

機械情報工学科　水内郁夫

http://www.jsk.t.u-tokyo.ac.jp/~ikuo/lec/soft3/

(ID:soft3, pass:MechanoI)

2008 年 12 月 22 日 (月)

1　Java アプレットのイベント処理

GUI において，ボタンを押した場合には，そのイベントを処理する手続きを登録する必要がある．そのようなイベント処理には，java.util パッケージの EventListner インタフェース，EventListner のサブインタフェースである java.awt.event パッケージの ActionListerner，ItemListener，AdjustmentListener インタフェースなどを実装する形をとる．

1.1　ボタンなどのイベント処理 ActionListener

次の例 (ButtonView) は，ボタンを表示し押されると Pressed となる例である．図 1 は，ButtonView の実行画面である．

```
1   /*<applet code='ButtonView'
2     width='150' height='100'>
3     </applet>*/
4   import java.applet.*;
5   import java.awt.*;
6   import java.awt.event.*;
7
8   public class ButtonView
9     extends Applet implements ActionListener{
10    Button button;
11    public void init() {
12      button = new Button("Button");
13      button.addActionListener(this);
14      add(button);
15    }
16
17    public void
18      actionPerformed(ActionEvent event) {
19      button.setLabel("Pressed");
20    }
21  }
```

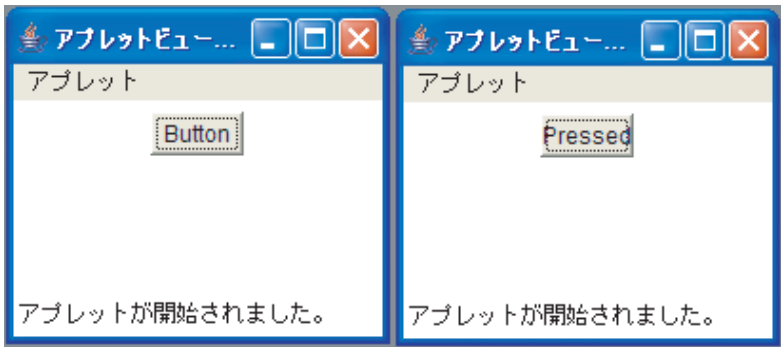


図 1: ButtonView の実行画面

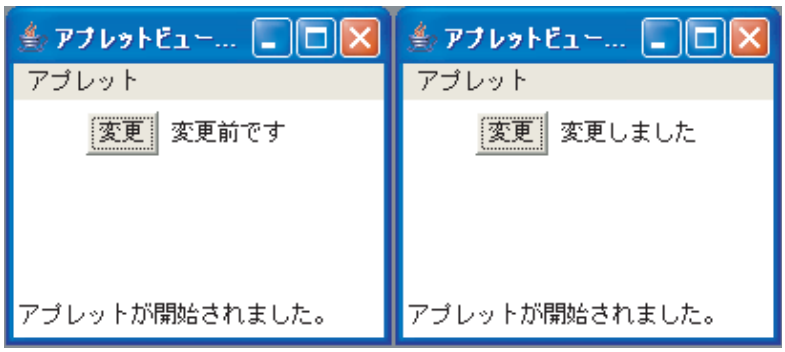


図 2: LabelView の実行画面

次の例 (LabelView) は，ボタンを押すと**ラベル**表示を行う例を示す．図 2 は，LabelView の実行の様子である．

```
1   /*<applet code='LabelView'
2     width='150' height='100'>
3     </applet>*/
4   import java.applet.*;
5   import java.awt.*;
6   import java.awt.event.*;
7
8   public class LabelView
9       extends Applet
10      implements ActionListener{
11      Button button;
12      Label label;
13      public void init() {
14          button = new Button("変更");
15          button.addActionListener(this);
16          add(button);
17          label = new Label("変更前です");
18          add(label);
19      }
20
21      public void
22      actionPerformed(ActionEvent event) {
23          label.setText("変更しました");
24      }
25  }
```

