Swingによるデスクトップアプリケーション開発（2進10進変換器）-JavaSE1.8

office・M
2024年10月4日 17:08

¥1,000 •••

Java8のSwing環境でデスクトップアプリケーションの開発方法を学ぶ講座をシリーズで提供しています。今回はWindowBuilder（Swingデザイナー）を使って2進数と10進数の相互変換を行う計算器アプリケーションを作成します。

2024年9月よりECLIPSEのバージョンを最新版（Version: 2024-06 (4.32.0)）に変更しました。

▼ 目次

外部設計

WindowBuilderのSwingデザイナーで図1のようなGUIを作成します。

図1. BinaryDecimalアプリケーションのGUI画面

内部設計

処理ロジック

変換ボタンをクリックすることでアクションイベントが発生し、ラジオボタンで指定された変換処理ロジックを実装したクラスが起動されます。

作成するクラス仕様
作成する**2進10進変換計算ロジック（BinaryDecimalBean.class）**の使い方とクラス図は以下のようになります。このクラスは通常のクラスとして作成してもかまいませんが、Webアプリで利用するなど後々の再利用を考慮して**JavaBeansの仕様**とします。

（仕様）
属性 (フィールド):
・binary: 2進数の値を保持する文字列 (String)
・decimal: 10進数の値を保持する文字列 (String)
コンストラクタ:
・BinaryDecimalBean(): フィールドを初期化するデフォルトコンストラクタ
メソッド:
・binaryDecimal(): binary の値を decimal に変換する
・decimalBinary(): decimal の値を binary に変換する
・setBinary(binary: String): binary 属性を設定する setter メソッド
・setDecimal(decimal: String): decimal 属性を設定する setter メソッド
・getBinary(): String: binary 属性を取得する getter メソッド
・getDecimal(): String: decimal 属性を取得する getter メソッド

（クラス図）

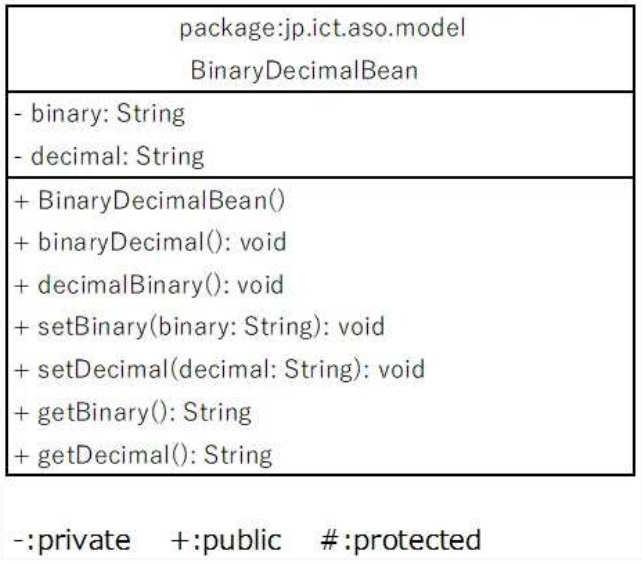

図2．BinaryDecimalBeanクラス図

実装準備

プロジェクトの作成

Eclipseのメニューバーより
ファイル→新規→Javaプロジェクト→「SwingBinaryDecimal」プロジェクトを作成→図３の内容で設定する

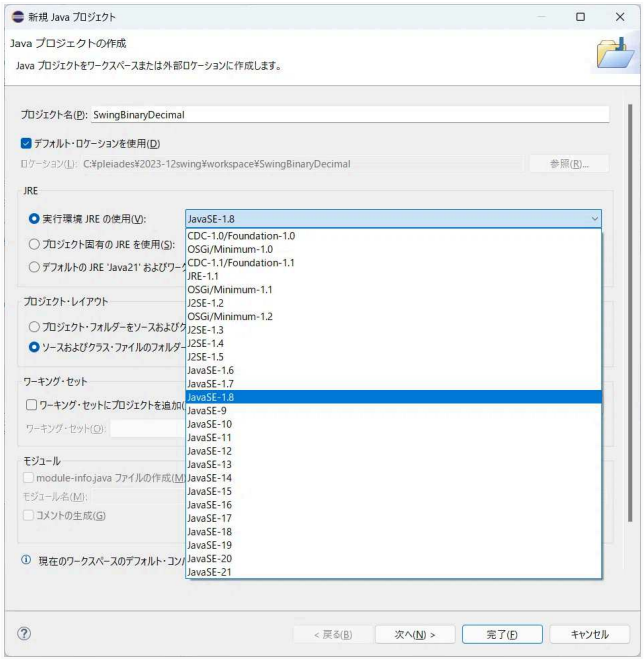

図3．SwingBinaryDecimalプロジェクト設定

※作成済みであればこの処理は必要ありません。

以下画面のスクリーンショットはライトテーマで取得します。
（ライトテーマの設定方法）
Eclipseのメニューバーより
　ウィンドウ → 設定 → 一般 → 外観 → ルック＆フィール → ライト
　→ 適用して閉じる → Eclipseの再起動がかかります

