Java と NetBeans でつくるグラフィックス・プログラム

Java 言語 ： サンマイクロシステム社で開発されたオブジェクト指向言語
Windows, Unix など多様なシステムのもとで利用可能
WEB アプリケーション (Java Applet など) の開発にも利用される

NetBeans IDE ： Java 言語でのソフトウェアの統合開発環境を提供する
グラフィカルにプログラムの開発が可能
無償で入手可能 (`http://www.netbeans.jp/`)

Java Applet – Web Page 上で実行される Java 言語で書かれたプログラム

1 Java Applet の作成

NetBeans IDE を使用して、Java Applet を作成する。

1. お絵書きプログラム
2. ボールの運動シミュレーション・プログラム

2 お絵書きプログラム

ここでは、NetBeans IDE を使用して、WEB ブラウザ上で動作する。簡単なお絵書きプログラムを作成する。完成したプログラムはホームページ上に置くこともできる。

作成する Applet の機能 図 1 に作成する Applet を示す。

- ウィンドウ内にマウスカーソルを移動させ、左ボタンを押した状態でマウスカーソルを動かすことで、線画を描く。
- 描画する線の太さや色を変更することができる。
- clear ボタンを押すと、それまでに描いた図が消去される。

プログラム作成の概要
上記のような動作をする Applet を完成させるには、次の作業が必要になる。

- clear ボタンなどを画面上の適切な位置に配置する。
- プログラムが起動すると、
  1. マウスボタンが押されるのを待ち、マウスカーソルの位置に応じて適切な処理をする。
  2. マウスボタンが押された状態でマウスカーソルが移動すると、移動にあわせて線を描く。

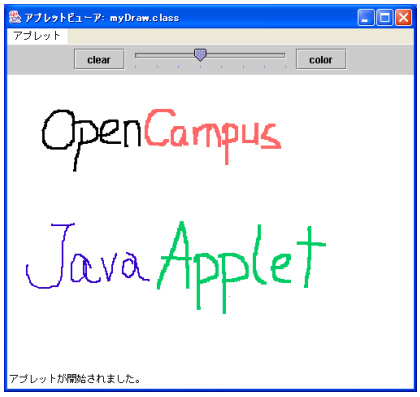

図 1: お絵書き Applet

といった動作をプログラムに組み込む。

統合開発環境では、このような作業を GUI(Graphical User Interface) のもとで比較的容易に行うことができる。

2.1 NetBeans IDE の起動

Windows 画面上で、図 2 に示すアイコンの上へマウスカーソルを移動させ左ボタンをダブルクリックする[1]。画面中央に

図 2: NetBeans IDE アイコン

図 3 が表示され、起動作業が開始される。

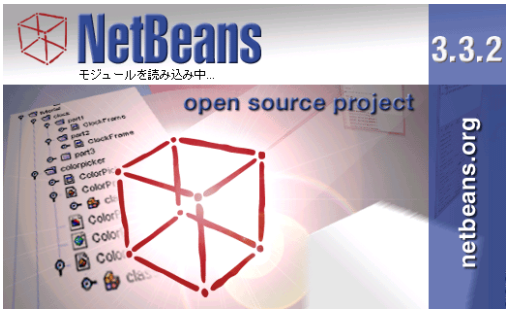

図 3: NetBeans の起動画面

しばらくすると、図 4 に示すウィンドウが開く。

2.2 名前の入力

Applet など Java のアプリケーションを新たに作成する場合は、作成するパッケージのテンプレート、名前などを入力する。

[1] ダブルクリック: 2 回連続してマウスボタンを軽く押す。

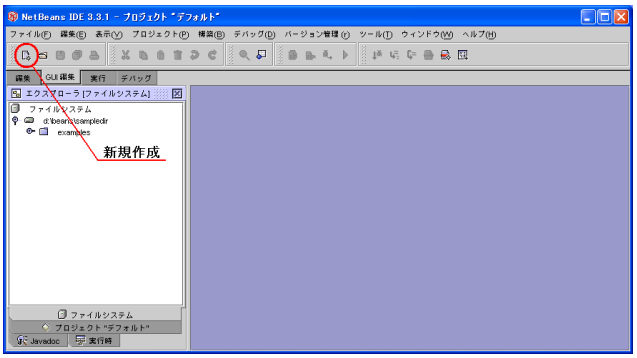

図 4: NetBeans(ウィンドウの一部)

図 4 に示す NetBeans ウィンドウ上部のツールバー左端のアイコン (図 4 中のまるで囲まれたアイコン) を左クリック、あるいは、メニューバーから [ファイル]–[新規] 選ぶ。すると、図 5 に示す新規ウィザード・ウィンドウが開かれる。

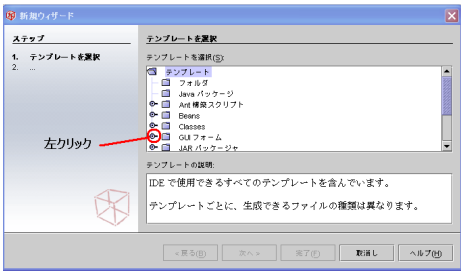

図 5: 新規ウィザード

これから Java Applet を作成するので、テンプレートより [GUI フォーム] を選ぶ。図 5 の GUI フォームの左側の丸いアイコンを左クリックすると、詳細な選択メニューが現れる。(図 6)

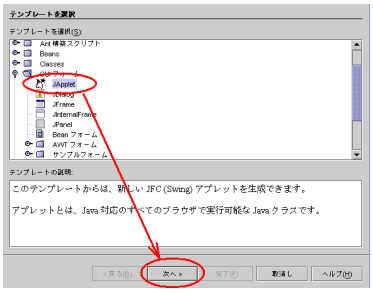

図 6: テンプレートメニュー

ここで、[JApplet] を選ぶと、JApplet という文字列の背景が灰色に変わる。この状態で [次へ] を左クリックする。(図 6)

作成場所の入力に移る。ここでは、作成する パッケージ (今回は Applet) に付ける名前を入力する。入力した名前のフォルダが作成され、このフォルダのなかに、必要なファイルが作成される。

「作成場所」の下の「名前」と表示されるボックス内を左クリックし、キーボードより myDraw と名前を入力する。別の名前を入力しても構わない。

今回は詳細な設定は行わないので、名前の入力後、[完了] を左クリックする。(図 7)

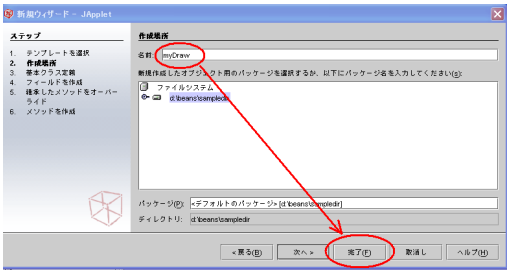

図 7: 作成場所の入力

ウィンドウは図 8 に示すように変わり、アプリケーションを作成するための、フォームエディタなどが表示される。

フォームエディタが表示されない場合は、「GUI 編集」タブをクリックして表示を切替える。

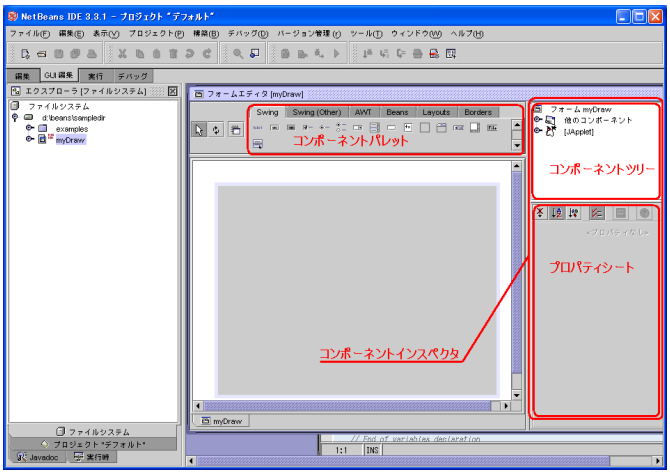

図 8: アプリケーションの作成画面

図 8 左側のエクスプローラ [ファイルシステム] には、これから作成するパッケージなどのファイル、ディレクトリ (フォルダ) の一覧がツリー形式で表示される。

2.3 コンポーネントの配置

図 8 に表示されたフォームエディタを使用して、Applet 上のボタンや描画領域の配置などを決める。

コンポーネントパレットに置かれているボタンやスライダーなどの GUI 要素 (コンポーネント) をフォームエディタ内に配置することで Applet 上での表示位置を指定する。後で、コンポーネントインスペクタを使用して、配置するコンポーネントの大きさなど詳細な設定を行う。

