第 1 回 Swing ゼミ

ゼミ担当者　：　村上　耕平，渡辺　崇文，嶋田　明奈
指導院生　：　西村　悟，山川　望
開催日　：　2006 年 5 月 18 日

ゼミ内容: 本ゼミでは，プログラミング演習に向けて，GUI 方式のプログラム作成に必要となる Swing について学ぶ．また，VisualEditor を利用して Swing を用いた GUI プログラムを作成する．

1　はじめに

本ゼミでは，Java を用いた GUI プログラムの作成において，最低限必要となる Swing の知識の取得を目的とする．Java には，AWT(Abstract Window Toolkit) と呼ばれる GUI 用ライブラリが用意されていたが，Java のバージョンアップが進み，ver.1.2 で大幅な強化がなされた. この時に，AWT と変わって標準化されたのが Swing である．Swing を使うと，あらゆるプラットフォームでまったく同じ外観と動作を持つアプリケーションを作成できるようになる. 本研究室では，作成したシステムのデモを発表する機会があり，その際に GUI プログラミングの知識が必要不可欠となる．そこで，本ゼミでは GUI プログラミングの利用が可能となるように Swing に関して学ぶ．また，Swing のプログラムを容易に作成するツールである VisualEditor の利用に関しても学ぶ.

2　Swing とは

Swing とは Sun Microsystems 社のプログラミング言語 Java に標準で付属するグラフィック関連のクラスライブラリであり，Java2(旧 JDK1.2) から標準搭載された JFC(Java Foundation Class) の一部である．Swing は，従来から提供されている AWT に比べて，以下の点が大きく改良されている．

- 実行環境によらない統一された GUI を提供
- 実行速度を高速化
- 多くのバグを解決
- HTML に対応

このようなことから，AWT に変わって Swing が標準化されている．

3　Swing のコンポーネントとレイアウト

Swing で GUI を作成する際に使用する，基本的なコンポーネントとレイアウトを紹介する．

3.1　JFrame

Swing を使ってアプリケーションを作成するために, まず JFrame を使ってウィンドウを作成する. そして，そのウィンドウの上にパネルやボタン，ラベルなどを貼り付けて，アプリケーションを作成する．

3.1.1　JFrame の作成

以下のプログラムで，フレームを作成する．

```
import javax.swing.*;

public class MyJFrame extends JFrame{
    public static void main(String[] args){
        MyJFrame test = new MyJFrame("MyJFrame");
        /* 「×」ボタンを押した時に Java アプリケーションを終了 */
        test.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        /* 実際に表示する */
        test.setVisible(true);
    }
        public MyJFrame(String title){
        setTitle(title);
       /* 表示位置を X=10, Y=10, サイズを 300 x 200 に変更する */
        setBounds( 10, 10, 300, 200);
    }
}
```

実行すると，Fig. 1 のようなウィンドウを作成することが出来る．

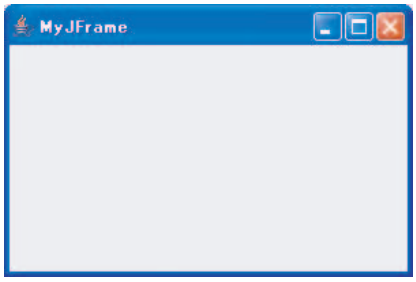

Fig. 1　JFrame (出典：自作)

3.2 JPanel

JPanel はボタンやラベルなどを貼り付けられるものである．ボタンやラベルをフレームに直接貼り付けることもできるが，フレームだけを使うとレイアウトが複雑になってしまう．そこで，ボタンやラベルなどはパネルに貼り付け，パネルをフレームに貼り付けると，より柔軟にレイアウトを設計することができる．

3.2.1 JPanel の作成

以下のプログラムでは，パネルをフレームに貼り付けている．

```
import javax.swing.*;

public class MyJPanel extends JFrame{
   public static void main(String[] args){
   MyJPanel test = new MyJPanel("MyJPanel");
   test.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       test.setVisible(true);
      }

   MyJPanel(String title){
        setTitle(title);
        setBounds( 10, 10, 300, 200);

/************   パネルの追加   ***************/
        /* パネルを作成 */
        JPanel p = new JPanel();
       /* パネルの背景色を青に */
        p.setBackground(Color.blue);
        /* パネルをフレームにはりつける */
        getContentPane().add(p);
/*****************************************/
    }
}
```

実行すると，Fig. 2 のようなウィンドウを作成することが出来る．

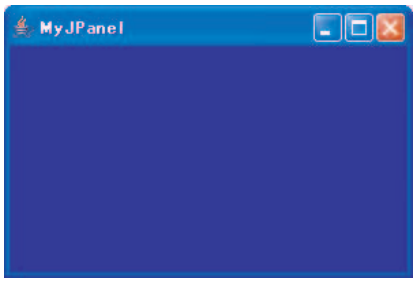

Fig. 2　JPanel (出典：自作)

3.3 JLabel

JLabel は画面上に文字を表示したい場合や画像を表示したい場合に利用するコンポーネントである．表示したい文字列をもったラベルを作成し，パネルに追加して使用する．

3.3.1 JLabel の作成

以下のプログラムでは，ラベルをパネルに追加し，そのパネルをフレームに貼り付けている．このプログラムでは，「Label's Test」という文字列をもったコンストラクタでラベルを作成し，フレームに追加している．

```
import javax.swing.*;
import java.awt.*;

public class MyJLabel extends JFrame{
    public static void main(String[] args){
     MyJLabel test = new MyJLabel("MyJLabel");
     test.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        test.setVisible(true);
      }

    MyJLabel(String title){
        setTitle(title);
        setBounds( 10, 10, 300, 200);

/************    ラベルの追加    ****************/
       /* ラベルの作成 */
        JLabel label = new JLabel("Label's Test");
        JPanel p = new JPanel();
        p.add(label);
        getContentPane().add(p);
/*****************************************/
    }
}
```