実装

Bean（クラス）の作成

Eclipseパッケージ・エクスプローラより
SwingBinaryDecimalプロジェクトを右クリック→新規→クラス
→以下の内容で作成

・BinaryDecimalBean.java
　　パッケージ：jp.ict.aso.model
　　名前：BinaryDecimalBean

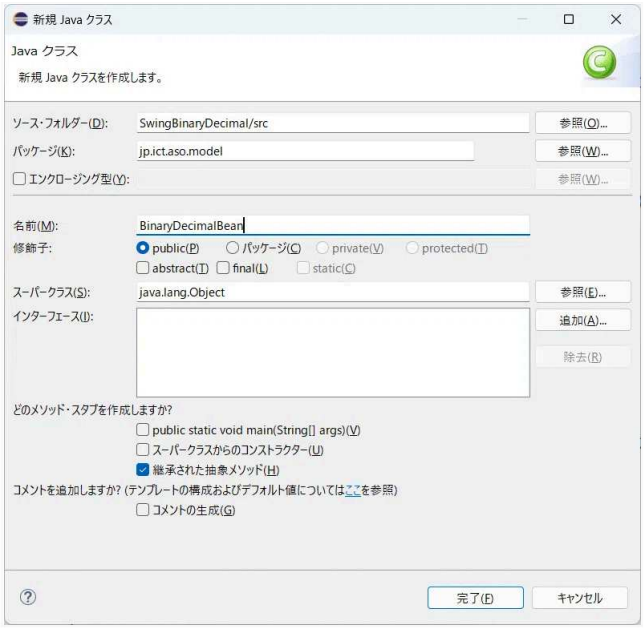

図4．BinaryDecimalBeanの作成

ソースコード：**図2のクラス図**のように作成します
　　変換計算ロジックのメソッドはヒントを参考にしてください

```
【ヒント】
    public void binaryDecimal() {
        //2進10進変換
        int decimalInt = Integer.parseInt(binary, 2);
        decimal=String.valueOf(decimalInt);
    }
    public void decimalBinary() {
        //10進2進変換
        int decimalInt=Integer.parseInt(decimal);
        binary = Integer.toBinaryString(decimalInt);
    }
```

ひな形の作成

WindowBuilderを用いてSwingアプリケーションのスケルトン（骨格）を自動生成させます。

Eclipseパッケージ・エクスプローラより
SwingBinaryDecimalプロジェクトを右クリック→新規→その他
→ WindowBuilder → Swingデザイナー → JFrameを選択 → 次へ

