Swingによるデスクトップアプリケーション開発（おみくじ）-JavaSE1.8

Swingによるデスクトップアプリケーション開発（おみくじ）-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）を使っておみくじを引くアプリケーションを作成します。

2025年9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。

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。おみくじを引いている（図1の中央）部分はアニメーションで表現します。

画像とアニメーションは事前に用意しておきます。今回は図１のような300×300ピクセルのおみくじ開始と運勢（大吉、中吉、小吉、大凶）の画像、及び　図1.1のようなお参りのアニメーション画像を用意します。

以下の「GIFアニメーションの作り方」を見ておしゃれなアニメーションを作ってください。

> ### GIFアニメーションの作り方
>
> GIFアニメーションの作り方.pdf
>
> ### 背景等画像のフリーサイト
>
> https://pixabay.com/illustrations/search/background/

図1．おみくじアプリの流れ

図1.1 お参りアニメーションの例

内部設計

処理ロジック

おみくじを引く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、アニメーション画面に遷移します。止めるボタンで運勢を文字と画像で表示します。運勢は４種類で以下の確率で出現するようにします。

大吉　10%
中吉　40%
小吉　40%
大凶　10%

作成するクラス仕様
作成する運勢計算ロジック（OmikujiBean.class）の仕様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）
・０から９までの乱数を発生させる
（Javaの例：int ran=new java.util.Random().nextInt(10);）
・乱数値が０ならば占い結果は「大凶」とします
・乱数値が１ならば占い結果は「大吉」とします
・乱数値が２or３or４or５ならば占い結果は「中吉」とします
・上記以外は「小吉」とします

（クラス図）

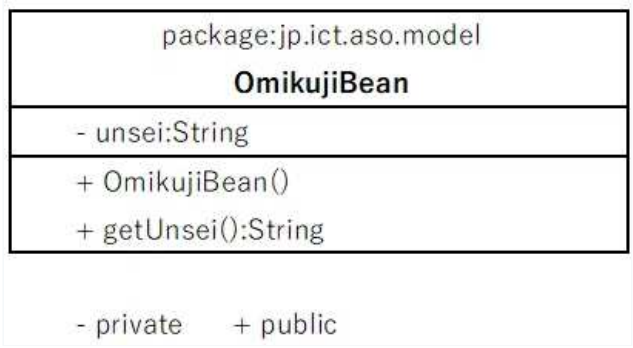

図2．OmikujiBeanクラス図

・JavaBeansの仕様に基づきOmikujiBean.javaを作成します。
・package名は jp.ict.aso.model とします。
・作成するクラスはSerializableインターフェースを実装します。
・変数はprivate宣言します。
・引数なしのコンストラクタを実装します。
・**コンストラクタ内に処理ロジックを実装**します。このとき処理結果は
　インスタンス変数に格納します。
・setterメソッドを実装します。
・getterメソッドを実装します。

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→「SwingOmikuji」プロジェクトを作成→図３の内容で設定する

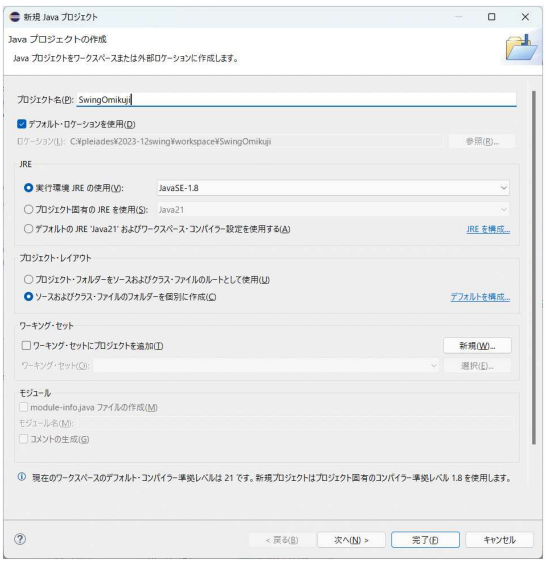

図3．SwingOmikujiプロジェクト設定

※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
SwingOmikuji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