図 9 に作成する Applet の GUI(Graphical User Interface) 画面でのボタンなどの配置を示す。中央 (center) に描画領域を

置き、上部 (north) に、消去ボタン、線の太さを変更するスライダー、および、線の色を変更するカラーメニューを表示させるためのボタンを配置する。今回は、south, west, east には何も配置しない。

このような GUI 画面の配置とするために、border layout というレイアウトにより、次のようにコンポーネントを配置する。

- north(北) にパネルを配置する。このパレットの中に以下の各種ボタンなどを配置する。
  - clear ボタン
  - 線の太さを変更するためのスライダー
  - カラーメニュー表示のためのボタン
- center(中央) に描画用パネルを置く。

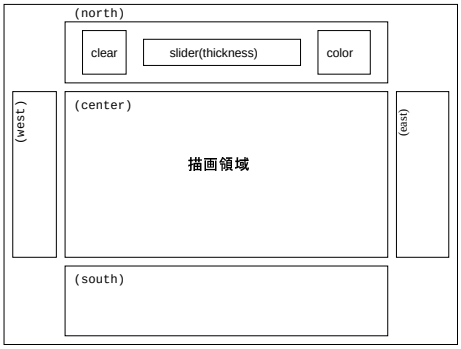

図 9: 作成する Applet のレイアウト

2.3.1 コンポーネント

図 8 中のコンポーネントパレットの内、ここで使用するものを図 10 に示す。

(a) Swing コンポーネント

(b) Swing(Other) コンポーネント

図 10: 使用するコンポーネントパレット

ここでは、(a) Swing コンポーネントパレット内の左から 2 つ目のボタン、右から 5 つ目のパネルを使用する。また、(b) Swing(Other) コンポーネントパレットでは、上段左端のスライダー (左右に動かすことで値を変えることができる)、右端のカラーチューザ (ColorChooser)、および、2 段目左から 3 つ目のダイアログを使用する。

2.3.2 コンポーネントの配置操作

まづ、パネル (Panel) を north(北) に配置する。続いて、このパネルのなかにボタンなど細かな部品 (コンポーネント) を配置する。

コンポーネントをアプリケーションに配置するには、まづ、コンポーネントパレット内の配置するコンポーネントを左クリックして選択し、その後、フォームエディタ上の配置する位置にマウスカーソルを移動させて左クリックする。

JPanel – ここでは、図 9 のようにコンポーネントを配置するので、まづ、図 11 に示すように、Swing タブを選択し、表示されるコンポーネントパレットより、JPanel を左クリックして、マウスカーソルをフォームエディタの上部で移動させて、左クリックする。

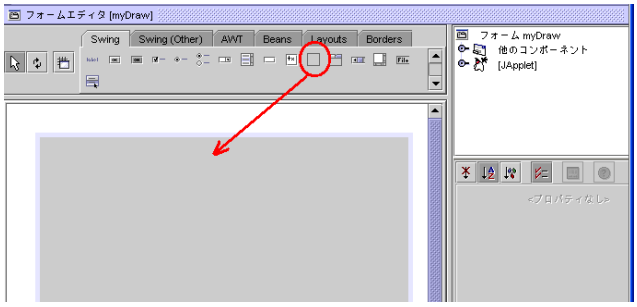

図 11: フォームエディタ上でのパネルの配置操作

画面が図 12 のように変わり、フォームエディタの上部 (north) に青色の長方形が表示され、コンポーネントツリーの JApplet の下に、BorderLayout と jPanel1[JPanel] が現れる。このように、フォームエディタ上に配置するコンポーネントには、その種類に応じた名前が付けられる。

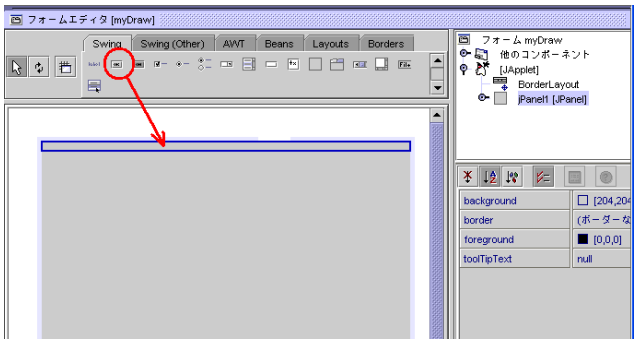

図 12: フォームエディタ上で north に配置されたパネル (jPanel1)

JButton – 次に、jPanel1 の上にボタンなどのコンポーネントを配置する。図 12 に示すように、Swing コンポーネントパレット内のボタン (JButton) をクリックした後に jPanel1 内へマウスカーソルを移動させて左クリックする。コンポーネン

トツリーに表示された jButton1 の位置より、これが、jPanel1 の内に置かれていることがわかる。また、jPanel1 内での配置が FlowLayout であることもわかる。(FlowLayout ではコンポーネントは横に整列して配置される。)

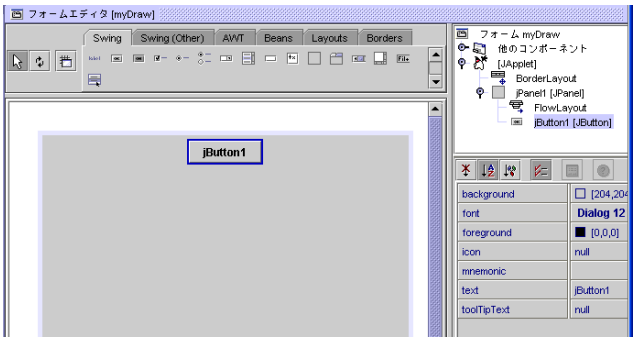

図 13: jPanel1 内に置かれたボタン (jButton1)

jButton1 と同様に、jPanel1 内に JSlider, JButton を配置する。スライダーの場合には、Swing(Other) タブをクリックして表示されるパレットより JSlider を選ぶ。JSlider および JButton には、それぞれ、jSlider1, jButton2 という名前が付けられる。

以上 3 つのコンポーネントを配置すると、図 14 に示すような表示になる。

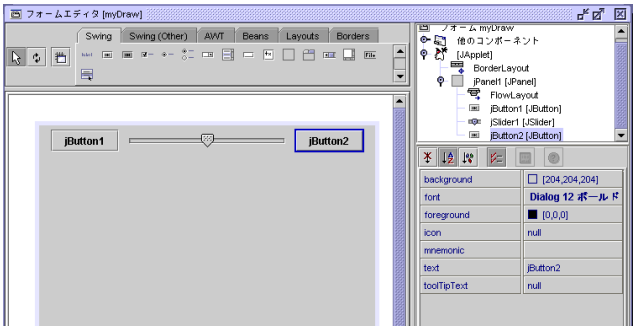

図 14: jPanel1 内でのコンポーネントの配置

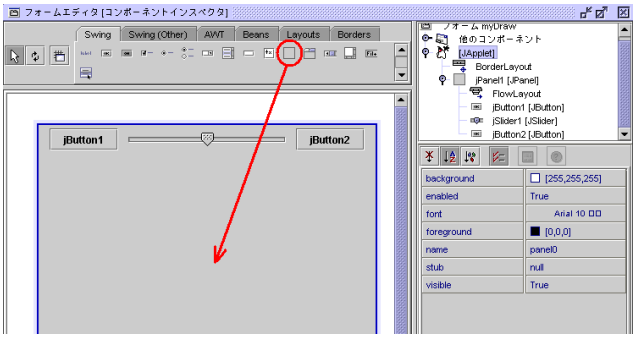

図 15: 描画パネルの配置

JPanel 図 15 に示すように、描画用のパネルを JApplet の center(中央) に配置する。

まづ、コンポーネントツリーの JApplet をクリックして、フォームエディタに JApplet 全体を表示させ、コンポーネントパレットより JPanel を選び、フォームエディタの中央 (center) 付近へマウスカーソルを移動させて、マウスボタンを左クリックする。

