Swingによるデスクトップアプリケーション開発（スロットマシン）-JavaSE1.8

office・M
2024年10月7日 10:39

¥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）を使ってスロットマシンのシミュレータを作成します。

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。スロットマシンの回転部分は図２のような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300×300ピクセル、または400×400ピクセルの0から9までの数字の画像とそれぞれの桁の回転アニメーションを用意します。

図1．スロットマシンシミュレータ

図2．スロットマシンの回転アニメーション

内部設計

処理ロジック

３つの乱数を発生させて当たりを判定するSlotmachineBeanクラスを提供します。数字あるいは絵文字の画像の回転のアニメーションをつけます。

提供クラス

提供されるスロットマシンの配当計算ロジック（**SlotmachineBean.class**）の仕様とクラス図は以下のようになります。

（仕様）
①メッセージ表示
・３つの乱数を発生させそれぞれインスタンス変数に格納する
・３組揃いでVerryGood、２組揃いでGood、のメッセージを表示する（ばらばらでは表示なし）
②持ち点と配当
・掛け点、持ち点のインスタンス変数を用意する
・初期値で持ち点は10点、掛け点は１点とする
・3組揃いで掛け点の5倍を持ち点にプラスする
・2組揃いで掛け点の2倍を持ち点にプラスする
・ばらばらで掛け点を持ち点よりマイナスする
・回転結果、掛け点、持ち点、メッセージ　の順にヘッダー部に表示する

（クラス図）

図3．SlotmachineBean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→「SwingSlotmachine」プロジェクトを作成→図４の内容で設定する

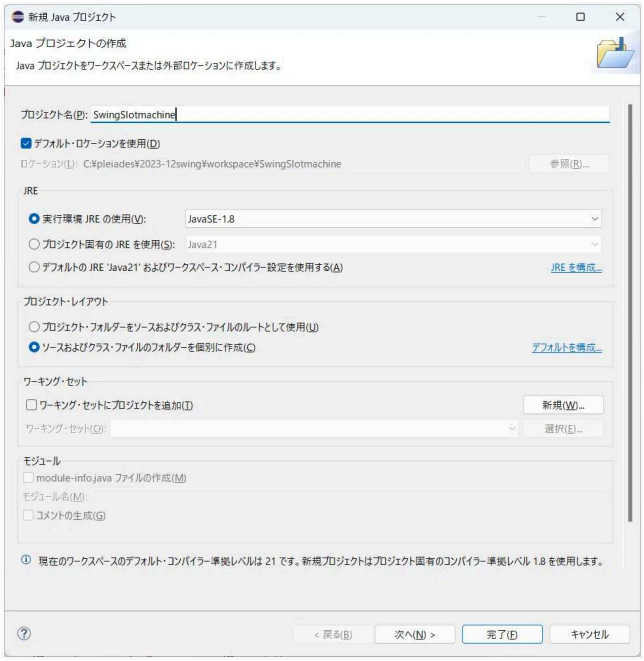

図4．SwingSlotmachineプロジェクト設定

※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

外部classの利用準備

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

ビルド・パスに追加

SlotmachineBean.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
プロジェクトを右クリック→プロパティ→「Javaのビルド・パス」を選択
→「ライブラリ」タブ→「外部クラス・フォルダーの追加」ボタンクリック
→「JavaBeans」を選択→最後に「適用して閉じる」をクリック

※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

図5．外部クラス・フォルダの追加

実装

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
SwingSlotmachine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：Slotmachine

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。

④「**Center**」位置のJPanel(**panel_2**)のプロパティのLayoutを**GridLayout**に変更し、Layoutの＋を展開した**columsの設定を３**にします。

⑤「**North**」位置のJPanel(**panel**)にGUI部品のJLabelとJtextFielsを２組、順番にパレットから配置します。JLabelとJTextFieldのtextプロパティを図６のように変更します。

⑥「**South**」位置のJPanel(**panel_1**)にGUI部品のJButtonを３つパレットから配置します。JButtonのtextプロパティを図６のように変更します。

⑦「**Center**」位置のJPanel(**panel_2**)にGUI部品のJLabelを３つパレットから配置します。それぞれのtextプロパティは空（””）しておきます。

