AiSee の Windows アプリ作成マニュアル
【基本編】

第 1 版

株式会社ベータ・ネット

作成者：石毛、大山、増子、伊藤

作成日：2022/10/28

[目次]

改訂履歴

版数	ページ番号	内容	作成日/改定日	作成者/改訂者
1			2022/10/28	

1　この資料の目的

この資料では、初めて AiSee に触れる方を想定し、AiSee の基本的な操作方法を紹介いたします。

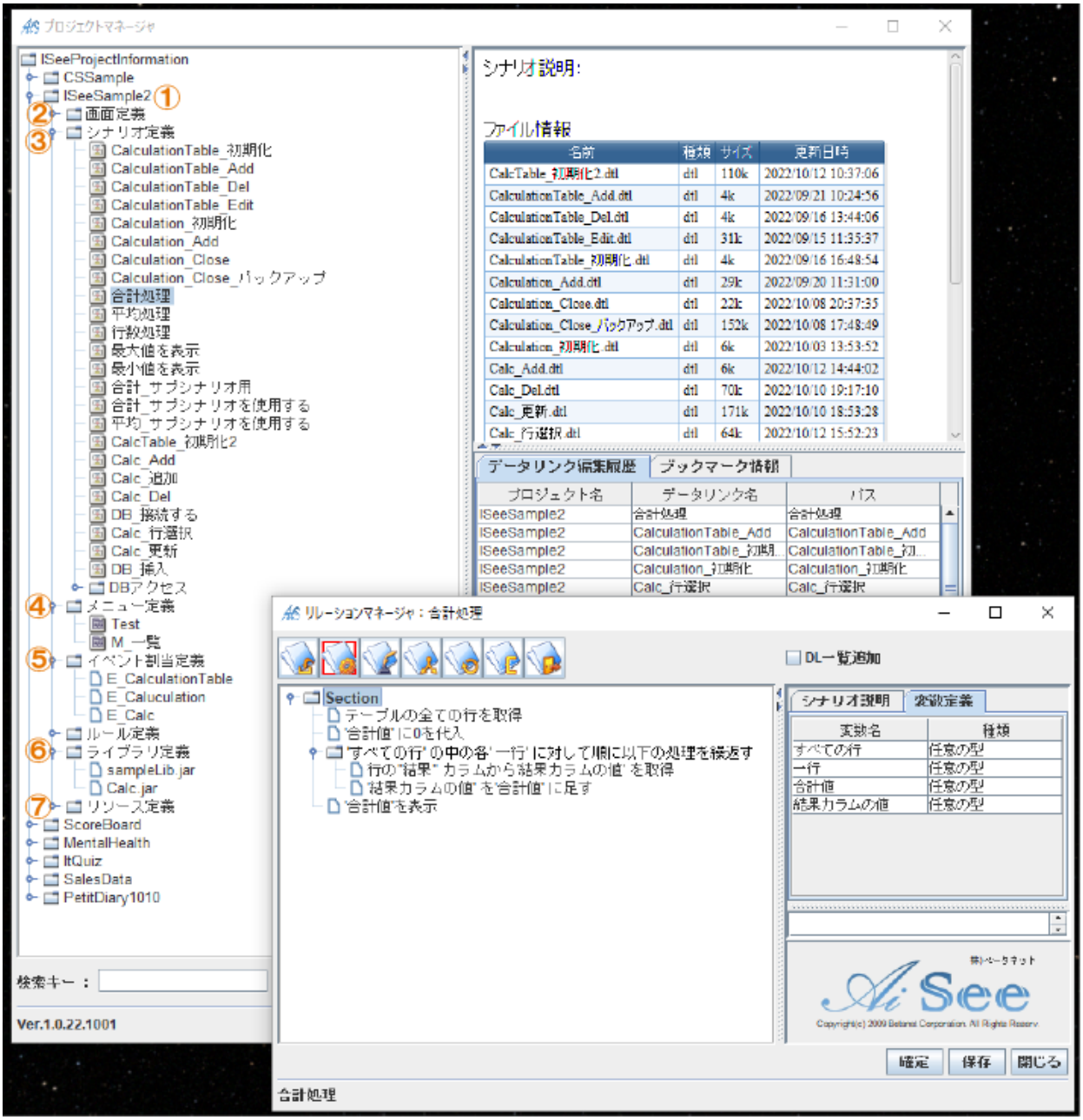
実際の作成例を通して、Windows アプリケーションを作成するための手順を中心に記載しています。

2　使用するアプリケーション

必要なアプリケーション	用途
AiSee	AiSee の Windows アプリ開発ツール（IDE)
NetBeans	画面表示用 Jar ファイル作成のために使用するアプリ

2.1　AiSee の画面全体の説明

まずは、これから操作することになる AiSee の画面の全体像を紹介します。
細かい内容は次章以降、順番に説明していきますので、現状は理解ができなくても問題ありません。
そのままアプリ作成に進んでください。



①プロジェクトフォルダ

アプリごとに作成されるフォルダです。図には複数のプロジェクトのファイルが表示されています。

AiSee では何種類ものアプリを同時作成したり、内容を他のアプリに流用したりする事も可能です。

②画面定義

画面イメージが格納されます。

③シナリオ定義

ボタンが押された時や、アプリが表示された時に何が起こるか等、実際の処理内容が記載される部分です。アプリが動く仕組みがここで記述されるため、この部分の作成が作業の大きなウェイトを占めます。

④メニュー定義

「右クリックで画面にはないメニューが表示される」といった設定を任意で加えることができます。

⑤イベント割当定義

「どのボタンを押すとどの処理が発動するか」など処理をボタンなどの部品に紐づけます。

⑥ライブラリ定義

別ソフトで作成した画面イメージのファイルである jar ファイル（〇〇〇jar）を登録します。

⑦リソース定義

データベースを使うアプリを作る場合のみ、データベースのファイル（〇〇〇.db）を登録します。

3　AiSee を使った Windows アプリ開発の全体像

アプリの開発から確認まで、全ての作業フェーズを紹介します。

開発環境設置

- 開発環境及びソフトをインストールする

  AiSee ウインドウズの開発ツールをインストールする

  NetBeans 及び JDK をインストールする

  ※NetBeans のインストールについては本資料では省略します。

アプリの画面設計作業

- NetBeans を利用して、アプリの画面を作成する

  ※本資料では省略、アプリの画面は事前準備されています。

アプリの開発作業

AiSee のウインドウズ開発ツール（IDE）で、下記の手順を行う

- プロジェクトを作成する
- アプリ画面の Jar ファイルを登録する
- シナリオを作成する
- イベントを定義する

- メニューを定義する

デバッグ

- 起動設定を行う
- テスト実行して画面動作を確認する

Windows アプリの完成

3.1　サンプルアプリの完成図

この手順書では、下図の「Calculation」と「CalculationTable」の 2 つの画面を使用し四則演算と、計算データを表へ格納する処理を行うアプリを作成します。

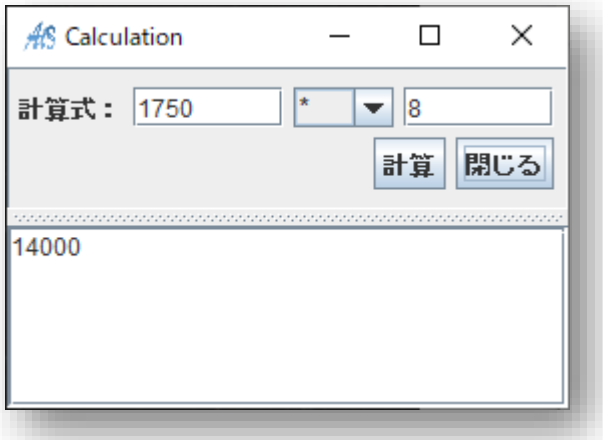
① 実行すると、「CalculationTable」が現れる (まだ数字等のデータがない、空の状態)

②「CalculationTable」の追加ボタンを押すと「Calculation」が現れる

③ 「Calcuation」の計算式の部分に数字を入力し、計算ボタンを押すと、下に計算結果が表示される

＜ Calculation　Table＞

＜ Calculation ＞



【このアプリの仕組み】

①「CalculationTable」が現れる（まだ数字等のデータがない、空の状態）

②「CalculationTable」の追加ボタンを押すと「Calculation」が現れる

③　「Calcuation」の計算式の部分に数字を入力し、計算ボタンを押すと、下に計算結果が表示される

④　「Calculation」の閉じるボタンを押すと、「Calculation」に入力してあった数字、演算子、計算結果が

「CalculationTable」の方に追加され、表示される

※上記が完了しましたら、修正ボタンや削除ボタンへの機能の追加もお試しください。

4　AiSee Windows アプリの作成手順

4.1　環境設置手順

①　AiSeeIDE をインストールする

②　NetBeans をインストールする

※今回は作成済みの Jar ファイルをダウンロードしていただくため、不要です。

4.2　画面作成

用意されたサンプルアプリ画面のファイル(Jar ファイル)を、以下よりダウンロードしてください。

Jar ファイルのダウンロード

※ここではアプリの画面の作成や NetBeans についての詳細説明は行いません。

尚、NetBeans という別のソフトを使用してアプリの画面を作成する場合の操作については下記リンクを参照してください。

NetBeans でアプリの画面を作成する方法

4.3　アプリの作成手順

Windows アプリを作成する手順を紹介します。

4.3.1 プロジェクトファイルの作成

まずは、AiSeeIDE(以下 AiSee)を起動してください。

次にプロジェクト名を決定・入力し、作業スペースを作成します。

この段階で入力した「プロジェクト名称」は後から変更ができないため、慎重に検討されることをおすすめします。

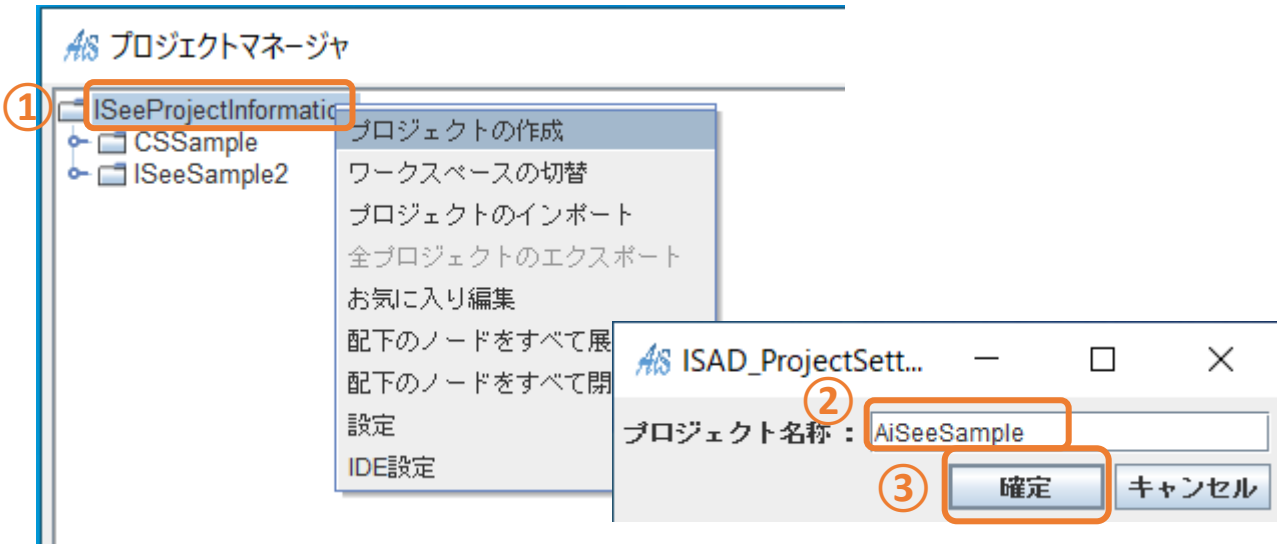
(1)新しいプロジェクトの作成手順【図 1】

①「ISeeProjectInfomation」右クリック

②プロジェクト名「AiSeeSample」を入力

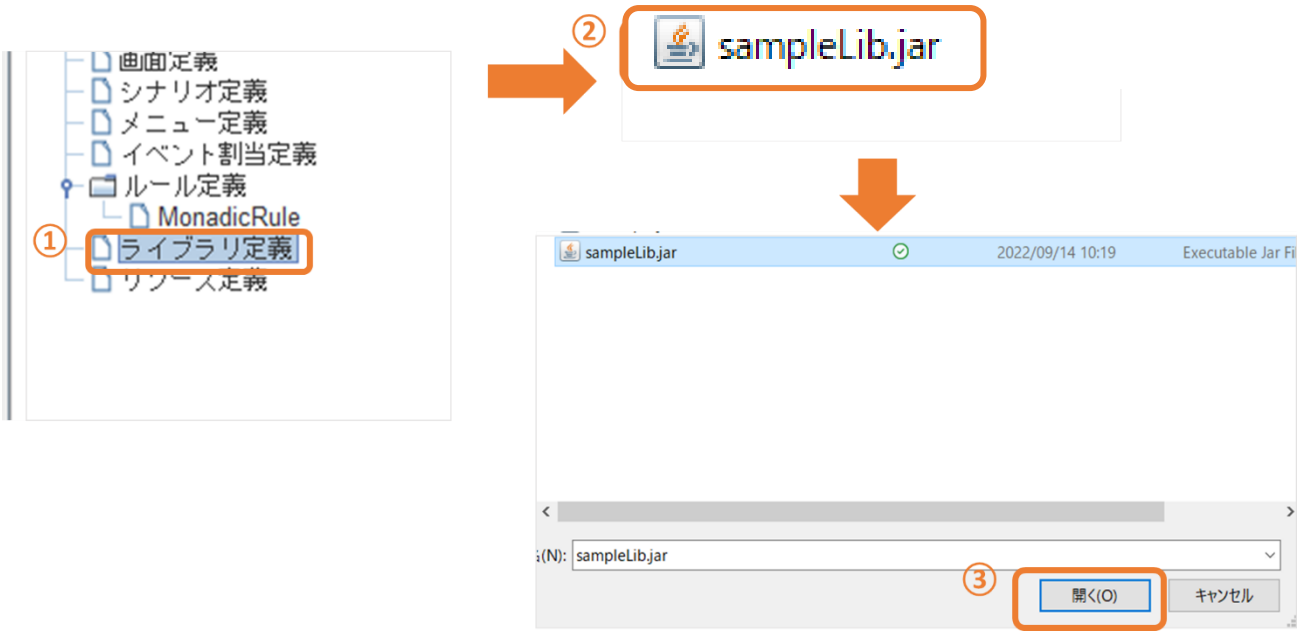
③「確定」をクリック

【図 1】



(２)ライブラリ定義【図 2】
①「ライブラリ定義」を右クリックし「ライブラリ登録」をクリック
②ダウンロードしたファイル「sampleLib.jar」を選択
③「開く」ボタンクリック
ライブラリ定義直下に「 sampleLib.jar 」が反映されていれば OK

【図 2】



(３)画面の登録【図 3～図 4】

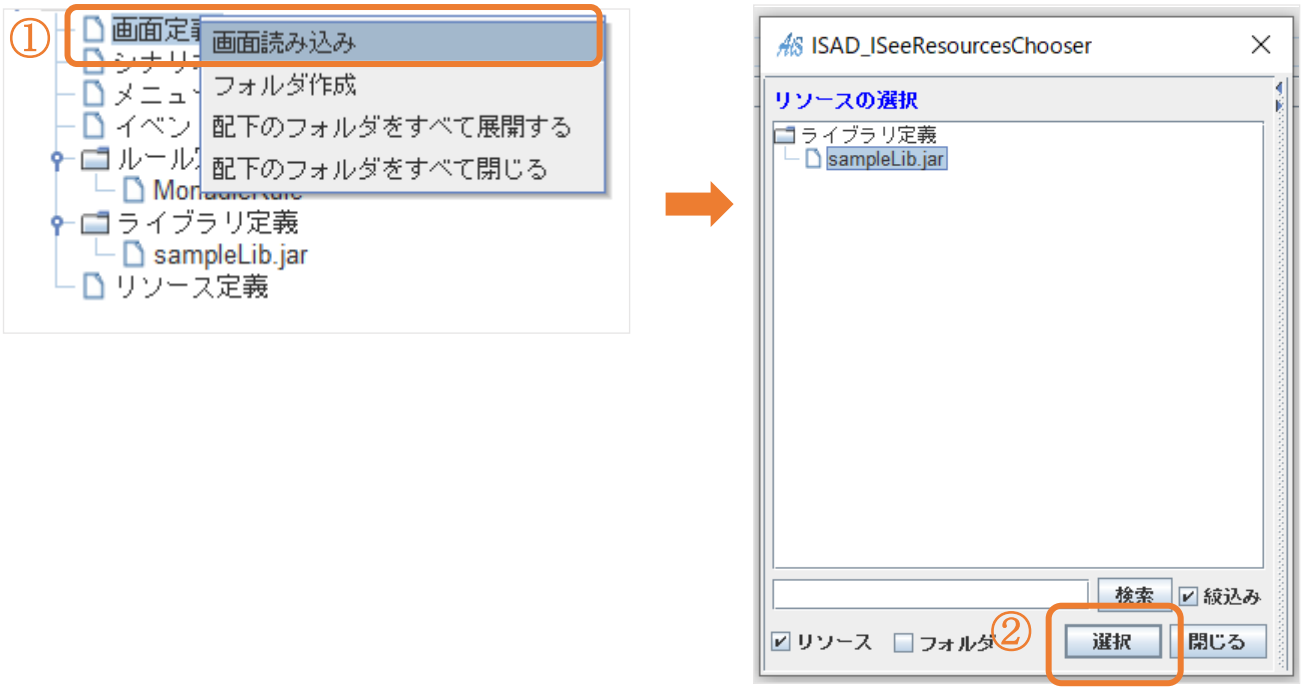
①「画面定義」を右クリック、「画面読み込み」クリック

②「sampleLib.jar」を選択して、右下「選択」ボタンを押下

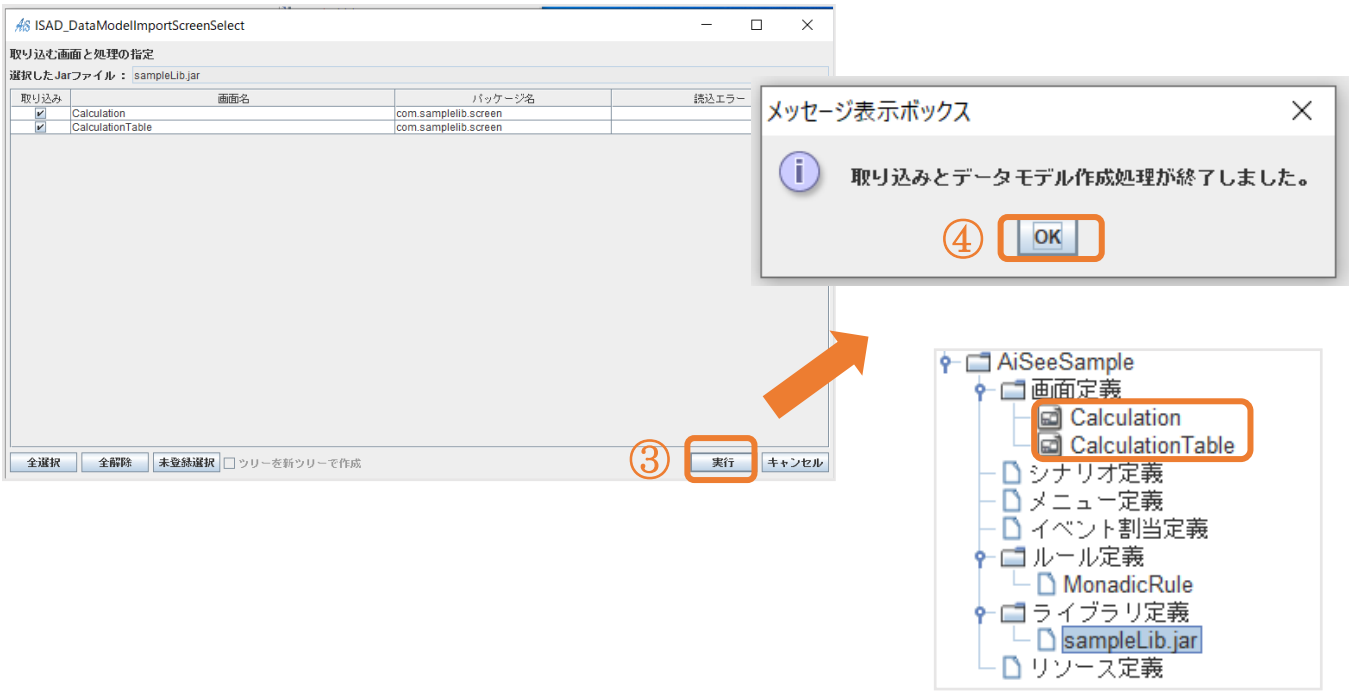
③　右下「実行」ボタン押下→「OK」押下

④「画面定義」直下に「Calculation」と「CalculationTable」が追加されていれば OK

【図 3】



【図 4】



5　プロジェクト内容の作成手順

ここからは、「AiSeeSample」というプロジェクトの内容を作成していきます。

もし作成中に困った事があった場合、Q&A に解決方法の記載がないかご確認ください。

5.1 シナリオの作成

「シナリオ」とは処理のことです。

シナリオ作成によって、計算する、変数を作る等、多種多様なアプリの動作を作っていきます。

1 つのシナリオの内部に、足し算や引き算、ループや条件付け分岐（もし〇〇なら△△）など

具体的処理内容が記載できます。複数のシナリオを組み合わせて、目的の処理全体が完了するよう作成します。

例えば、画面を初期化するシナリオであれば、アプリが開くときの画面のサイズを規定したり、

テキストフィールドを空の状態にリセットしておくといった処理を 1 つのシナリオの内部に複数記載します。

ここではまず、シナリオの「項目名」のみを先に作成します。

作成する内容は、以下の画像の通りです。

※シナリオの並び順は、処理の順番ではないため、この通りにならなくとも問題はありません。

シナリオの「内容」の作成手順については下記リンク先を参照してください。

イベント毎のシナリオの作成

シナリオ作成【図 5】

「シナリオ定義」右クリックして「シナリオ新規作成」を押下→シナリオ名を入力

「CalculationTable_初期化」とし「OK」ボタンを押下

【図 5】

5.2　イベント定義の指定

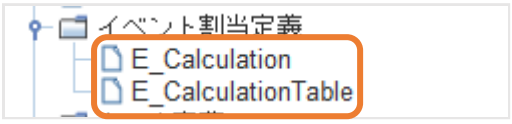
イベントとは、「OO をしたら指定したシナリオ処理が実行される」というものです。