JColorChooser – 最後に、描画する線の色を変更するためのカラーチューザ (JColorChooser) を配置する。このコンポーネントは、jButton2 がクリックされたときに、新しく別のウィンドウ内で表示されるようにするため、JDialog コンポーネントの機能を利用する。以下に述べる操作により、JApplet とは別のウィンドウとして JDIalog を置き、この中に JColorChooser を配置する。

1. コンポーネントパレットより、Swing(Other) タブをクリックし表示されるパレットより JDialog を選んだ後にフォームエディタ内をクリックするとコンポーネントツリー内の「他のコンポーネント」の下に jDialog1 が表示される。

2. 続いて、コンポーネントツリー内の jDialog1 アイコンをクリックすると、フォームエディタの表示が、JApplet から jDialog1 に変わるので、フォームエディタにはコンポーネントを何も含まない状態が表示される。

3. JColorChooser を選択して、jDialog1 を表示している空の状態のフォームエディタ内でマウスボタンをクリックすると、図 16 の表示に変わる。

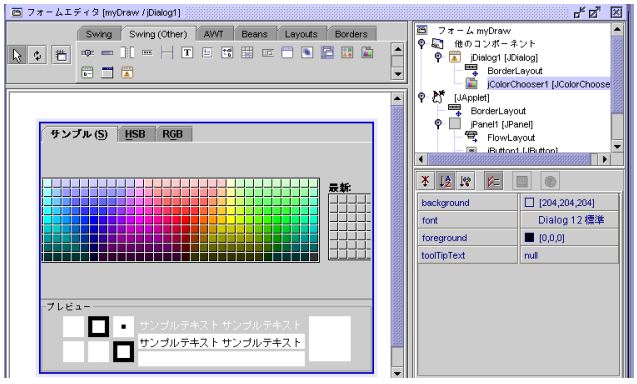

図 16: 描画パネルの配置

図 16 のコンポーネントツリーでコンポーネントの配置を確認すると、他のコンポーネントとして jDialog1 が配置され、この中に jColorChooser1 が置かれている。JApplet の内に BorderLayout で jPanel1 と jPanel2 が置かれている。jPanel1 の内には FlowLayout で jButton1, jSlider1, jButton2 置かれている。これらを clear ボタン、線の太さを変えるスライダ、ColorChooser を表示するためのボタンとして使用する。

2.4 イベント処理

作成するアプリケーションにおいて、ボタンを押す、マウスを動かす、キーボードのキーを押すなどの入力操作をイベントと呼ぶ。イベントに対応して、画面を消去する、線を描くなどの応答をアクションと呼んでいる。

ここでは、各イベントとそれに応答するアクションを登録する。

図 17 に例示するように、イベントとアクションの登録操作は次のようになる。

1. 画面左上の「接続モード」ボタンをクリックする。
2. 続いて、イベントの発生元をクリックする。
3. さらに、アクションの対象 (ターゲット) をクリックする。

clear – jButton1 をクリックした時に (jButton1 コンポーネントの上でマウスボタンが押されるイベントが発生した場合に)、jPanel2 上に描かれている図を消去する応答 (アクション) を登録する。

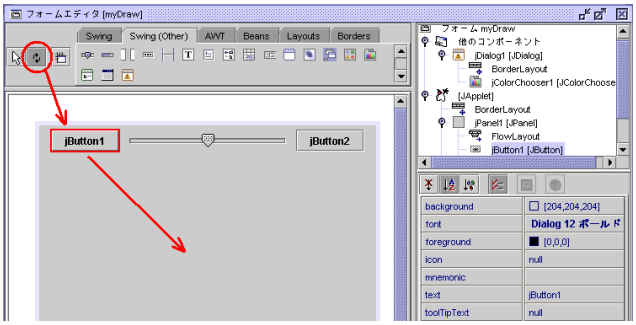

図 17: イベントの登録

図 17 に示すように、「接続モード」ボタンをクリックし、次に jButton1 の上でマウスの左ボタンをクリックする。この操作で、jButton1 の境界線が赤色に変わる。境界色が赤になることでこのコンポーネントに対するイベントを処理する操作であることが確認できる。続いて、jPanel2 の上でマウスボタンを左クリックすると、図 18 が表示される。

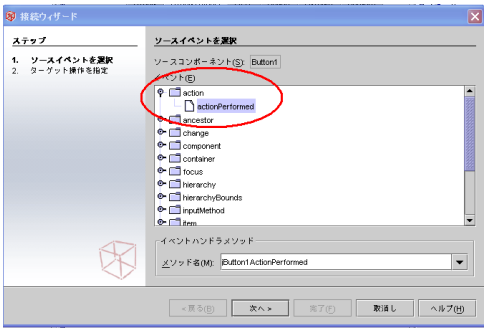

図 18: イベントの登録画面

ここでは、action の左側の短い横棒が付いた丸いアイコンをクリックして詳細を表示し、actionPerformed を選ぶ (actionPerformed の背景色が灰色に変わる)。この状態で、[次へ] をクリックし、次の操作画面へ移る。(図 19)

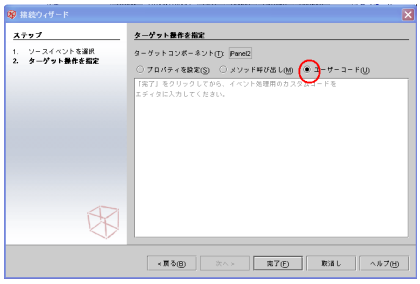

図 19: ターゲットの登録画面

jPanel2 に描かれた図を消去する操作は、直接にプログラムコードを入力することで登録するので、図の「ユーザーコード」の左側の丸いボタンを図のように選択した後に [完了] をクリックする。

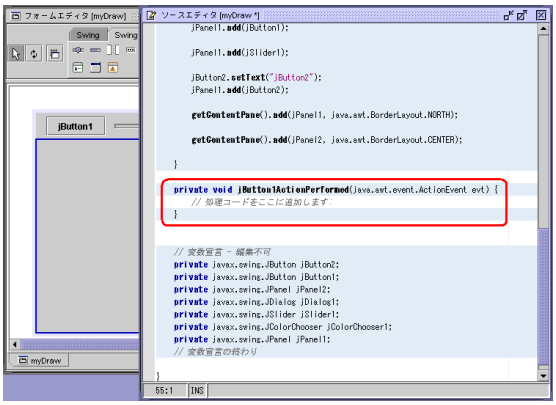

図 20: ソースエディタ

画面が切り替わり図 20 に示すように、ソースエディタ内の jButton1ActionPerformed という名前の関数の中にカーソルが置かれた状態でソースエディタが表示される。ここに、このイベントに対するアクション (jPanel2 内の図を消去する) を定義する。

```
// 処理コードをここに追加します
```

とある行を次の行に書き換える:

```
Graphics g = jPanel2.getGraphics();
Dimension s = jPanel2.getSize();
g.clearRect(0, 0, s.width, s.height);
jPanel2.setVisible(true);
```

図 21 に枠で囲って置き換える部分を示す。図の背景色が付いているが行は、エディタに対して保護されており書き換えることができない。

図 21: アクションの登録

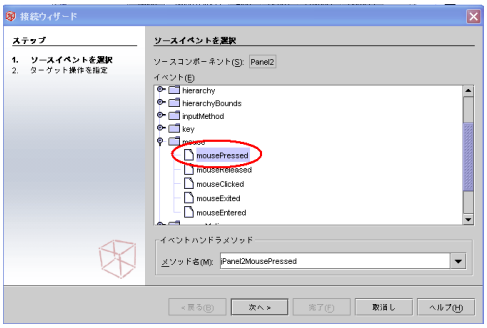

図 23: mousePressed イベントの登録画面

color – フォームエディタを表示して、jButton2 ボタンが押された時に jColorChooser1 を含む jDialog1 が表示されるようにイベントとアクションを登録する。

