



Swingによるデスクトップアプリケーション開発（マネーシミュレータ）-JavaSE1.8



office・M

2024年9月29日 10:05

...

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）を使ってマネーシミュレータを作成します。元本投資・積立投資・貸付返済の3つのシミュレーション機能を持っています。計算ロジックはWebアプリケーション開発で利用した独自クラスを再利用します。

『Swingによるデスクトップアプリケーション開発（元本投資）-JavaSE1.8』を終了していることが前提条件です。

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

GUI設計のポイント

内部設計

処理ロジック

提供クラス

実装準備

プロジェクトの作成

外部classの利用準備

ビルド・パスに追加

実装

GUI実装

イベント実装

提出課題

単独起動

実行可能JARファイル

最後に

外部設計

『Swingによるデスクトップアプリケーション開発（元本投資）-JavaSE1.8』の外部設計をもとにしてWindowBuilderのSwingデザイナーで図1のようなGUIを作成します。

図1. マネーシミュレータ画面

以下のように元本投資、積立投資、貸付返済のシミュレーション機能を持たせます。

マネーシミュレータ

元本・積立額 or 借... 100 万円

年率 or 手数料率 0.3 %

運用期間 or 返済金額 10 年・月 or 円

☒ 元本投資(年) ☐ 積立投資(月) ☐ 貸付返済(月)

野村証券(元本投資) 年複利 四捨五入 非課税

期間	運用収益	元本+運用収益
1年目	¥3,000	¥1,003,000
2年目	¥6,009	¥1,006,009
3年目	¥9,027	¥1,009,027
4年目	¥12,054	¥1,012,054
5年目	¥15,090	¥1,015,090
6年目	¥18,135	¥1,018,135
7年目	¥21,189	¥1,021,189
8年目	¥24,253	¥1,024,253
9年目	¥27,326	¥1,027,326
10年目	¥30,408	¥1,030,408

計算 クリア

図1.1 元本投資シミュレーション画面

マネーシミュレータ

元本・積立額 or 借... 5 万円

年率 or 手数料率 0.3 %

運用期間 or 返済金額 10 年・月 or 円

☐ 元本投資(年) ☒ 積立投資(月) ☐ 貸付返済(月)

金融庁(積立投資) 月複利 四捨五入 非課税

期間	元本+運用収益
1ヶ月目	¥50,000
2ヶ月目	¥100,013
3ヶ月目	¥150,038
4ヶ月目	¥200,076
5ヶ月目	¥250,126
6ヶ月目	¥300,189
7ヶ月目	¥350,264
8ヶ月目	¥400,352
9ヶ月目	¥450,452
10ヶ月目	¥500,565

計算 クリア

図1.2 積立投資シミュレーション画面

月数	残金
1か月目	¥982,500
2か月目	¥964,781
3か月目	¥946,840
4か月目	¥928,675
5か月目	¥910,283
6か月目	¥891,661
7か月目	¥872,806
8か月目	¥853,716
9か月目	¥834,387
10か月目	¥814,816

図1.3 貸付返済シミュレーション画面

GUI設計のポイント

実行経過の履歴はhtml形式でテキストフィールドに保存されています。よって、表示する際はhtmlのタグを解釈できる**JEditorPane**を使います。プロパティの**contentType**を **text/html** に設定することでhtmlによる文字飾り等が利用できるようになります。

シミュレーションを行う場合、運用期間（or返済金額）の入力値によって結果の出力量が変わってきます。よって、JEditorPaneは**JScrollPane**内で縦スクロールが可能なように配置します。その際Window枠全体で表示されるよう中央パネルのレイアウトは**BoxLayout**に変更しておきます。

『Swingによるデスクトップアプリケーション開発（元本投資）-JavaSE1.8』をもとに上部パネルのテキスト領域をラジオボタンに変更してシミュレーションの機能を選択するようにします。入力のテキストフィールドは変更せずに、機能ごとに意味を持たせることにします。

内部設計

元本投資の計算と履歴表示を行う外部クラス（InvestmentSimBean.class）を『MVCモデルによるWebアプリケーション開発（No 5.元本投資）-EE8』の記事より取得してください。Eclipseのビルドパスに追加することで、当該クラスを利用できるようになります。

積立投資の計算と履歴表示を行う外部クラス（AccumulationSimBean.class）を『MVCモデルによるWebアプリケーション開発（No 6.積立投資）-EE8』の記事より取得してください。Eclipseのビルドパスに追加することで、当該クラスを利用できるようになります。