ここでは「Calculation の計算ボタンを押すと Calculation_Add が実行される」設定、

「Calculation の閉じるボタンを押すと Calculation_Close が実行される」設定、

「CalculationTable の追加ボタンを押すと Calculation が表示される」etc…の設定を行います。

下記の画像の通りに作成します。

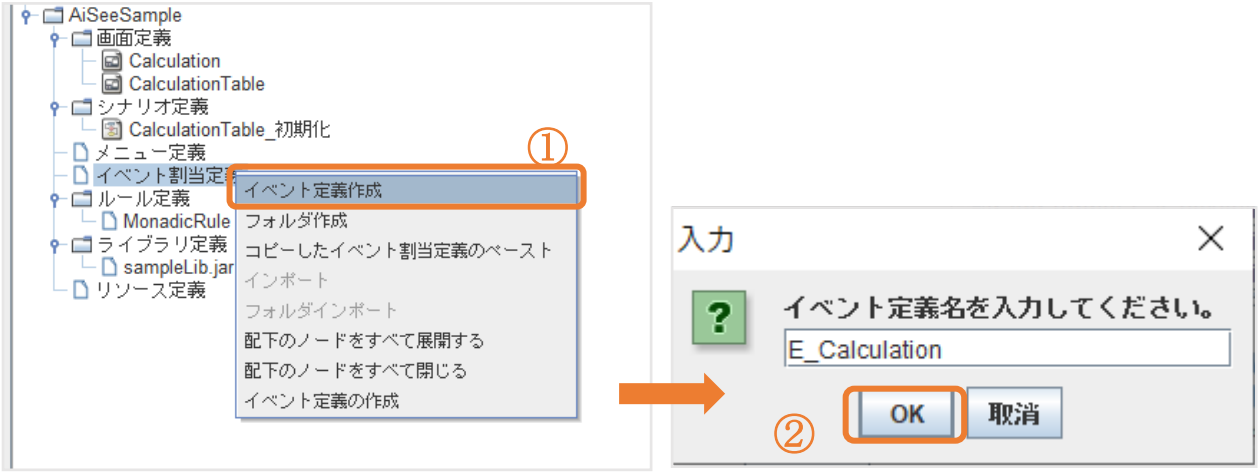


イベント定義作成

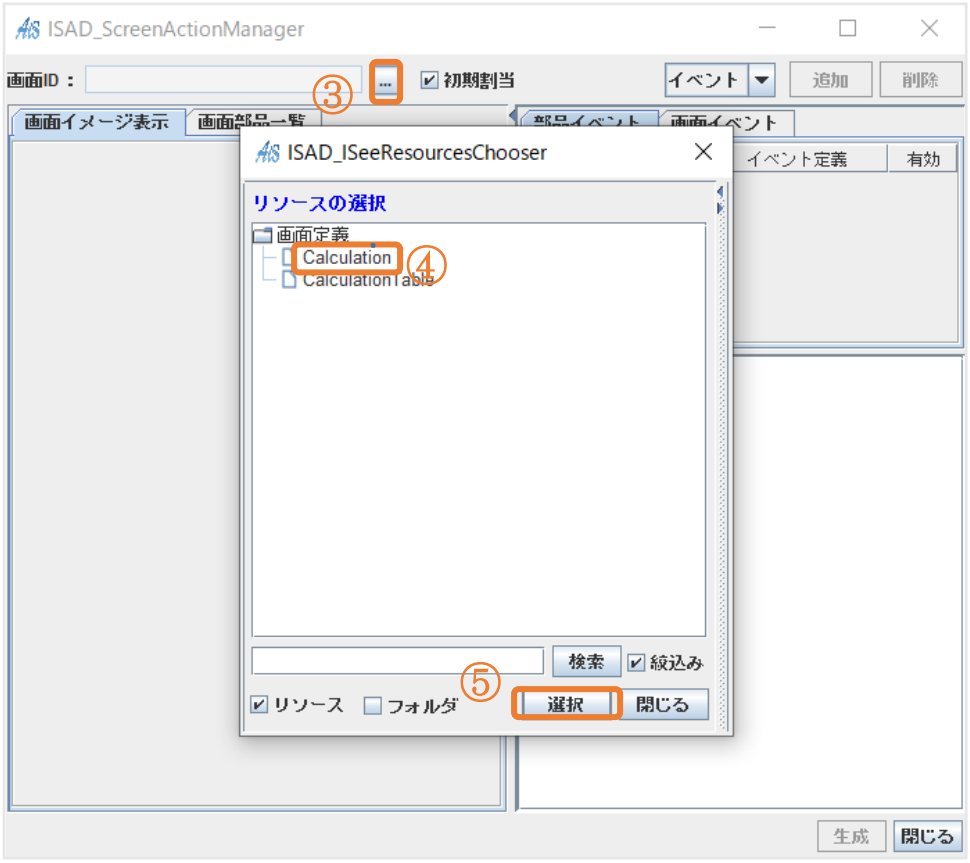
「イベント割当定義」を右クリックして「イベント定義作成」

「イベント名」を入力して「OK」ボタン

【図 6】

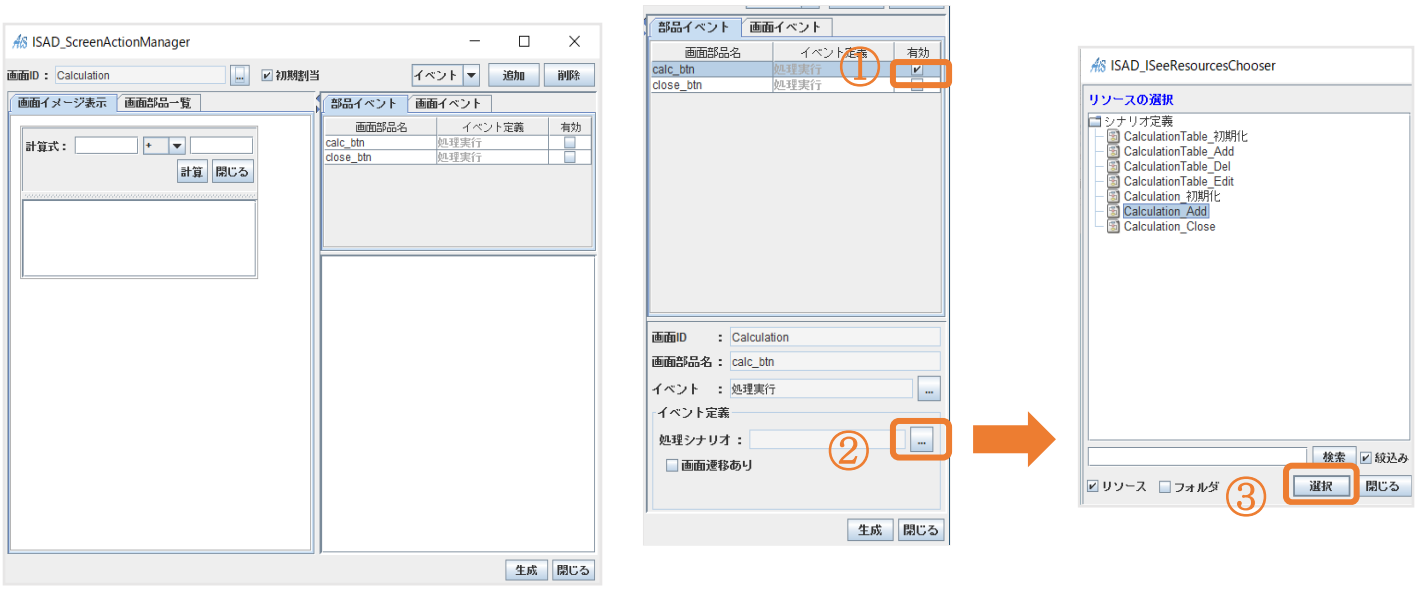


【図 7】



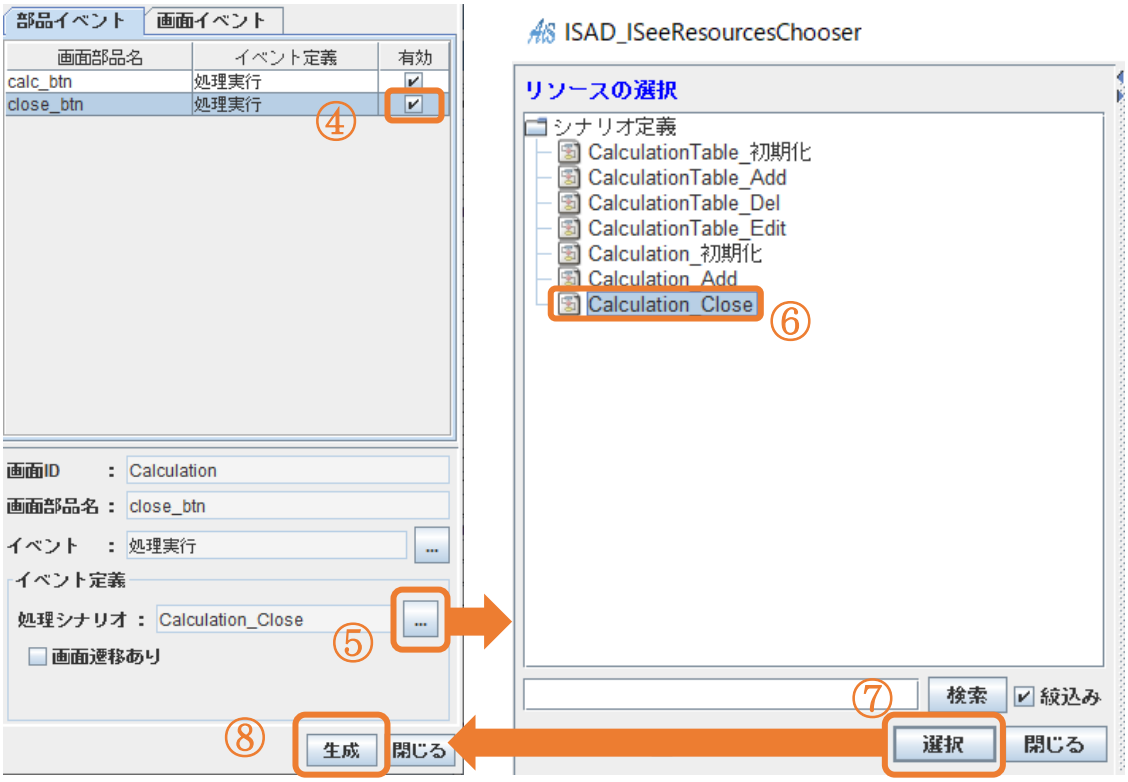
E_CalculationTable を作成する際は④で「CalculationTable」を選択してください。

【図 8】Calculation の設定



CalculationTable_Add の章から来た場合はこちらから戻れます。

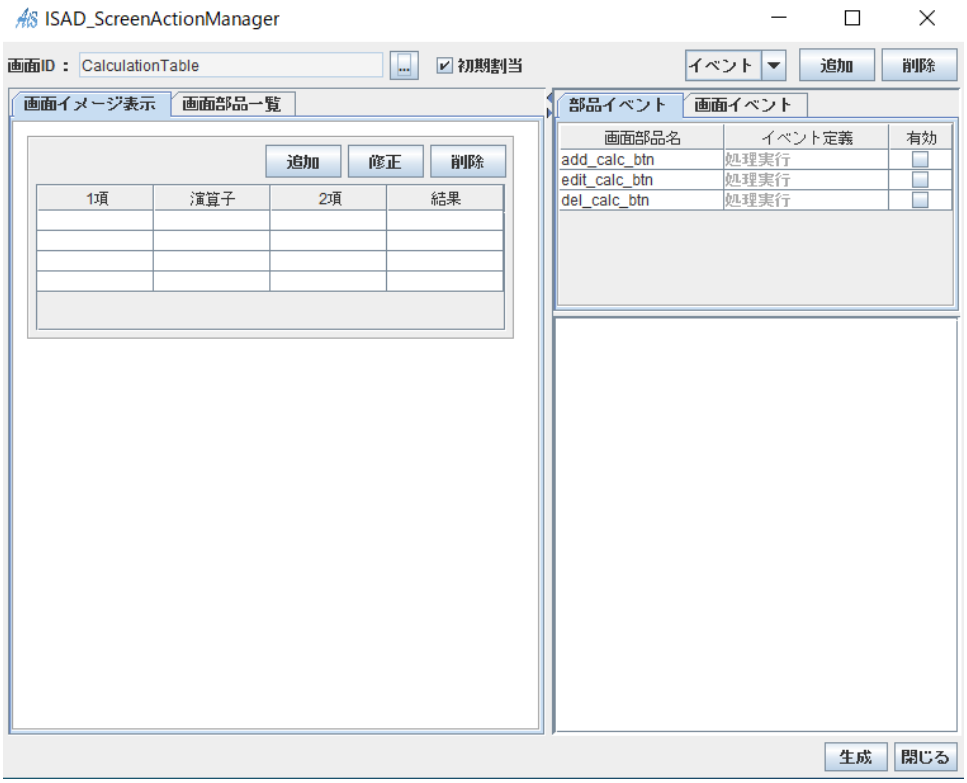
【図 9】



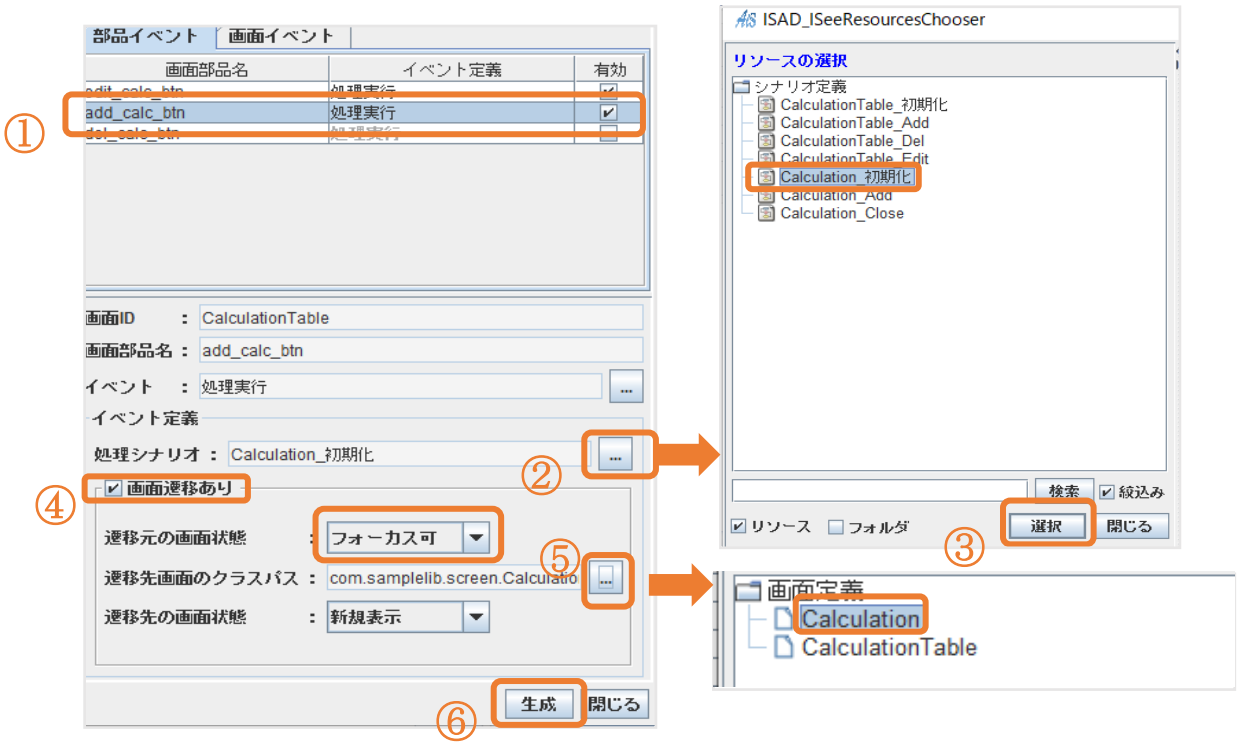
CalculationTable_Close の章から来た場合はこちらから戻れます。