先の clear と同様に、フォームエディタを利用して、jButton2 でのイベント actionPerformed に対するアクションとしてユーザーコードーを登録する。ソースエディタにより、jButton2ActionPerformed関数が次の内容になるよう 2 行目を書き換える:

```
private void jButton2ActionPerformed(...) {
    jDialog1.show();
}
```

描画 – jPanel2 内に図を描くために 2 つのイベントとアクションを登録する。

- jPanel2 内でマウスボタンが押された時にマウスカーソルの位置を記録する。また、このときの jColorChooser1 と jSlider1 の状態から描く線の色と太さを決めておく。
- jPanel2 内でマウスがドラッグ (ボタンが押された状態でカーソルが移動する) されたときに、マウスカーソルの以前の位置と現在の位置を結ぶ直線を描く。細かく直線を描き続ければマウスカーソルが移動した軌跡が残る。

mousePressed – jPanel2 内でマウスボタンが押されたときの処理:

図 22, 23 に示すような操作で、jPanel2 内でマウスボタンが押されたときの処理 (mousePressed) を登録する。

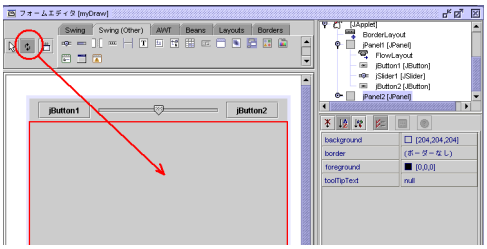

図 22: イベントの登録画面

この場合も、ターゲット操作は「ユーザーコード」を選ぶ。アクションの登録は次のようにする:

```
private void jPanel2MousePressed(...) {
  p = evt.getPoint();
  g = (Graphics2D)jPanel2.getGraphics();
  g.setStroke(new BasicStroke((float)jSlider1.getValue()));
  g.setColor(jColorChooser1.getColor());
}
```

このアクションの内容を簡単に説明する。

1. マウスがクリックされた点の座標を p に記録する。
2. jPanel2 に図を描くための Graphics オブジェクトを入手し g に記録する。
3. jSlider1 の値を Graphics オブジェクト g により描く線の太さとする。
4. jColorChooser で指定された色を g により描く線の色とする。

mouseDragged – jPanel2 内でマウスカーソルがドラッグされたときの処理:

mousePressed と同様にして次のアクションを登録する:

```
private void jPanel2MouseDragged(...) {
    p0 = p;
    p = evt.getPoint();
    g.draw(new Line2D.Double(p0.x, p0.y, p.x, p.y));
    jPanel2.setVisible(true);
}
```

このアクションの内容を簡単に説明する。

1. p0 に以前のマウスカーソルの位置を記録する。
2. このイベントが発生した時点でのマウスカーソルの座標を p に記録する。
3. Graphics オブジェクト g により p0 から p まで直線を描く。
4. jPanel2 の表示を更新する。

ソースコードの編集　– ソースコードの先頭部分と末尾部分を追加し、以下にアンダーラインで示す行を追加する。

先頭部分

```
/**
 *
 * @author  ...
 */
import java.awt.*;
import java.awt.geom.*;

public class myDraw extends javax... {

    /** Creates new form myDraw */
    public myDraw() {
```

末尾部分

```
    // 変数宣言 - 編集不可
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JSlider jSlider1;
    private javax.swing.JDialog jDialog1;
    private javax.swing.JColorChooser jColorChooser1;
    private javax.swing.JPanel jPanel1;
    // 変数宣言の終わり
    private Graphics2D g;
    private Point p, p0;
    private int d;
}
```

2.5　コンポーネントの詳細設定

以下のようにボタンに表示される文字列などを変更する。

1. フォームエディタ、あるいは、コンポーネントツリーの jButton1 をクリックし、プロパティシートに表示される jButton1 のプロパティの内、text の値 jButton1 をクリックする。

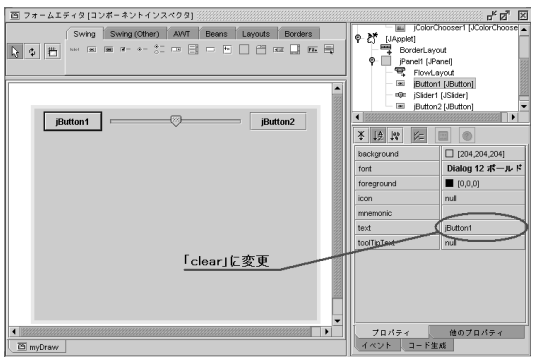

図 24: jButton1 の text プロパティの変更

キーボードからの入力により、この値を clear に変更する。

2. 同様にして、jButton2 の text プロパティ値を color に変更する。

3. jSlider1 のプロパティを下記のように変更する。

maximum	12
minimum	1
minorTickSpacing	1
paintTicks	True
value	3

4. jPanel2 の背景色を灰色から白に変更する。

background	[255,255,255]

5. jDialog1 のコンポーネントシートの「コード生成」タグをクリックし、表示される「初期化後コード」の空白となっているプロパティ値の上でマウスを左クリックすると下図のような表示に変わる。

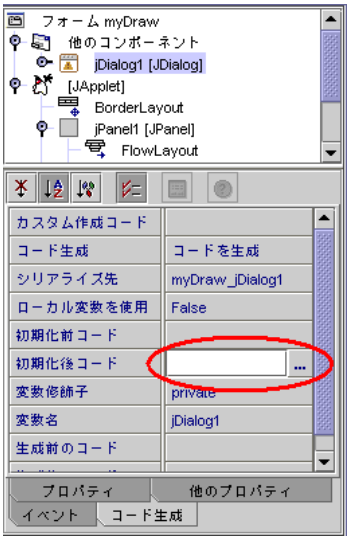

図 25: コードの追加

ここで、「...」の箇所を左クリックして表示されるウィンドウ内に次のコードを書き込む。

```
jDialog1.setSize(420, 300);
```

図 26: 初期化後コードの追加

2.6 Applet の実行

アプリケーションが完成したので、実行に移る。図 27 に示す三角形のアイコンを左クリックすると自動的に実行までに必要な処理が行われる。

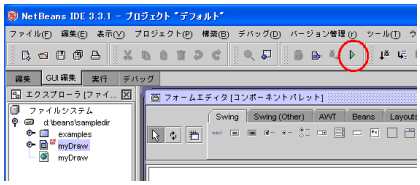

図 27: 実行

途中で入力ミスなどによる問題が発生すると、そのことを通知がコードエディタ中の行番号などとともに表示される。このような場合には、コードエディタにより問題の箇所を修正する。

標準では WEB ページが狭く、描画画面全体が表示されないので、一旦、実行を終了し、ページサイズを変更する。図 28 に示すエクスプローラ中の myDraw アイコンをクリックするとソースエディタ内に WEB ページのソーステキストが表示されるので、図中のアンダーラインで示した行を書き換えて、サイズを調節する。

```
<APPLET code="myDraw.class" width=420 height=300></APPLET>
```

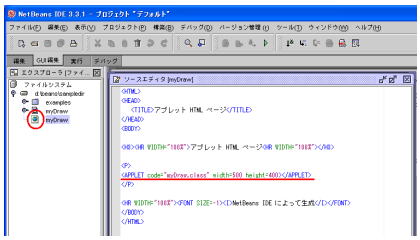

図 28: myApplet.html の修正

3 ボールの運動シミュレーション・プログラム

ここでは、NetBeans IDE を使用して、WEB ブラウザ上で動作する。簡単なボールの運動をシミュレートするプログラムをつくる。

作成する Applet の機能 図 29 に作成する Applet を示す。

- ボールがウィンドウ内を壁で反射しながら運動する。
- 重力を考慮する場合と重力が無い場合の 2 通りについてのシミュレーションを行う。
- スタート、ストップボタンで運動の開始と停止を切替える。
- 運動開始時の水平、および、垂直方向の初期速度をスライダーにより変更できる。(停止状態)