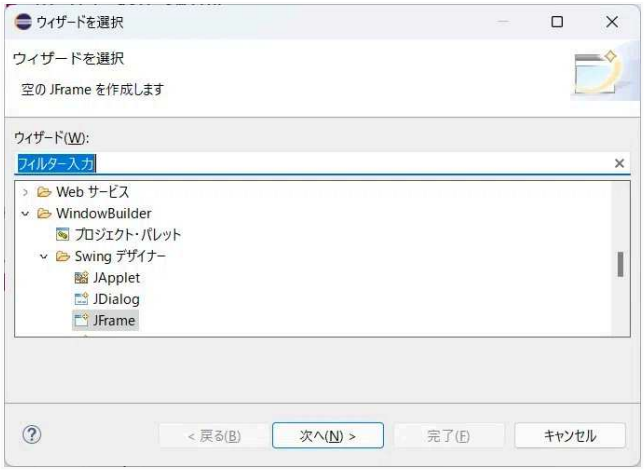

図5．JFrameウィザードの選択

以下の内容で作成

パッケージ：jp.ict.aso.swing

名前：BinaryDecimal

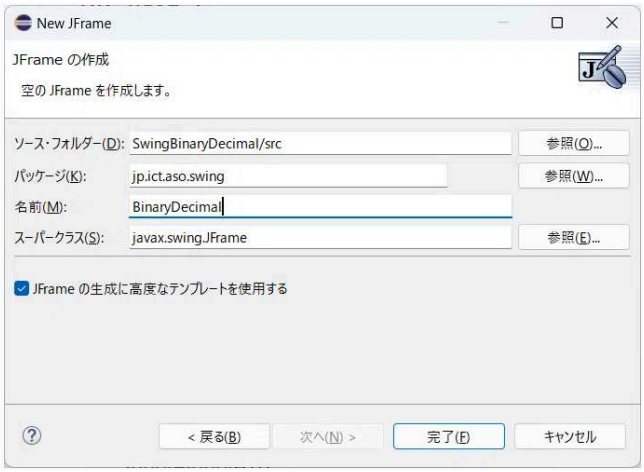

図6．Swingアプリケーションの生成

GUI実装

自動生成されたプログラム（スケルトン）からGUIのデザインを実装します。

パレットと構造（コンポーネント、プロパティ）のViewを利用するのがコツです。デザインイメージは設定反映の参考としてとらえた方が良いでしょう。

①画面中央下部にあるデザインタブでソースコード編集画面からSwingデザイナーに切り替えます。

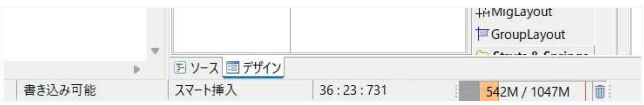

図7．Swingデザイナーに切り替え

②contentPaneのLayoutプロパティをBorderLayoutに設定します。

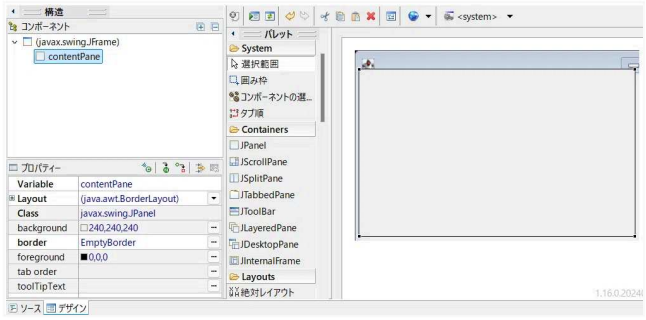

図8．コンテンツペインの設定

③contentPaneの「North」の位置にGUI部品のJPanelをパレットから配置します。JPanelのLayoutプロパティはFlowLayoutのままです。

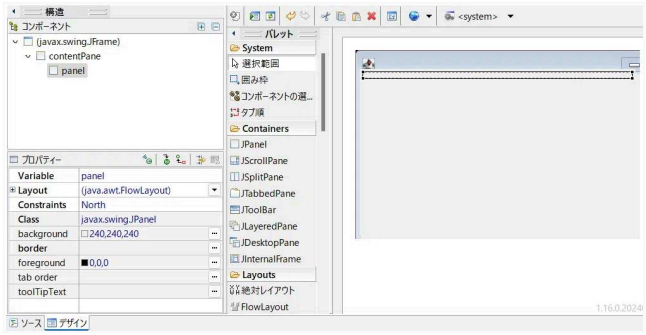

図9．JPanelの設定

④JPanelにGUI部品のJRadioButton、JTextFieldを順番にパレットから配置します。JRadioButtonのtextプロパティを図10のように変更します。

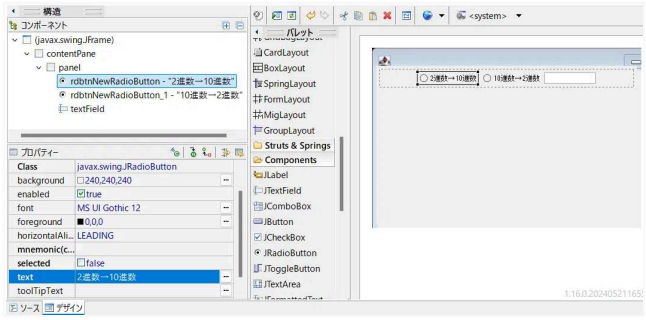

図10．ラジオボタンとテキストフィールドの配置及びテキストの設定

⑤contentPaneの「South」の位置にGUI部品のJPanelをパレットから配置します。

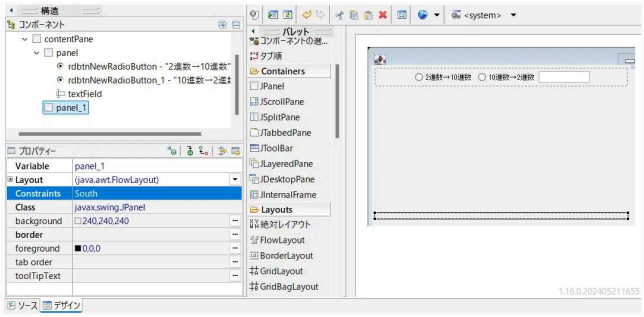

図11．South位置にパネルを配置

⑥contentPaneの「South」の位置のJPanelにGUI部品のJButtonを２つパレットから配置します。あわせてtextプロパティもそれぞれ「変換」と「クリア」に変更します。