同じ手順で「E_CalculationTable」を作成してください。

【図 10】CalculationTable の設定

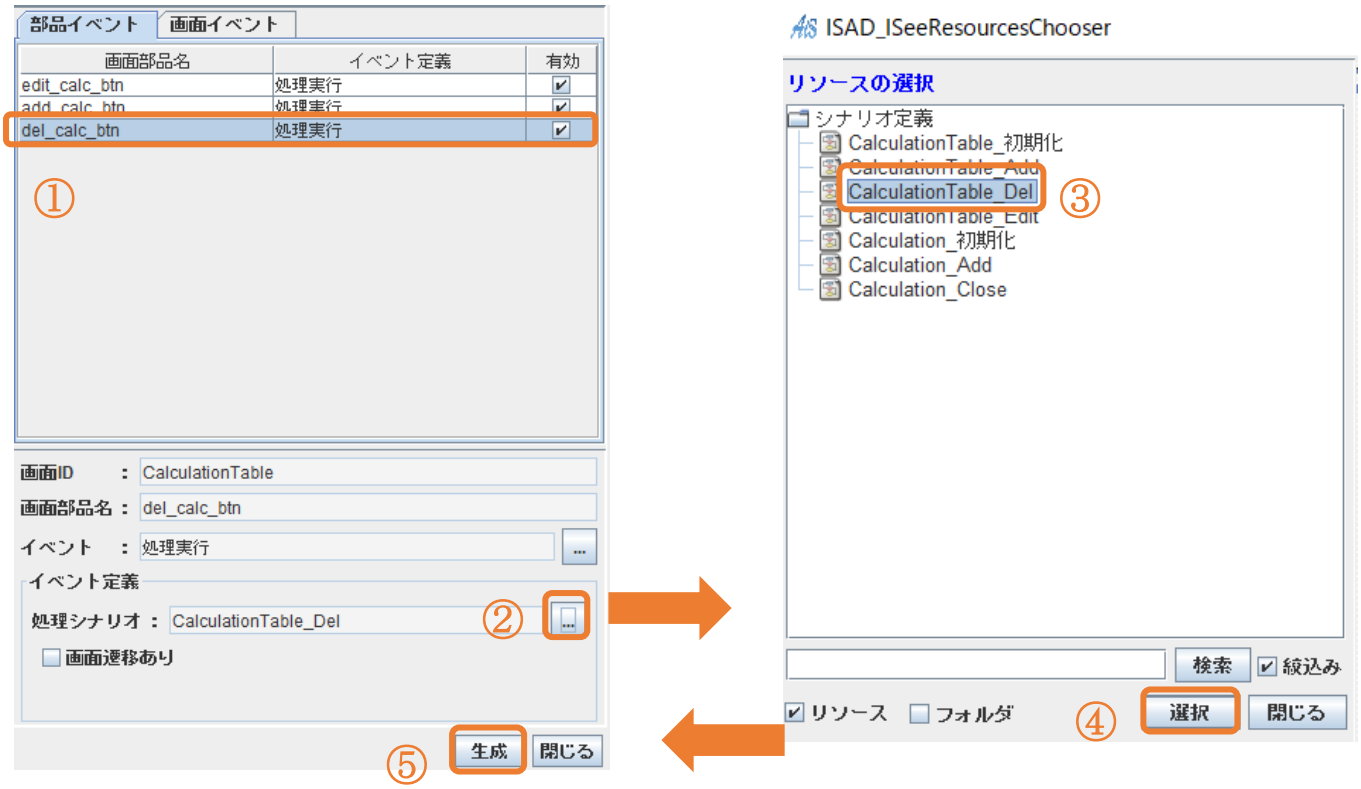


【図 11】



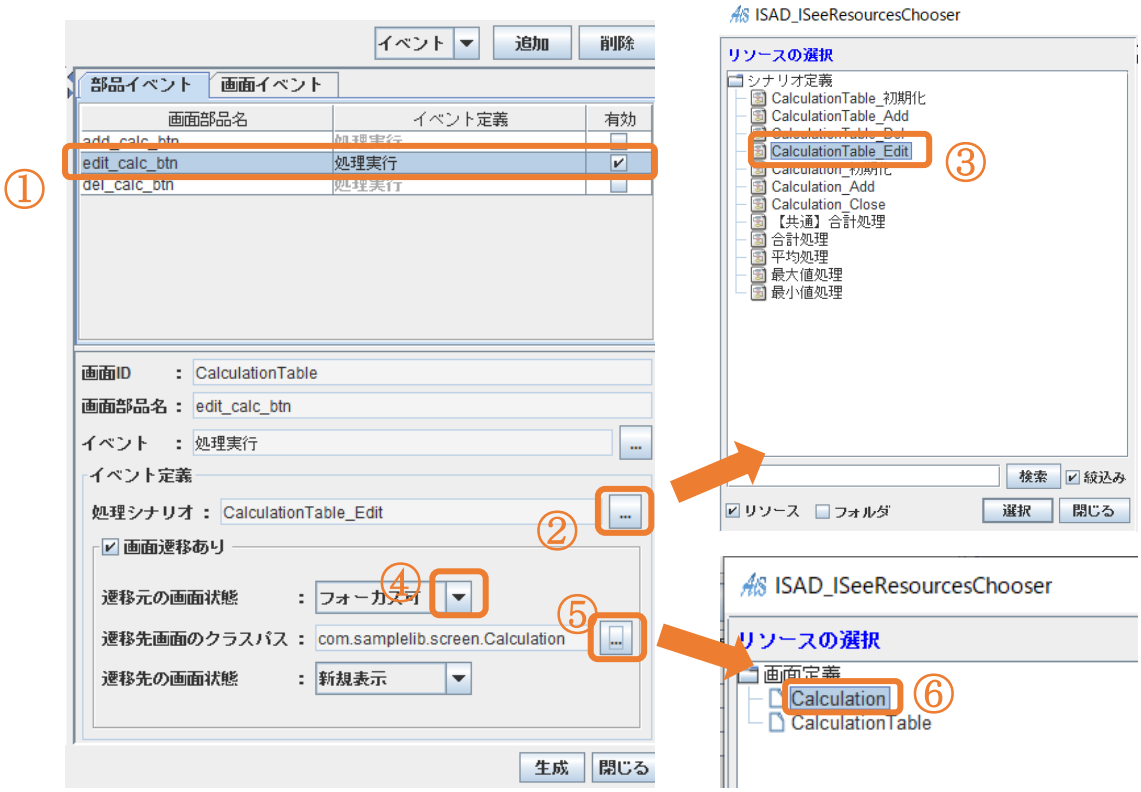
Calculation_初期化の章から来た場合はこちらから戻れます。

【図 12】



CalculationTable_Del の章から来た場合はこちらから戻れます。

【図 13】



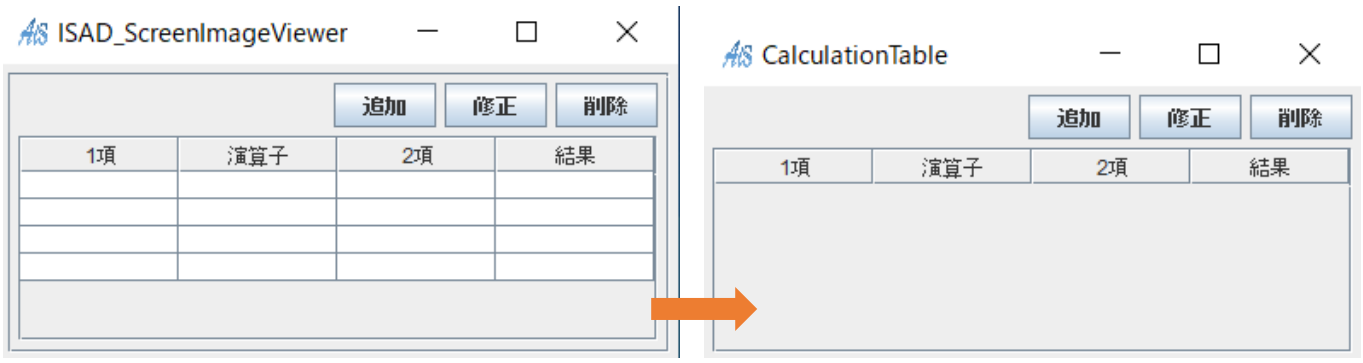
CalculationTable_Edit の章から来た場合はこちらから戻れます。

5.3 イベント毎のシナリオの内容サンプル

ここからは、１つ１つのシナリオの内部に、さらに具体的な処理を記述していきます。
シナリオの内部に、様々な種類の「リレーション」や「代入処理」などを作成しますが、
これらは上から下へ順番に処理されていくため、並び順にも注意しながら作成してください。
※並び順の変更：作成した処理を右クリックすると下や上へ移動できます。

5.3.1 CalculationTable_初期化 とテスト実行

「CalculationTable」のデフォルトで作成されているテーブルを削除する実装

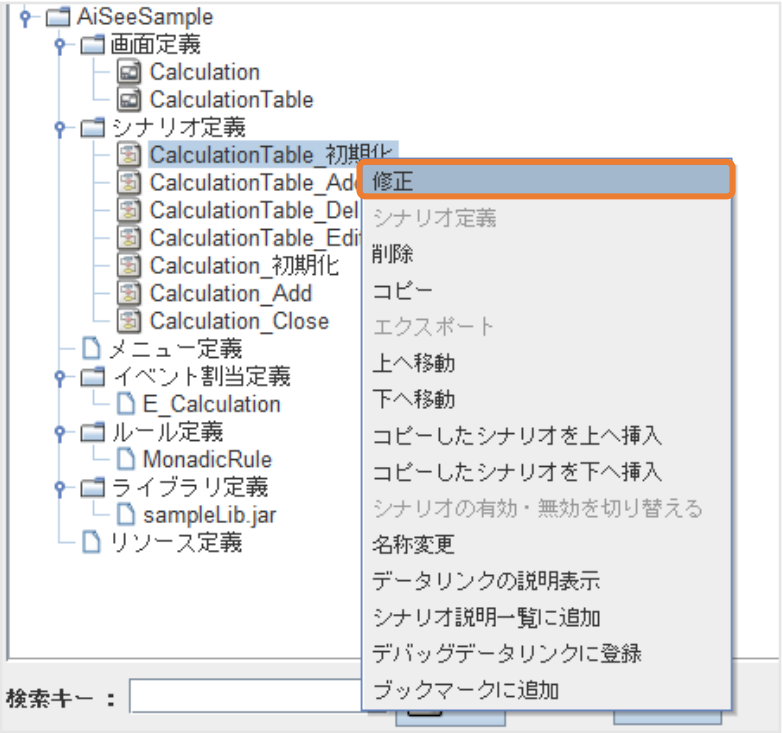


手順１：テーブルすべての行を削除する

手順２：テスト実行

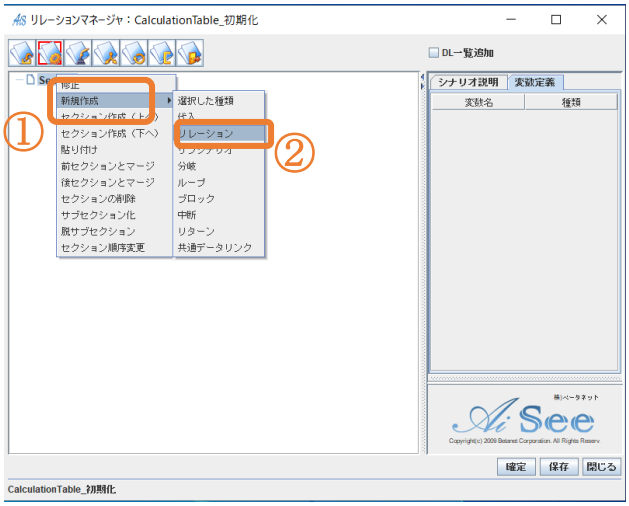
手順 1：テーブルすべての行を削除する 【図 14～図 17】

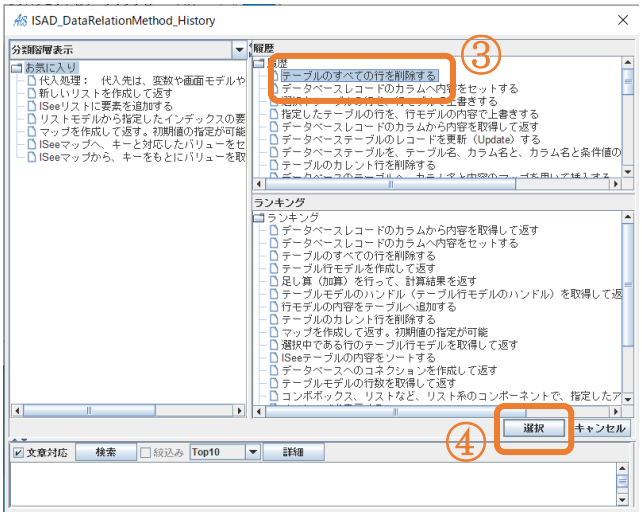
【図 14】



「CalculationTable_初期化」右クリックで「修正」選択

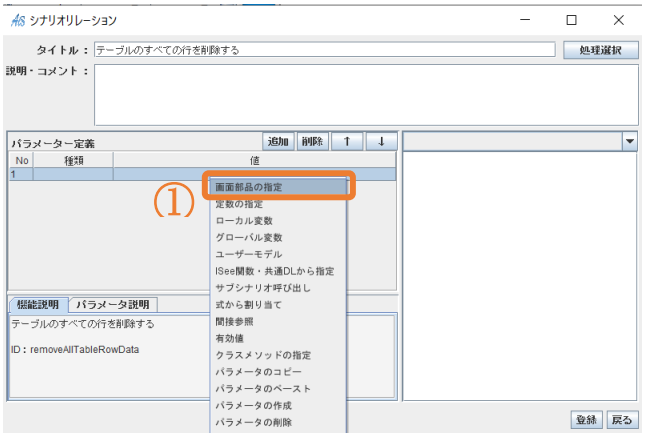
【図 15】

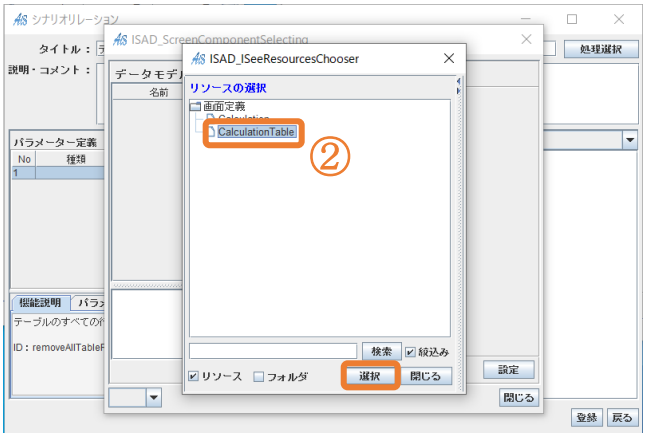


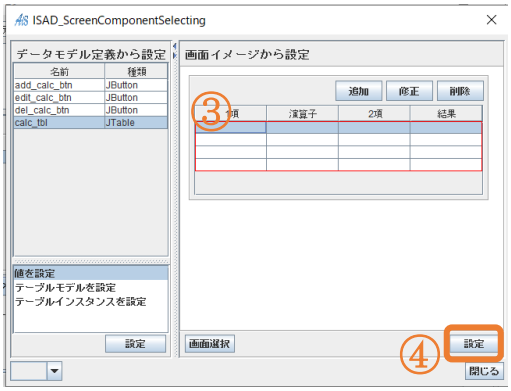


「テーブルのすべての行を削除する」を選択。
ない場合は左下の検索欄で「テーブルのすべての行を削除する」を入力して検索ボタン。

【図 16】



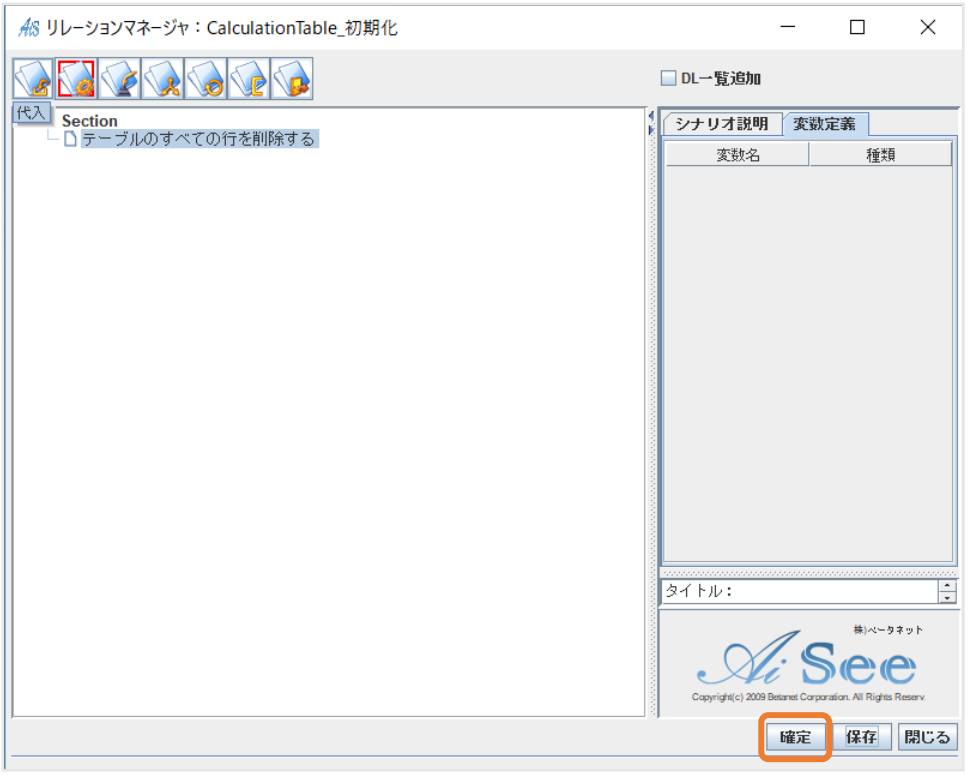




①パラメータ定義の「画面部品を指定」を右クリック

パラメータ定義で入れるものは「パラメータ説明」のタブをクリックすると確認できます。
リレーションによってパラメータ定義で選択する内容が変わるので都度確認をお願いいたします。

【図 17】


リレーションマネージャ：CalculationTable_初期化
DL一覧追加
代入
Section
テーブルのすべての行を削除する
シナリオ説明
変数定義
変数名
種類
タイトル：
確定
保存
閉じる

手順２：テスト実行

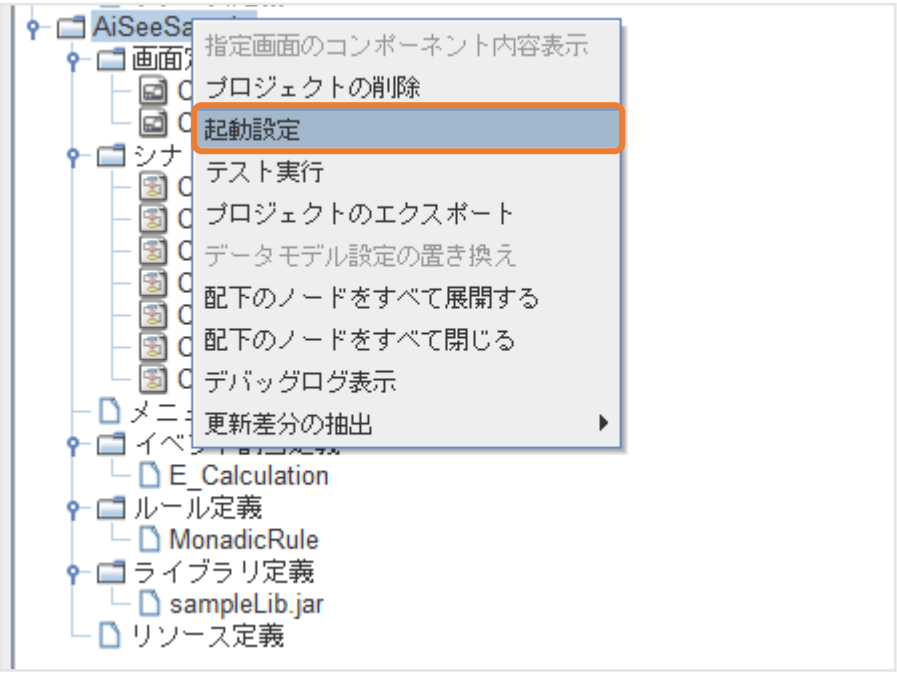
１で行った初期化が反映されているかの確認を行います。

テスト実行は適宜行いエラーや反映漏れが無いか確認をしましょう。

以下【図 17～図 18】の起動設定が終わったあと、**F5 キー**を押すとテスト実行されます。

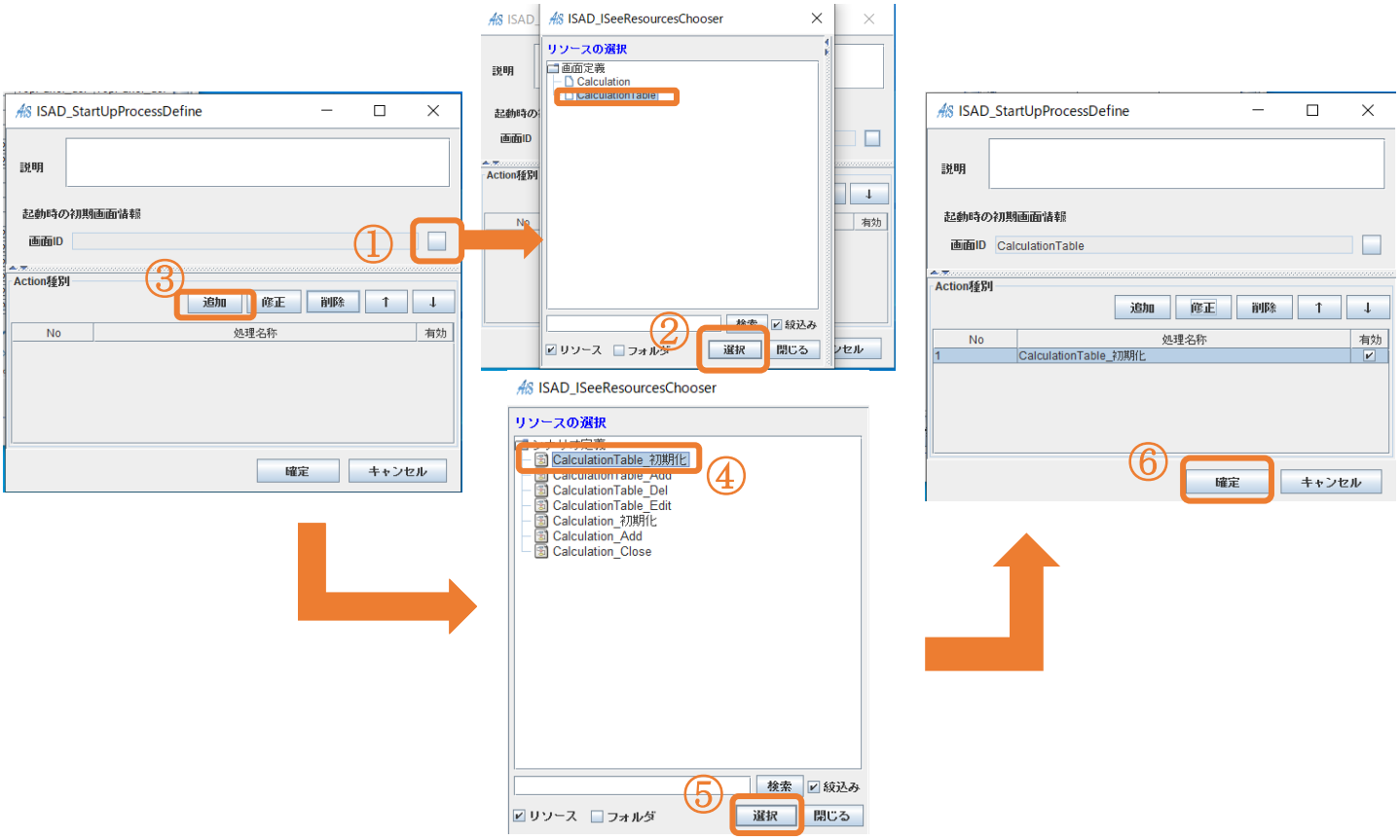
プロジェクトフォルダ「AiSeeSample」の直下フォルダ内のどこかを選択した状態で F5 キーを押して下さい。