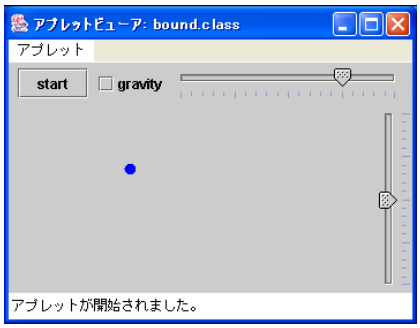

図 29: ボールの運動シミュレーション

プログラム作成の概要

上記のような動作をする Applet を完成させるまでの作業を以下に示す。

- start ボタンなどを画面上の適切な位置に配置する。
- 時間を追って、ボールの位置を計算し、画面上に描く機能をソースコードに書き込む。

統合開発環境では、このような作業を GUI(Graphical User Interface) のもとで比較的容易に行うことができる。

3.1 NetBeans IDE の起動

Windows 画面上で、図 30 左に示すアイコンの上へマウスカーソルを移動させ左ボタンをダブルクリックする[2]。画面中央に図 30 右が表示され、起動作業が開始される。

しばらくすると、図 31 に示すウィンドウが開く。

[2] ダブルクリック: 2 回連続してマウスボタンを軽く押す。

図 30: NetBeans のアイコンと起動画面

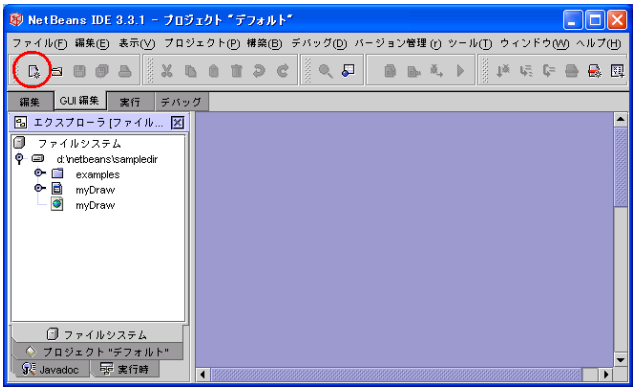

図 31: NetBeans(ウィンドウの一部)

3.2 名前の入力

Applet など Java のアプリケーションを新たに作成する場合は、作成するパッケージのテンプレート、名前などを入力する。

図 31 に示す NetBeans ウィンドウ上部のツールバー左端のアイコン (図中のまるで囲まれたアイコン) を左クリック、あるいは、メニューバーから [ファイル]–[新規] 選ぶ。すると、図 32 に示すような新規ウィザード・ウィンドウが開かれる。

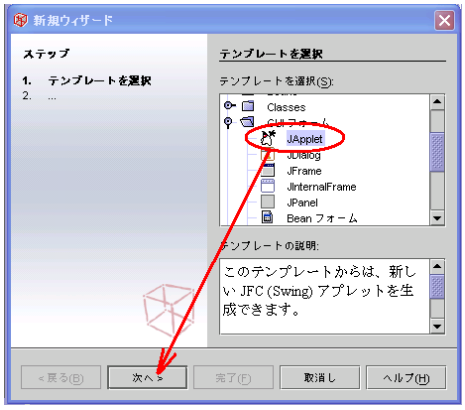

図 32: 新規ウィザード

これから Java Applet を作成するので、テンプレートより [GUI フォーム] を選ぶ。図 32 の GUI フォームの左側の丸いアイコンを左クリックすると、詳細な選択メニューが現れ、図 32 の表示となる。ここで、[JApplet] を選ぶと、JApplet という文字列の背景が灰色に変わる。この状態で [次へ] を左クリックする。

作成場所の入力に移る。ここでは、作成する パッケージ (今回は Applet) に付ける名前を入力する。入力した名前のフォルダが作成され、このフォルダのなかに、必要なファイルが作成される。

「作成場所」の下の「名前」と表示されるボックス内を左クリックし、キーボードより bound と名前を入力する。別の名前を入力しても構わない。

今回は詳細な設定は行わないので、名前の入力後、[完了] を左クリックする。(図 33)

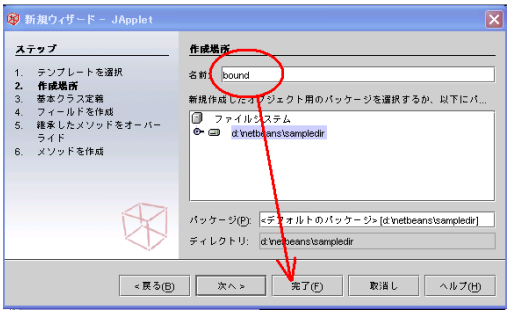

図 33: 作成場所の入力

ウィンドウは図 34 に示すように変わり、アプリケーションを作成するための、フォームエディタなどが表示される。

フォームエディタが表示されない場合は、「GUI 編集」タブをクリックして表示を切替える。

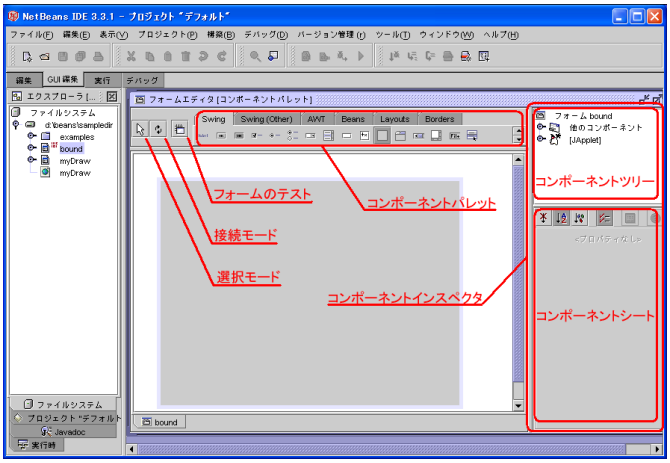

図 34: アプリケーションの作成画面

図 34 中に各部分の名称を示す。

選択モード – 作成するウィンドウ画面上にボタンなどの要素 (コンポーネント) を配置する場合にこのアイコンをクリックする。

接続モード – 作成するアプリケーションの実行時に、ボタンなどをクリックすると表示の変更や処理を行うなど、実行時

の操作 (イベント) と処理 (アクション) の関係を定義する際にこのアイコンをクリックする。

フォームのテスト – コンポーネントの配置を確認するときにこのアイコンをクリックする。

コンポーネントパレット – 画面上に配置するコンポーネントをここから選択する。

コンポーネントツリー – 配置されたコンポーネントの画面上での相互関係を表示する。

コンポーネントシート – 配置されたコンポーネントの持つ各種パラメータ (サイズ、背景色など) の値を表示する。

図 34 左側のエクスプローラ [ファイルシステム] には、これから作成するパッケージなどのファイル、ディレクトリ (フォルダ) の一覧がツリー形式で表示される。

3.3 コンポーネントの配置

図 34 に表示されたフォームエディタを使用して、Applet 上のボタンや描画領域の配置などを決める。

コンポーネントパレットに置かれているボタンやスライダーなどの GUI 要素 (コンポーネント) をフォームエディタ内に配置することで Applet 上での表示位置を指定する。後で、コンポーネントインスペクタを使用して、配置するコンポーネントの大きさなど詳細な設定を行う。

図 35 に作成する Applet の GUI(Graphical User Interface) 画面でのボタンなどの配置を示す。中央 (center) に描画領域を置き、上部 (north) に、スタートボタン、重力の ON/OFF ボックス (チェックボックス)、および、水平方向の初期速度を変えるためのスライダーを配置する。右側 (east) に垂直方向の初期速度を調整するためのスライダーを置く。描画画面は左右を水平方向とし、重力は下方向に加わるとしてボールの運動をシミュレートする。今回は、south, west には何も配置しない。