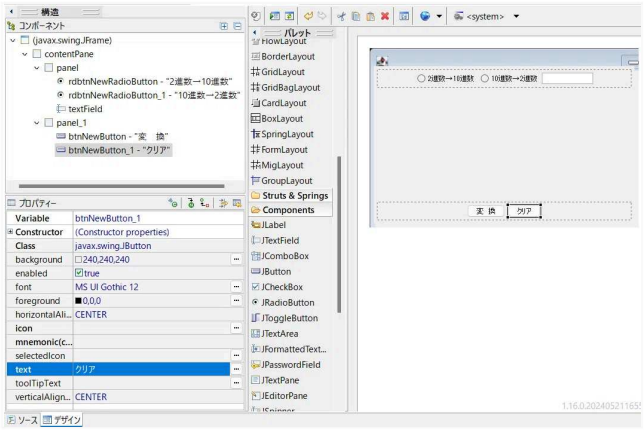

図11．South位置のパネルにボタン配置

⑦ボタンにイベントリスナーを対応付けます。パレットのSwingActions内にある「新規」のリスナーを選択してボタンをクリックすることで対応付けられます。

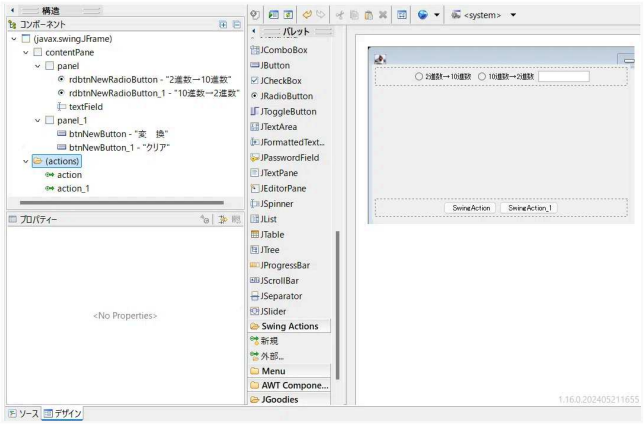

図12．イベントリスナーの対応付け

⑧contentPaneの「Center」の位置にGUI部品のJPanelをパレットから配置します。

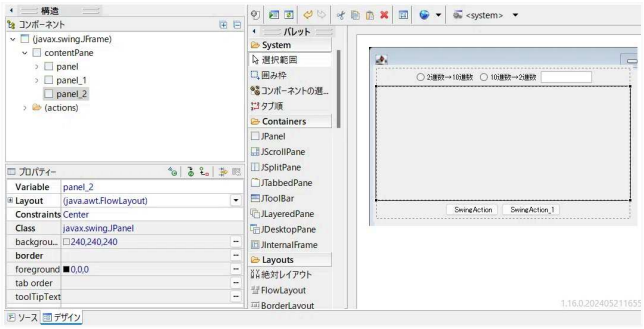

図13．Center位置にパネルを配置

⑨contentPaneの「Center」の位置のJPanelにGUI部品のJScrollPaneを配置しプロパティのholizontalScrollBarPolicyとverticalScrollBarPolicyをそれぞれ
　HORIZONTAL_SCROLLBAR_ALWAYS
　VERTICAL_SCROLLBAR_ALWAYS
に変更します。