【図 18】

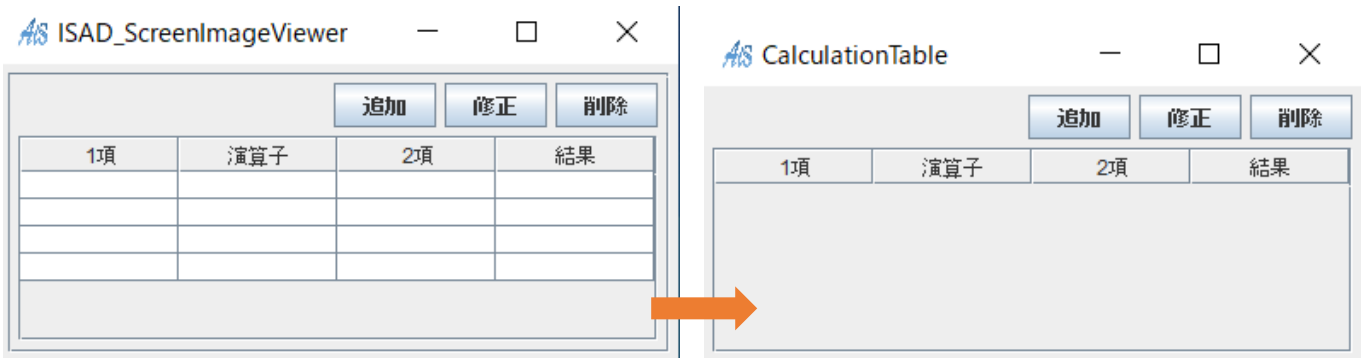


「AiSeeSample」を右クリックで「起動設定」を選択

【図 19】



【図 20】

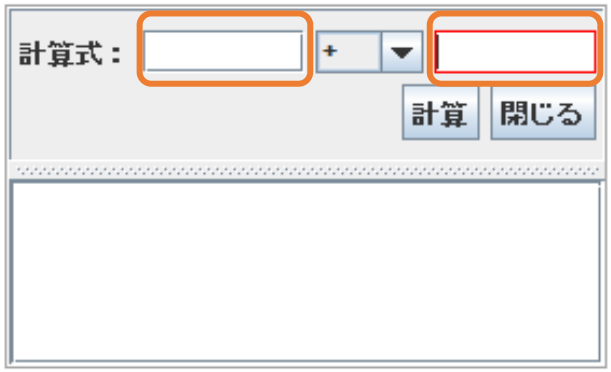


テーブルの表示が消えていれば OK です

5.3.2　Calculation_初期化

Calculation 画面の１項・２項の欄(オレンジ枠)を起動時に空の状態で起動したいので下記の手順で定義する。

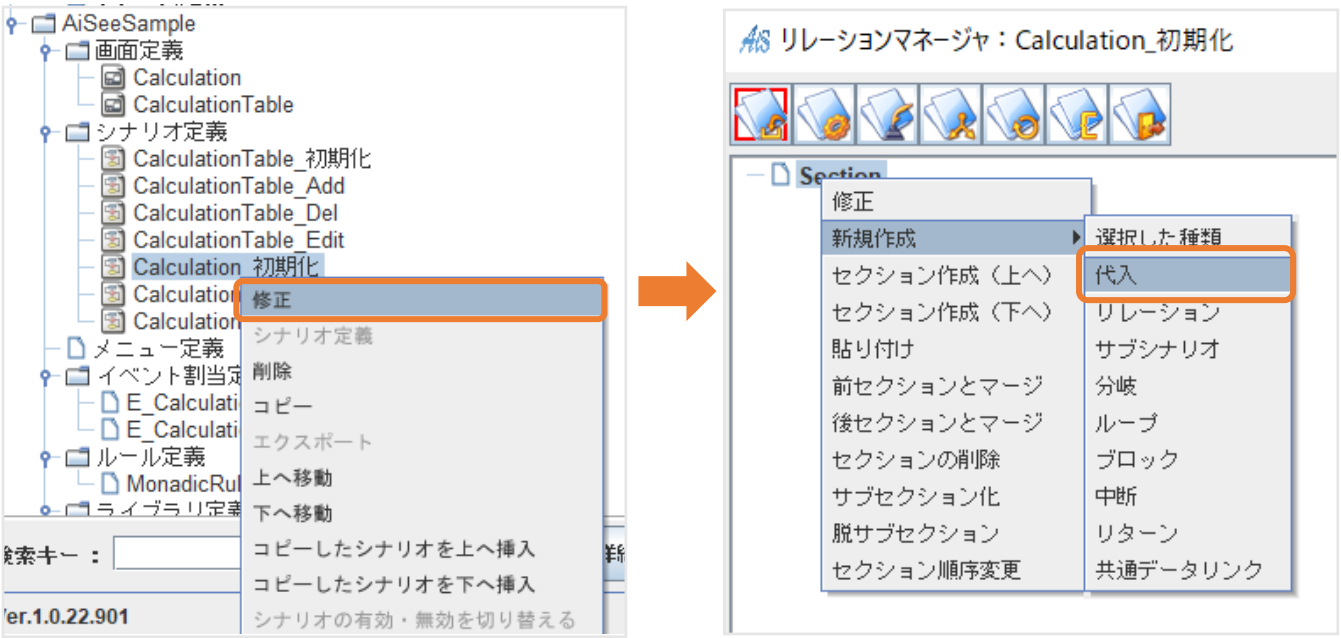
まずはイベント割当定義が出来ているか確認お願いします。



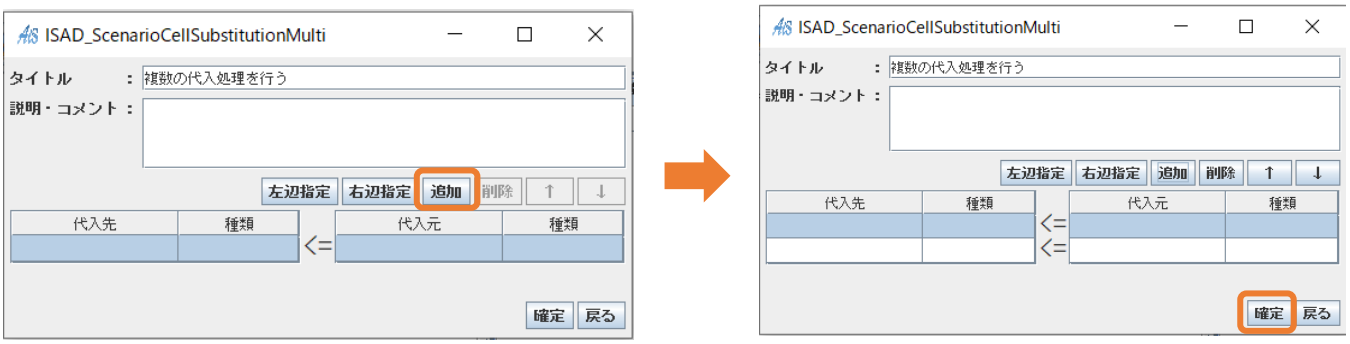
手順１：１項・２項の欄を起動時に空の状態にする

「Calculation_初期化」を右クリックで「修正」

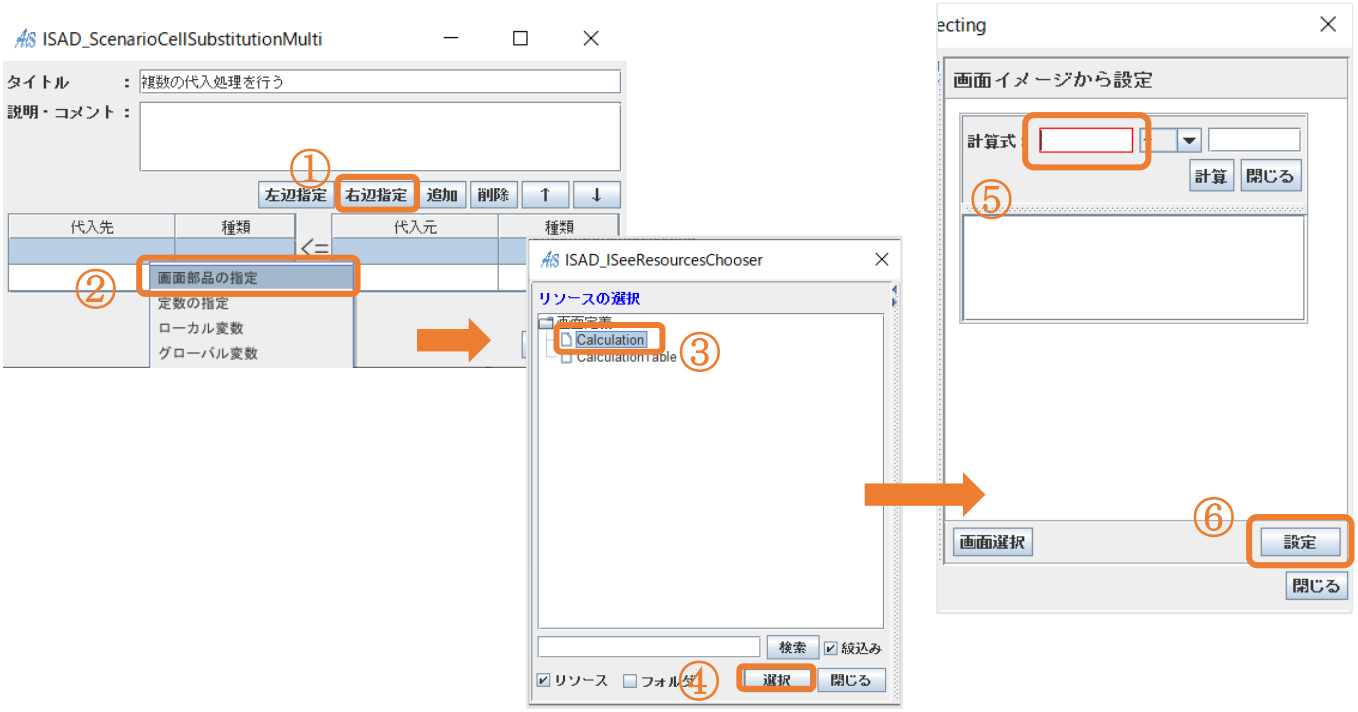
【図 21】



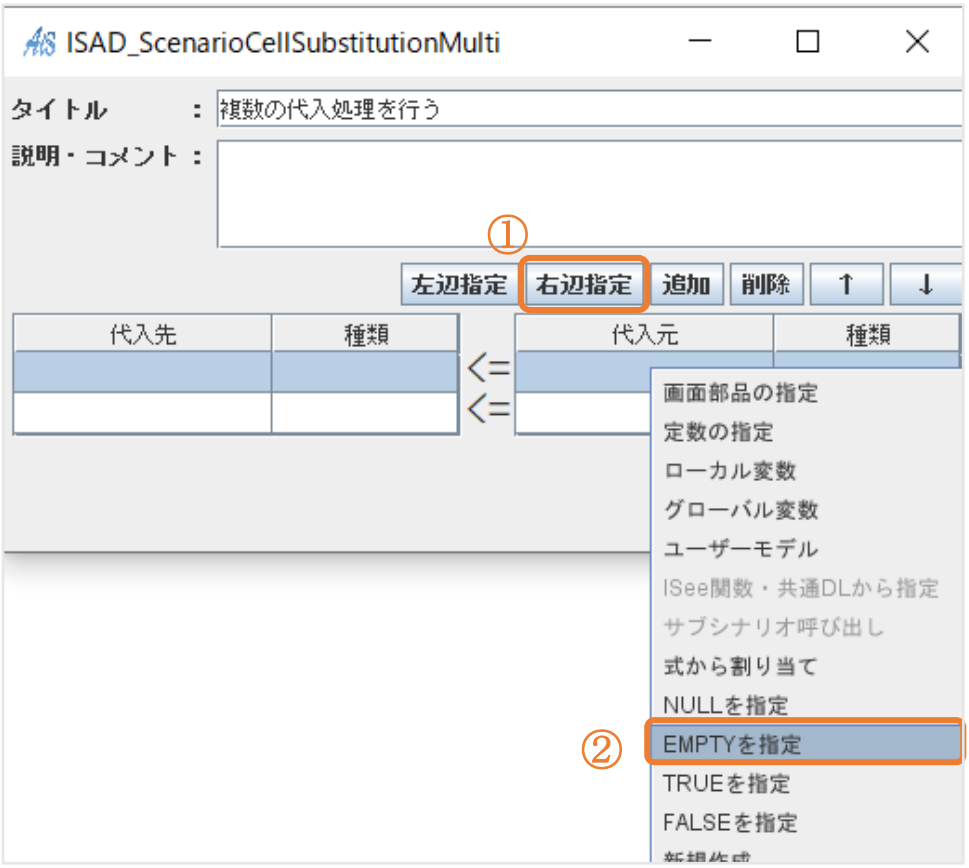
【図 22】


ISAD_ScenarioCellSubstitutionMulti
タイトル ： 複数の代入処理を行う
説明・コメント ：
左辺指定
右辺指定
追加
削除
代入先
種類
代入元
種類
確定
戻る

【図 23】

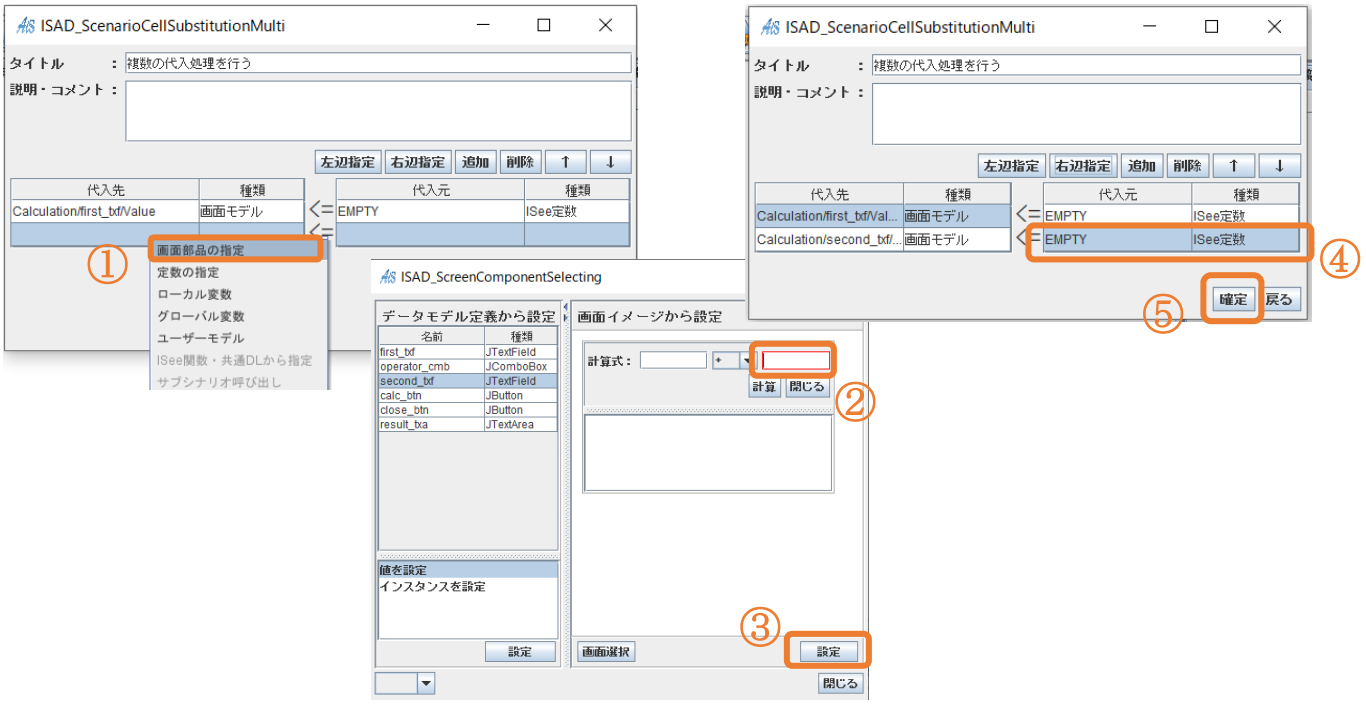

ISAD_ScenarioCellSubstitutionMulti
タイトル ： 複数の代入処理を行う
説明・コメント ：
①
左辺指定
右辺指定
追加
削除
代入先
種類
代入元
種類
②
画面部品の指定
定数の指定
ローカル変数
グローバル変数
ISAD_ISeeResourcesChooser
リソースの選択
Calculation
Calculation Table
③
検索
絞込み
リソース
フォルダ
④
選択
閉じる
ecting
画面イメージから設定
計算式
計算
閉じる
⑤
⑥
画面選択
設定
閉じる

【図 24】



EMPTY = 空　という意味です。

【図 25】



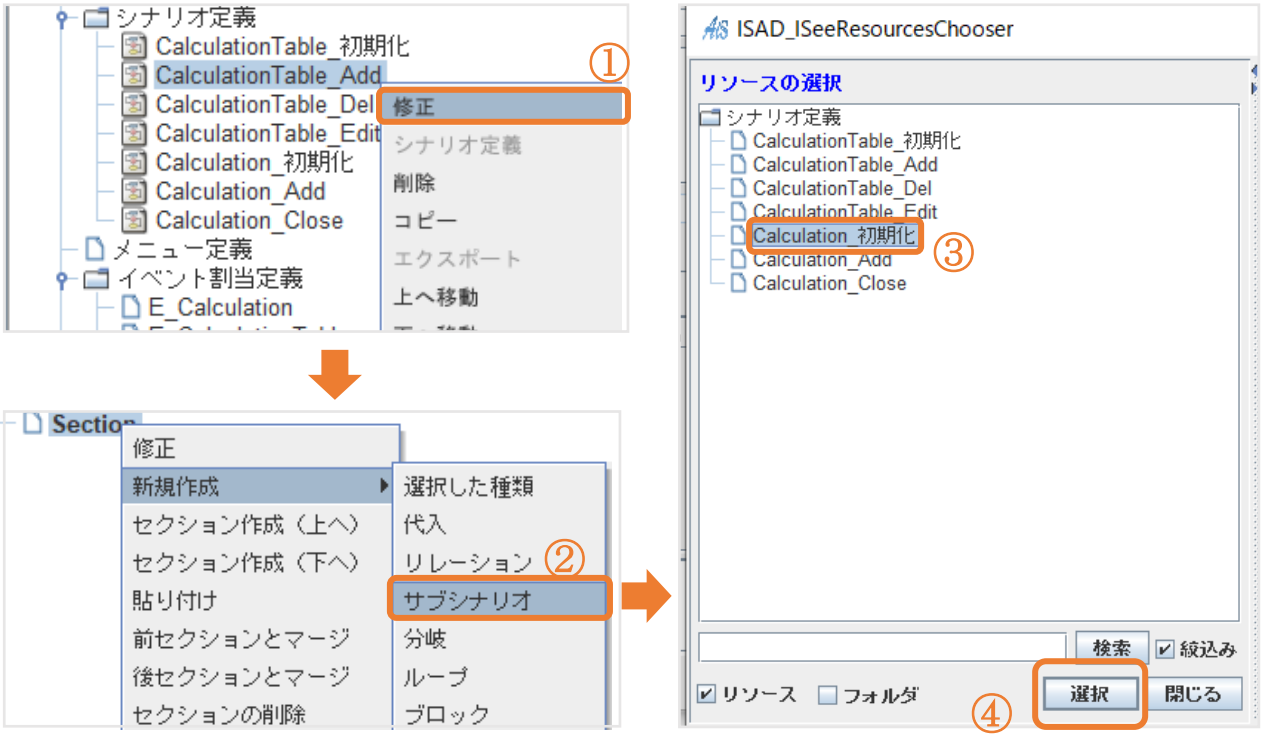
ここで画面イメージから設定で選択するコンポーネントで命名されている「oo_txf 」などは NetBeans で画面を作成時に作った名前が反映されています。