このような GUI 画面の配置とするために、border layout というレイアウトを使用し、次のようにコンポーネントを配置する。

- north(北) にパネルを配置する。このパレットの中に以下の各種ボタンなどを配置する。
  - start/stop ボタン
  - gravity チェックボックス
  - 水平方向のスライダー
- east(右側) に垂直方向のスライダーを置く。
- center(中央) に描画用パネルを置く。

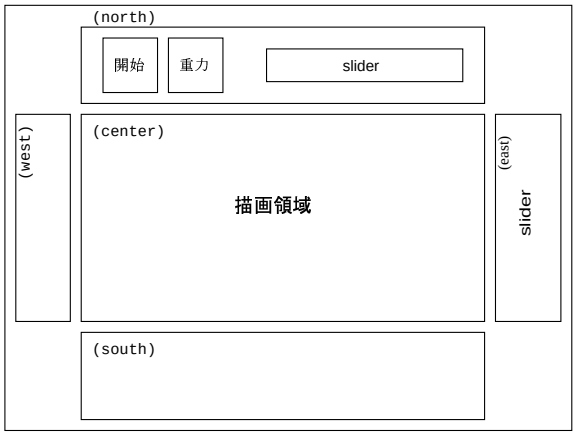

図 35: 作成する Applet のレイアウト

3.3.1 コンポーネント

図 34 中のコンポーネントパレットの内、ここで使用するものを図 10 に示す。

(a) Swing コンポーネント

(b) Swing(Other) コンポーネント

(c) Beans コンポーネント

図 36: 使用するコンポーネントパレット

ここでは、
(a) Swing コンポーネントパレット内の左から 3 つ目のトグルボタン、右から 5 つ目のパネルを使用する。
(b) Swing(Other) コンポーネントパレットでは、上段左端のスライダー (左右に動かすことで値を変えることができる) を使用する。
(c) Beans コンポーネントパレット中の時計のアイコンで表示された Timer をアニメーションを実現するために使用する。

3.3.2 コンポーネントの配置操作

まづ、パネル (Panel) を north(北) に配置する。続いて、このパネルのなかにボタンなど細かな部品 (コンポーネント) を配置する。

コンポーネントをアプリケーションに配置するには、まづ、コンポーネントパレット内の配置するコンポーネントを左ク

リックして選択し、その後、フォームエディタ上の配置する位置にマウスカーソルを移動させて左クリックする。

JPanel(1) – ここでは、図 35 のようにコンポーネントを配置するので、まづ、図 37 に示すように、Swing タブを選択し、表示されるコンポーネントパレットより、JPanel を左クリックして、マウスカーソルをフォームエディタの上部で移動させて、左クリックする。

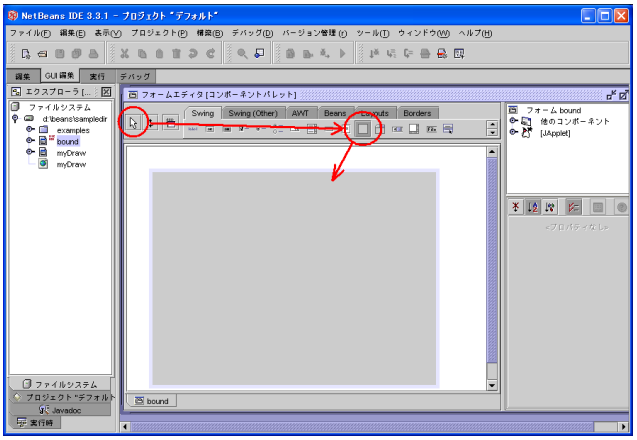

図 37: フォームエディタ上でのパネルの配置操作

画面が図 38 のように変わり、フォームエディタの上部 (north) に青色の長方形が表示され、コンポーネントツリーの JApplet の下に、BorderLayout と jPanel1[JPanel] が現れる。このように、フォームエディタ上に配置するコンポーネントには、その種類に応じた名前が付けられる。

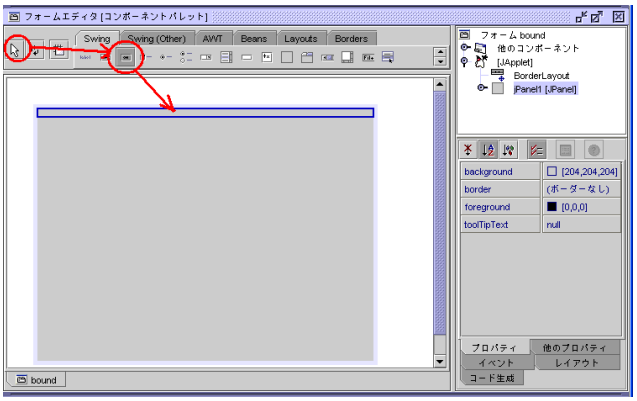

図 38: フォームエディタ上で north に配置されたパネル (jPanel1)

JToggleButton – 次に、jPanel1 の上にボタンなどのコンポーネントを配置する。図 38 に示すように、Swing コンポーネントパレット内のトグルボタン (JToggleButton) をクリックした後に jPanel1 内へマウスカーソルを移動させて左クリックする。

ToggleButton は ON/OFF 状態を持つボタンであるから、これをアニメーションの start/stop に使用する。

コンポーネントツリーに表示された jToggleButton1 の位置より、これが、jPanel1 の内に置かれていることがわかる。また、jPanel1 内での配置が FlowLayout であることもわかる。(FlowLayout ではコンポーネントは横に整列して配置される。)

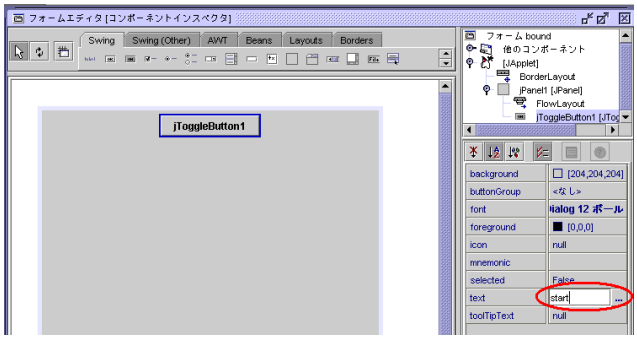

図 39: jPanel1 内に置かれたトグルボタン (jToggleButton1)

図 39 に示すコンポーネントシート中に表示されている jToggleButton1 の属性値のうち text の属性値を「start」に変更する。このために、まづ、図中の丸印で囲った属性値をマウスで左クリックする。図のように表示が変わるので、キーボードよりバックス内に「start」と入力する。

JCheckBox – 同様に jPanel1 内にチェックボックス (JCheckBox) を配置し、そのテキスト値を「gravity」と変更する。CheckBox はチェックの「有り/無し」を状態として持つので、ここをチェックした場合にボールの運動に重力を考慮することとする。

jToggleButton1 と同様にして、このチェックボックスのテキスト属性値を「gravity」に変更する。

JSlider(1) – 次に、同じく、jPanel1 内に水平方向の初期速度を設定するスライダーを置く。Swing(other) タブをクリックして表示されるコンポーネントより、スライダー (JSlider) を jPanel1 内に配置する。

この jSlider1 の属性値を下表のように変更する。

属性	値	意味
majorTickSpacing	10	目盛の間隔
maxmum	20	最大値
minimum	−20	最小値
mimorTickSpacing	2	目盛の間隔
paintTicks	True	目盛を表示する
value	10	初期値

JSlider(2) – JApplet の右側 (east) に上下方向の初期速度を設定するスライダーを配置する。図 40 に示すように、コンポーネントツリー内の JApplet を左クリックしてフォームエディタに作成する Applet 全体を表示した後、コンポーネント

パレット内の Swing(Other) タブをクリックして表示されるコンポーネントよりスライダー (JSlider) を選び、フォームエディタの右端をクリックする。