実行すると，Fig. 3 に示すように，文字列を表示することが出来る．

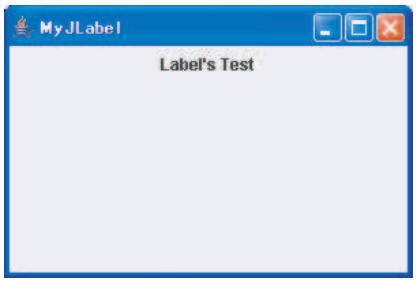

Fig. 3　JLabel (出典：自作)

3.4 JTextField

JTextField はユーザからの文字入力用に利用する．例えば，アンケート画面で住所や名前の入力欄を作る場合などに利用する．ラベルと同様に，パネルに追加して使用する．

3.4.1 JTextField の作成

以下のプログラムでは，テキストフィールドをパネルに追加し，そのパネルをフレームに貼り付けている．

```
import javax.swing.*;

public class MyJTextField extends JFrame{
    public static void main(String[] args){
     MyJTextField test = new MyJTextField("MyJTextField");
        test.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        test.setVisible(true);
    }

    MyJTextField(String title){
        setTitle(title);
        setBounds( 10, 10, 300, 200);

/************    テキストフィールドの追加    **************/
        /* テキストの文字桁を 15 に */
        JTextField text = new JTextField("Test", 15);
        JPanel p = new JPanel();
        p.add(text);
        getContentPane().add(p);
/*********************************************************/
    }
}
```

実行すると，Fig. 4 のような文字入力画面を作成することが出来る．

Fig. 4　JTextField (出典：自作)

3.5 JButton

JButton は，ユーザからの操作を受けつけるためのコンポーネントであり，フレームやパネルに貼り付けて使用する．

3.5.1 JButton の作成

以下のプログラムでは，ボタンをパネルに追加し，そのパネルをフレーム貼り付けている．

```
import javax.swing.*;

public class MyJButton extends JFrame{
    public static void main(String[] args){
     MyJButton test = new MyJButton("MyJButton");
        test.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        test.setVisible(true);
    }

    MyJButton(String title){
        setTitle(title);
        setBounds( 10, 10, 300, 200);

/************   ボタンの追加   **************/
        /* ボタンの作成 */
        JButton btn = new JButton("Button");
        JPanel p = new JPanel();
        p.add(btn);
        getContentPane().add(p);
/*****************************************/
    }
}
```

実行すると，Fig. 5 のようなボタンを作成することが出来る．

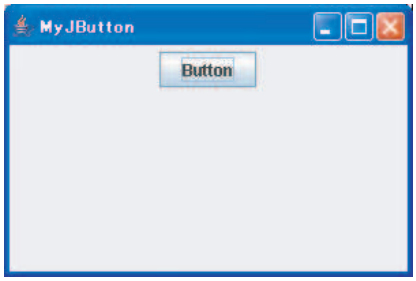

Fig. 5　JButton (出典：自作)

3.6 JCheckBox

JCheckBox は，ある項目に対して,「はい」または「いいえ」の選択をするときに使うボタンであり，フレームやパネルに貼り付けて使用する．

3.6.1 JCheckBox の作成

以下のプログラムでは，チェックボックスをパネルに追加し，そのパネルをフレームに貼り付けている．

```
import javax.swing.*;

public class MyJCheckBox extends JFrame{
    public static void main(String[] args){
        MyJCheckBox test = new MyJCheckBox("MyJCheckBox");
        test.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        test.setVisible(true);
    }

    MyJCheckBox(String title){
        setTitle(title);
        setBounds( 10, 10, 300, 200);

/************   チェックボックスの追加   **************/
       /* チェックボックスの作成 */
        JCheckBox check1 = new JCheckBox("House");
        JCheckBox check2 = new JCheckBox("Car");
        JCheckBox check3 = new JCheckBox("TV & Audio");
        JCheckBox check4 = new JCheckBox("PC");

        JPanel p = new JPanel();

        p.add(check1);
        p.add(check2);
        p.add(check3);
        p.add(check4);

        getContentPane().add(p);
/*****************************************************/
    }
}
```

実行すると，Fig. 6 のようなチェックボックスを作成することが出来る．

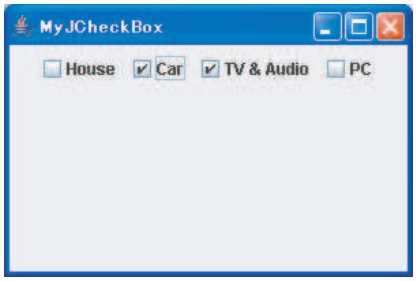

Fig. 6　JCheckBox (出典：自作)

3.7 JRadioButton

JRadioButton は，複数の選択肢の中から，1 つの項目を選択するときに使うボタンであり，フレームやパネルに貼り付けて使用する. ラジオボタンを使用する際は，複数の選択肢をグループ化するために，ButtonGroup というコンポーネントと組み合わせて使う点に注意する.