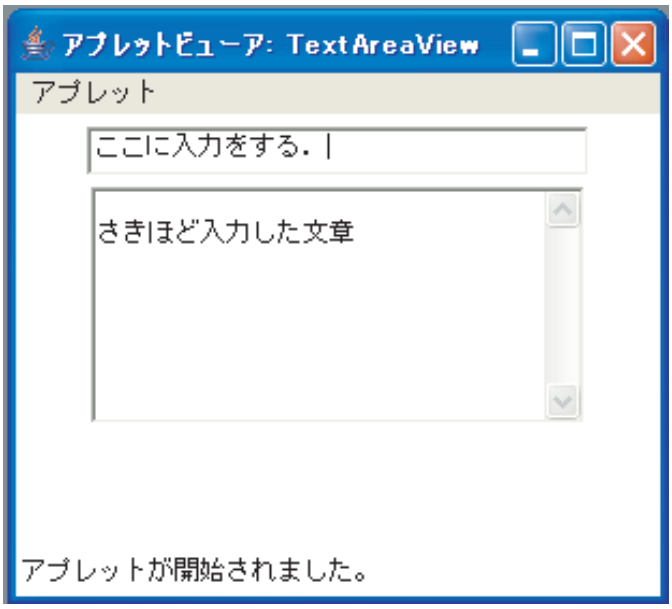


図 3: TextArea の実行画面

次の例 (TextAreaView) は，一行の文字列を入力する**テキストフィールド (TextField)** と，複数行のテキストを入力できる**テキストエリア (TextArea)** の例を示す．図 3 は TextAreaView の実行の様子である．

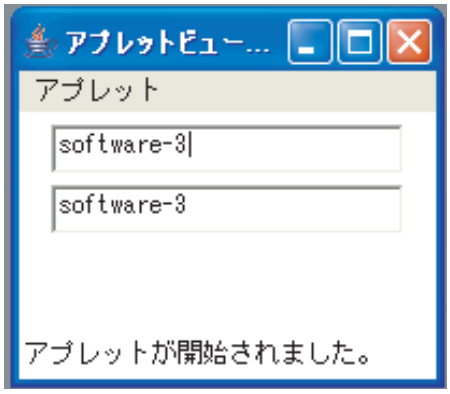
```
1   /*<applet code='TextAreaView'
2     width='300' height='200'>
3     </applet>*/
4   import java.applet.*;
5   import java.awt.*;
6   import java.awt.event.*;
7
8   public class TextAreaView
9       extends Applet
10      implements ActionListener{
11      TextField field;
12      TextArea area;
13      public void init() {
14          field = new TextField(30);
15          area = new TextArea(6, 30);
16          field.addActionListener(this);
17          add(field);
18          add(area);
19      }
20
21      public void
22      actionPerformed(ActionEvent event) {
23          area.append(field.getText() + "\n");
24          field.setText("");
25      }
26  }
```

1.2 チェックボックスなどの処理 ItemListener

```
/*<applet code='CheckboxView'
  width='150' height='100'>
  </applet>*/
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class CheckboxView
    extends Applet implements ItemListener{
    Checkbox checkbox;
    public void init() {
        checkbox = new Checkbox("表示");
        checkbox.addItemListener(this);
        add(checkbox);
    }

    public void
    itemStateChanged(ItemEvent event) {
        if ( checkbox.getState() ) {
            setBackground(Color.black);
            checkbox.setBackground(Color.black);
            checkbox.setForeground(Color.white);
        } else {
            setBackground(Color.white);
            checkbox.setBackground(Color.white);
            checkbox.setForeground(Color.black);
        }
    }
}
```

1.3 テキストフィールドの処理 TextListener

```
/*<applet code='TextFieldView'
  width='150' height='100'>
  </applet>*/
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class TextFieldView
    extends Applet
    implements TextListener{
  TextField field,  field2;
  public void init() {
    field = new TextField(20);
    field2 = new TextField(20);
    field.addTextListener(this);
    add(field);
    add(field2);
  }

  public void
  textValueChanged(TextEvent event) {
    field2.setText(field.getText());
  }
}
```



2 Mouseイベント の処理

マウスの入力を受け取る applet の例を示す. 以下のように, MouseTest クラスを, MouseListener, MouseMotionListner インタフェースを実装する形で Applet クラスのサブクラスとして定義する. MotionListner インタフェースを持てば mouse が applet 内に入った場合, 抜けた場合, ボタンが押された場合, ボタンが離れた場合, 押してすぐ話された場合のそれぞれで, mouseEntered, mouseExited, mousePressed, mouseReleased, mouseClicked というメソッドが呼び出されることになるので, ユーザがそのメソッドのうち必要なものだけ MouseTest クラスに定義をする.

MouseMotionListener インタフェースをもてば, マウスが押されて引きづられた場合, マウスが動いている場合のそれぞれで mouseDragged, mouseMoved が呼び出される.

2.1 MouseTest.html

```
1   <html>
2     <head>
3       <title>MouseTest</title>
4     </head>
5   <body>
6   <hr>
7   <applet code=MouseTest.class
8      width=500 height=500>
9     alt=" &lt;APPLET&gt; "
10    </applet>
11  <hr>
12  <a href="MouseTest.java">The source</a>.
13  </body>
14  </html>
```

HTML ファイルを開く際に, セキュリティに関する警告が出る場合は許可をする.

