Swingによるデスクトップアプリケーション開発（シーザ暗号）-JavaSE1.8

Swingによるデスクトップアプリケーション開発（シーザ暗号）-JavaSE1.8

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）を使って単一換字式暗号（シーザ暗号）を行うプログラムを作成します。暗号化ロジック（Bean）の部品は提供します。こうした部品として作成されたクラスファイルを活用する方法を学びます。チーム開発での役割分担を想定しています。**

2025年8月よりECLIPSEのバージョンを最新版（Version: 2025-06 (4.36.0)）に変更しました。

設計仕様

【単一換字式暗号の仕様】
・暗号化アルゴリズム：特定の文字を辞書順に特定の数だけ前（復号する場合は後ろ）にある文字と置きかえます。この時循環シフトを行います（ローテーションします）。
・暗号化キー：辞書順にずらす数値（特定の数）のことです。とりわけカエサルが実際に用いた３のとき**シーザー暗号**といいます。

（例）暗号化キーが３の時
　ENCODE（暗号化）　D → A，a → x，2 → 9
　DECODE（復　号）　A → D，x → a，9 → 2

※今回の仕様では大文字、小文字、数字に対応させます（数字に対応させるためkeyの最大値は10とします）

【Model（処理ロジック）の仕様】
CaesarBean.classが提供されます。
・Beanは引数なしのコンストラクタで実体化します。
・ロジックのメソッドを実行することで暗号化(encode)・復号(decode)の
　変換処理が行われます。
・結果はフィールド変数(encodeText)(decodeText)内に格納されます。

（実行例）暗号化キーが３の時
　INPUT :This is a copy machine of 90 XEROX
　ENCODE:Qefp fp x zlmv jxzefkb lc 67 UBOLU
　DECODE:This is a copy machine of 90 XEROX

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。

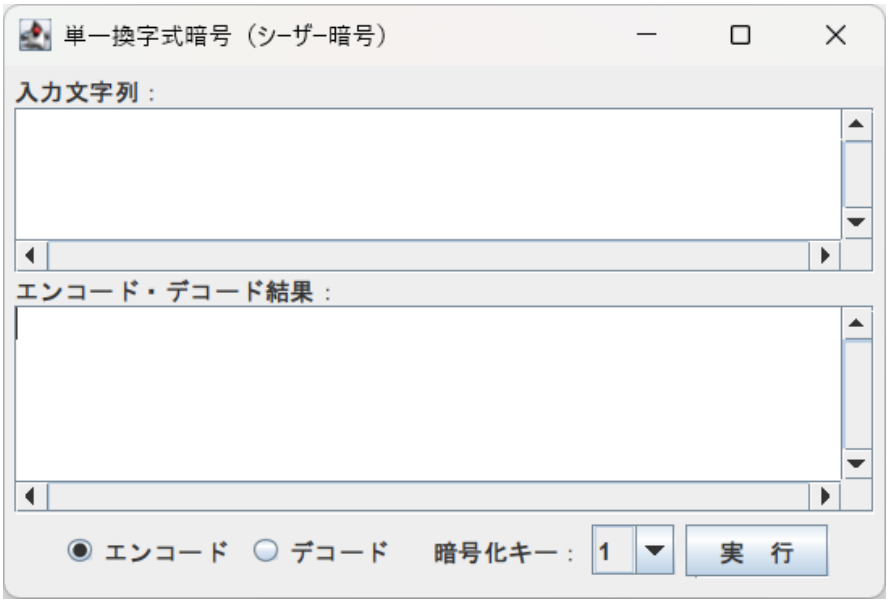

図1．円ドル変換アプリケーションのGUI画面

内部設計

処理ロジック

単一換字式暗号の暗号化及び復号を行うクラスファイルは提供されます。

提供される暗号化・復号計算ロジック（CaesarBean.class）の使い方とクラス図は以下のようになります。

（使い方）

- CaesarBeanは引数なしのコンストラクタで実体化します。
- ロジックのメソッドを実行することで暗号化(encode)・復号(decode)のそれぞれの変換処理が行われます。
- 結果はフィールド変数(encodeText)(decodeText)内にそれぞれ格納されます。