3.7.1 JRadioButton の作成

以下のプログラムでは，ラジオボタンをパネルに追加し，そのパネルをフレームに貼り付けている.

```
import javax.swing.*;

public class MyJRadioButton extends JFrame{
    public static void main(String[] args){
     MyJRadioButton test = new MyJRadioButton("MyJRadioButton");
        test.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        test.setVisible(true);
    }

    MyJRadioButton(String title){
        setTitle(title);
        setBounds( 10, 10, 300, 200);

/************   ラジオボタンの追加   **************/
        /* ラジオボタンの作成 */
        JRadioButton radio1 = new JRadioButton("House");
        JRadioButton radio2 = new JRadioButton("Car");
        JRadioButton radio3 = new JRadioButton("TV & Audio");
        JRadioButton radio4 = new JRadioButton("PC");

        /* ボタングループの作成 */
        ButtonGroup group = new ButtonGroup();

        /* ボタングループにラジオボタンを追加 */
        group.add(radio1);
        group.add(radio2);
        group.add(radio3);
        group.add(radio4);

        JPanel p = new JPanel();

        p.add(radio1);
        p.add(radio2);
        p.add(radio3);
        p.add(radio4);

        getContentPane().add(p);
/************************************************/
    }
}
```

実行すると，Fig. 7 のようなラジオボタンを作成することが出来る.

Fig. 7　JRadioButton (出典：自作)

3.8 BorderLayout

BorderLayout とは，コンポーネントを上下左右・中央に配置するためのレイアウトマネージャである．Java の GUI プログラミングでは，このようなレイアウトマネージャを用いることによって，効果的なレイアウトを実現している．ここでは，レイアウトマネージャの中で最も基本となる BorderLayout について述べる．

3.8.1 BorderLayout

以下のプログラムでは，ボタンを 5 つ作成し，それらをボーダーレイアウトを指定したパネルに追加し，そのパネルをフレームに貼り付けている．

```
mport javax.swing.*;
import java.awt.*;

public class MyBorderLayout extends JFrame{
    public static void main(String[] args){
     MyBorderLayout test = new MyBorderLayout("MyBorderLayout");
        test.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        test.setVisible(true);
    }

    MyBorderLayout(String title){
        setTitle(title);
        setBounds( 10, 10, 300, 200);

        JButton btn1 = new JButton("North");
        JButton btn2 = new JButton("South");
        JButton btn3 = new JButton("West");
        JButton btn4 = new JButton("East");
        JButton btn5 = new JButton("Center");

        JPanel p = new JPanel();

/************   ボーダーレイアウトによるボタンの配置    **************/
        /* パネルにおいてボーダーレイアウトを設定 */
        p.setLayout(new BorderLayout());

        p.add(btn1,BorderLayout.NORTH);
        p.add(btn2,BorderLayout.SOUTH);
        p.add(btn3,BorderLayout.WEST);
        p.add(btn4,BorderLayout.EAST);
        p.add(btn5,BorderLayout.CENTER);

        getContentPane().add(p);
/*********************************************************************/
    }
}
```

実行すると，Fig. 8 に示すようにボタンの配置をレイアウトすることが出来る．

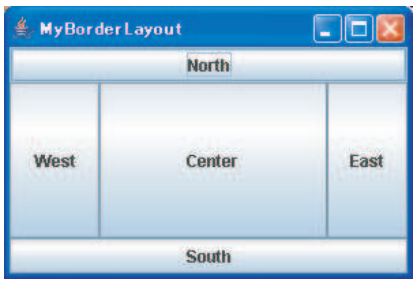

Fig. 8　BorderLayout (出典：自作)

4　Visual Editor

4.1　Visual Editor とは

VisualEditor(VE) とは，Swing と SWT(Standart Widget Toolkit) の GUI を作成するための Eclipse プラグインである．VE を使用すると，Microsoft の VisualBasic のように，パレット (ツールボックス) に並んでいるコンポーネントをフレームに貼り付けるという方法で，容易かつ視覚的に GUI を作成することができる．

4.2　Visual Editor のダウンロード方法

VE に必要な媒体は、VE 自体も含めて以下の 4 つである．(バージョンは 2006 年 5 月の最新版である．)

1. Eclipse3.0.1
   オープンソースの統合ソフトウェア開発環境の一つである．

2. EMF(Eclipse Modeling Framework2.0.1)
   モデリングのフレームワークとコードの生成機能を Eclipse に実装する技術である．EMF のダウンロードは，参考文献 3 の URL にアクセスし，Fig. 9 に示した場所より emf-sdo-xsd-SDK-2.1.2.zip をダウンロードする．次に zip ファイルを解凍し，そのフォルダの中に入っている plugin と features を自分の Eclipse フォルダ内の plugin と features に上書きする．

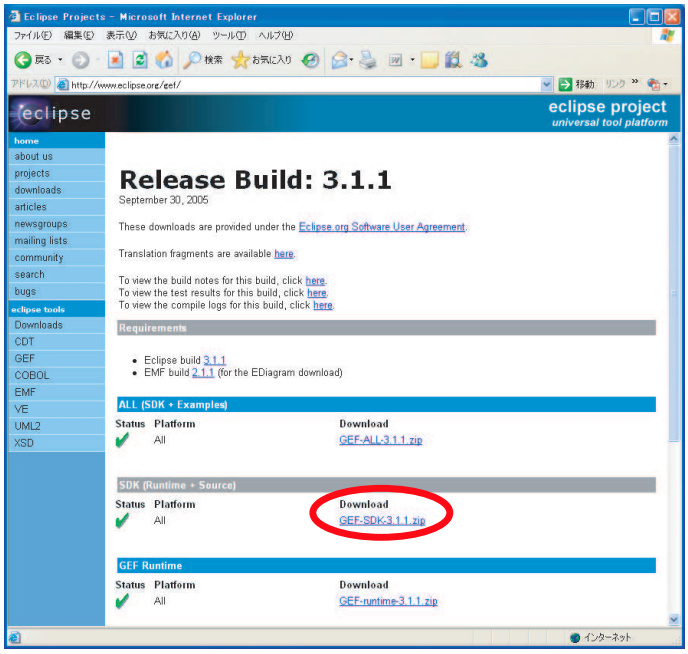

Fig. 9　EMF のダウンロード画面 (出典：自作)