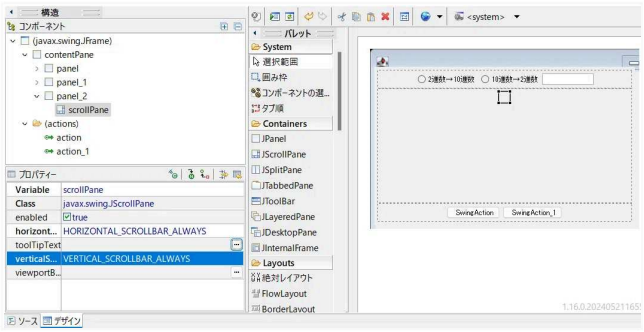

図13．Center位置にスクロールバーを配置

⑩JScrollPane内にJTextAreaを配置します。このとき**デザインイビュー内でViewportの位置**に配置します。

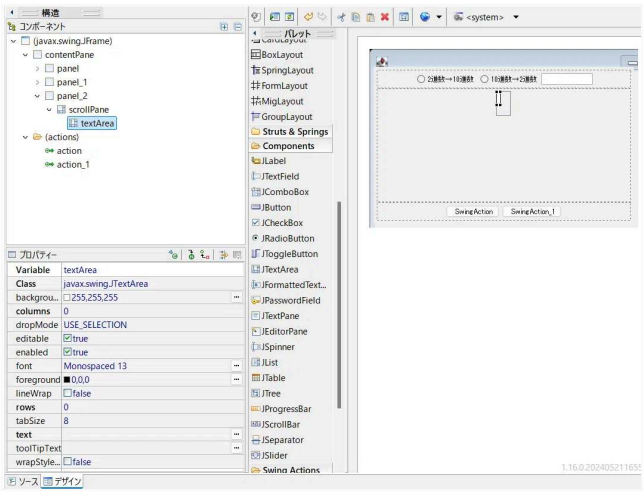

図14．スクロール可能なテキストエリアを配置

⑪JTextAreaのプロパティーでcolumsを30にrowsを8に設定しtextに変換結果と入力します。

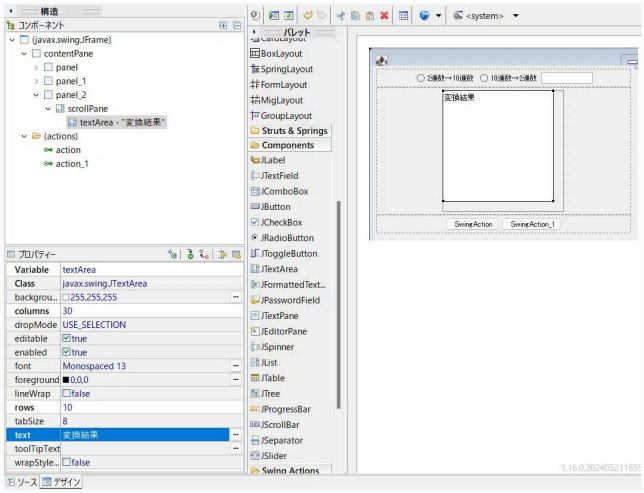

図15．テキストエリアのプロパティ設定

イベント実装

ソースタブに変更します。

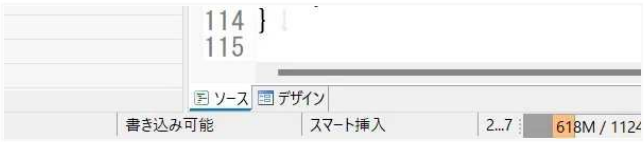

図16．ソースタブに変更

①「変換ボタン」のイベントのソース部分を変更します。

```
private class SwingAction extends AbstractAction {
    public SwingAction() {
        putValue(NAME, "変　換");
        putValue(SHORT_DESCRIPTION, "2進←→10進相互変換を行います");
    }
    public void actionPerformed(ActionEvent e) {
        if(rdbtnNewRadioButton.isSelected()) {
            //2進10進変換
            BinaryDecimalBean bdb = new BinaryDecimalBean();
            bdb.setBinary(textField.getText()); //2進の値セット
            bdb.binaryDecimal();                //計算
            textArea.append(String.valueOf(bdb.getDecimal())+"¥n");//表示
        }
        if(rdbtnNewRadioButton_1.isSelected()) {
            //10進2進変換
            BinaryDecimalBean bdb = new BinaryDecimalBean();
            bdb.setDecimal(textField.getText()); //10進の値セット
            bdb.decimalBinary();                 //計算
            textArea.append(String.valueOf(bdb.getBinary())+"¥n");//表示
        }
    }
}
```

図17．変換ボタンのイベント内容

②「クリアボタン」のイベントのソース部分を変更します。

```
private class SwingAction_1 extends AbstractAction {
    public SwingAction_1() {
        putValue(NAME, "クリア");
        putValue(SHORT_DESCRIPTION, "表示領域をクリアします");
    }
    public void actionPerformed(ActionEvent e) {

        txtrn.setText(null);

    }
}
```

図18．クリアボタンのイベント内容

③フィールド変数を変更します。//kokoの部分を追加します。

```
public class BinaryDecimal extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField textField;
    private final Action action = new SwingAction();
    private final Action action_1 = new SwingAction_1();

    private JRadioButton rdbtnNewRadioButton;   //koko
    private JRadioButton rdbtnNewRadioButton_1; //koko
    private JTextArea textArea;                 //koko

    /**
     * Launch the application.
     */
```

図19．フィールド変数の変更

④ローカル変数の宣言になっている部分を変更します。//kokoの部分を変更します。

```
    JPanel panel = new JPanel();
    contentPane.add(panel, BorderLayout.NORTH);

    rdbtnNewRadioButton = new JRadioButton("2進数→10進数");    //koko
    panel.add(rdbtnNewRadioButton);

    rdbtnNewRadioButton_1 = new JRadioButton("10進数→2進数"); //koko
    panel.add(rdbtnNewRadioButton_1);
```