　　パッケージ：**jp.ict.aso.model**
　　名前：**OmikujiBean**

ソースコード：**図2のクラス図**のように作成します
　　運勢計算のロジックはヒントを参考にしてください

【ヒント】

```
int fortune=new java.util.Random().nextInt(10);
switch (fortune) {
 case 0:
  unsei="大凶";break;
 case 1:
  unsei="大吉";break;
 case 2:
 case 3:
 case 4:
 case 5:
  unsei="中吉";break;
 default:
  unsei="小吉";break;
}
```

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
SwingOmikuji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

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

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。

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

③contentPaneの「North」「South」「Center」の順番でGUI部品のJPanelをパレットから配置します。各パネルのプロパティのConstrainsは「**panel**が**North**」「**panel_1**が**South**」「**panel_2**が**Center**」となります。

④JPanelのLayoutプロパティはFlowLayoutのままです。

⑤「North」位置のJPanel(panel)にGUI部品のJButtonとJLabelを２つ順番にパレットから配置します。JButtonとJLabelのtextプロパティを図４のように変更します。

⑥「South」位置のJPanel(panel_1)にGUI部品のJButtonを２つパレットから配置します。JButtonのtextプロパティを図４のように変更します。

⑦「Center」位置のJPanel(panel_2)にGUI部品のJLabelをパレットから配置します。

ここまでのGUIの設計状況（部品配置）は図４のようになります。

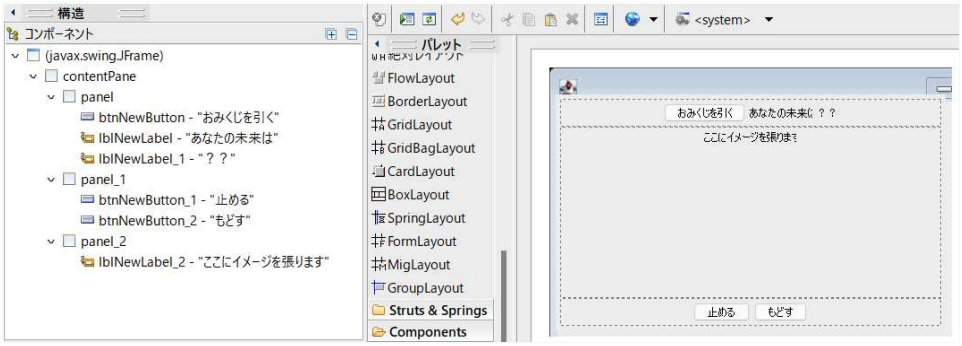

図4．部品配置

⑧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。

最終的なデザインは図5のようになります。

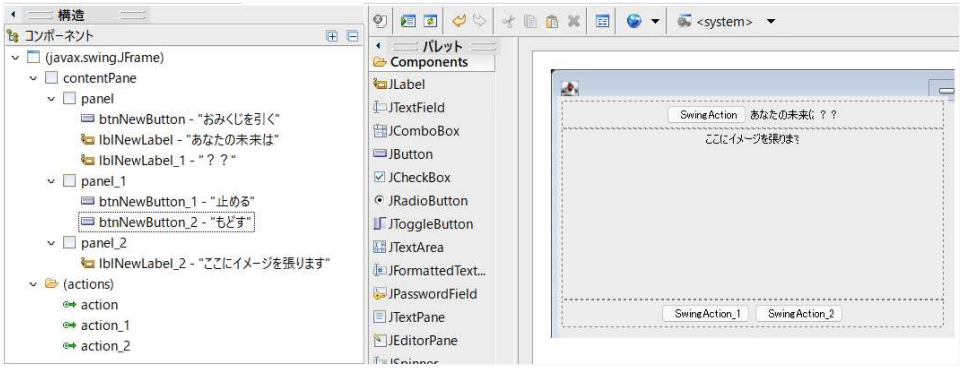

図5．最終デザイン

イベント実装

ソースタブに変更します。

①「おみくじを引くボタン」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, "おみくじを引く");
        putValue(SHORT_DESCRIPTION, "おみくじを引くアニメーションを実行します");
    }
    public void actionPerformed(ActionEvent e) {
            image = new ImageIcon(this.getClass().getResource("images/omairi.gif"));
            lblNewLabel_2.setIcon(image);