3. GEF(Graphical Editing Framework3.0.1)
モデルをグラフィカルに編集するアプリケーションを作成する為に使用するフレームワークのことである．GEF のダウンロードは，参考文献 4 の URL にアクセスし，Fig. 10 に示した場所より GEF-SDK-3.1.1.zip をダウンロードする．次に zip ファイルを解凍し，そのフォルダの中に入っている plugin と features を自分の Eclipse フォルダ内の plugin と features に上書きする．

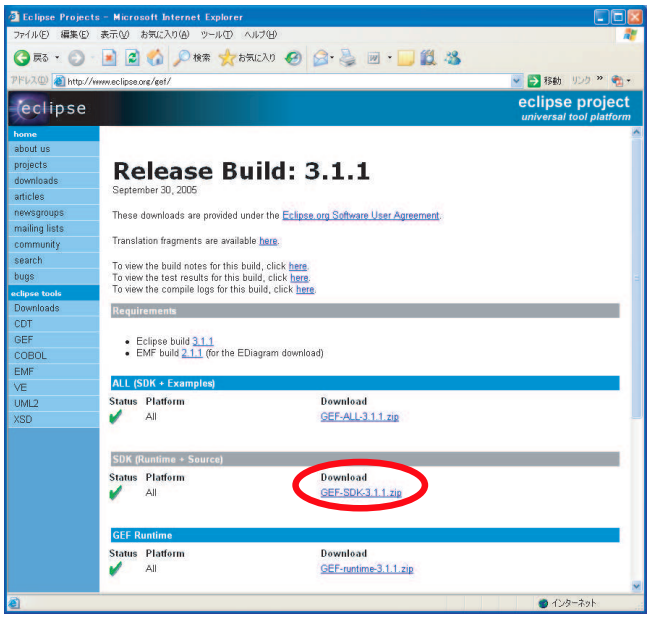

Fig. 10　GEF のダウンロード画面 (出典：自作)

4.　Visual Editor1.0
VE 本体である．VE のダウンロードは，参考文献 5 の URL にアクセスし，Fig. 11 に示した場所より VE-SDK-1.1.0.1.zip と VE-runtime-1.x.zip のどちらかをダウンロードする．次に zip ファイルを解凍し，そのフォルダの中に入っている plugin と features を自分の Eclipse フォルダ内の plugin と features に上書きする．

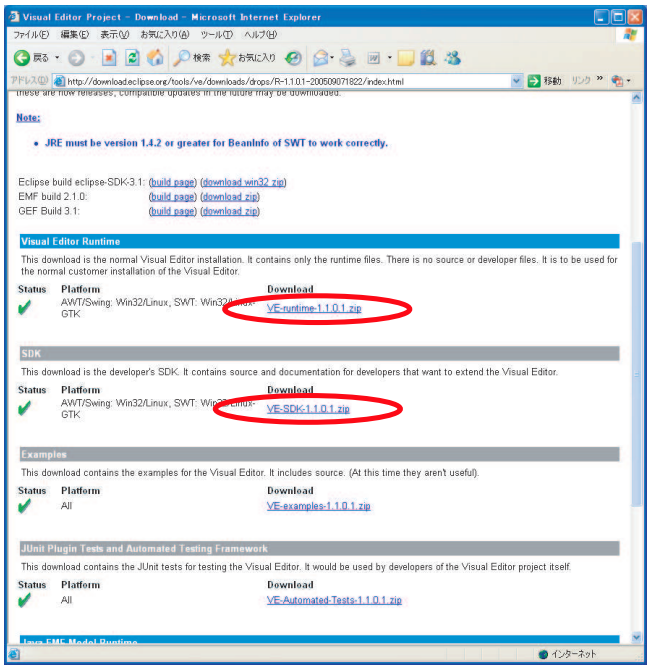

Fig. 11　VE のダウンロード画面 (出典：自作)

5　Visual Editor の利用

5.1　Visual Editor の画面構成

本項では，Visual Editor の画面構成について述べる．Visual Editor には，サンプルアプリケーションがいくつか含まれているので，これを起動して画面構成を確認する．

Eclipse を起動し，「ファイル」→「新規」→「サンプル」を選択する (Fig. 12).

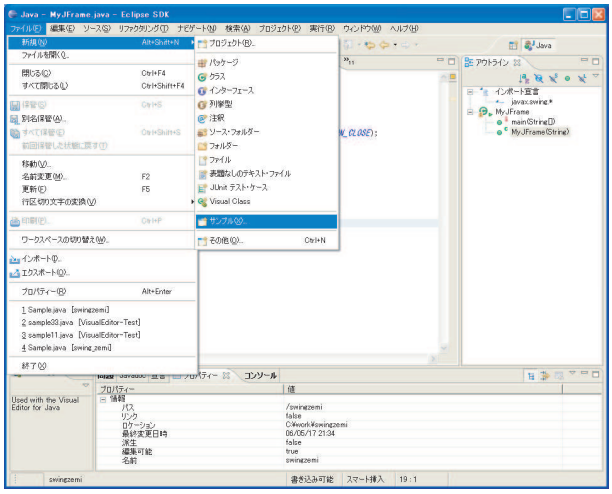

Fig. 12　サンプルアプリケーションの起動 1 (出典：自作)

次に，表示されたダイアログのツリーから「サンプル」→「Swing」→「BasicSwingComponents」を選択し，「次へ」をクリックする (Fig. 13).