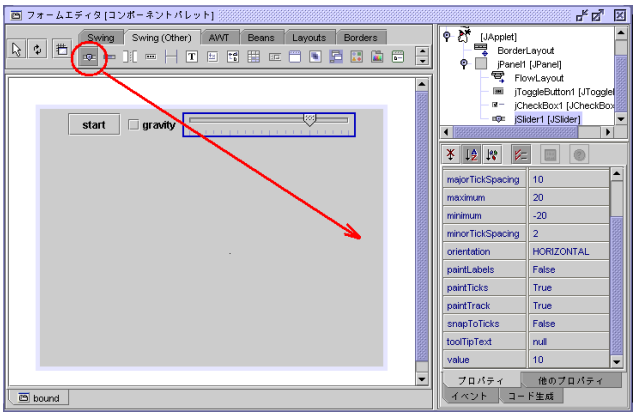

図 40: スライダー (jSlider2)

この後、jSlider1 と同様の方法で、スライダーの方向、その他 jSlider2 の属性値を下表のように変更する。

属性	値	意味
majorTickSpacing	10	目盛の間隔
maxmum	20	最大値
minimum	−20	最小値
minorTickSpacing	2	目盛の間隔
orientation	VERTICAL	方向
paintTicks	True	目盛を表示する
value	10	初期値

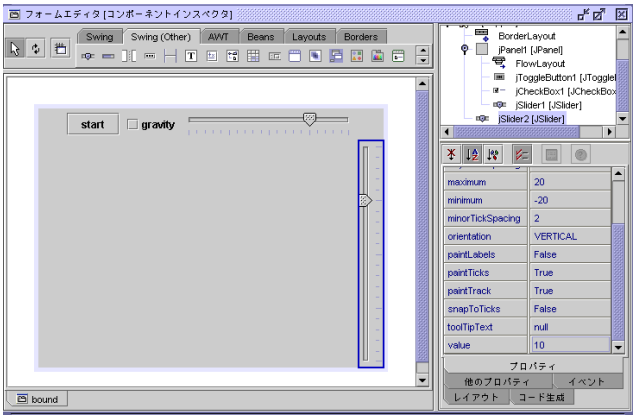

図 41: jSlider2 のコンポーネントシート

JPanel(2) – jPanel1 などと同様に、次の操作で、ボールの動きを表示するパネルを中央に配置する。

1. 「選択モード」アイコンを左クリックした後
2. コンポーネントパレットの JPanel アイコンをクリックする
3. 次に、フォームエディタ内の中央 (center) 付近をクリックする。
4. この描画パネルの周囲に境界線を描くために、jPanel2 のコンポーネントシートの border の値を (ボーダーなし) から [MatteBorder] に変更する。このために図の丸で囲った部分にマウスカーソルを移動させ左クリックする。このとき「..」と表示される箇所を再度左クリックすると新しいウィンドウが開き、境界の表示形式の選択が可能となるので、MatteBorder を選択し、「完了」をクリックしてこのウィンドウを閉じる。

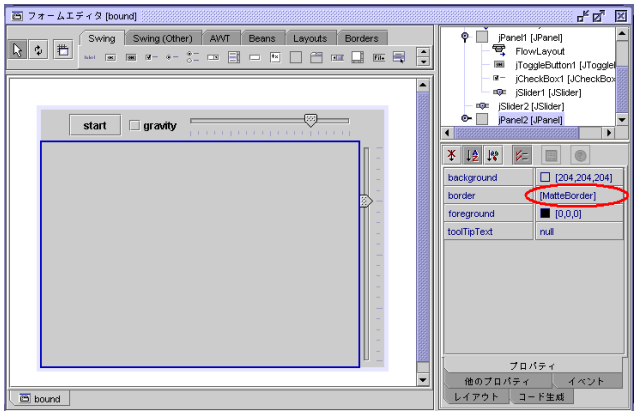

図 42: ボールの描画パネル (jPanel2)

3.4 イベント処理

作成するアプリケーションにおいて、ボタンを押す、マウスを動かす、キーボードのキーを押すなどの入力操作をイベントと呼ぶ。イベントに対応して、画面を消去する、線を描くなどの応答をアクションと呼んでいる。

ここでは、各イベントとそれに応答するアクションを登録する。

スタートボタン (jToggleButton1) – 「start」ボタンを押すと、ボタンの表示が「stop」に変わり、もう一度、このボタンを押すと表示が「start」に戻るように設定する。このためには、ボタンが押されたというイベントに対して、表示を切替えると言うアクションを接続する。

図 43 に示すように次の操作を行い、アクションを登録する。

1. 「接続モード」アイコンをクリックする。
2. 次に、イベントの発生元である start ボタン (jToggleButton1) をクリックする。この操作でボタンが赤色の枠で囲まれて表示される。
3. アクションを定義するために、フォームエディタ内でマウスを左クリックする。

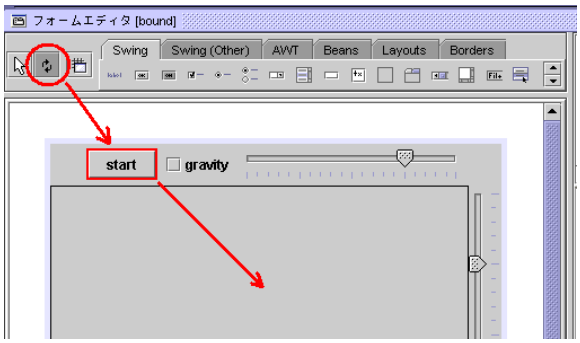

図 43: jToggleButton1 で発生したイベントに対する処理操作

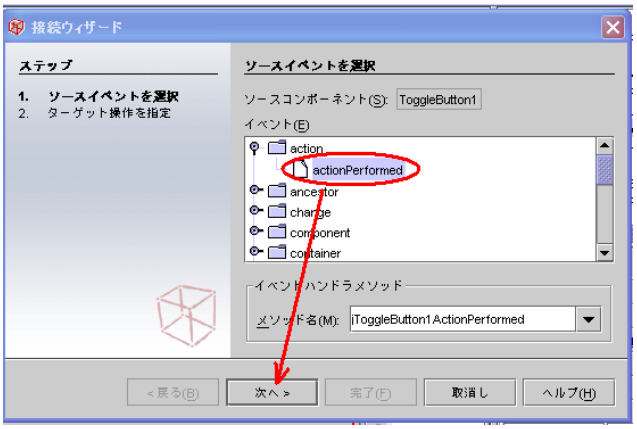

図 44: イベントの登録

図 44 に示す接続ウィザードが表示される。

ここで、ボタンが押されたことに対する処理をしたいので、action より actionPerformed を選択する。もし、actionPerformed がこのウィザード内に表示されていない場合には、action の左横の小さな丸印をクリックすると、actionPerformed が表示される。

actionPerformed をクリックして、図のように背景色が変わると、メソッド名のボックスに「 jToggleButton1ActionPerformed 」と表示される。この状態で「次へ」を左クリックする。

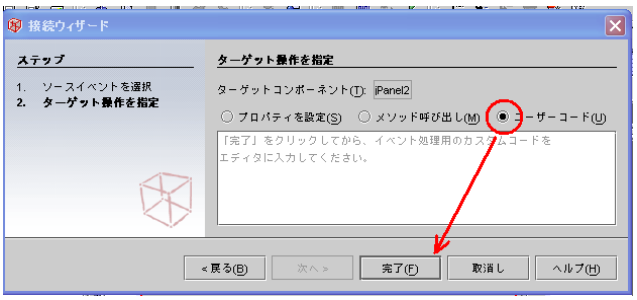

図 45: アクションのターゲット操作

今回は、アクションのコードを直接キーボードから入力するので、図 45 に示すようにユーザーコードの左横の小さな丸をクリックした後に「完了」をクリックして次の操作に移る。

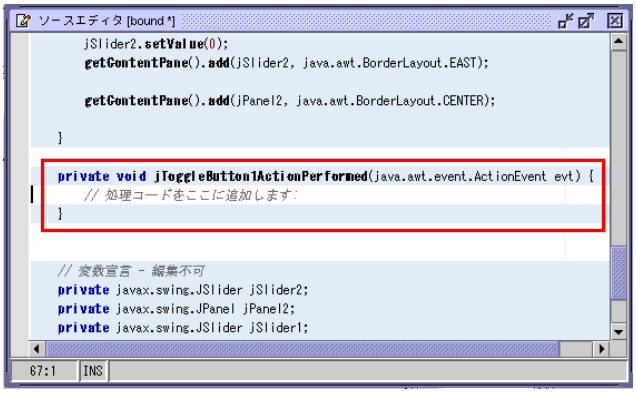

図 46: ソースエディタ

図 46 に示すソースエディタが表示されたところで、このエディタ内の

// 処理コードをここに追加します.