              //ボタンの使用可否
              btnNewButton.setEnabled(false);
              btnNewButton_1.setEnabled(true);
              btnNewButton_2.setEnabled(false);
    }
}
```

図6．おみくじを引くボタンのイベント内容

②「止めるボタン」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, "止める");
        putValue(SHORT_DESCRIPTION, "おみくじアニメーションを止めて占います");
    }
    public void actionPerformed(ActionEvent e) {
        OmikujiBean ob = new OmikujiBean();
        if("大吉".equals(ob.getUnsei())) {
            image = new ImageIcon(this.getClass().getResource("images/great.png"));

        }else if("中吉".equals(ob.getUnsei())) {
            image = new ImageIcon(this.getClass().getResource("images/middle.png"));

        }else if("小吉".equals(ob.getUnsei())) {
            image = new ImageIcon(this.getClass().getResource("images/small.png"));

        }else {
            image = new ImageIcon(this.getClass().getResource("images/bad.png"));

        }
         lblNewLabel_2.setIcon(image);
         lblNewLabel_1.setText(ob.getUnsei());

         //ボタンの使用可否
         btnNewButton.setEnabled(false);
         btnNewButton_1.setEnabled(false);
         btnNewButton_2.setEnabled(true);
    }
}
```

図7．止めるボタンのイベント内容

③「もどすボタン」のイベントのソース部分を変更します。

```
private class SwingAction_2 extends AbstractAction {
    public SwingAction_2() {
        putValue(NAME, "もどす");
        putValue(SHORT_DESCRIPTION, "最初の画面に戻ります");
    }
    public void actionPerformed(ActionEvent e) {
            image = new ImageIcon(this.getClass().getResource("images/omikuji.png"));
            lblNewLabel_2.setIcon(image);
            lblNewLabel_1.setText("？？");

          //ボタンの使用可否
          btnNewButton.setEnabled(true);
          btnNewButton_1.setEnabled(false);
          btnNewButton_2.setEnabled(false);

    }
}
```

図8．もどすボタンのイベント内容

④import文を追加しフィールド変数を変更します。//kokoの部分を追加します。

```
import javax.swing.AbstractAction;
import java.awt.event.ActionEvent;
import javax.swing.Action;

import javax.swing.ImageIcon;  //koko
import jp.ict.aso.model.OmikujiBean;   //koko

public class Omikuji extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private final Action action = new SwingAction();
    private final Action action_1 = new SwingAction_1();
    private final Action action_2 = new SwingAction_2();
    private ImageIcon image;        //koko
    private JLabel lblNewLabel_1;   //koko
    private JLabel lblNewLabel_2;   //koko
    private JButton btnNewButton;   //koko
    private JButton btnNewButton_1; //koko
    private JButton btnNewButton_2; //koko


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

図9．import文とフィールド変数の変更

⑤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);

    btnNewButton = new JButton("おみくじを引く");  //koko
    btnNewButton.setAction(action);
    panel.add(btnNewButton);

    JLabel lblNewLabel = new JLabel("あなたの未来は");
    panel.add(lblNewLabel);

    lblNewLabel_1 = new JLabel("？？");          //koko
    panel.add(lblNewLabel_1);

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

    btnNewButton_1 = new JButton("止める");      //koko
    btnNewButton_1.setAction(action_1);
    panel_1.add(btnNewButton_1);

    btnNewButton_2 = new JButton("もどす");      //koko
    btnNewButton_2.setAction(action_2);
    panel_1.add(btnNewButton_2);

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

    lblNewLabel_2 = new JLabel("");              //koko
    panel_2.add(lblNewLabel_2);
```