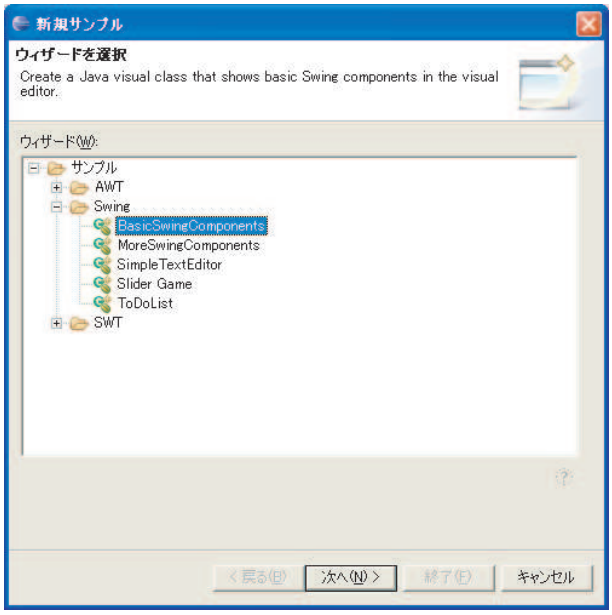

Fig. 13　サンプルアプリケーションの起動 2 (出典：自作)

Fig. 14 のような画面が表示されるので，パッケージの部分に「sample」と入力し，「終了」をクリックする．すると，サンプルアプリケーションが起動する．Visual Editor を使用する際に知っておくべき画面構成を Fig. 15 に示す.

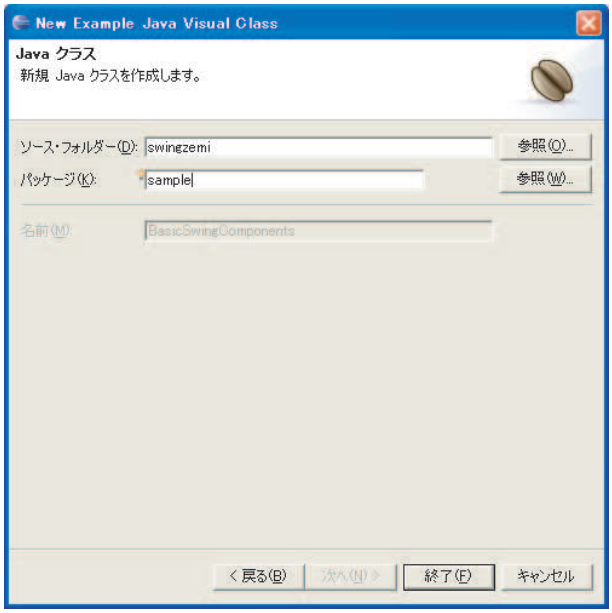

Fig. 14　パッケージ名の入力画面 (出典：自作)

1. フォームデザイナ
   GUI コンポーネントを使った画面が表示される場所である．基本的にはこの場所にコントロールパレットからコンポーネントを貼り付けていくことにより GUI アプリケーションを作成する．

2. エディタビュー
   プログラムのソースコードが表示されるビューである．この部分でプログラムを直接編集し，その変更を即座にフォームデザイナに反映することができる．

3. プロパティービュー
   各コンポーネントのプロパティを設定するためのビューである．ここでは必要に応じてコンポーネントの形やレイアウトなどの詳細を設定する．

4. コントロールパレット
   Swing のコンポーネントやコンテナが並んだパレットである．ここから使用したいコンポーネントを選択し，フォームデザイナにドラッグして任意の位置，任意の大きさで配置することができる．

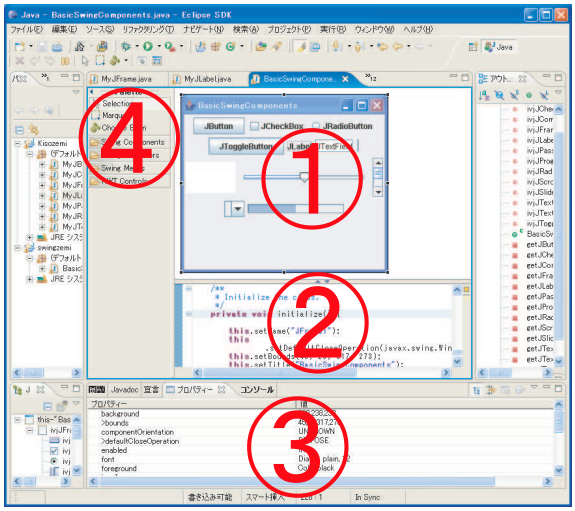

Fig. 15　Visual Editor の画面構成 (出典：自作)

5.2 Visual Editor を用いたアプリケーションの作成

Eclipse を起動し,「ファイル」→「新規」→「プロジェクト」を選択すると，ウィザードの選択画面が現れるので,「Java プロジェクト」を選択して「次へ」をクリックする．プロジェクトの作成画面が現れるので，Fig. 16 に示すようにプロジェクト名を「swingzemi」と入力し,「終了」をクリックする．