貸付返済の計算と履歴表示を行う外部クラス（LoanSimBean.class）を『MVCモデルによるWebアプリケーション開発（No 7.貸付返済）-EE8』の記事より取得してください。Eclipseのビルドパスに追加することで、当該クラスを利用できるようになります。

処理ロジック

- ・ラジオボタンの選択で元本投資、積立投資、貸付返済のシミュレーション機能を選択します。
- ・元本（借入金）、利率（手数料率）、運用期間（返済金額）の入力領域はそれぞれの機能で共有します。
- ・計算ボタンをクリックすることでアクションイベントを発生させます。
- ・イベント内で入力値からそれぞれの機能の外部クラスに引数を与えて実体化しシミュレーションを行います。
- ・シミュレーション結果は表示領域に張り付けます。

提供クラス

提供される元本投資計算ロジック（InvestmentSimBean.class）、積立投資計算ロジック（AccumulationSimBean.class）、貸付返済計算ロジック（LoanSim.class）の使い方とクラス図は以下のようになります。

元本投資計算ロジック（InvestmentSimBean.class）

（使い方）

- ・InvestmentSimBeanクラスを**元本**と**利率**と**期間**の3つの引数をもつコンストラクタで実体化します。
- ・simulationメソッドを実行することでsimフィールドに運用収益の履歴がhtml文字列で表組されます。
- ・利用側のクラスはgetSim()メソッドで運用収益の履歴を取得します。

（クラス図）

package:jp.ict.aso.model	
InvestmentSimBean	
- ganpon:int	//入力単位は円（元本）
- riritu:double	//入力単位は%の実数値（利率）
- kikan:int	//入力単位は年（期間）
- sim:String	//運用収益の履歴（html形式）
+ InvestmentSimBean(ganpon:int,riritu:double,kikan:int):	
+ simulation():void	//シミュレーション実施
+ getSim():String	//結果の取得
+ getGanpon():int	//設定された元本取得
+ getRiritu():double	//設定された利率取得
+ getKikan():int	//設定された期間取得
-:private +:public #:protected	

積立投資計算ロジック（AccumulationSimBean.class）

（使い方）

- ・ AccumulationSimBeanクラスを**積立金額**と**想定利回り**と**積立期間**の3つの引数をもつコンストラクタで実体化します。
- ・ simulationメソッドを実行することでsimフィールドに運用収益の履歴がhtml文字列で表組されます。
- ・ 利用側のクラスはgetSim()メソッドで運用収益の履歴を取得します。

（クラス図）

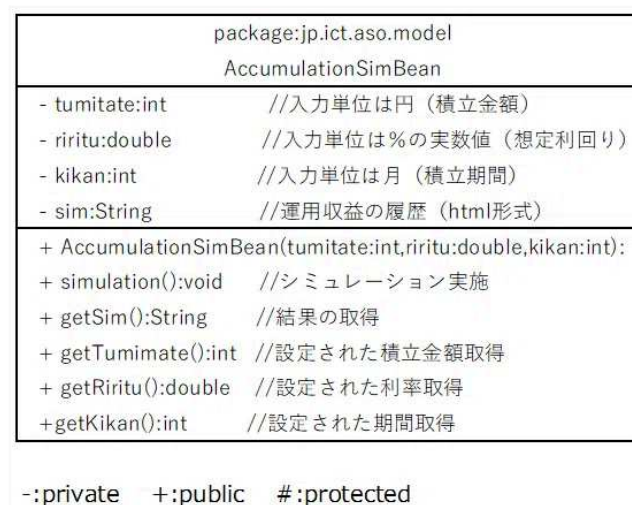


図2. AccumulationSimBeanクラス図

貸付返済計算ロジック（LoanSim.class）

（使い方）

- ・ LoanSimBeanクラスを**借入残額**と**手数料率（年利率）**と**返済金額**の3つの引数をもつコンストラクタで実体化します。
- ・ simulationメソッドを実行することでsimフィールドに借入残金の履歴がhtml文字列で表組されます。
- ・ 利用側のクラスはgetSim()メソッドで借入残金の履歴を取得します。

（クラス図）

package:jp.ict.aso.model	
LoanSimBean	
- zankin:int	//入力単位は円（借入残額）
- riritu:double	//入力単位は%の実数値（年利率）
- hensai:int	//入力単位は円（返済額）
- sim:String	//借入残金の履歴（html形式）
+ LoanSimBean(zankin:int,riritu:double,hensai:int):	
+ simulation():void	//シミュレーション実施
+ getSim():String	//結果の取得
+ getZankin():int	//設定された借入残額取得
+ getRiritu():double	//設定された利率取得
+ getHensai():int	//設定された返済額取得
-:private +:public #:protected	