図20．ローカル変数の宣言変更

```
    textArea = new JTextArea();     //koko
    textArea.setText("変換結果");
    textArea.setRows(10);
    textArea.setColumns(30);
    scrollPane.setViewportView(textArea);
```

図21．ローカル変数の宣言変更

⑤実行確認します。エディタの画面内で右クリック → 実行 → Javaアプリケーションで実行されます。

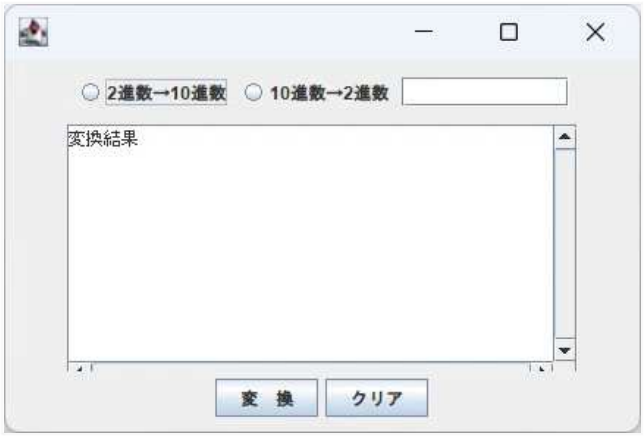

図22．起動画面

⑥テキストフィールドに100を入力してそれぞれの変換結果が表示されるか確認します。

図23．変換結果

変換結果は問題ないようですが、表示にいろいろと不具合があるようです。
とりあえず「タイトルがない」「画面の大きさが任意に変えられてしまう」「ラジオボタンが同時に押されてしまう」「ラジオボタンはデフォルトで10進→2進を選択しておく」「テキストエリア内の変換結果の文字列の表示の後に改行が必要」の４点を修正します。

実装変更

①フレームにタイトルを追加します。
②画面（フレーム）の大きさを変更して固定します。
③ラジオボタンは一つだけ選択できるようにグループ化します。
④ラジオボタンのデフォルト選択を設定します。
⑤テキストエリア内の文字列の改行を指定します。

⑥実行して動きを確認します。これで完成しましたが、**エラー処理は実装されていません**。

図24．2進10進相互変換器アプリーケーション完成

エラー処理

2進数10進数の相互変換可能な数値はint型の取りうる範囲とします。Javaのint型は**32ビット**になるので**10進数では-2147483648～2147483647**の値を変換することが可能になります。また、Javaでは負数を表現するために先頭ビットが符号ビットとなるため**2進数は31ビット以内で入力**することにします（2進数で負の数の変換は考慮しません）。

この範囲を超えた入力でエラーメッセージを出力するようプログラムを変更してください。当然、数字以外の入力は不可です。

入力チェックのクラス **BinaryValidator.java** と **DecimalValidator.java**を作成して、判断します。

BinaryValidatorクラス
（仕様）
入力された文字列が**31ビット以内の2進数**かどうかを判定します。
@param binary 入力された2進数を表す文字列
@return true なら有効な31ビット以内の2進数、false なら無効な値

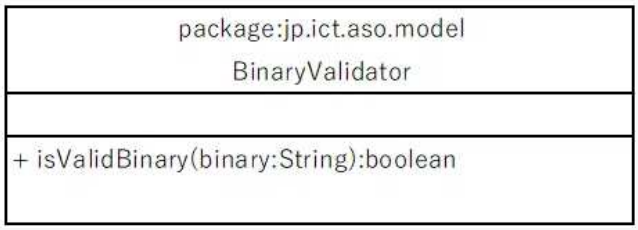

図25．BinaryValidatorクラス図

DecimalValidatorクラス
（仕様）
入力された文字列が**-2147483648～2147483647の範囲にある10進数**かどうかを判定します。
@param decimal 入力された10進数を表す文字列
@return true なら有効な範囲の10進数、false なら無効な値

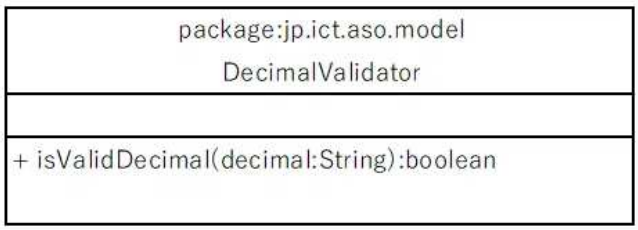

図26．DecimalValidatorクラス図

イベントリスナにエラーチェック処理を加えて本当に完成です。

図27．エラー処理実装済み2進10進相互変換器アプリーケーション

単独起動

実行可能JARファイル

①せっかくですので、単独で起動できるアプリケーションにエクスポートしましょう。Java1.8以上のJREの環境がWindowsのPCにインストールされていればダブルクリックで起動できます。