2.2 MouseTest.java

```
1   /**
2    * MouseTest.java
3    */
4
5   import java.awt.event.MouseEvent;
6   import java.awt.event.MouseListener;
7   import java.awt.event.MouseMotionListener;
8   import java.awt.Graphics;
9   import java.lang.Math;
10
11  public class MouseTest
12      extends java.applet.Applet
13      implements MouseListener,
14                 MouseMotionListener {
15
16      int mx, my;
17
18      public void init() {
19          setSize(500, 500);
20          addMouseListener(this);
21          addMouseMotionListener(this);
22      }
23
24      public void destroy() {
```

```
25          removeMouseListener(this);
26          removeMouseMotionListener(this);
27      }
28
29      public void paint(Graphics g) {
30          g.drawRect(0, 0,
31                     getSize().width - 1,
32                     getSize().height - 1);
33          g.drawRect(mx, my,
34                     (getSize().width/10) - 1,
35                     (getSize().height/10) - 1);
36          g.drawString(toString(),20,20);
37          g.drawString(paramString(),20,40);
38      }
39
40      /*
41       * MouseMotionListner methods
42       */
43      public void mouseDragged(MouseEvent e) {
44          mx = e.getX();
45          my = e.getY();
46          repaint();
47      }
48
49      public void mouseMoved(MouseEvent e) {
50          mx = (int)(Math.random()*1000) %
51              (getSize().width -
52                  (getSize().width/10));
53          my = (int)(Math.random()*1000) %
54              (getSize().height -
55                  (getSize().height/10));
56          repaint();
57      }
58
59      /*
60       * MouseListner methods
61       */
62      public void mouseClicked(MouseEvent e) {
63          AudioClip ac=Applet.newAudioClip(getClass().getResource("tada.wav"));
64          ac.play();
65      }
66      public void mouseEntered(MouseEvent e) {
67          mx = e.getX();
68          my = e.getY();
69          repaint();
70      }
71      public void mouseExited(MouseEvent e) {
72          mx = e.getX();
73          my = e.getY();
74          repaint();
75      }
76      public void mousePressed(MouseEvent e) {
77          mx = 0;
78          my = 0;
79          repaint();
80      }
81
82      public void mouseReleased(MouseEvent e) {
83          mx = e.getX();
84          my = e.getY();
85          repaint();
86      }
87
88  }
```

3 課題

課題は，ikuo-soft3-08@jsk.t.u-tokyo.ac.jp 宛てに提出してください．メールのサブジェクトは，「`soft3-kadai:20081222:xxxxx`」(xxxxx は学生証番号) とする．締切は来年 1 月 13 日 (火) とする (多少の遅れは許容)．その他，質問・要望・意見などがあれば，遠慮なく ikuo@jsk.t.u-tokyo.ac.jp にメールをどうぞ．

3.1 課題 1

図 4 に示されるような 2 次元の迷路の地図を表現するクラス等のデータ構造を定義し，ある迷路のデータがある時，その中を動き回るプログラムを作成せよ．プログラミング言語は問わない．具体的には，以下のような機能が考えられるだろう．

- 2 次元の格子状の空間の地図を記憶する機能．マスの間の状態は，壁有り，壁無し，不明，である．記憶形式 (データ構造) を検討しよう．4 つのリンクを持つノードのクラスを定義する?
- 特定の場所の壁の状態 (壁有り，壁無し，不明) をセットする機能．
- 特定の場所の壁の状態を調べる機能．
- 2 次元の地図を表示する機能．コンソールに表示する形にするか，グラフィカルに表示する方法にするか，表示法を検討しよう．
- 地図をファイルに書き出す機能．ファイルから読み出す機能．ファイルの形式は自由．ただし，Java applet は，ファイルの入出力ができないので，Java applet の形であれば，この機能は不要．

図 5 は，この課題を Java applet で実現してみた例である．200 行弱の Java プログラムである．Save ボタンは applet なのでうまく働かない．

余力のある人は，**アドバンスト課題**として，課題 1 のプログラムで，迷路のデータ (不明無し) が与えられた時，スタートからゴールまでの経路を探索するプログラムを作成しよう．