ここまでのGUIの設計状況（部品配置）は図６のようになります。

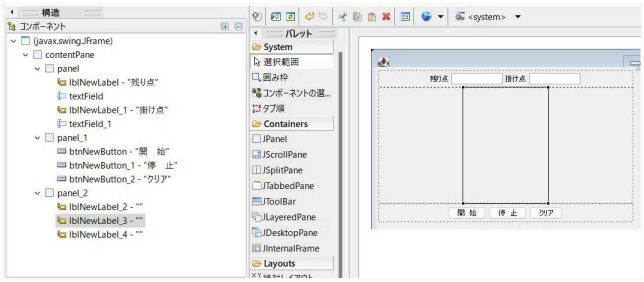

図6．部品配置

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７のようになります。

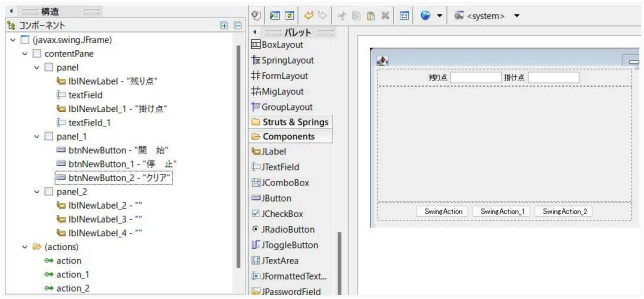

図7．最終デザイン

イベント実装

ソースタブに変更します。

①「開始ボタン」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, "開　始");
        putValue(SHORT_DESCRIPTION, "回転を開始します");
    }
    public void actionPerformed(ActionEvent e) {
        //回転開始
        image = new ImageIcon(this.getClass().getResource("images/number1.gif"));
        lblNewLabel_2.setIcon(image);
        image = new ImageIcon(this.getClass().getResource("images/number2.gif"));
        lblNewLabel_3.setIcon(image);
        image = new ImageIcon(this.getClass().getResource("images/number3.gif"));
        lblNewLabel_4.setIcon(image);

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

図8．開始ボタンのイベント内容

②「停止ボタン」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, "停　止");
        putValue(SHORT_DESCRIPTION, "回転を止めます");
    }
    public void actionPerformed(ActionEvent e) {
        //ローリングの設定と実行
        SlotmachineBean sb=new SlotmachineBean();
        sb.setZanten(Integer.parseInt(textField.getText()));
        sb.setKaketen(Integer.parseInt(textField_1.getText()));
        sb.slotRolling();
        //停止時の画像の設定
        image = new ImageIcon(this.getClass().getResource("images/"+sb.getNumA()+".png"));
        lblNewLabel_2.setIcon(image);
        image = new ImageIcon(this.getClass().getResource("images/"+sb.getNumB()+".png"));
        lblNewLabel_3.setIcon(image);
        image = new ImageIcon(this.getClass().getResource("images/"+sb.getNumC()+".png"));
        lblNewLabel_4.setIcon(image);
        //残り点・掛け点の更新
        textField.setText(String.valueOf(sb.getZanten()));
        textField_1.setText(String.valueOf(sb.getKaketen()));

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

図9．停止ボタンのイベント内容

③「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_2 extends AbstractAction {
    public SwingAction_2() {
        putValue(NAME, "クリア");
        putValue(SHORT_DESCRIPTION, "初期画面にもどします");
    }
    public void actionPerformed(ActionEvent e) {
        //初期状態の画像の設定
        image = new ImageIcon(this.getClass().getResource("images/7.png"));
        lblNewLabel_2.setIcon(image);
        image = new ImageIcon(this.getClass().getResource("images/7.png"));
        lblNewLabel_3.setIcon(image);
        image = new ImageIcon(this.getClass().getResource("images/7.png"));
        lblNewLabel_4.setIcon(image);
        //残り点・掛け点の初期化
        textField.setText("10");
        textField_1.setText("1");

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

図10．クリア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 javax.swing.Action;
import javax.swing.ImageIcon;   //koko
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;

import jp.ict.aso.model.SlotmachineBean;    //koko

public class Slotmachine extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField textField;
    private JTextField textField_1;
    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_2;  //koko
    private JLabel lblNewLabel_3;  //koko
    private JLabel lblNewLabel_4;  //koko
    private JButton btnNewButton;  //koko
    private JButton btnNewButton_1; //koko
    private JButton btnNewButton_2; //koko
```

図11．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_1 = new JPanel();
contentPane.add(panel_1, BorderLayout.SOUTH);

btnNewButton = new JButton("開　始");   //koko
btnNewButton.setAction(action);
panel_1.add(btnNewButton);

btnNewButton_1 = new JButton("停　止"); //koko
btnNewButton_1.setAction(action_1);
panel_1.add(btnNewButton_1);

btnNewButton_2 = new JButton("クリア"); //koko
btnNewButton_2.setAction(action_2);
panel_1.add(btnNewButton_2);

JPanel panel_2 = new JPanel();
contentPane.add(panel_2, BorderLayout.CENTER);
panel_2.setLayout(new GridLayout(0, 3, 0, 0));

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

lblNewLabel_3 = new JLabel(""); //koko
panel_2.add(lblNewLabel_3);

lblNewLabel_4 = new JLabel(""); //koko
panel_2.add(lblNewLabel_4);
```