図3. LoanSimBeanクラス

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→「SwingMoney」プロジェクトを作成する

→図4の内容で設定する

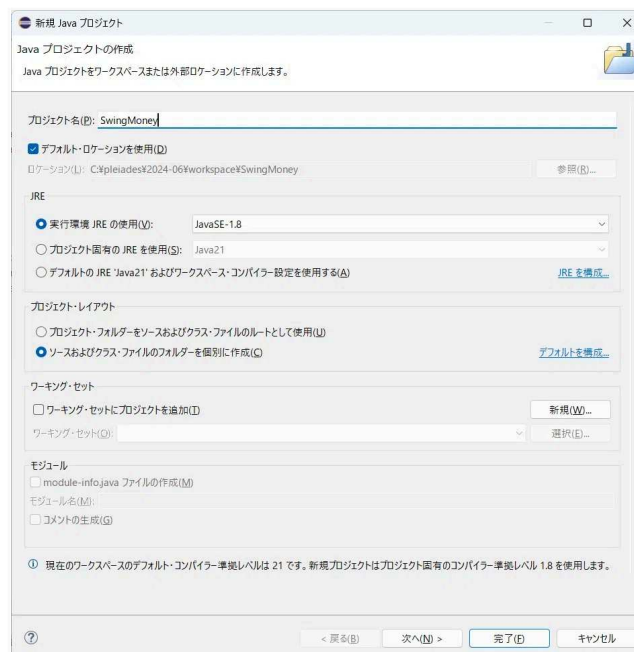


図4. SwingMoneyプロジェクト設定

※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「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**InvestmentSimBean.class**、**AccumulationSimBean.class**、**LoanSim.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:¥提供Bean ¥jp ¥ict ¥aso ¥model ¥〇〇〇.class）

ビルド・パスに追加

InvestmentSimBean.class、AccumulationSimBean.class、LoanSim.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
→「提供Bean」を選択→最後に「適用して閉じる」をクリック

※図4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

GUI実装

『Swingによるデスクトップアプリケーション開発（元本投資）-JavaSE1.8』で作成したプログラムを変更して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図6のデザインイメージようにGUIを変更します。

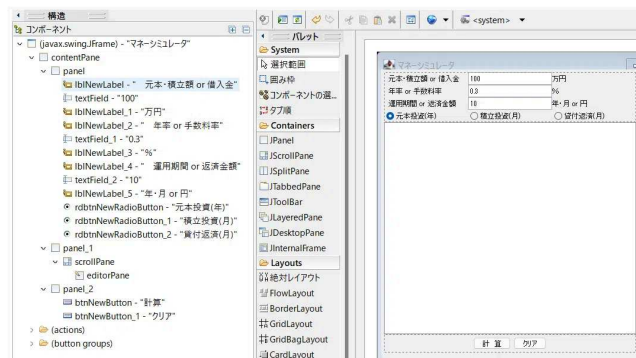


図6. 変更後の画面デザイン

イベント実装

ソースタブに変更します。

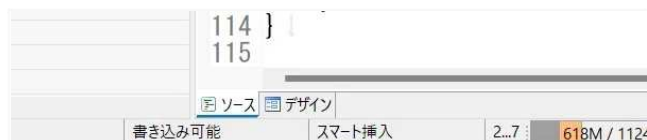


図7. ソースタブに変更

①「計算ボタン」のイベントのソース部分を変更します。ラジオボタンの選択状況により実体化するBeanを変更します。

以下の元本投資のソースコードをヒントに考えましょう！

```
if(rdbtnNewRadioButton.isSelected()) {  
    //元本・利率・運用期間をBeanの仕様通りに変換  
    int ganpon = Integer.parseInt(textField.getText())*10000;//単位が万円  
    double riritu = Double.parseDouble(textField_1.getText())/100.0;//単位が%  
    int kikan = Integer.parseInt(textField_2.getText());  
    //一旦表示領域クリア  
    sim="";  
}
```

```
editorPane.setText(sim);

// シミュレーション結果の作成
InvestmentSimBean isb = new InvestmentSimBean(ganpon,riritu,kikan);
isb.simulation();
sim=isb.getSim();
editorPane.setText(sim);
}
```

元本投資のシミュレーション結果は参照先 1 に示す「野村証券のマネーシミュレータ（みらい電卓）」のWebサイトと全く同じ結果になりますので以下のurlで確認してください。

（参照先 1）

利息元加処理：年複利 元加方式：四捨五入 期間：10年
利率：0.3% 元本：100万円 野村証券
※実行結果とぴったり同じになります
<https://www.nomura.co.jp/hajimete/simulation/unyou.html>