（クラス図）

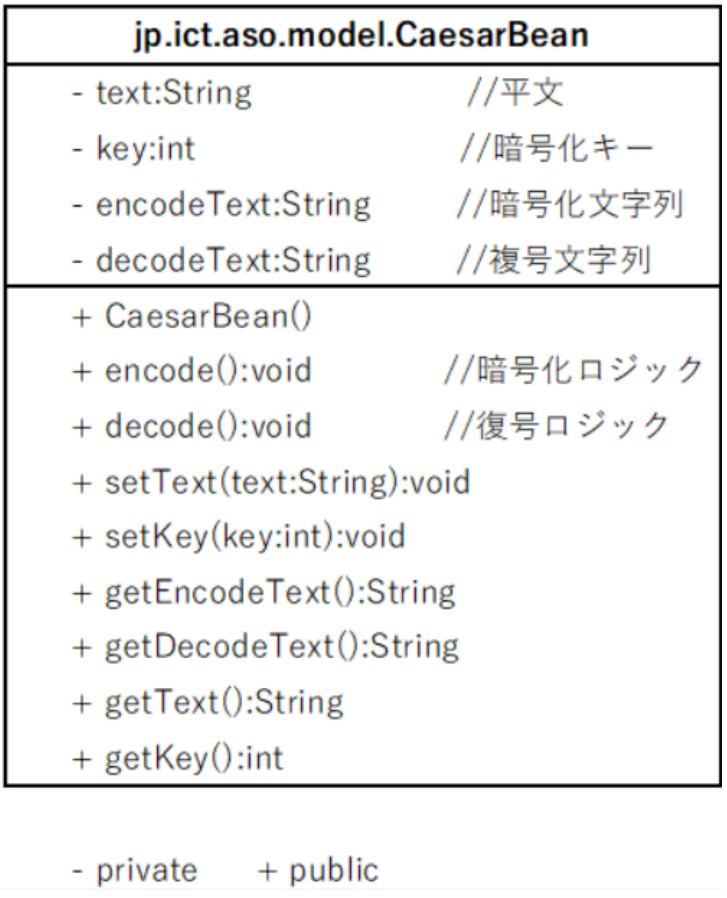

図2．CaesarBeanクラス図

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→**「SwingCaesar」プロジェクト**を作成→図３の内容で設定す

る

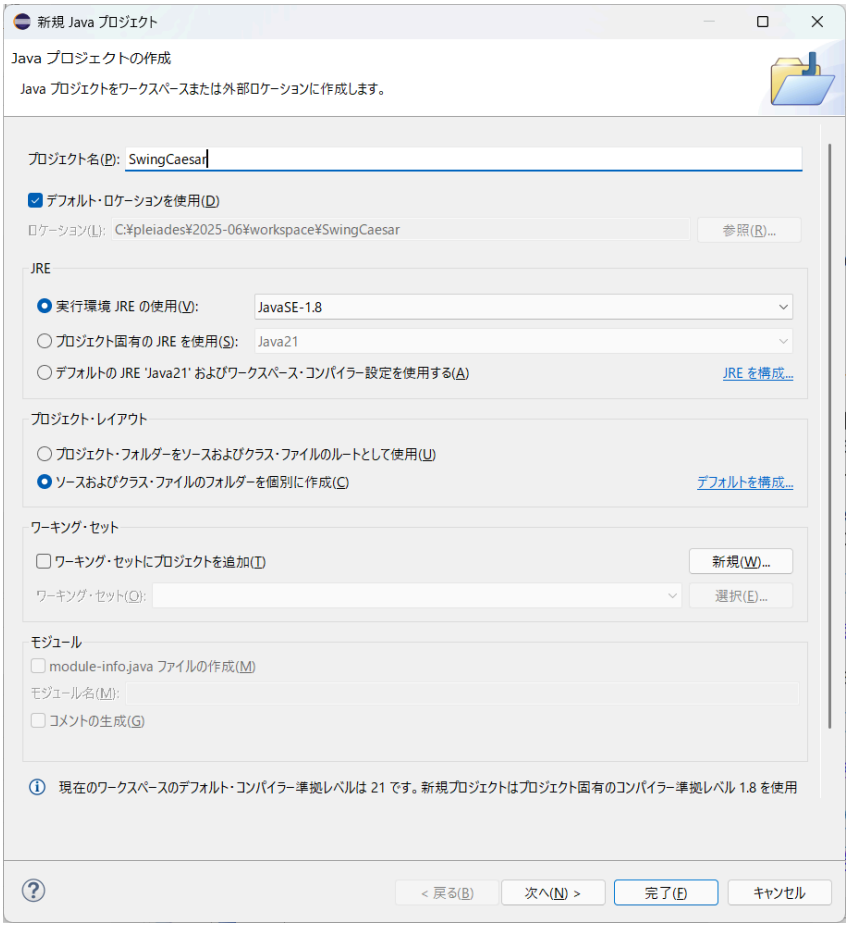

図3．SwingCaesarプロジェクト作成

※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

modelの準備

(1)単一換字式の暗号化・復号計算ロジック用JavaBeansクラスを使用する準備をします

以下Windows環境を想定しています。
事前に「提供Beanフォルダ」を作成しておきます（例　**c:￥提供Bean**）。
提供Beanフォルダ内にCaesarBean.classを保存しておきます。
その際、図４のように**パッケージの階層**に従ってください。
（例　c:￥提供Bean￥jp￥ict￥aso￥model￥CaesarBean.class）