図12．ローカル変数の宣言変更

⑥作成済みのスロットマシンの数字とアニメーションの画像を配置します。

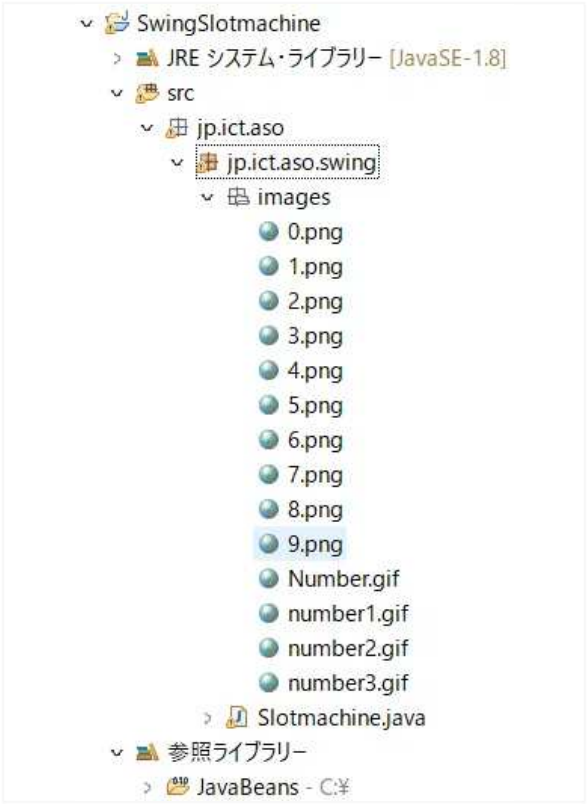

図13．画像とアニメーションの配置

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。

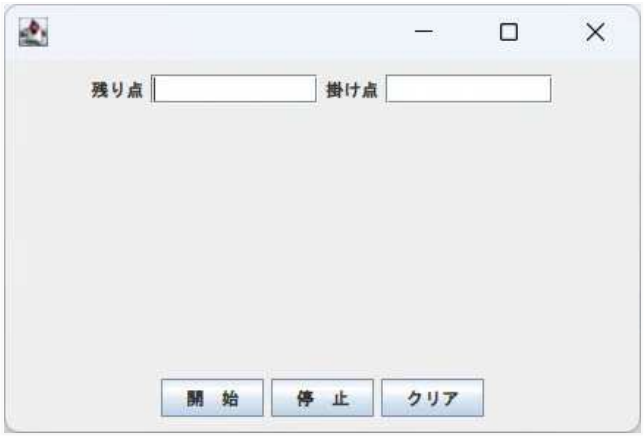

図14．スロットマシン起動画面

⑧残り点と掛け点を入力してボタンをクリックしアプリの処理の流れを確認します。

図15．スロットマシンの開始

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」「起動時に残り点と掛け点が入っていない」「起動時にすべてのボタンが有効になっている」「ラベルや点数の表示が画像に対して小さい」「残り点を任意に変更できる」の８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起動時に画像を表示します。
④起動時は開始ボタンとクリアボタンを有効にします。
⑤フレーム以外のテキストのフォントは18ポイントに設定します。
⑥起動時及びクリア時の残り点を10、掛け点を１で設定します。
⑦残り点のテキストフィールドは編集不可に設定します。

実行して動きを確認します。これで完成です。

図16．スロットマシン完成

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
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

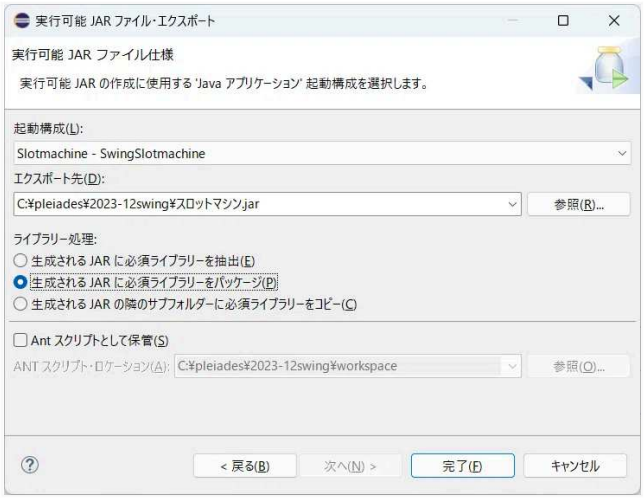

図17．実行可能JARファイルの設定

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。

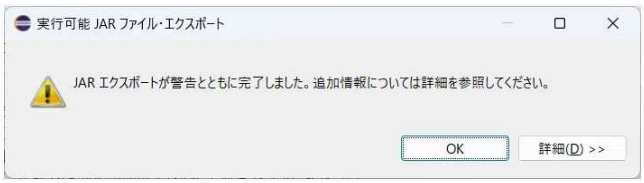

図18．警告ダイアログ

作成されたjarファイルをダブルクリックするとアプリが起動します。

図19．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！

ソースコード例と画像例

以下に各プログラムのソースコードの例（本文内では【ヒント】になっている部分の完全なソースコードの例）を示しますので、実装の参考にしてください。

また、プログラム中で使用するSlotmachineBean.class及びイメージ画像やアニメーションを次のセクションに置きますので、ダウンロードして使用してください。

SlotmachinBean.class

SlotmachineBean.class

1.65 KB

ファイルダウンロードについて

ダウンロード

images777.zip

images777.zip

1.81 MB

ファイルダウンロードについて

ダウンロード