積立投資のシミュレーション結果は参照先 2 に示す「カシオ」のWebサイトと全く同じ結果になりますので以下のurlで確認してください。

（参照先 2）

利息元加処理：月複利 元加方式：四捨五入 期間：1 年
利率：0.3% 積立額：5 万円 非課税 カシオ
※実行結果とぴったり同じになります
<https://keisan.casio.jp/exec/system/1254841870>

また、金融庁のWEBサイトでも確認可能です。

（参照先 3）

金融庁 資産運用シミュレーション
https://www.fsa.go.jp/policy/nisa2/moneyplan_sim/index.html

貸付返済のシミュレーション結果は参照先 4 に示す「アプラス」のWebサイトと全く同じ結果になりますので以下のurlで確認してください。

（参照先 4）

アプラス（新生銀行） 元利定額リボルビング払いシミュレーション
<https://www.aplus.co.jp/creditcard/revo/simulation/index.html>

②ラジオボタンの選択状況によりシミュレーションが変更されることを確認して完成です。

マネーシミュレータ

元本・積立額 or 借入金: 100 万円

年率 or 手数料率: 15 %

運用期間 or 返済金額: 30000 年・月 or 円

☐ 元本投資(年)
 ☐ 積立投資(月)
 ☒ 貸付返済(月)

35か月目	¥237,487
36か月目	¥210,455
37か月目	¥183,085
38か月目	¥155,373
39か月目	¥127,315
40か月目	¥98,906
41か月目	¥70,142
42か月目	¥41,018
43か月目	¥11,530
最終返済額:	¥11,674
最終返済回:	44回
総返済額:	¥1,301,674

計算 クリア

図8. マネーシミュレータ完成

提出課題

以下の条件は「NISAで長期積み立てを行うシミュレーション」の例です。
条件に従い今回のプログラムでシミュレーションを行い結果を提出してください。

(条件)

毎月積み立て金額：5,000円(0.5万円)

想定利回り(年率)：3.0%

積立期間：20年(240カ月)

※金融庁の「つみたてシミュレータ」によるとざっくりと164万円になるようですが、今回のプログラムを使って1円の単位まで計算してください。

毎月の積立金額: 0.5 万円

想定利回り(年率): 3 %

積立期間: 20 年

計算する

将来の運用資産額

164 万円

図9. 金融庁つみたてシミュレータの例

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
Swing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

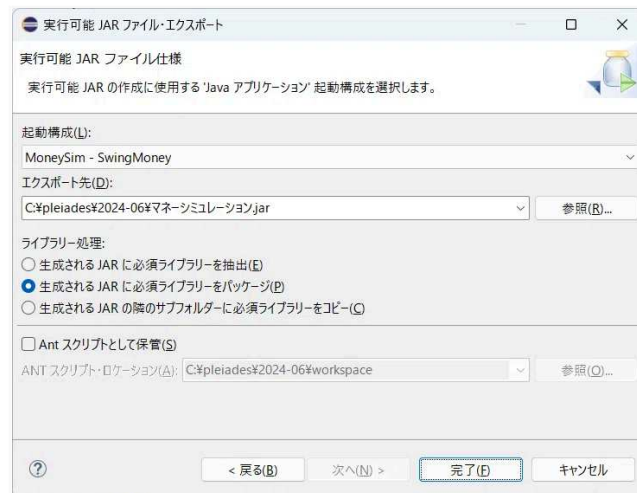


図10. 実行可能JARファイルの設定

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。

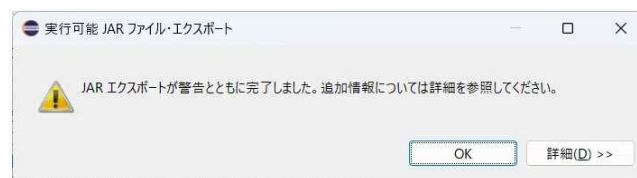


図11. 警告ダイアログ

作成されたjarファイルをダブルクリックするとシミュレーションが起動します。



図12. 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元本投資、積立投資、貸付返済のシミュレーション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。

ただし、このアプリを実装するにはそれぞれのシミュレーションを実行するクラスファイルが必要です。このクラスは

『MVCモデルによるWebアプリケーション開発（No 5 .元本投資）-EE8』 『MVCモデルによるWebアプリケーション開発（No 6 .積立投資）-EE8』 『MVCモデルによるWebアプリケーション開発（No 7 .貸付返済）-EE8』

の3つの記事から取得可能です。

また、『Swingによるデスクトップアプリケーション開発（元本投資）-JavaSE1.8』を終了していることが前提条件です。