余力の無い人も，何かしらのプログラムを作成して，メールで送ってください．

3.2 課題 2

ここまでのソフトウェア第三の講義について，感想や意見を書いてください．特に詳しく説明して欲しかった項目等あれば，それも教えてください．

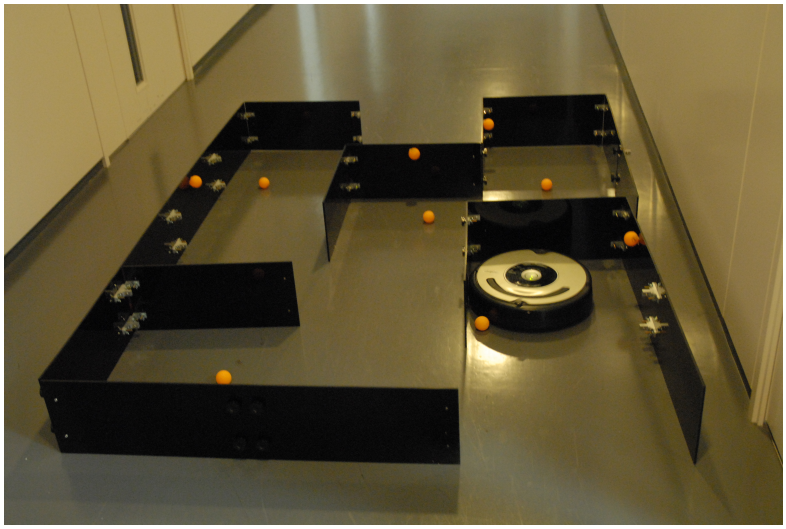


図 4: 課題の迷路のイメージ

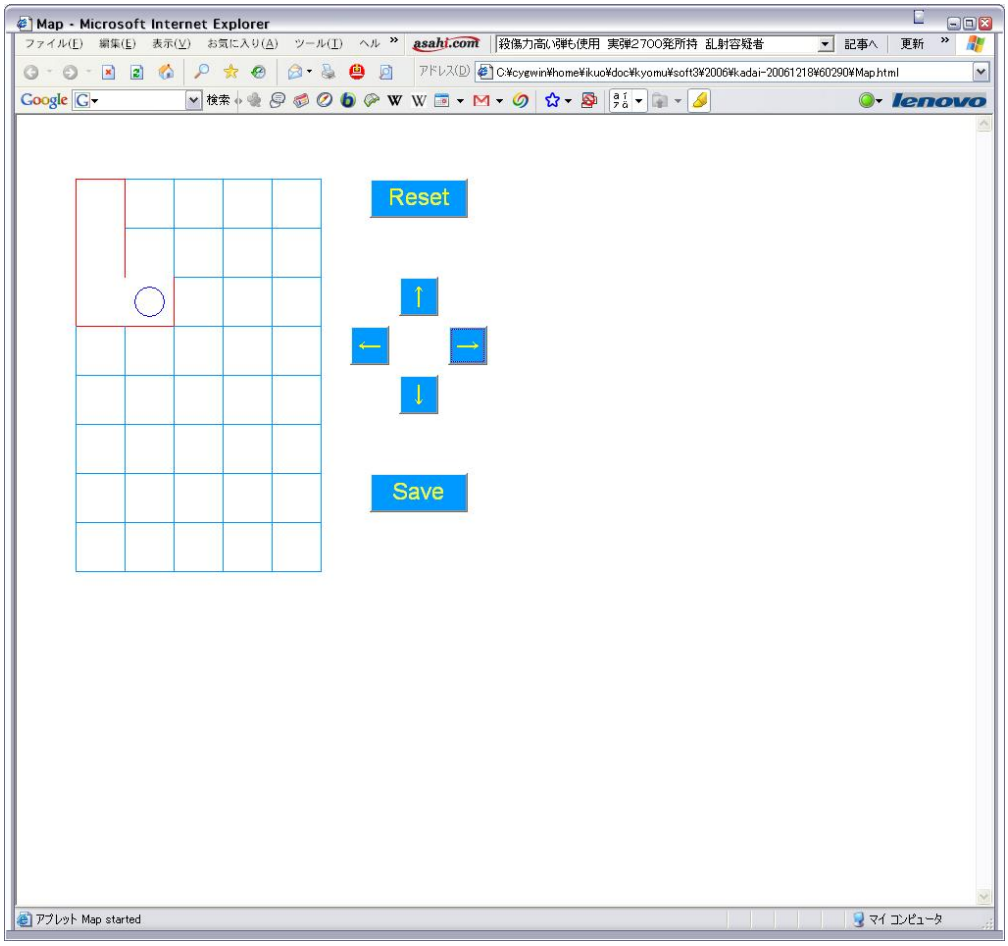


図 5: 課題のプログラムの実行サンプル