5.3.3 CalculationTable_Add

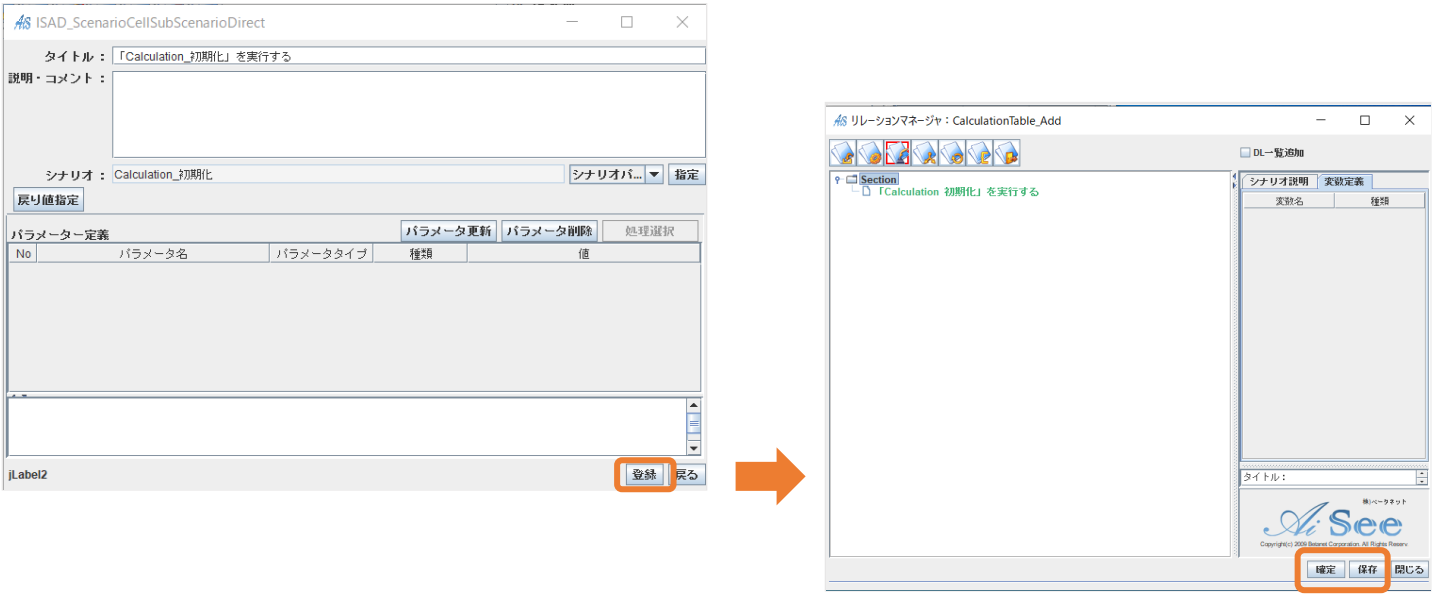
「サブシナリオ」機能を使ってリレーションを使いまわす方法。

手順１：サブシナリオで「Calculation_ 初期化」を実行するを設定

【図 26】



【図 27】



5.3.4 Calculation_Add

演算子の分岐設定の実装

まずはイベント割当定義が出来ているか確認お願いします。

手順１：分岐処理を行う

手順２：計算をする

手順３：計算結果を表示する

手順４：テスト実行

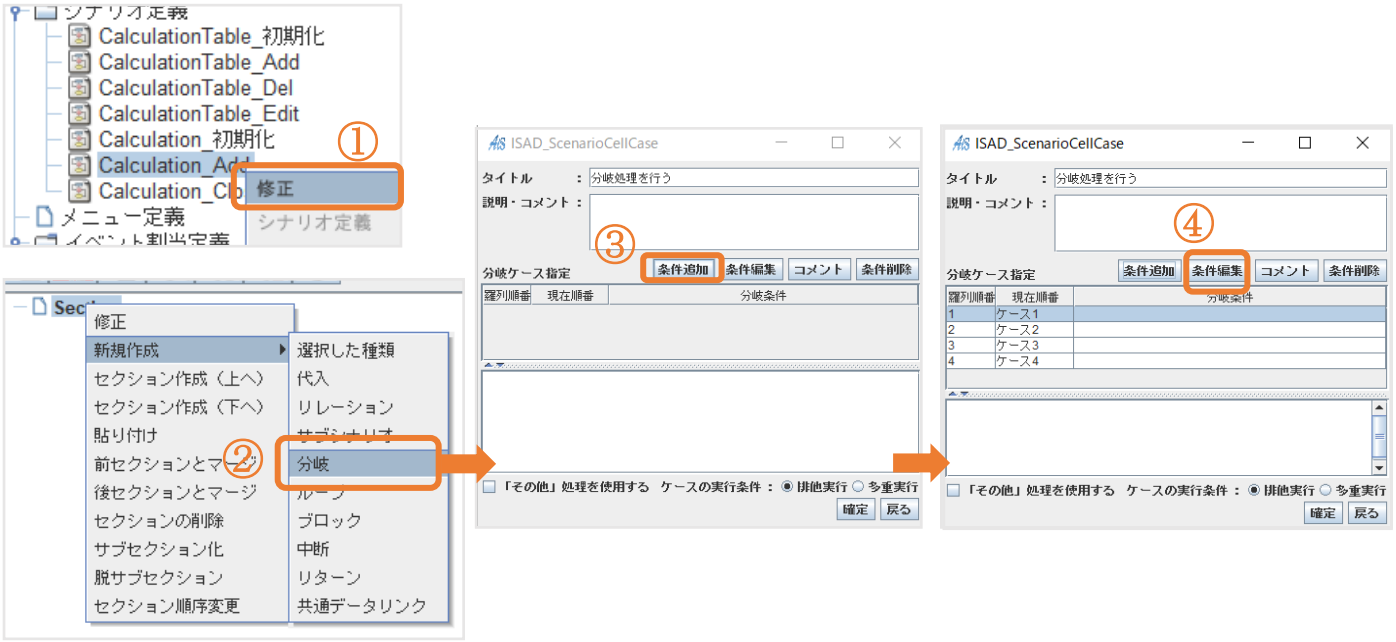
手順１：分岐処理を行う

計算の演算子の分岐点を作成します。

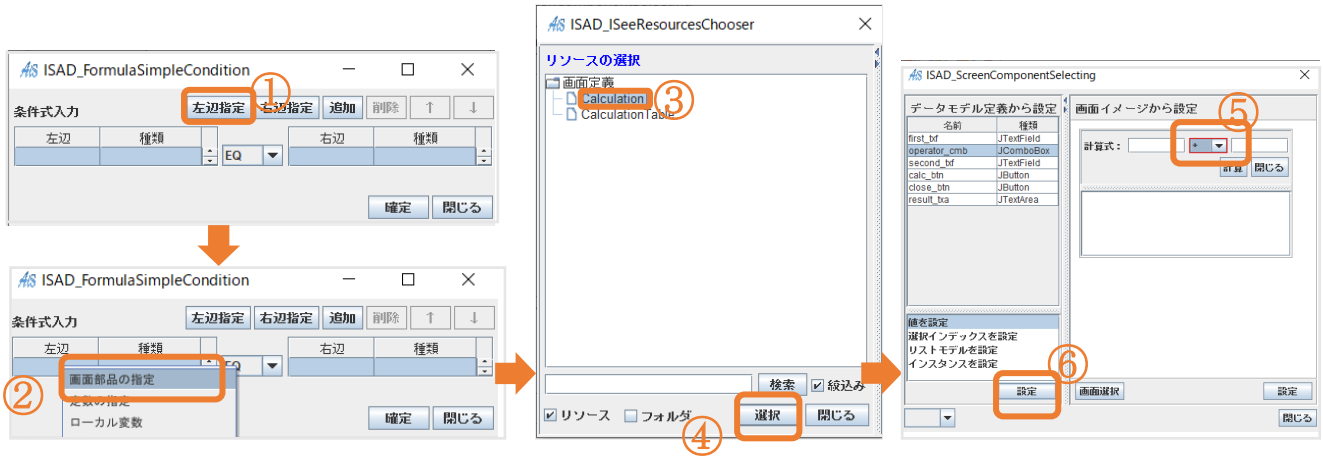
「+」「-」「*」「/」だったら～までを作成します。

③「条件追加」を４回押すとこの状態になります。

【図 28】

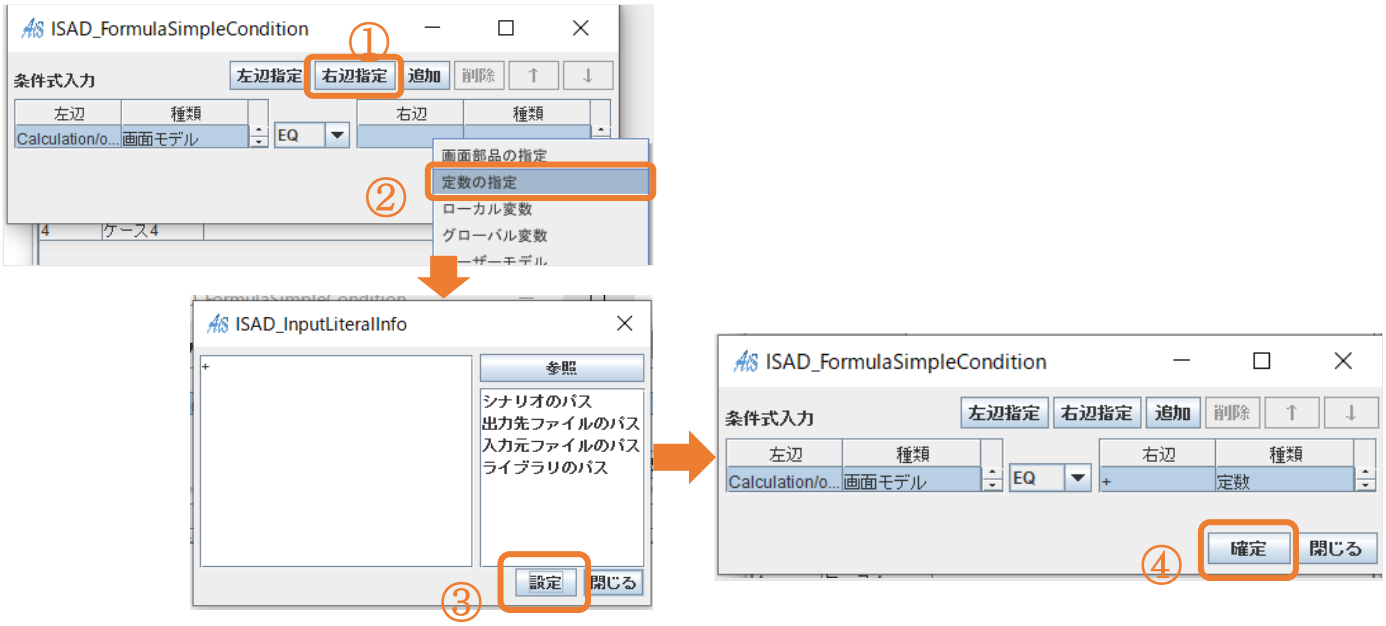


【図 29】



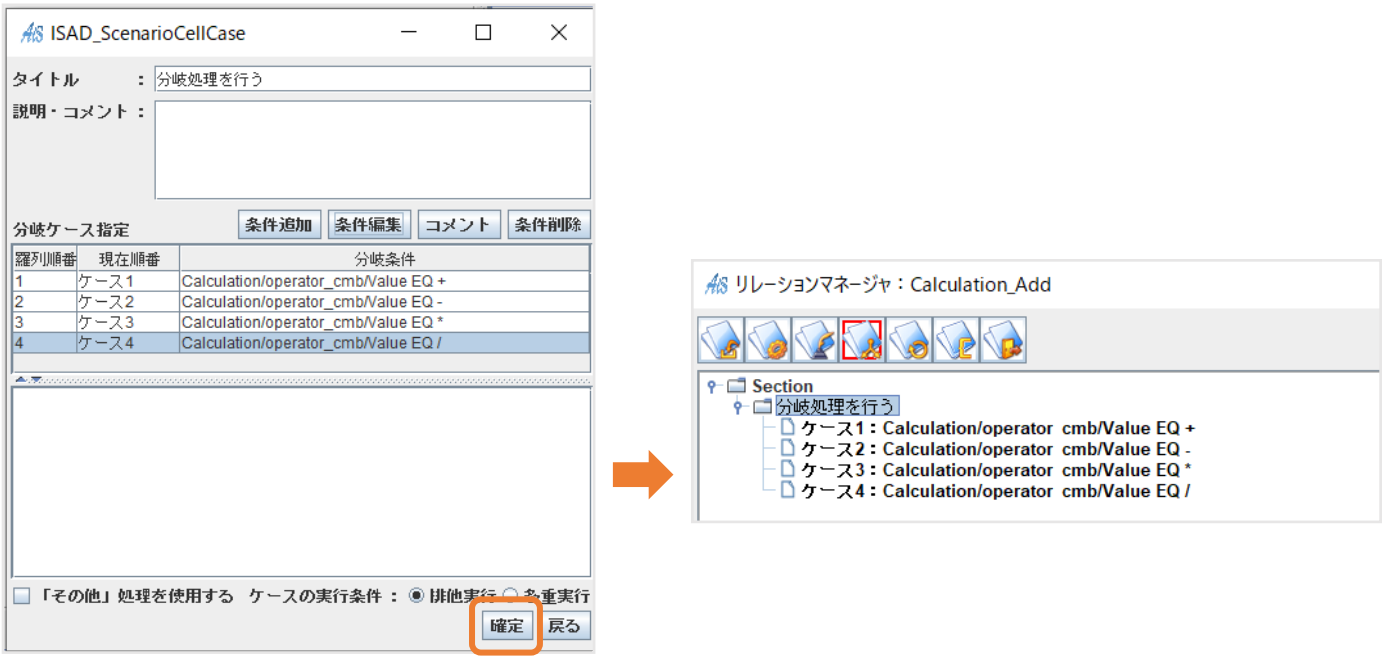
※この時「画面イメージから設定」直下の「設定」は押さないようにしてください

【図 30】



「-」「*」「/」も同様の操作で設定してください。

【図 31】


ISAD_ScenarioCellCase
タイトル ： 分岐処理を行う
説明・コメント ：
分岐ケース指定
条件追加
条件編集
コメント
条件削除
羅列順番 現在順番 分岐条件
1 ケース1 Calculation/operator_cmb/Value EQ +
2 ケース2 Calculation/operator_cmb/Value EQ -
3 ケース3 Calculation/operator_cmb/Value EQ *
4 ケース4 Calculation/operator_cmb/Value EQ /
「その他」処理を使用する ケースの実行条件 ： 排他実行 多重実行
確定
戻る
リレーションマネージャ：Calculation_Add
Section
分岐処理を行う
ケース1 : Calculation/operator cmb/Value EQ +
ケース2 : Calculation/operator cmb/Value EQ -
ケース3 : Calculation/operator cmb/Value EQ *
ケース4 : Calculation/operator cmb/Value EQ /