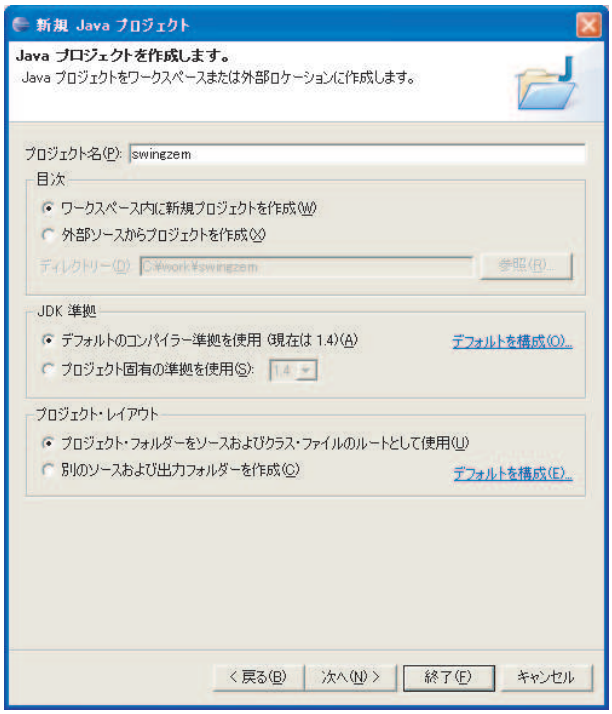

Fig. 16　プロジェクトの作成画面 (出典：自作)

次に，作成した「swingzemi」プロジェクトの上で右クリックし,「新規」→「その他」を選択する．そして，表示されたダイアログのツリーから「Java」→「Swing」→「JFrame Visual Class」を選択する．すると，Fig. 17 に示したクラス名入力画面が表示されるので，クラス名「Sample」を入力する．また，ここで public static void main と継承のチェックボックスにチェックが入っているのを確認し,「終了」をクリックする．これで Fig. 18 に示すようにフレームが作成される．

Sample.java には main メソッドが生成されるが，デフォルトは空である．そのため，Fig. 19 に示すようにエディタビューで main メソッド内に「new Sample();」と記述して main を実装し，動作確認がとれるようにしておく．

次に，Fig. 20 に示すようにフォームデザイナのフレームをクリックして選択し，プロパティビューで「visible」プロパティの値を true，また,「defaultCloseOperation」プロパティーの値を EXIT に設定して，×ボタンクリックでアプリケーションが終了するようにする．さらに，フォームデザイナのパネルをクリックして「layout」プロパティの値を null にして，コンポーネントを任意の位置に配置できるようにする．

5.3 ボタンの作成

ボタンは，Fig. 21 に示すように「パレットビュー」→「SwingComponents」→「Button」を選択し，フレーム上に左クリックでドラッグすることにより作成する．アクションリスナーを追加したい場合は，Fig. 22 に示すように表示されているボタンの上で右クリックし,「Events」→「Add Events」をクリックし，追加したいイベントを選択して「終了」をクリックする．

5.4 テキストフィールドの作成

テキストフィールドもボタンと同じように，Fig. 23 に示すようにパレットビューから選択して作成する．

5.5 ラベルの作成

ラベルもボタンと同じように，Fig. 24 に示すようにパレットビューから選択して作成する．

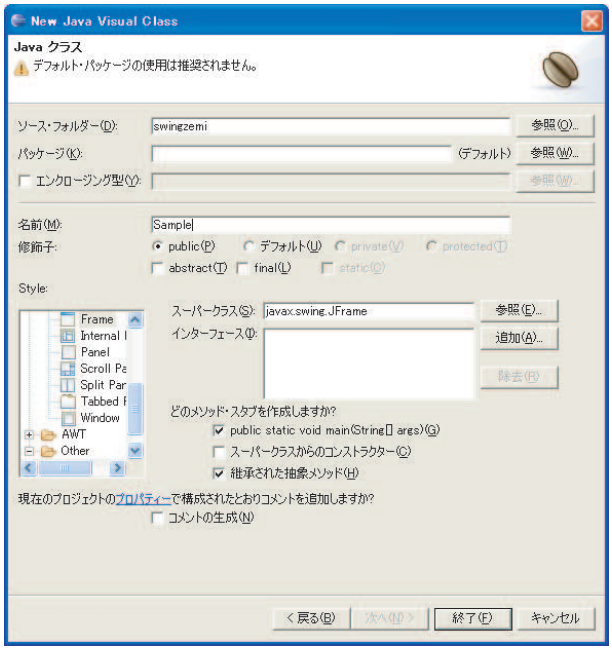

Fig. 17　クラスの作成 (出典：自作)

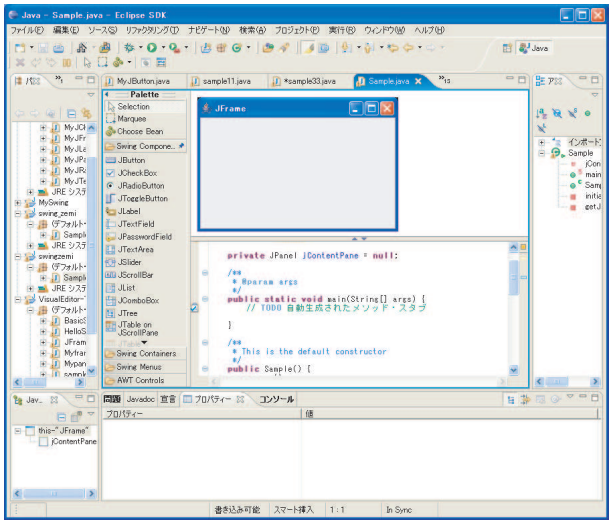

Fig. 18　フレームの作成 (出典：自作)