Eclipseパッケージ・エクスプローラより
SwingBinaryDecimalプロジェクトを右クリック → エクスポート
→ Java → 実行可能JARファイル → 次へ
→ 以下のように設定する → 完了

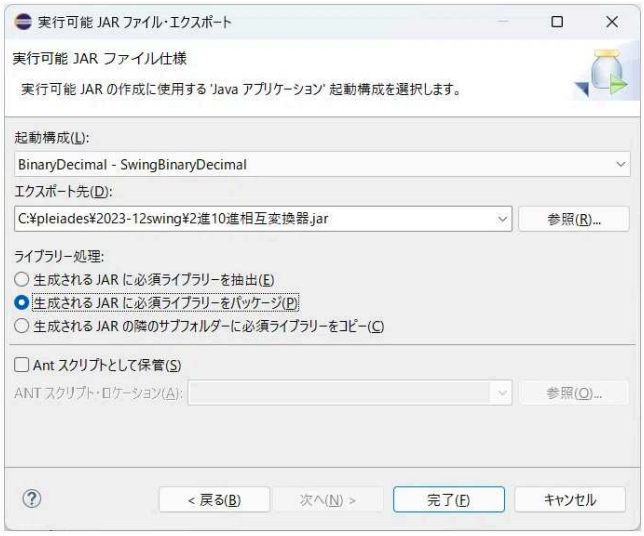

図28．実行可能JARファイルの設定

以下のような警告が出る場合がありますが、気にしません。

図29．警告ダイアログ

作成されたjarファイルをダブルクリックするとアプリが起動します。

図30．実行可能JARファイル

ソースコード例

以上でSwingデザイナーを使って2進数10進数の相互変換を行うデスクトップアプリケーションを作成できました。

以下にすべてのプログラムのソースコード例を示しますので実装の参考にしてください。

BinaryDecimalBean.java

```
package jp.ict.aso.model;
import java.io.Serializable;
public class BinaryDecimalBean implements Serializable{
    String binary;
    String decimal;
public BinaryDecimalBean() {
    binary="0";
    decimal="0";
}
public void binaryDecimal() {
    //2進10進変換
    int decimalInt = Integer.parseInt(binary, 2);
    decimal=String.valueOf(decimalInt);
```

```
}
public void decimalBinary() {
    //10進2進変換
    int decimalInt=Integer.parseInt(decimal);
    binary = Integer.toBinaryString(decimalInt);
}
public void setBinary(String binary) {
    this.binary=binary;
}
public void setDecimal(String decimal) {
    this.decimal=decimal;
}
public String getBinary() {
    return binary;
}
public String getDecimal() {
    return decimal;
}
}
```