手順２：計算をする

画像内ではケース１の足し算の設定方法を行っています。

「+」「-」「*」「/」だったら～**どうする**を作成します。

ケース２～４も同様に操作を行います。出来上がると【図 32】のようになります。

選択するリレーションは下記です。

ケース１「足し算(加算)を行って、計算結果を返す」

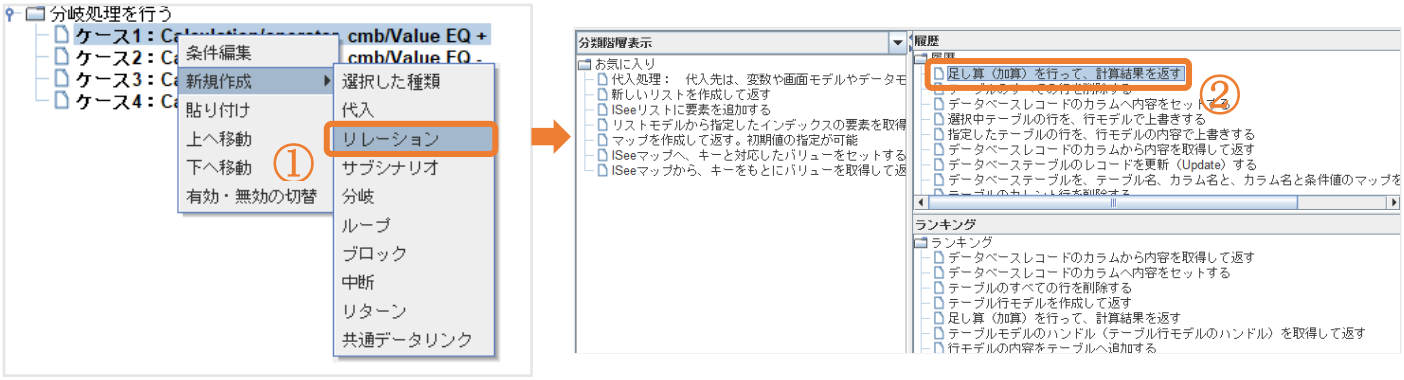
ケース２「引き算(減算)を行う関数」

ケース３「掛け算(乗算)を行い、結果を返す」

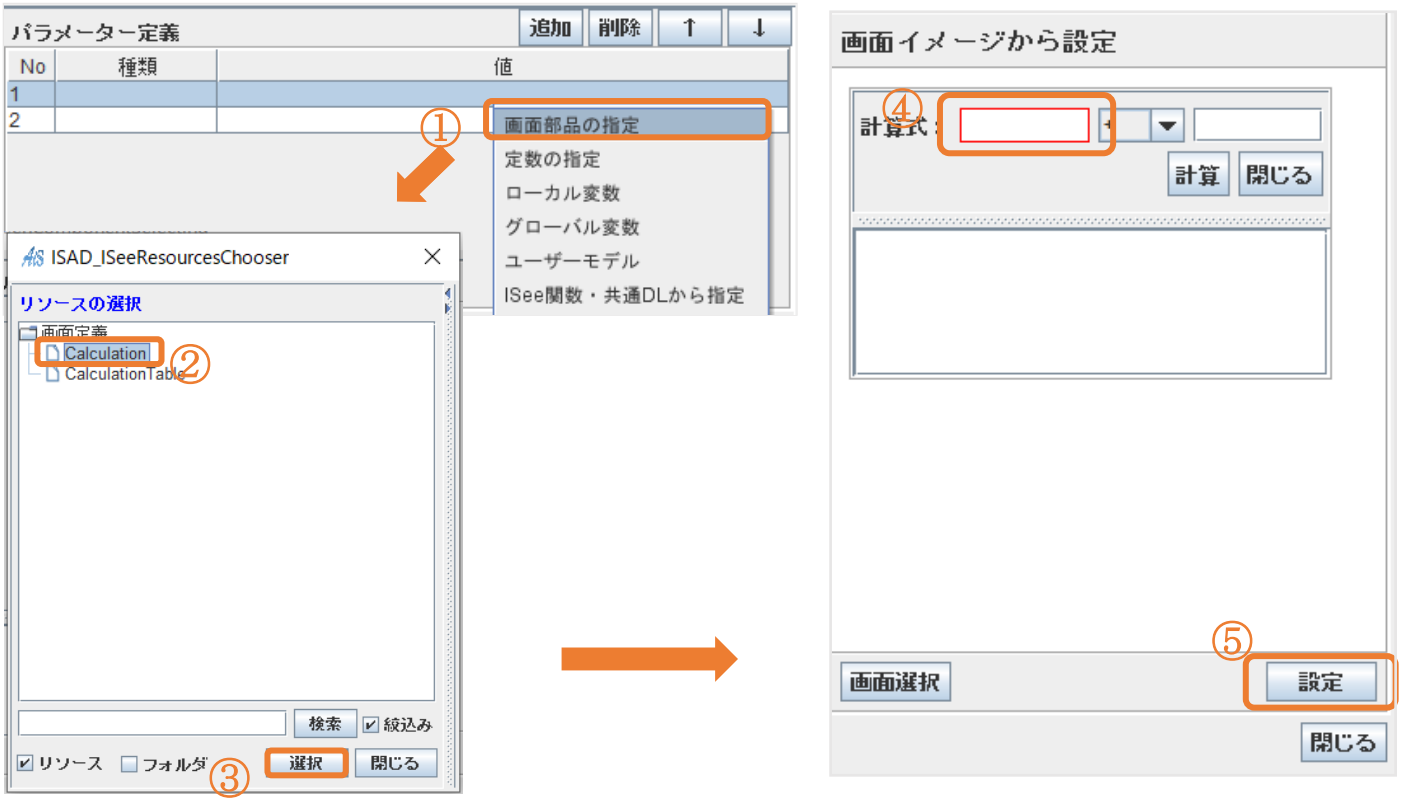
ケース４「割り算(除算)を行い、計算結果を返す」

ローカル変数「計算結果」は未作成なので作成してください。【図 35】

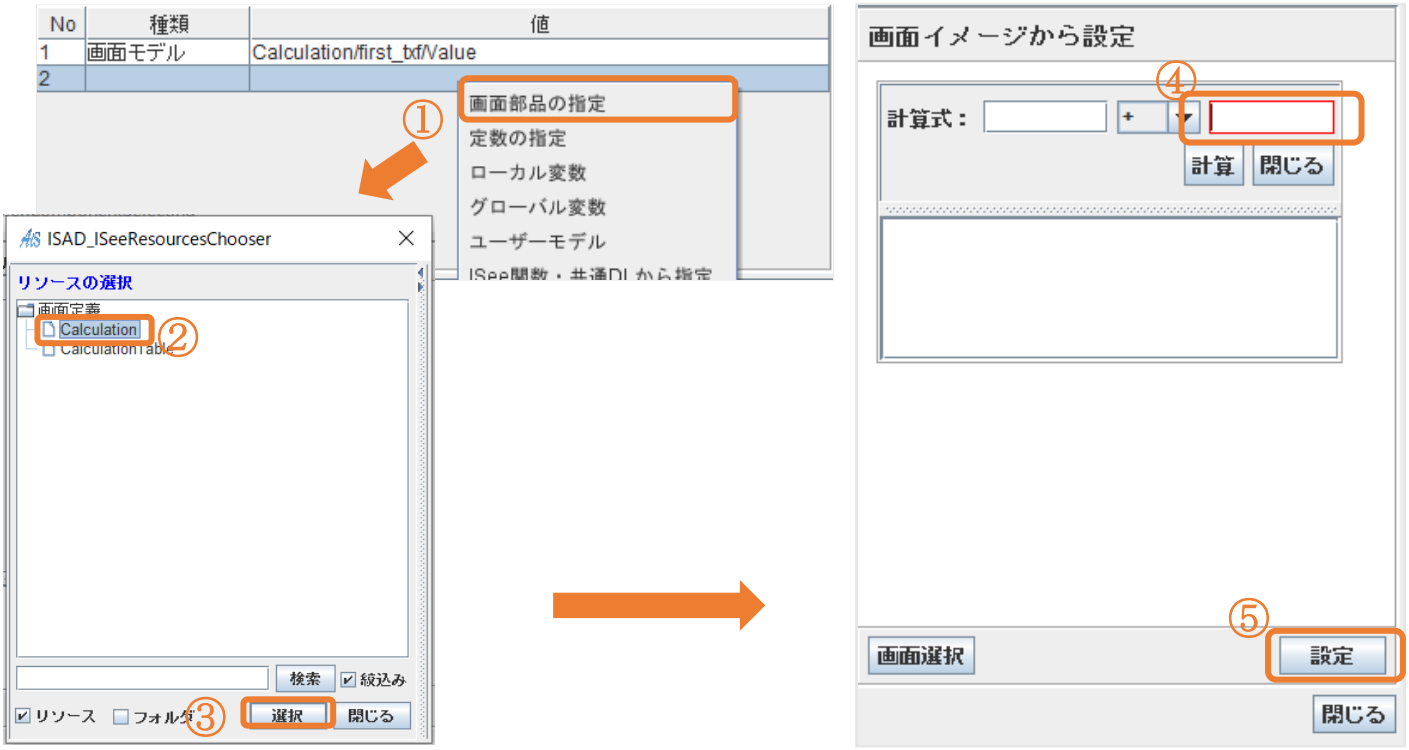
【図 32】


分岐処理を行う
ケース1：
ケース2：
ケース3：
ケース4：
cmb/Value EQ +
条件編集
新規作成
貼り付け
上へ移動
下へ移動
有効・無効の切替
①
選択した種類
代入
リレーション
サブシナリオ
分岐
ループ
ブロック
中断
リターン
共通データリンク
分類階層表示
お気に入り
代入処理：　代入先は、変数や画面モデルやデータモ
新しいリストを作成して返す
ISeeリストに要素を追加する
リストモデルから指定したインデックスの要素を取得
マップを作成して返す。初期値の指定が可能
ISeeマップへ、キーと対応したバリューをセットする
ISeeマップから、キーをもとにバリューを取得して返
履歴
足し算（加算）を行って、計算結果を返す
②
データベースレコードのカラムへ内容をセットする
選択中テーブルの行を、行モデルで上書きする
指定したテーブルの行を、行モデルの内容で上書きする
データベースレコードのカラムから内容を取得して返す
データベースレコードのレコードを更新（Update）する
データベーステーブルを、テーブル名、カラム名と、カラム名と条件値のマップを
ランキング
ランキング
データベースレコードのカラムから内容を取得して返す
データベースレコードのカラムへ内容をセットする
テーブルのすべての行を削除する
テーブル行モデルを作成して返す
足し算（加算）を行って、計算結果を返す
テーブルモデルのハンドル（テーブル行モデルのハンドル）を取得して返す
行モデルの内容をテーブルへ追加する

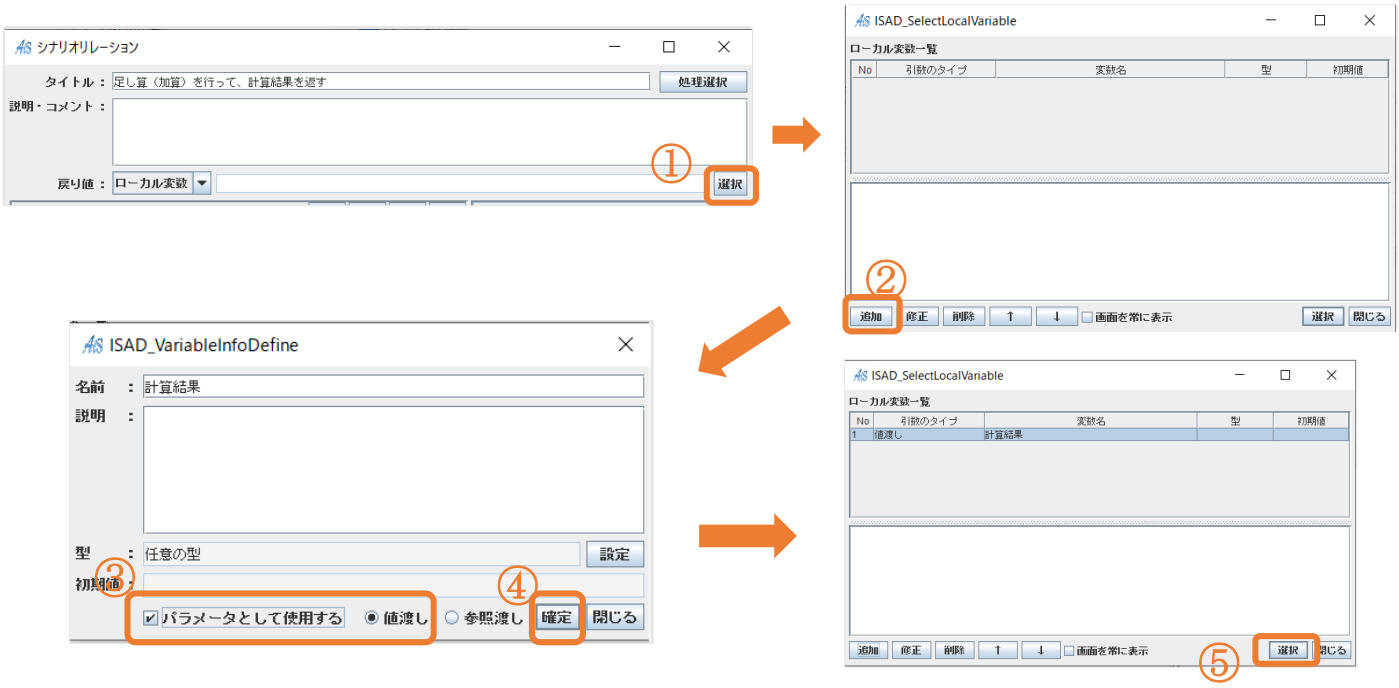
【図 33】


パラメーター定義
追加
削除
No
種類
値
1
2
①
画面部品の指定
定数の指定
ローカル変数
グローバル変数
ユーザーモデル
ISee関数・共通DLから指定
ISAD_ISeeResourcesChooser
リソースの選択
画面定義
Calculation
CalculationTable
②
検索
絞込み
リソース
フォルダ
③
選択
閉じる
画面イメージから設定
④
計算式：
計算
閉じる
⑤
画面選択
設定
閉じる

【図 34】

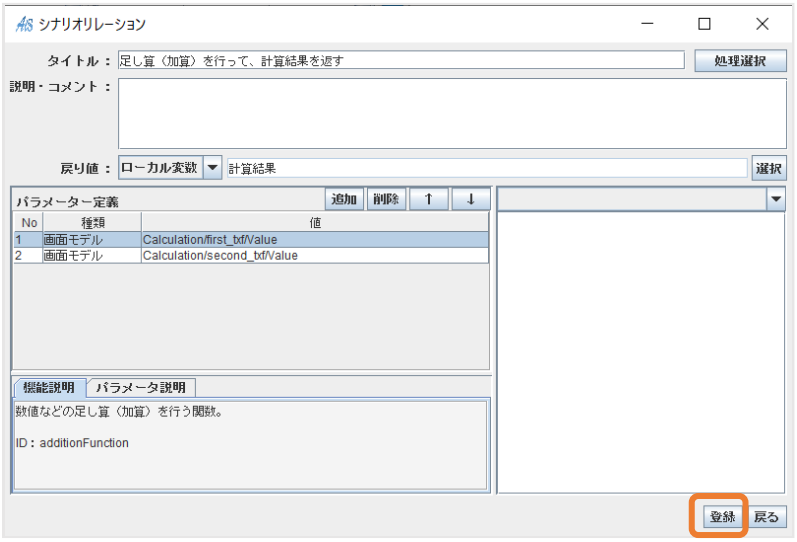


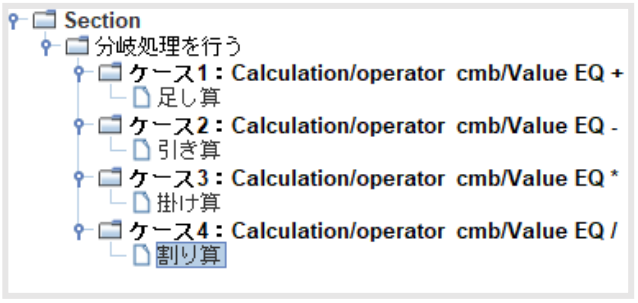
【図 35】



「パラメータとして使用する」にチェックを入れてください。

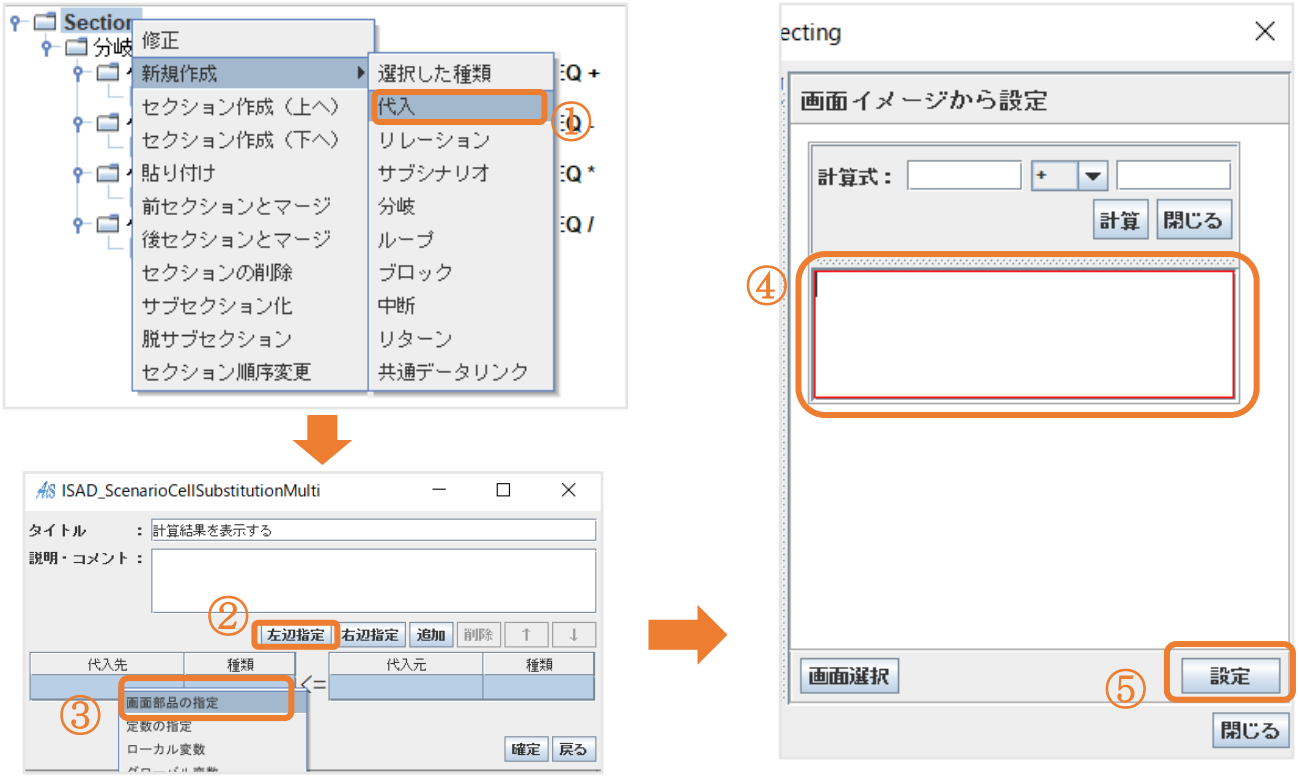
【図 36】


シナリオリレーション
タイトル：
足し算（加算）を行って、計算結果を返す
処理選択
説明・コメント：
戻り値：
ローカル変数
計算結果
選択
パラメーター定義
追加
削除
No
種類
値
1
画面モデル
Calculation/first_txf/Value
2
画面モデル
Calculation/second_txf/Value
機能説明
パラメータ説明
数値などの足し算（加算）を行う関数。
ID：additionFunction
登録
戻る

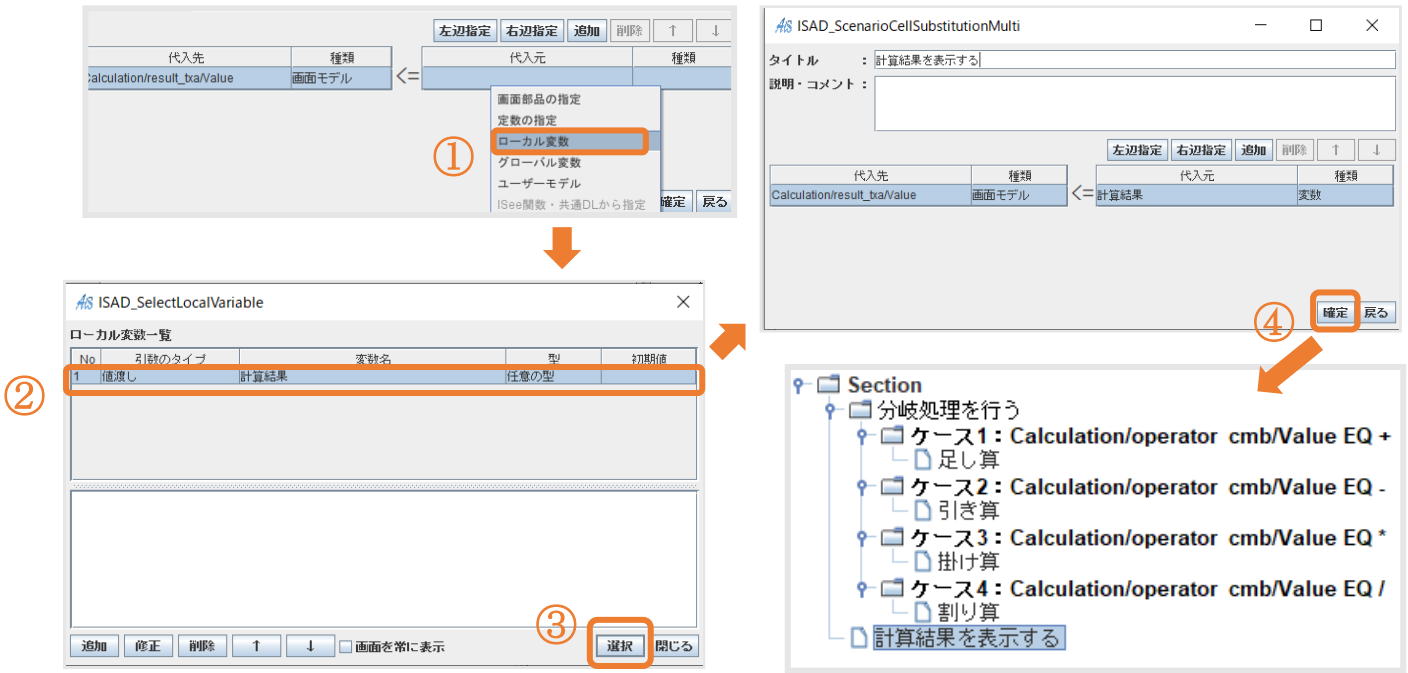

Section
分岐処理を行う
ケース1：Calculation/operator cmb/Value EQ +
足し算
ケース2：Calculation/operator cmb/Value EQ -
引き算
ケース3：Calculation/operator cmb/Value EQ *
掛け算
ケース4：Calculation/operator cmb/Value EQ /
割り算