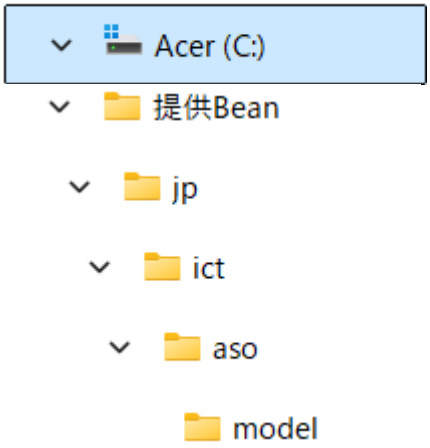

図4．提供Beanの階層

(2)CaesarBean.classをEclipseのビルド・パスに追加します

Eclipseパッケージ・エクスプローラより
SwingCaesarプロジェクトを右クリック→ビルド・パス→ビルド・パスの構成→「ライブラリ」タブ→「外部クラス・フォルダーの追加」→「提供Bean」を指定します→最後に「適用して閉じる」をクリック

※**Eclipse2025の場合**→caesarプロジェクトを右クリック→プロパティ→Java
　のビルド・パスで設定します。
※図５のように一度設定されていれば再度設定する必要はありません。
※うまく読み込めない場合は一旦「提供Bean」を除去してから再度追
　加するか名前を変えてみます。

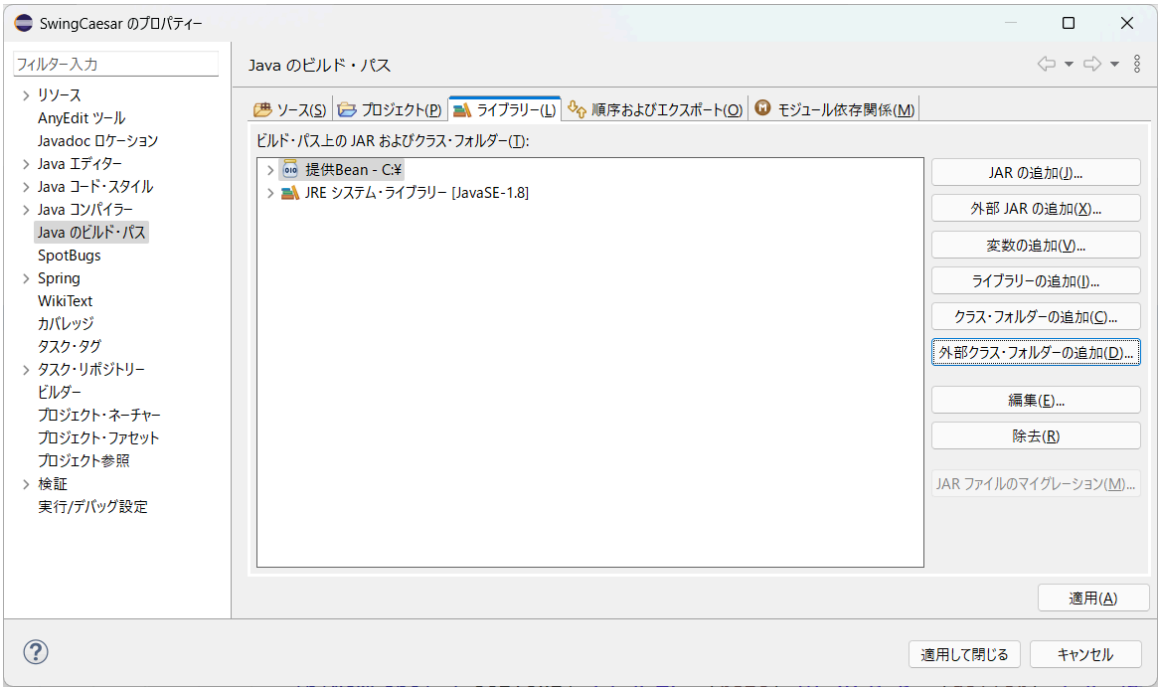

図5．Javaのビルドパス追加画面

GUI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
SwingCaesar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