と表示されている行を次の最初と最後の行を除いたもので置き換える.

```
private void jToggleButton1ActionPerformed(...) {
  if( jToggleButton1.isSelected() ) {
      vx = jSlider1.getValue();
      vy = -jSlider2.getValue();
      jToggleButton1.setText("stop");
  } else {
      jToggleButton1.setText("start");
  }
}
```

図 46 のソースエディタ内で、背景色が付いているが行は、エディタに対して保護されており書き換えることができない。

入力を終えたところで，図 47 に示すソースエディタ右上の右から 3 つ目のアイコンをクリックしてソースエディタをアイコン化し，後ろに隠れているフォームエディタを表示させる。

図 47: jToggleButton1 に対するアクション

3.5 Timer 処理

ボールの動きをシミュレートするには、一定時間間隔でボールの移動にあわせて画面を書き換える動作が必要になる。この

アニメーションを実現するために、Beans の Timer コンポーネントを利用する。

フォームエディタを表示し、

1. 「選択モード」アイコンをクリックした後、
2. コンポーネントパレットの Beans タブをクリックして
3. 表示される Timer を選択する (クリックする)。
4. フォームエディタ内の適当な場所でマウスをクリックすると、図 48 に示すように、コンポーネントツリーの「他のコンポーネント」の下に「timer1[Time]」という表示がアイコンとともに追加される。
5. ここで、コンポーネントシート内の Delay 値を 200 に書き換える。

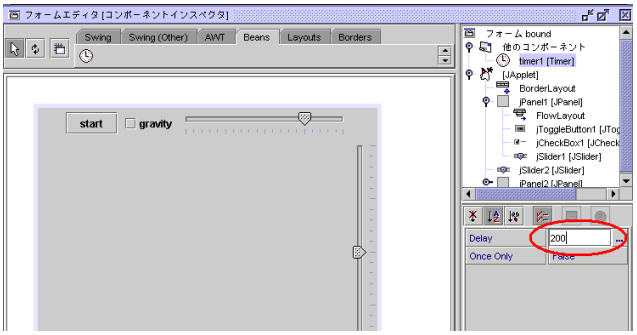

図 48: Timer コンポーネント (Beans)

上で書き換えた Delay は、タイマーの割り込み時間を決めるもので、ミリ秒を単位とした値である。したがって、ここでは 0.2 秒間隔でアニメーションのコマを進めることになる。

次に、タイマーによる割り込み処理を登録する。これからアニメーションの各コマ毎のボールの動作と、ボールの描画方法を登録する。

図 49 に示すように、フォームエディタ内の「接続モード」アイコンをクリックした後にコンポーネントツリー内のタイマをクリックする。

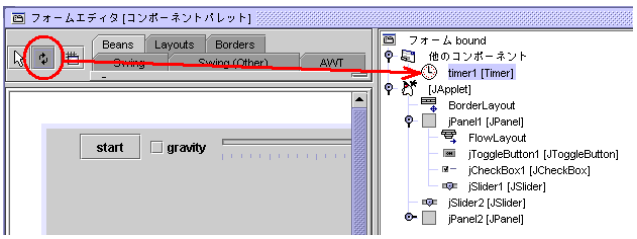

図 49: ソースエディタ

図 50 に示すようにソースイベントを timer の下の onTime イベントを選択し、スタートボタンの場合と同様に、ターゲット操作指定においてユーザーコードを選択し以下に示すソースコードをソースエディタにより入力する。

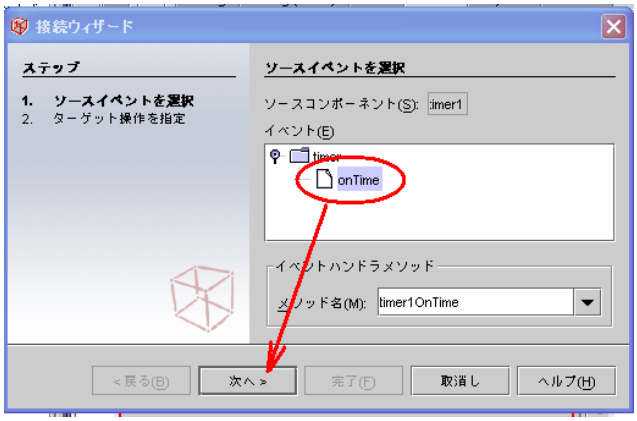

図 50: onTime イベント

```
private void timer1OnTime(...) {
  if(jToggleButton1.isSelected()) {
      if(g == null) gInit();
      x += vx;
      if(x<0) {
         x = -x;
         vx = -vx;
      } else if(x > w) {
         x = 2*w - x;
         vx = -vx;
      }
      vx *= a;

      if(jCheckBox1.isSelected()) vy++;
      y += vy;
      if(y < 0) {
         y = -y;
         vy = -vy;
      } else if(y > h) {
         y = 2*h-y;
         vy = -vy;
      }
      vy *= a;
      repaint();
  }
}
```

timer1OnTime(タイマー処理)

上記の他に、次のコードを追加する必要がある。

先頭部に追加するコード :

(アンダーラインで示す行を入力する。)

```
/**
 *
 * author kinoshita@
 */
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;

public class bound1 extends ...  {
```

末尾部に追加するコード :

と最後の行

```
    }
```

の間に以下のソースコードを追加する。

ソースコードの後ろの部分に以下のアンダーラインで示す行を 除いた行 を書き加える。

```
    private javax.swing.JToggleButton jToggleButton1;
// 変数宣言の終わり
    public void run() {
    }

    double x, y, vx, vy;
    double a = 0.998;
    Graphics2D g = null;
    int w, h;

    public void gInit() {
        g = (Graphics2D)jPanel1.getGraphics();
        Dimension s = jPanel1.getSize();
        w = s.width - 10;
        h = s.height - 10;
        x = 100;
        y = 50;
    }

    public void paint(Graphics g0) {
        if(g == null) gInit();
        super.paint(g0);
        g.setColor(Color.blue);
        g.fill(new Ellipse2D.Double(x,y,10,10));
        jPanel1.setVisible(true);
    }
}
```

3.6 Applet の実行

アプリケーションが完成したので、実行に移る。図 51 に示す三角形のアイコンを左クリックすると自動的に実行までに必要な処理が行われる。

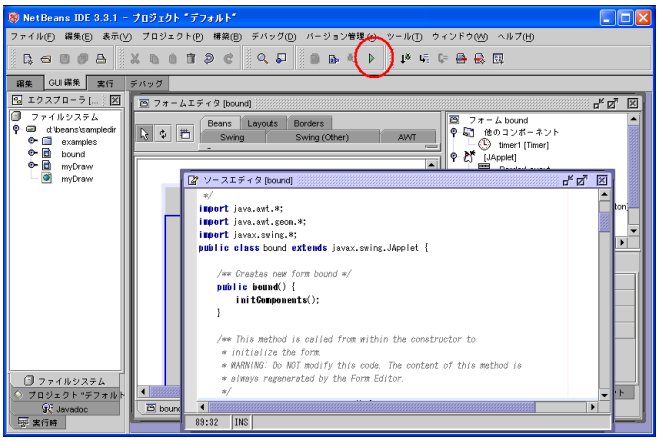

図 51: 実行

途中で入力ミスなどによる問題が発生すると、そのことを通知がコードエディタ中の行番号などとともに表示される。このような場合には、コードエディタにより問題の箇所を修正する。