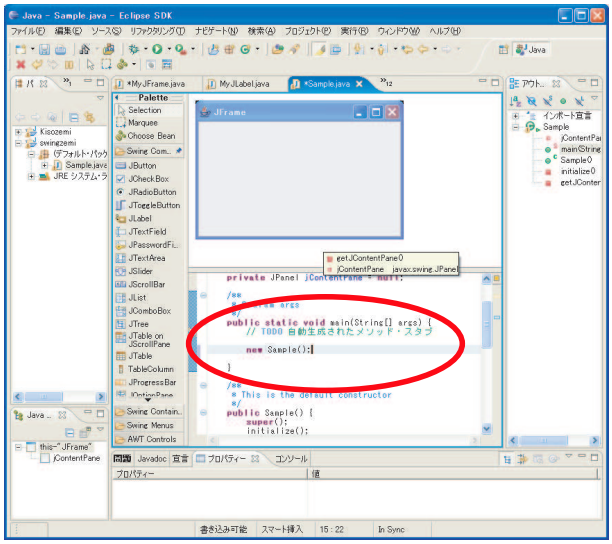

Fig. 19　main メソッドの実装 (出典：自作)

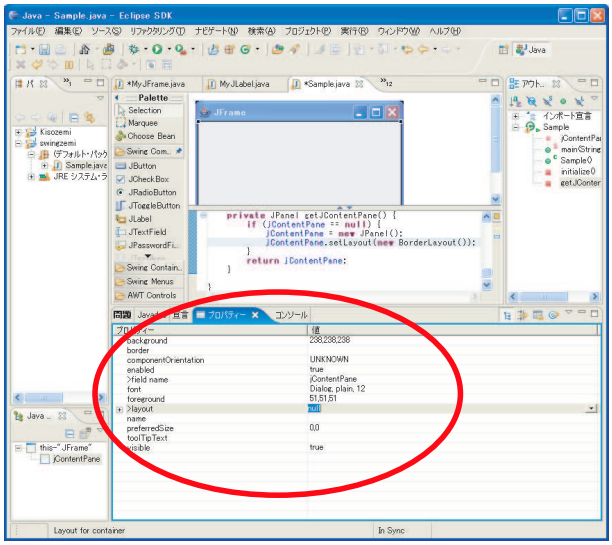

Fig. 20　プロパティの値の変更 (出典：自作)

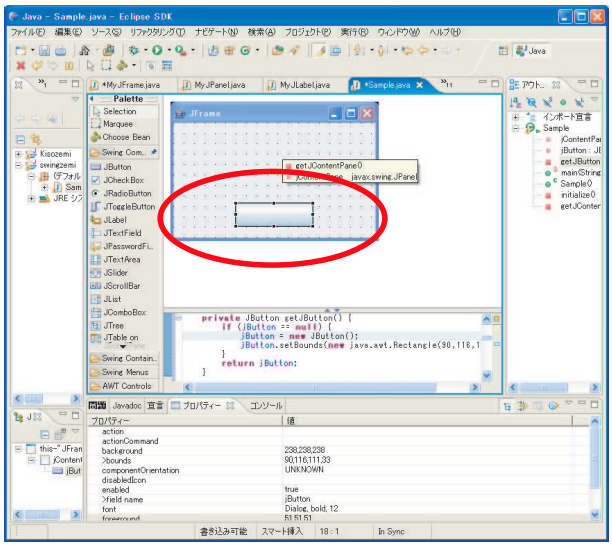

Fig. 21　ボタンの作成 (出典：自作)

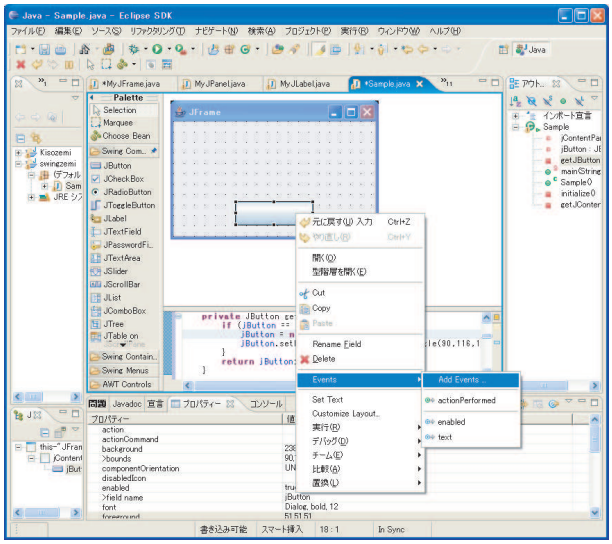

Fig. 22　アクションリスナーの追加 (出典：自作)