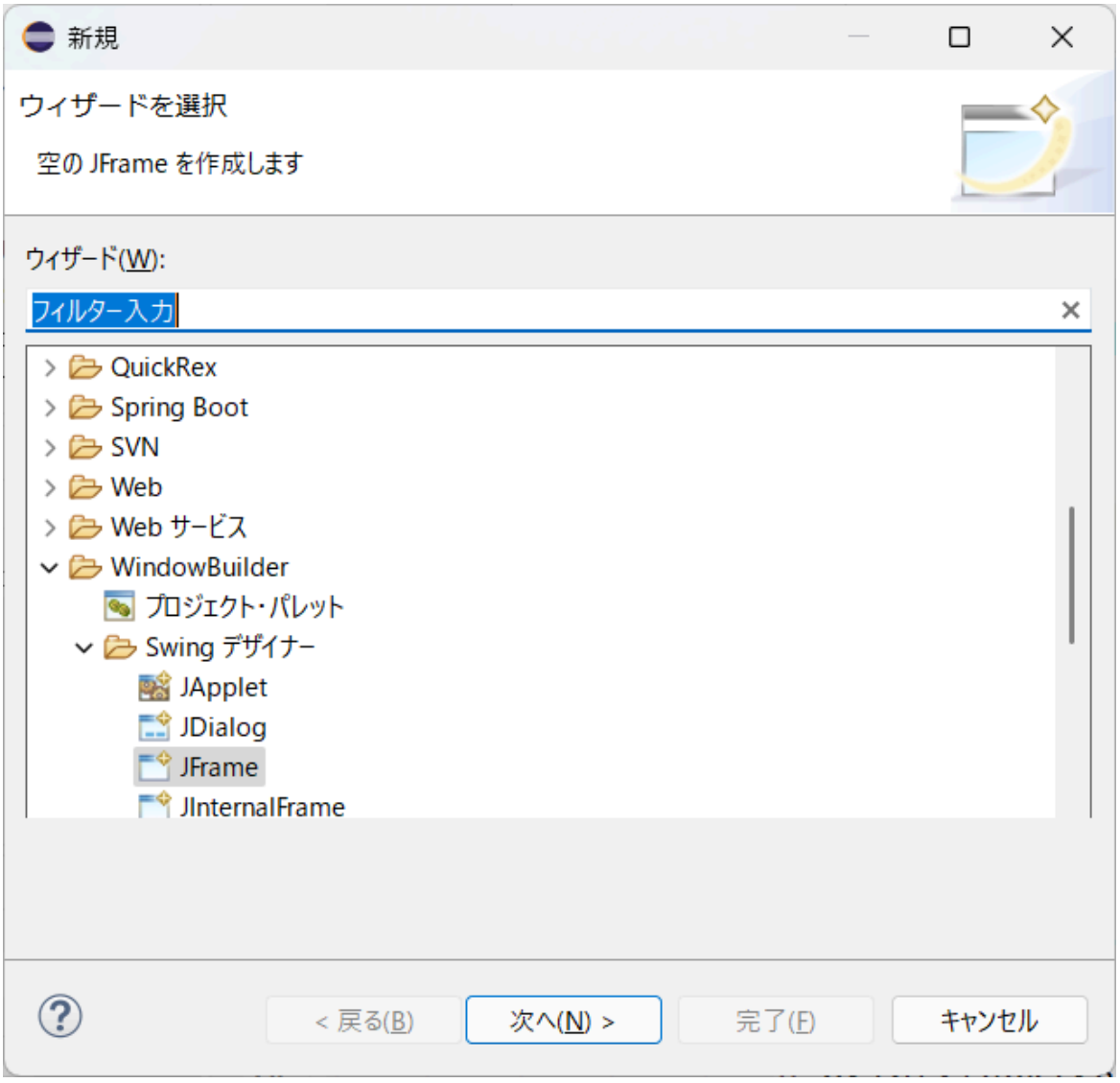

図6．JFrameウィザードの選択

以下の内容で作成
　パッケージ：jp.ict.aso.swing
　名前：Cipher

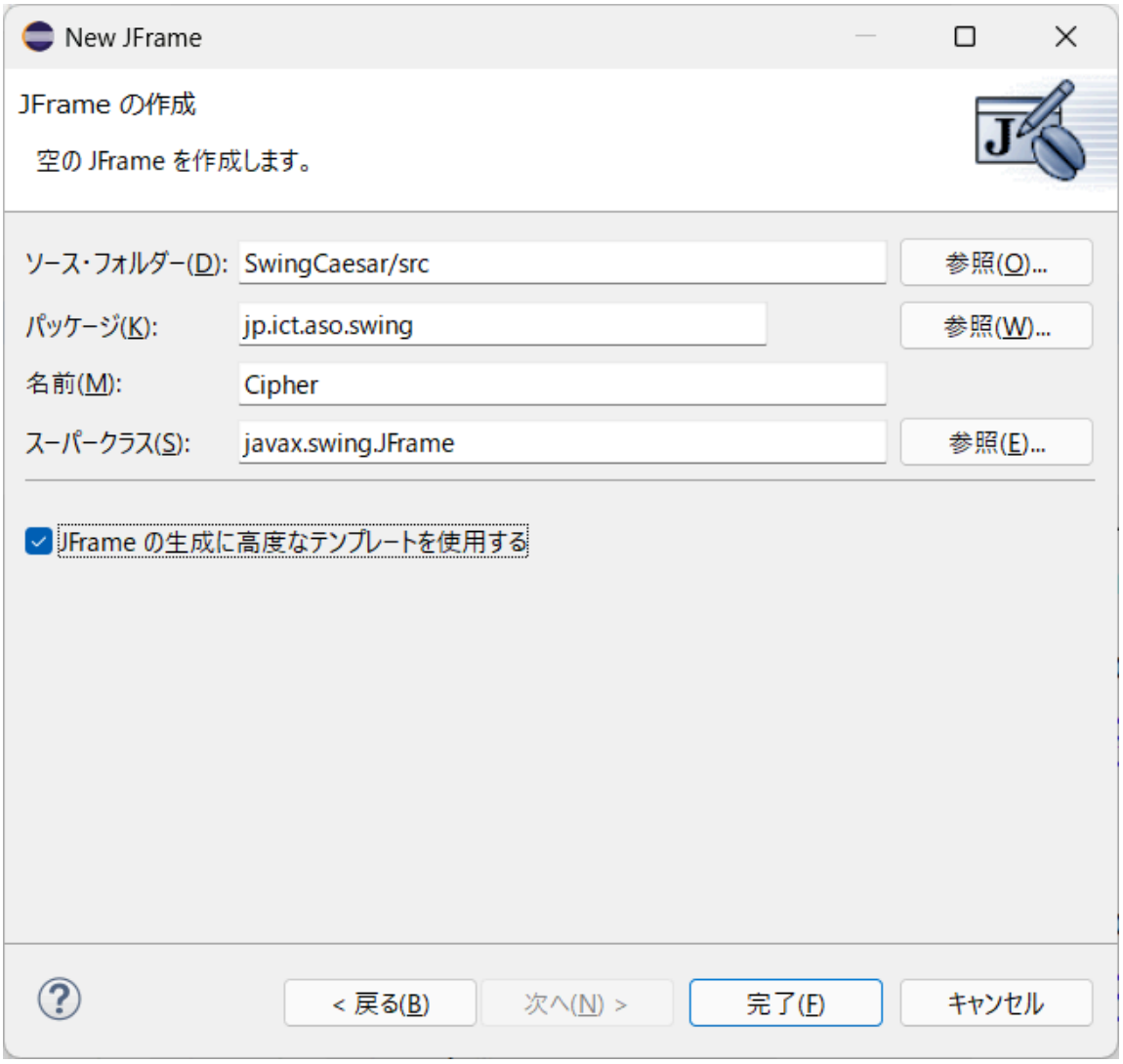

図7．Swingアプリケーションの生成

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。

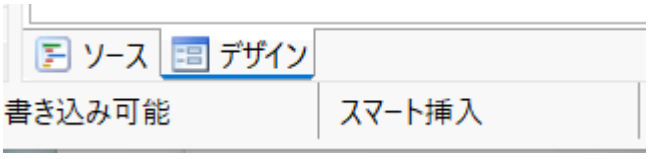

図8．Swingデザイナーに切り替え

②図9.を参考にデザイン設計します。

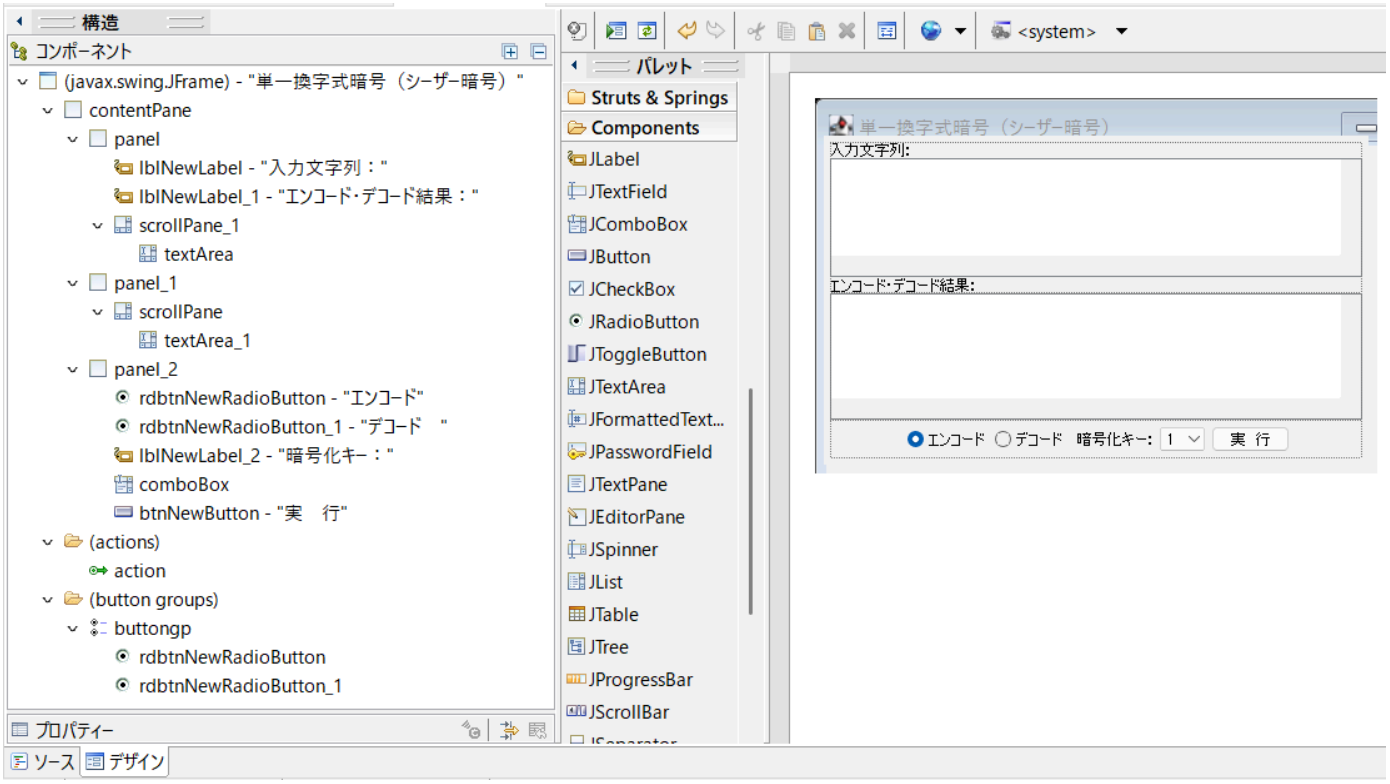

図9．GUIデザイン設計

イベント実装

ソースタブに変更します。

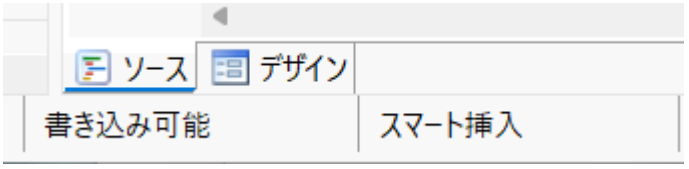

図10．ソースタブに変更

①「変換結果」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, "実　行");
        putValue(SHORT_DESCRIPTION, "暗号化・復号を行います");
    }
    public void actionPerformed(ActionEvent e) {

        String input = textArea.getText();
        String encdec = textArea_1.getText();
        int key = comboBox.getSelectedIndex()+1;

        if(rdbtnNewRadioButton.isSelected()) { //エンコード
            CaesarBean cb = new CaesarBean();
            cb.setText(input);
            cb.setKey(key);
            cb.encode();
            textArea_1.append(cb.getEncodeText()+"¥n");
        }else {                                 //デコード
            CaesarBean cb = new CaesarBean();
            cb.setText(input);
            cb.setKey(key);
            cb.decode();
            textArea_1.append(cb.getDecodeText()+"¥n");
        }
    }
}
```

図11．単一換字式暗号の実行結果ボタンのイベント内容

②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 Cipher extends JFrame {

    private JPanel contentPane;
    private final Action action = new SwingAction();

    private JTextArea textArea;                    //koko
    private JTextArea textArea_1;                  //koko
    private JRadioButton rdbtnNewRadioButton;      //koko
    private JRadioButton rdbtnNewRadioButton_1;    //koko
    private JComboBox comboBox;                    //koko
    /**
     * Launch the application.
```