図10．ローカル変数の宣言変更

⑥作成済みのおみくじアニメーションと運勢結果の画像を配置します。

図11．画像とアニメーションの配置

⑦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12．おみくじアプリ起動画面

⑧ボタンをクリックしてアプリの処理の流れを確認します。

図13．おみくじアプリの流れ

処理の流れ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」「画像の全体が表示されていない」「起動時に画像が表示されていない」「起動時にすべてのボタンが有効になっている」の５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）の大きさを画像全体が表示できるように大きくします。
③画面の大きさを変えられないように固定します。
④起動時に画像を表示します。
⑤起動時はおみくじを引くボタンだけを有効にします。

実行して動きを確認します。これで完成です。

図14．おみくじアプリ完成

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
SwingOmikuji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

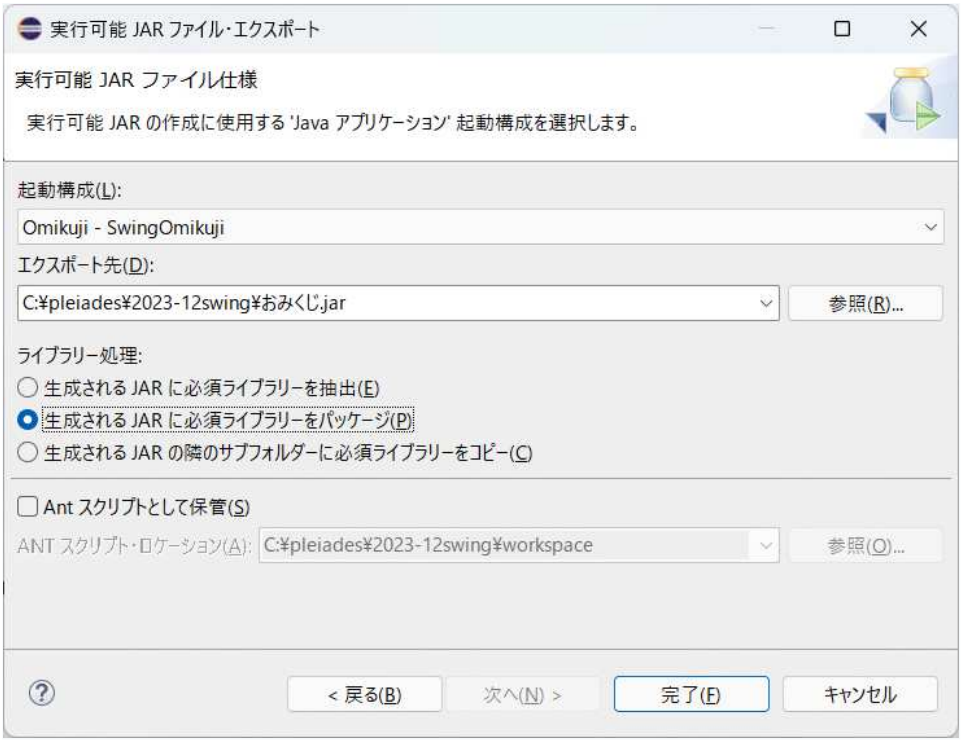

図15．実行可能JARファイルの設定

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。

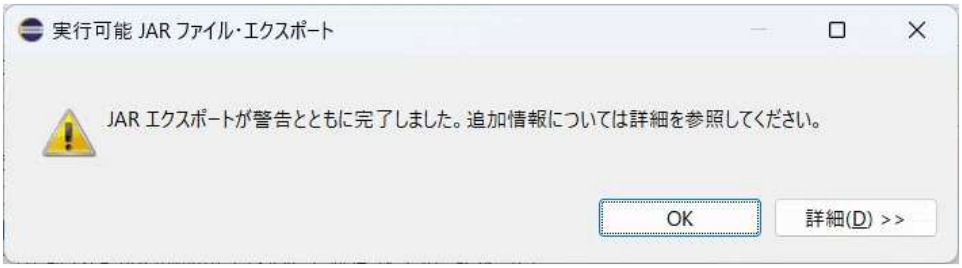

図16．警告ダイアログ

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17．実行可能JARファイル

最後に

今回は、GUIデザイン部分の設定の詳細は提示していません。また、実装変更部分のソースコード例も載せていません。自由に工夫してみてください！