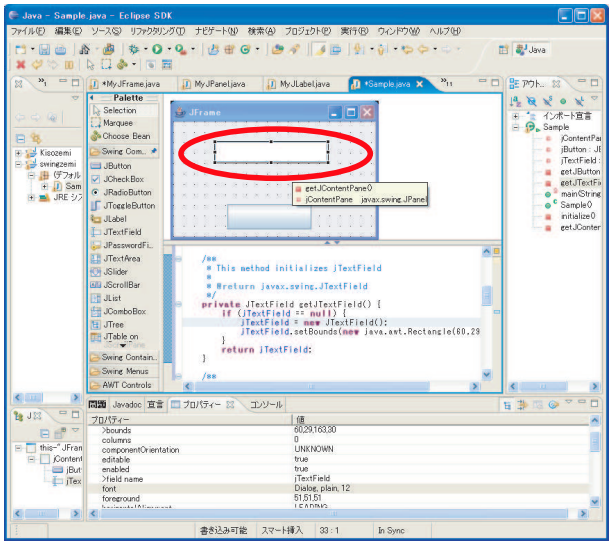

Fig. 23　テキストフィールドの作成 (出典：自作)

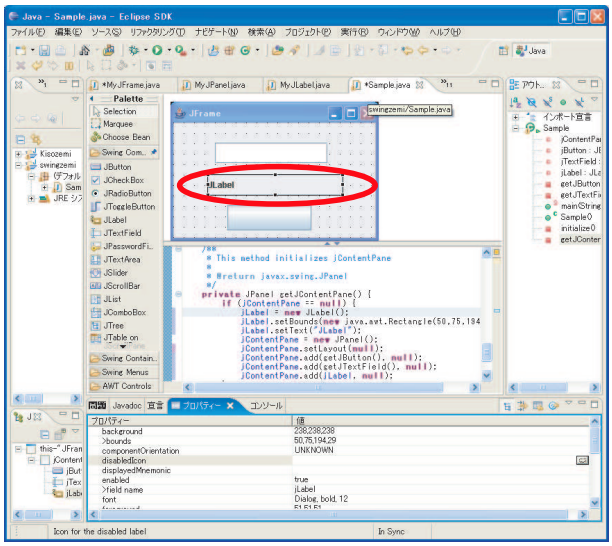

Fig. 24　ラベルの作成 (出典：自作)

5.6　チェックボックスの作成

チェックボックスもボタンと同じように，Fig. 25 に示すようにパレットビューから選択して作成する．

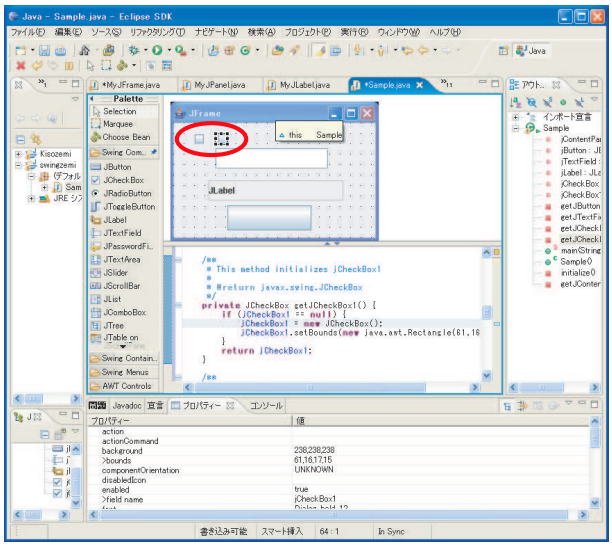

Fig. 25　ラベルの作成 (出典：自作)

5.7　ラジオボタンの作成

ラジオボタンもボタンと同じように，パレットビューから選択して作成する．だだし，ボタンのグループ化を行う場合は，エディタビューで直接プログラムの変更を行う．具体的には，getJContentPane メソッド内に，以下のコードを追加する．また，javax.swing.ButtonGroup クラスをインポートする．

```
import javax.swing.ButtonGroup;
```

```
ButtonGroup buttonGroup = new ButtonGroup();
buttonGroup.add(jRadioButton);
buttonGroup.add(jRadioButton1);
```

Fig. 26 に示すように，ラジオボタンを作成することができる．

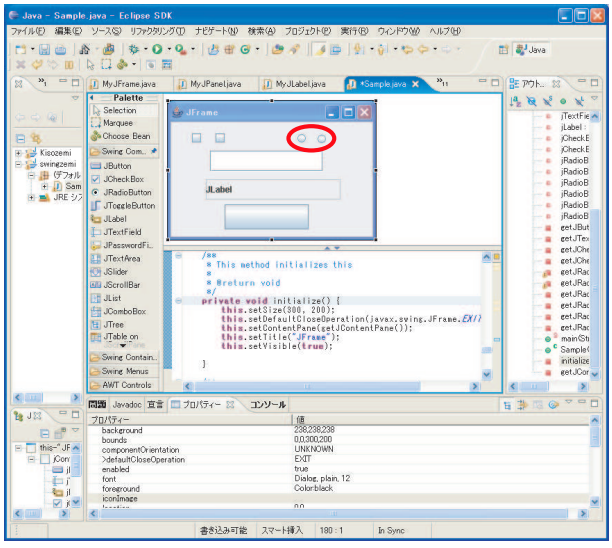

Fig. 26　ラジオボタンの作成 (出典：自作)

参考文献

1) javadrive
   http://www.javadrive.jp/

2) Visual Editor メモ
   http://www.okisoft.co.jp/esc/Eclipse3/visualeditor/index.html

3) emf-sdo-xsd-SDK-2.1.2.zip
   http://download.Eclipse.org/tools/emf/scripts/downloads.php?s=2.1.2/R200601191349

4) GEF-SDK-3.1.1.zip
   http://www.Eclipse.org/gef/

5) VE-SDK-1.1.0.1.zip
   http://download.Eclipse.org/tools/ve/downloads/drops/R-1.1.0.1-200509071822/index.html