図12．フィールド変数の変更

③ローカル変数の宣言になっている部分を変更します
あわせてラジオボタンをグループ化します。//kokoの部分を変更します。

```
textArea = new JTextArea(); //koko
textArea.setRows(4);
scrollPane_1.setViewportView(textArea);
textArea_1 = new JTextArea();   //koko
textArea_1.setRows(4);
scrollPane.setViewportView(textArea_1);

comboBox = new JComboBox(); //koko
comboBox.setModel(new DefaultComboBoxModel(new String[] {"1"
panel_2.add(comboBox);

rdbtnNewRadioButton = new JRadioButton("エンコード");   //koko
rdbtnNewRadioButton.setSelected(true);
panel_2.add(rdbtnNewRadioButton);

rdbtnNewRadioButton_1 = new JRadioButton("デコード□"); //koko
panel_2.add(rdbtnNewRadioButton_1);

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

図13．ローカル変数の宣言変更

④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14．実行方法

⑤入力は「This is a pen」暗号化キー「3」のエンコードの実行で結果を確認します。

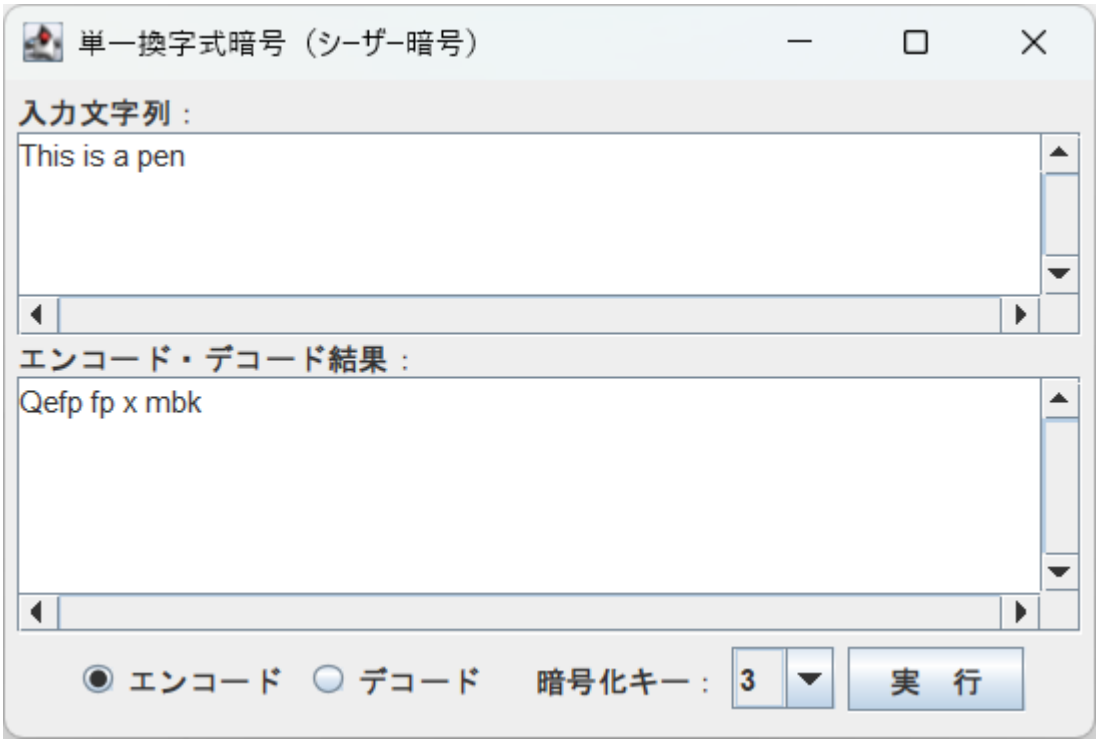

図15．エンコード実行

⑥「エンコード結果」を入力欄にコピーして暗号化キー「3」のデコードの実行で元に戻るか確認します。

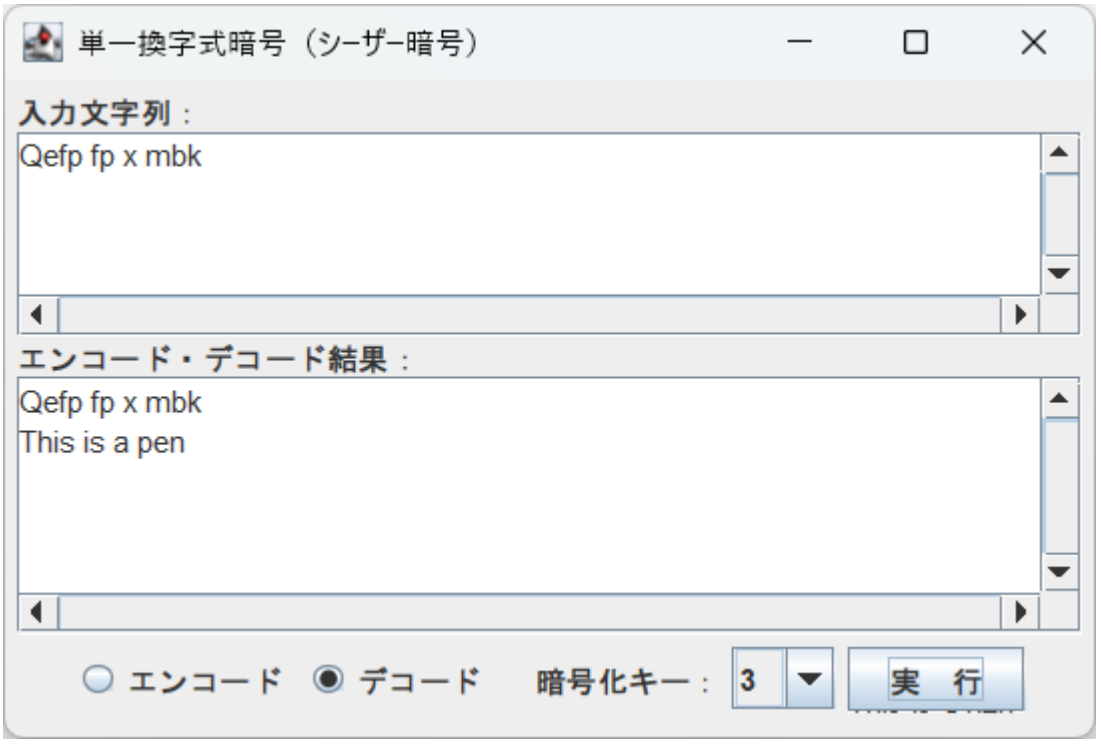

図16．デコード実行

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
SwingCaesar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

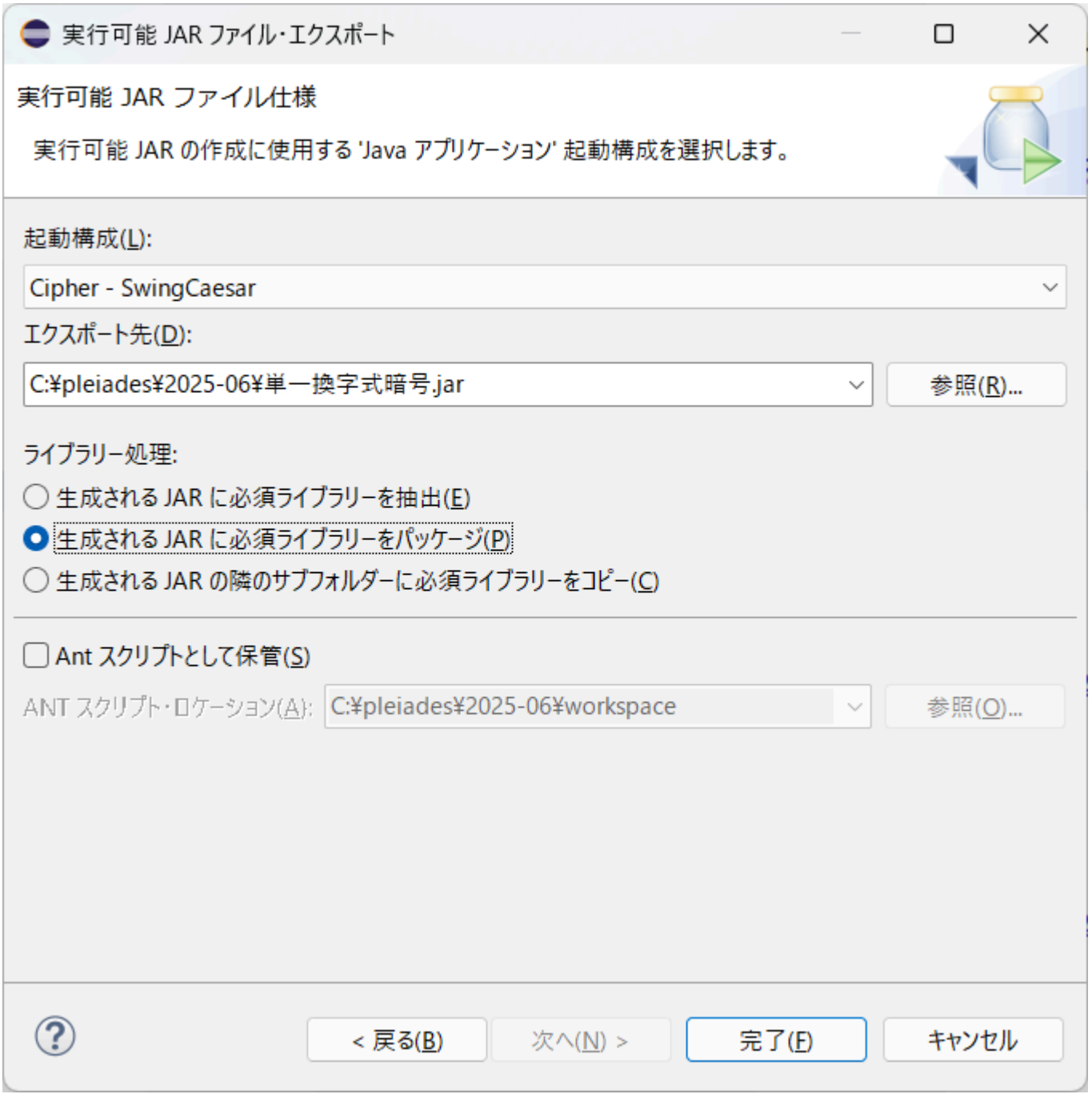

図17．実行可能JARファイルの設定

以下のような警告ダイアログが出る場合がありますが無視します。

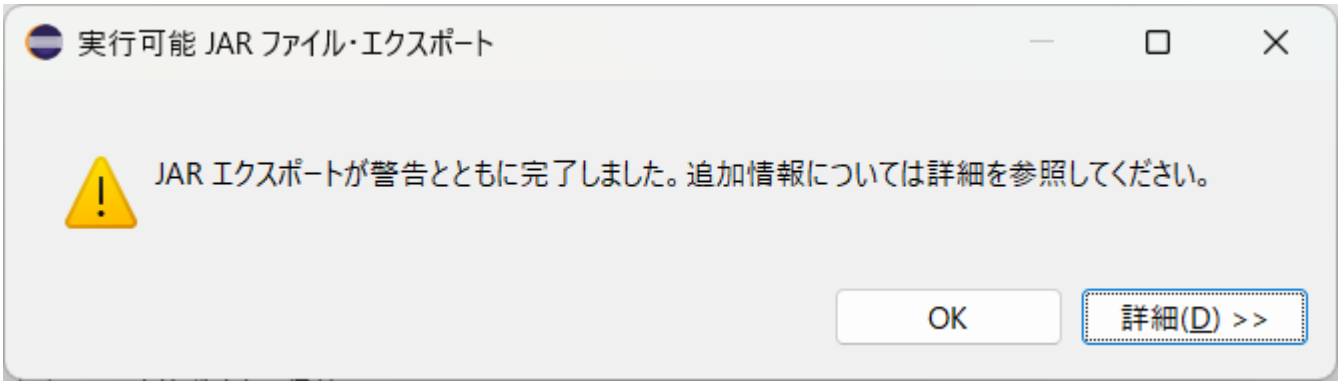

図18．警告ダイアログ

作成されたjarファイルをダブルクリックすると単一換字式暗号アプリが起動します。

図19．実行可能JARファイル

最後に

以上でSwingデザイナーを使って単一換字式暗号のエンコード・デコードを行うデスクトップアプリを作成できました。