手順３：計算結果を表示する

【図 37】



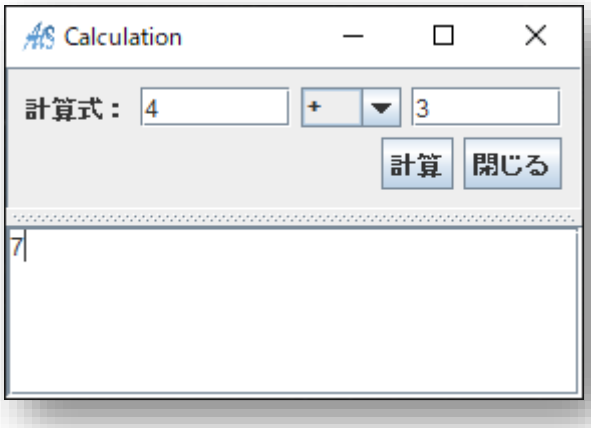
【図 38】



ケース 2 以降のローカル変数はケース 1 のときに作ったものをそのまま使用してください。

ここで F5 キーを押し、**テスト実行**を行いましょう。

問題なければ下図のように計算結果が表示されるようになります。



5.3.5 Calculation_Close

Calculation で計算した内容を CalculationTable に反映させる実装を行います。

まずはイベント割当定義が出来ているか確認お願いします。

手順１：行モデルを作成する

手順２：「１項」「演算子」「2 項」「結果」の内容をカラムへセットする

手順３：「行モデル」の内容をテーブルへ追加する

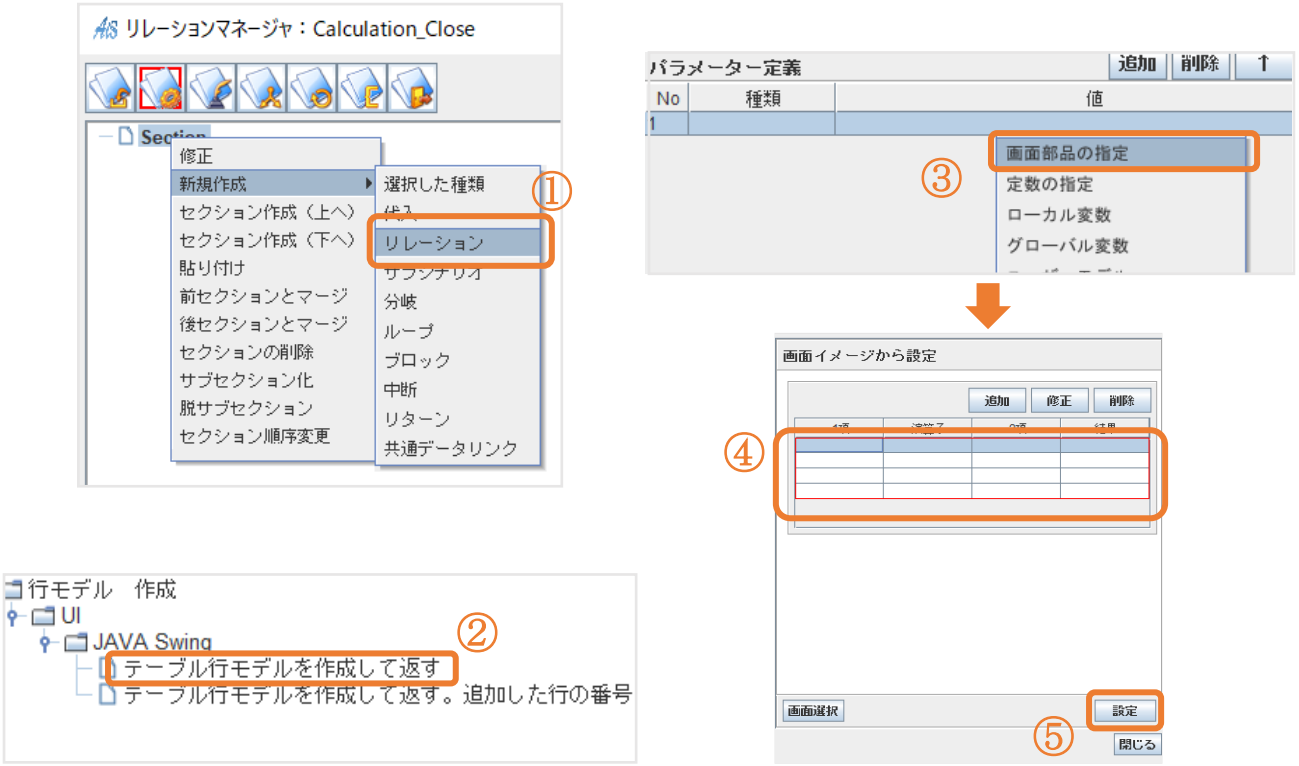
手順４：テスト実行

手順１：行モデルを作成する

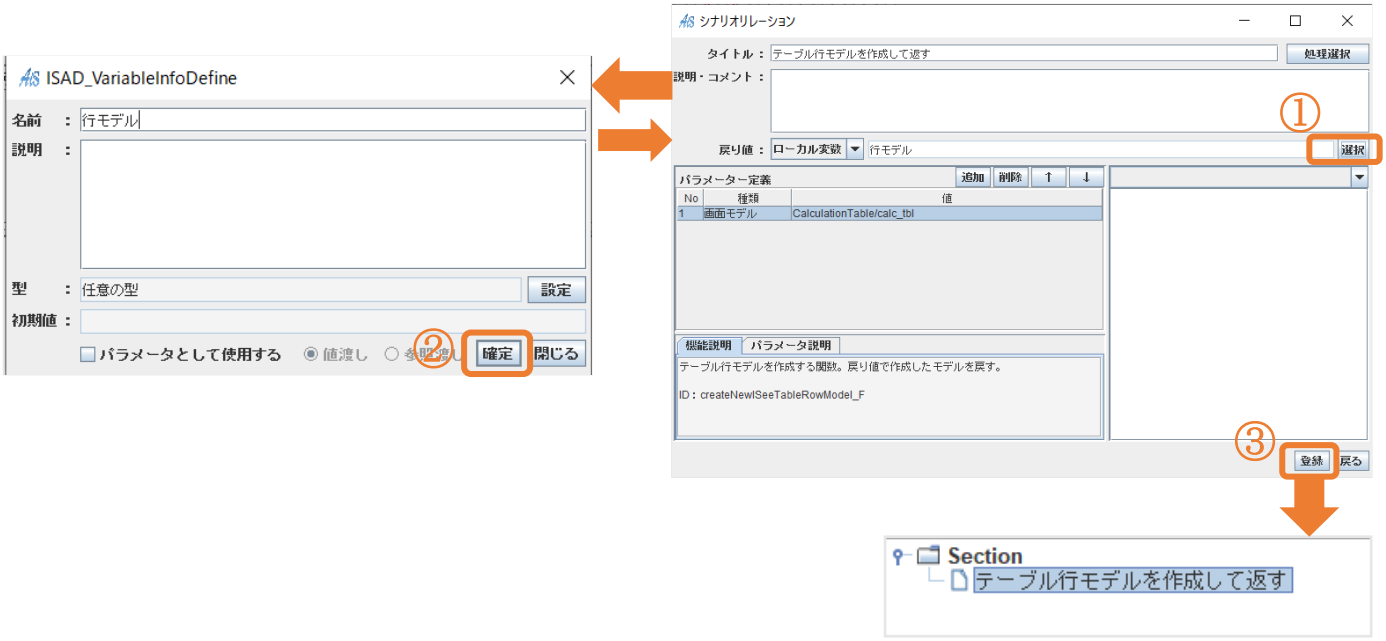
「戻り値」の設定が必要になります。

戻り値とは呼び出した結果を保管する箱のようなものです。

【図 39】



【図 40】



シナリオリレーション内①の「選択」をクリックして、「行モデル」を入力して「確定」をクリックしてください。

手順２：「１項」「演算子」「２項」「結果」の内容をカラムへセットする

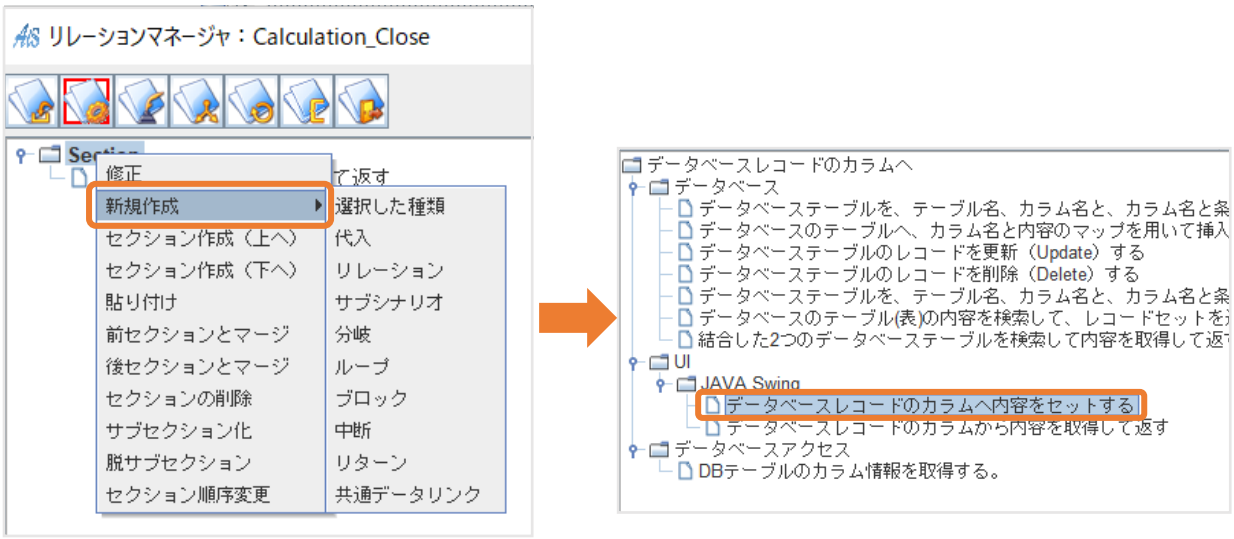
「演算子」「２項」「結果」は同じ手順で作成します。

下記は「１項」「演算子」の設定操作手順です。

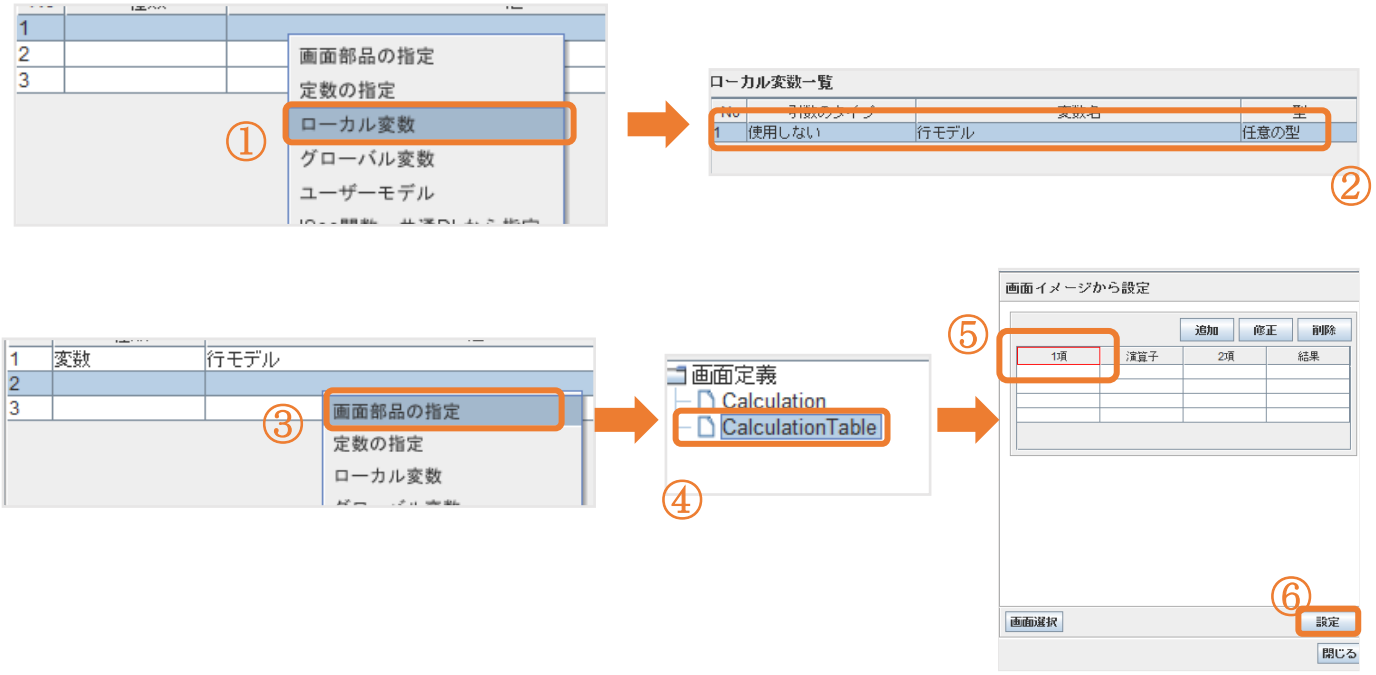
「２項」、「結果」に関しては【図 41】⑤の選択位置を「２項」と「結果」で選択位置を変更してください。

コピー機能を使って修正を行うと短時間での作業が出来て効率的です。【図 43～45】

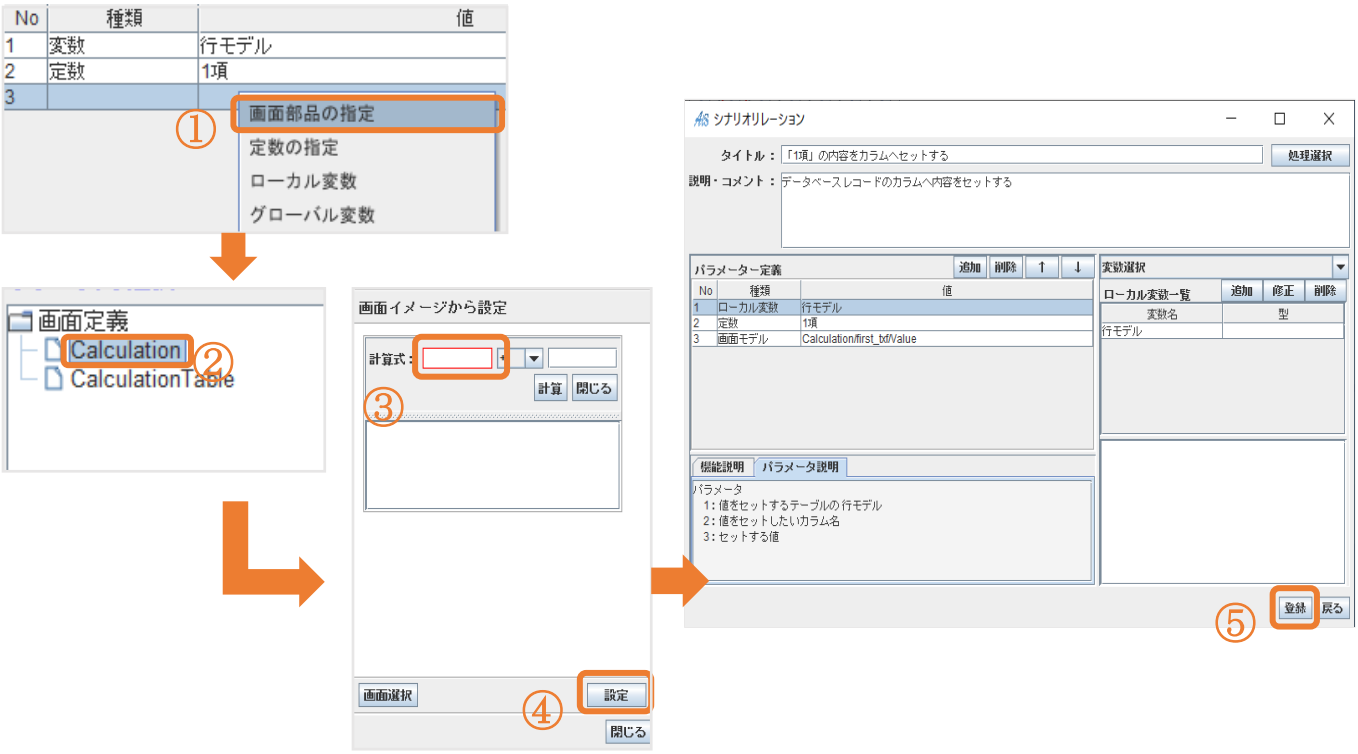
【図 41】



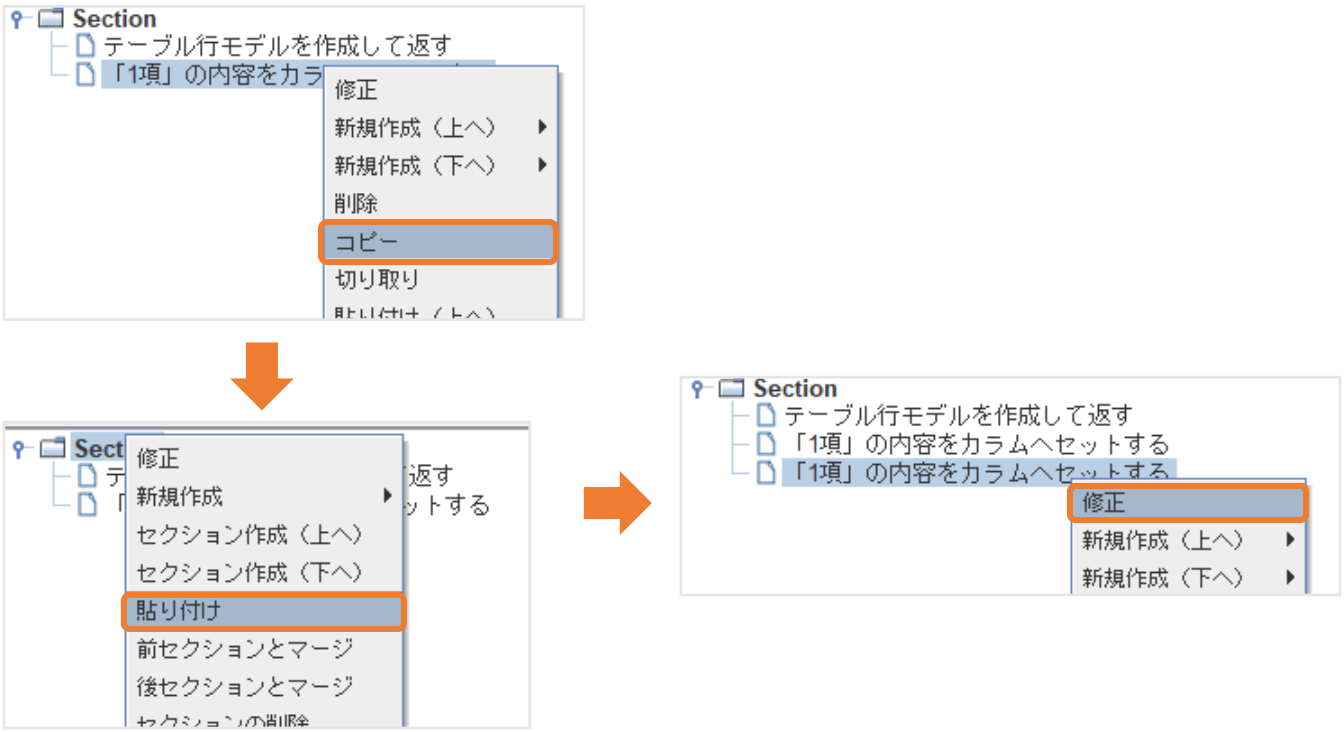
【図 42】


画面部品の指定
定数の指定
ローカル変数
グローバル変数
ユーザーモデル
①
ローカル変数一覧
使用しない
行モデル
任意の型
②
変数
行モデル
③
画面定義
Calculation
CalculationTable
④
画面イメージから設定
追加
修正
削除
1項
演算子
2項
結果
⑤
画面選択
設定
⑥
閉じる

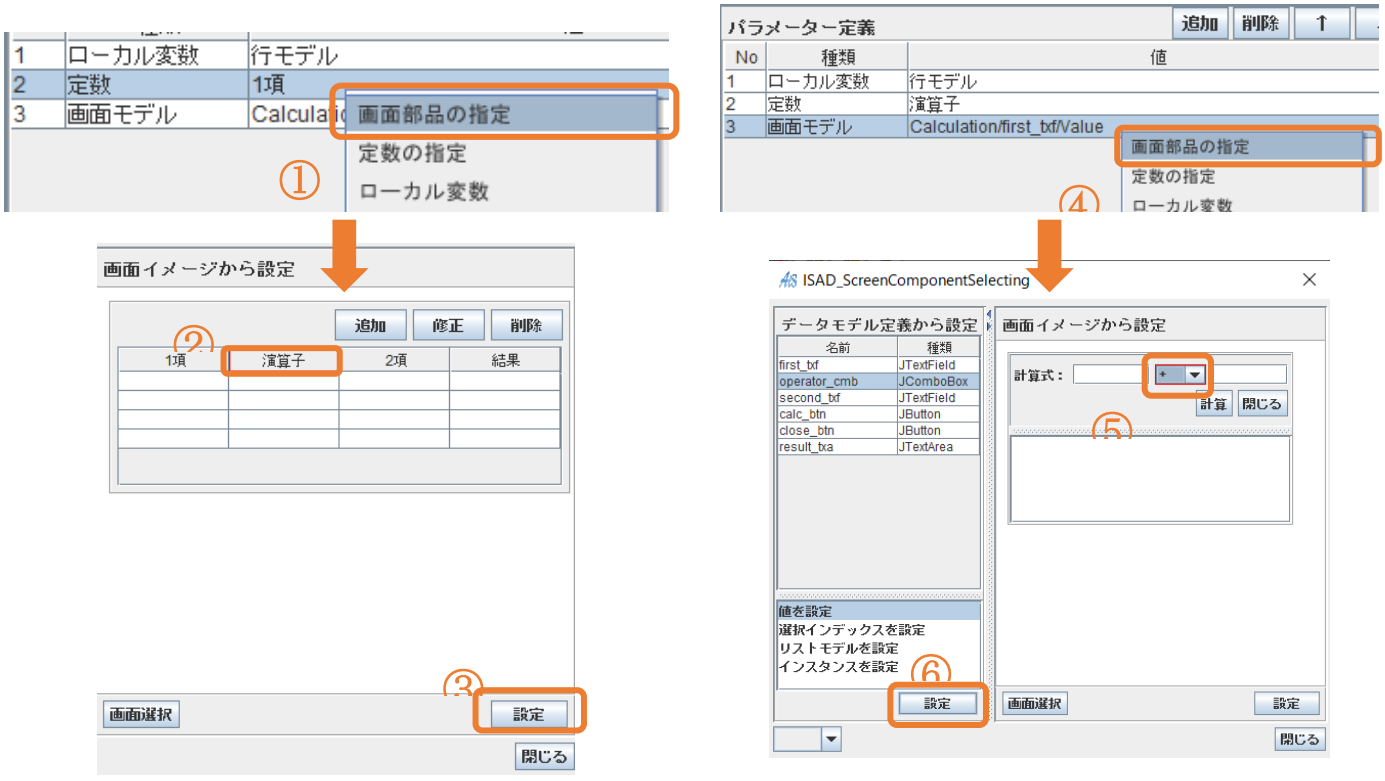
【図 43】


No
種類
値
変数
行モデル
定数
1項
画面部品の指定
定数の指定
ローカル変数
グローバル変数
①
画面定義
Calculation
CalculationTable
②
画面イメージから設定
計算式：
計算
閉じる
③
画面選択
設定
④
閉じる
シナリオリレーション
タイトル：
「1項」の内容をカラムへセットする
処理選択
説明・コメント：
データベースレコードのカラムへ内容をセットする
パラメーター定義
追加
削除
種類
値
ローカル変数
行モデル
定数
1項
画面モデル
Calculation/first_txt/Value
変数選択
ローカル変数一覧
変数名
型
機能説明
パラメータ説明
パラメータ
1：値をセットするテーブルの行モデル
2：値をセットしたいカラム名
3：セットする値
登録
戻る
⑤