BinaryDecimal.java

```
package jp.ict.aso.swing;

import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;

import javax.swing.AbstractAction;
import javax.swing.Action;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.ScrollPaneConstants;
import javax.swing.border.EmptyBorder;

import jp.ict.aso.model.BinaryDecimalBean;
import jp.ict.aso.model.BinaryValidator;
import jp.ict.aso.model.DecimalValidator;

public class BinaryDecimal extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField textField;
    private final Action action = new SwingAction();
    private final Action action_1 = new SwingAction_1();
```

```
private JRadioButton rdbtnNewRadioButton;   //koko
private JRadioButton rdbtnNewRadioButton_1; //koko
private JTextArea textArea;                 //koko

/**
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                BinaryDecimal frame = new BinaryDecimal();
                frame.setVisible(true);
                frame.setResizable(false);      //koko

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

/**
 * Create the frame.
 */
public BinaryDecimal() {
    setTitle("2進10進相互変換器");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 350);      //koko
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

    setContentPane(contentPane);
    contentPane.setLayout(new BorderLayout(0, 0));

    JPanel panel = new JPanel();
    contentPane.add(panel, BorderLayout.NORTH);

    rdbtnNewRadioButton = new JRadioButton("2進数→10進数");  //koko
    panel.add(rdbtnNewRadioButton);

    rdbtnNewRadioButton_1 = new JRadioButton("10進数→2進数");    //koko
    rdbtnNewRadioButton_1.setSelected(true);    //koko
    panel.add(rdbtnNewRadioButton_1);

    ButtonGroup buttongp = new ButtonGroup();   //koko
    buttongp.add(rdbtnNewRadioButton);          //koko
    buttongp.add(rdbtnNewRadioButton_1);        //koko

    textField = new JTextField();
    panel.add(textField);
    textField.setColumns(10);

    JPanel panel_1 = new JPanel();
    contentPane.add(panel_1, BorderLayout.SOUTH);

    JButton btnNewButton = new JButton("変　換");
```

```
        btnNewButton.setAction(action);
        panel_1.add(btnNewButton);

        JButton btnNewButton_1 = new JButton("クリア");
        btnNewButton_1.setAction(action_1);
        panel_1.add(btnNewButton_1);

        JPanel panel_2 = new JPanel();
        contentPane.add(panel_2, BorderLayout.CENTER);

        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS)
        scrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_ALW
        panel_2.add(scrollPane);

        textArea = new JTextArea();     //koko
        textArea.setText("●変換範囲\r\n２進数：31ビット以内（負の数は考慮しません）\r\n10進数：-2
        textArea.setRows(10);
        textArea.setColumns(30);
        scrollPane.setViewportView(textArea);
    }

    private class SwingAction extends AbstractAction {
        public SwingAction() {
            putValue(NAME, "変　換");
            putValue(SHORT_DESCRIPTION, "2進↔10進相互変換を行います");
        }
        public void actionPerformed(ActionEvent e) {
            if(rdbtnNewRadioButton.isSelected()) {
                //入力値のチェック
                BinaryValidator bv = new BinaryValidator();
                if(bv.isValidBinary(textField.getText())) { //true
                    //2進10進変換
                    BinaryDecimalBean bdb = new BinaryDecimalBean();
                    bdb.setBinary(textField.getText()); //2進の値セット
                    bdb.binaryDecimal();                //計算
                    textArea.append(String.valueOf(bdb.getDecimal())+"\n");//表示
                }else {
                    textArea.append("エラー: 入力された値 " + textField.getText()
                    + " は31ビット以内の2進数ではありません。\n");
                }
            }
            if(rdbtnNewRadioButton_1.isSelected()) {
                //入力値のチェック
                DecimalValidator dv = new DecimalValidator();
                if(dv.isValidDecimal(textField.getText())) { //true
                    //10進2進変換
                    BinaryDecimalBean bdb = new BinaryDecimalBean();
                    bdb.setDecimal(textField.getText()); //10進の値セット
                    bdb.decimalBinary();                //計算
                    textArea.append(String.valueOf(bdb.getBinary())+"\n");//表示
                }else {
                    textArea.append("エラー: 入力された値 " + textField.getText()
                    + " は-2147483648～2147483647の範囲にある10進数ではありません。\n");
                }
            }
        }
    }
```

```
    private class SwingAction_1 extends AbstractAction {
        public SwingAction_1() {
            putValue(NAME, "クリア");
            putValue(SHORT_DESCRIPTION, "表示領域をクリアします");
        }
        public void actionPerformed(ActionEvent e) {

            textArea.setText(null);

        }
    }
}
```

BinaryValidator.java

```
package jp.ict.aso.model;

public class BinaryValidator {

    /**
     * 入力された文字列が31ビット以内の2進数かどうかを判定します。
     *
     * @param binary 入力された2進数を表す文字列
     * @return true なら有効な31ビット以内の2進数、false なら無効な値
     * intのリテラルの範囲
     *  （int型の変数は -2147483648～2147483647までの数値データを格納）
     */
    public boolean isValidBinary(String binary) {
        // 入力値がnullまたは空文字列でないか確認
        if (binary == null || binary.isEmpty()) {
            return false;
        }

        // 入力値が2進数（0または1）であり、かつ長さが1～31文字であるかをチェック
        return binary.matches("[01]{1,31}");
    }
}
```

DecimalValidator.java

```
package jp.ict.aso.model;

public class DecimalValidator {

    /**
```

```java
     * 入力された文字列が-2147483648～2147483647の範囲にある10進数かどうかを判定します。
     *
     * @param decimal 入力された10進数を表す文字列
     * @return true なら有効な範囲の10進数、false なら無効な値
     * intのリテラルの範囲
     *  （int型の変数は -2147483648～2147483647までの数値データを格納）
     */
    public boolean isValidDecimal(String decimal) {
        // 入力値がnullまたは空文字列でないか確認
        if (decimal == null || decimal.isEmpty()) {
            return false;
        }

        try {
            // 文字列を整数に変換し、-2147483648～2147483647の範囲に収まっているかをチェック
            int value = Integer.parseInt(decimal);
            return value >= -2147483648 && value <= 2147483647;
                   } catch (NumberFormatException e) {
            // 数値への変換が失敗した場合（例：文字列に数字以外の文字が含まれているなど）はfalseを返
            return false;
        }

    }
}
```