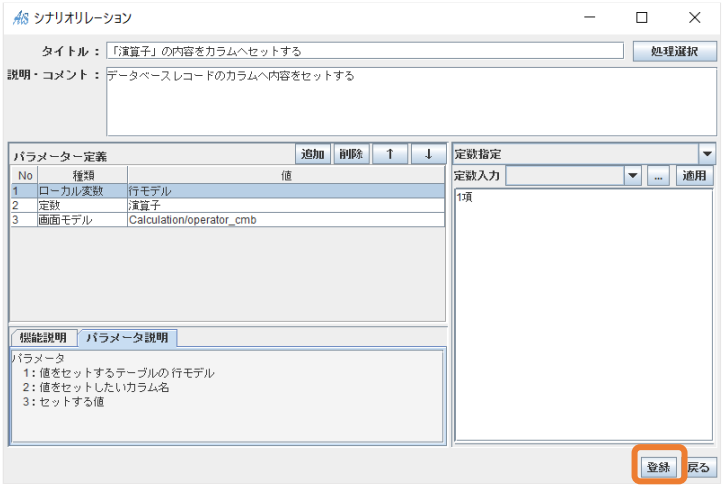
【図 44】

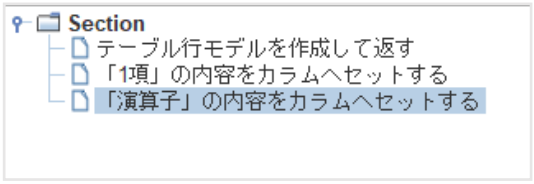

Section
テーブル行モデルを作成して返す
「1項」の内容をカラ
修正
新規作成（上へ）
新規作成（下へ）
削除
コピー
切り取り
修正
新規作成
セクション作成（上へ）
セクション作成（下へ）
貼り付け
前セクションとマージ
後セクションとマージ
返す
トする
Section
テーブル行モデルを作成して返す
「1項」の内容をカラムへセットする
「1項」の内容をカラムへセットする
修正
新規作成（上へ）
新規作成（下へ）

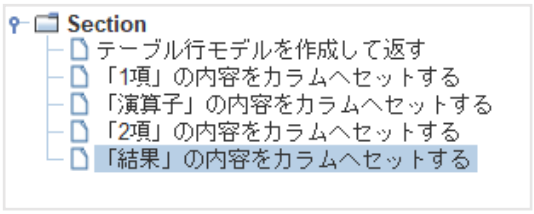
【図 45】


1 ローカル変数 行モデル
2 定数 1項
3 画面モデル Calculati
画面部品の指定
定数の指定
ローカル変数
①
パラメーター定義
追加
削除
No 種類 値
1 ローカル変数 行モデル
2 定数 演算子
3 画面モデル Calculation/first_txf/Value
画面部品の指定
定数の指定
ローカル変数
④
画面イメージから設定
追加
修正
削除
②
1項
演算子
2項
結果
画面選択
③
設定
閉じる
ISAD_ScreenComponentSelecting
データモデル定義から設定
名前 種類
first_txf JTextField
operator_cmb JComboBox
second_txf JTextField
calc_btn JButton
close_btn JButton
result_txa JTextArea
値を設定
選択インデックスを設定
リストモデルを設定
インスタンスを設定
⑥
設定
画面イメージから設定
計算式：
計算
閉じる
⑤
画面選択
設定
閉じる

【図 46】

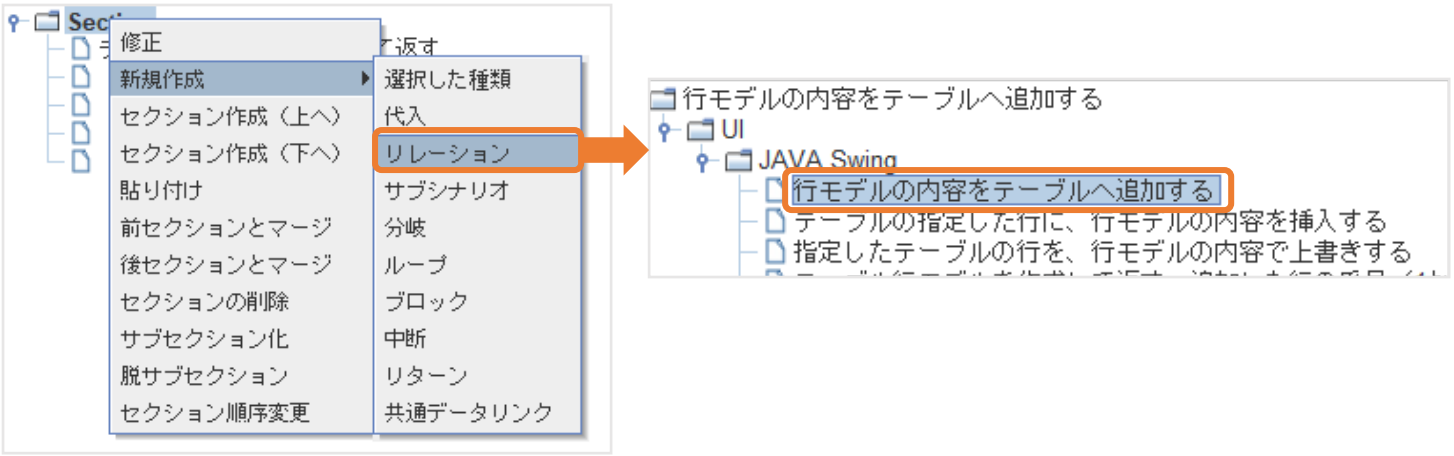

シナリオリレーション
タイトル：
「演算子」の内容をカラムへセットする
処理選択
説明・コメント：
データベースレコードのカラムへ内容をセットする
パラメーター定義
追加
削除
No
種類
値
1
ローカル変数
行モデル
2
定数
演算子
3
画面モデル
Calculation/operator_cmb
定数指定
定数入力
適用
1項
機能説明
パラメータ説明
パラメータ
1：値をセットするテーブルの 行モデル
2：値をセットしたいカラム名
3：セットする値
登録
戻る


Section
テーブル行モデルを作成して返す
「1項」の内容をカラムへセットする
「演算子」の内容をカラムへセットする

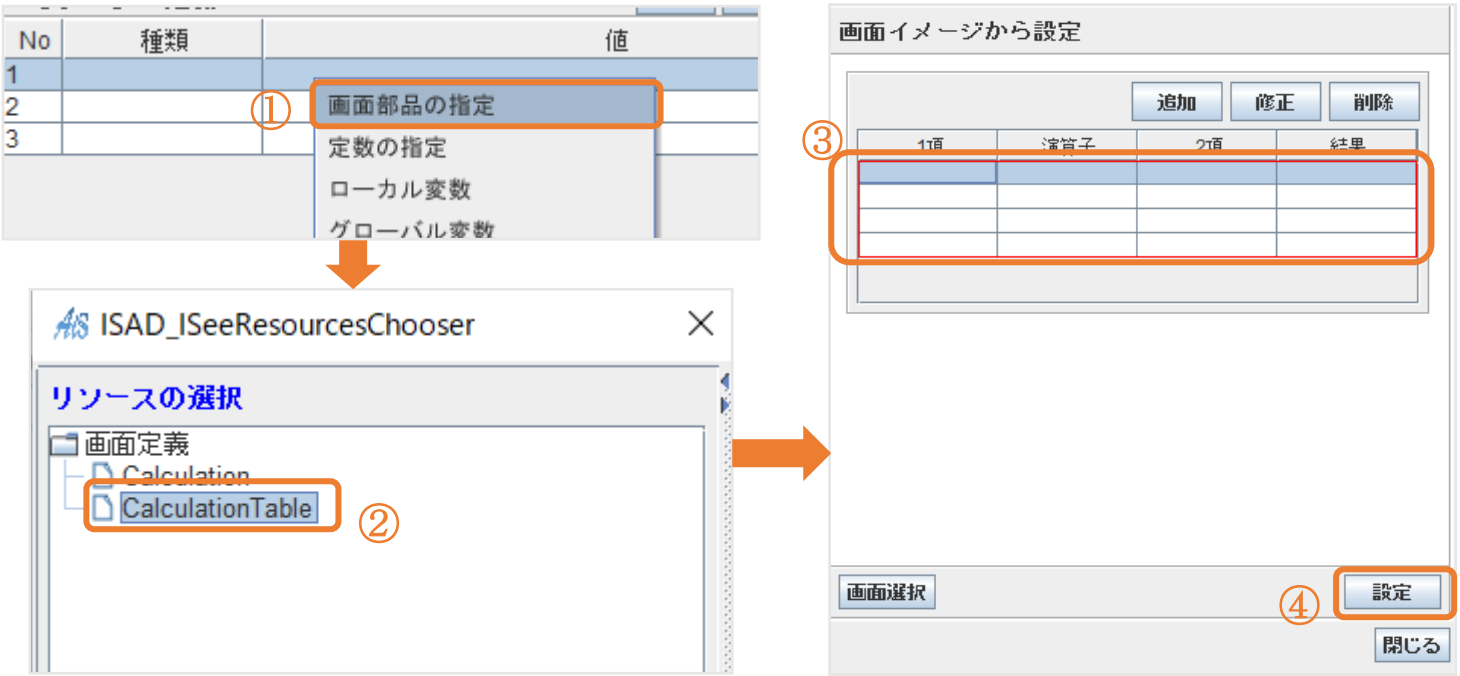

Section
テーブル行モデルを作成して返す
「1項」の内容をカラムへセットする
「演算子」の内容をカラムへセットする
「2項」の内容をカラムへセットする
「結果」の内容をカラムへセットする

手順３：「行モデル」の内容をテーブルへ追加する

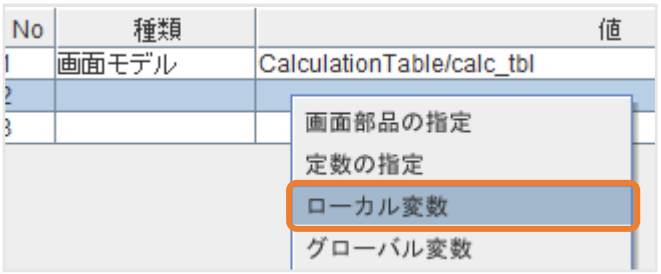
【図 47】

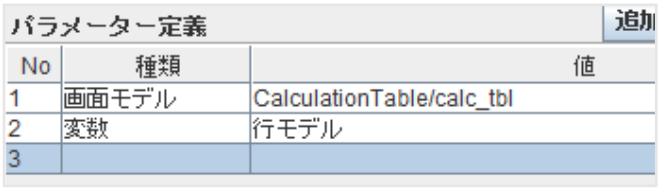


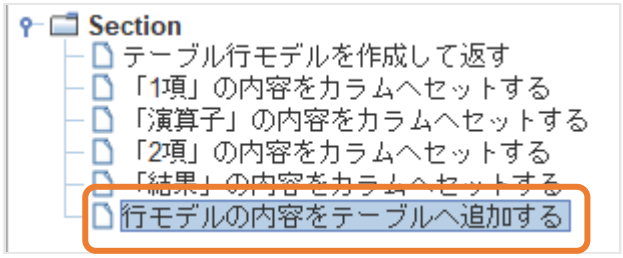
【図 48】



【図 49】







パラメータ定義 3 行目は任意項目の為、今回は未定義で問題ありません。

※図 49 に「ローカル変数」と「変数」がありますが、同じものです。

ここで F5 キーを押し、**テスト実行**を行いましょう。

問題がなければ「Calculation」で計算、閉じるボタンを押すと「CalculationTable」に反映されていると思います。

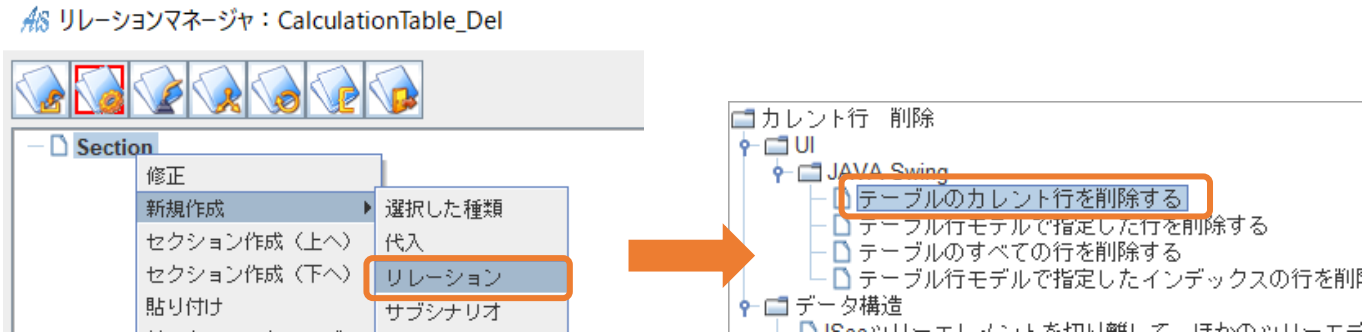
5.3.6 CalculationTable_Del

「CalculationTable」で選択した行の削除を行う実装

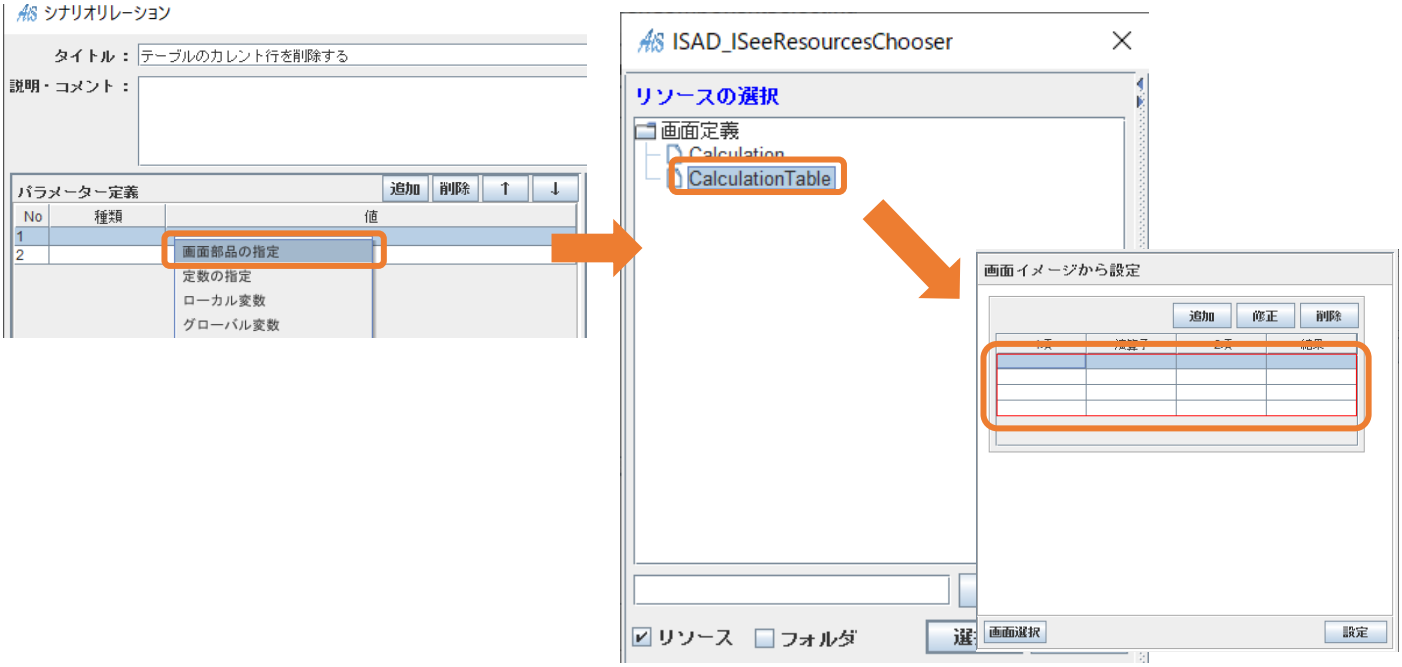
まずは[イベント割当定義]が出来ているか確認お願いします。

手順１：カレント行を削除する

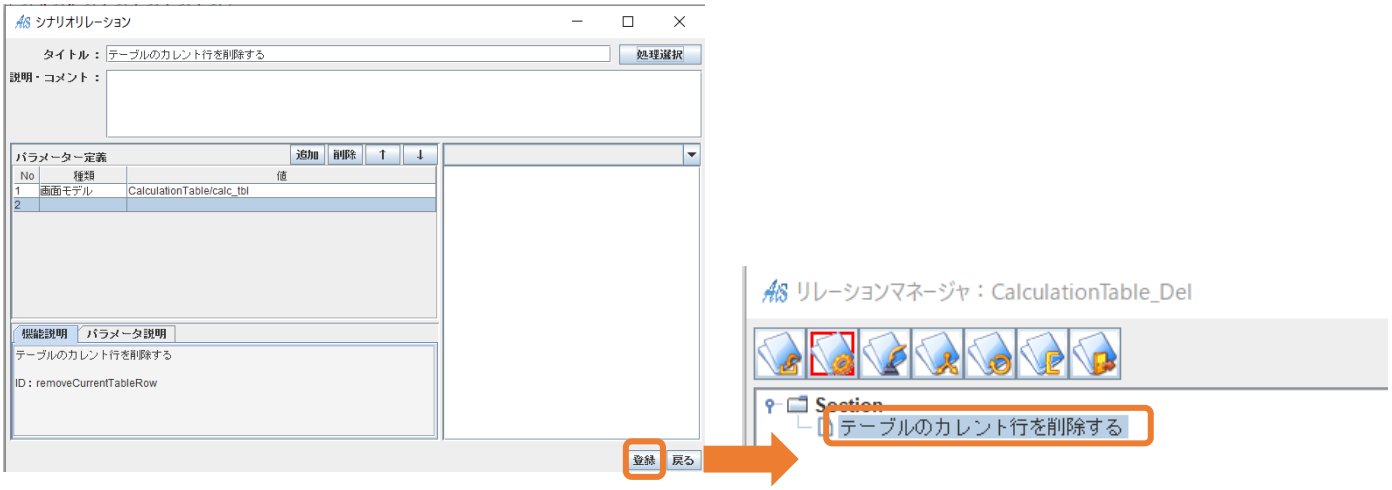
【図 50】



【図 51】



【図 52】



ここで F5 キーを押し、**テスト実行**を行いましょう。

問題がなければ「CalculationTable」で選択した行を削除することができます。

5.3.7 CalculationTable_Edit

「CalculationTable」で選択した行の修正を行う実装

まずはイベント割当定義が出来ているか確認お願いします。

手順１：選択中である行のテーブル行モデルを取得

手順２：変数「edit 値」から「1 項」の値を取得する

手順３：「value」から値を取得して計算式の左側に代入する

手順４：変数「edit 値」から「演算子」の値を取得する

手順５：コンボボックスを選択状態にする

手順６：変数「edit 値」から「 2 項」の値を取得する

手順７：「value」から値を取得して計算式の左側に代入する

手順８：変数から「結果」の値を取得する

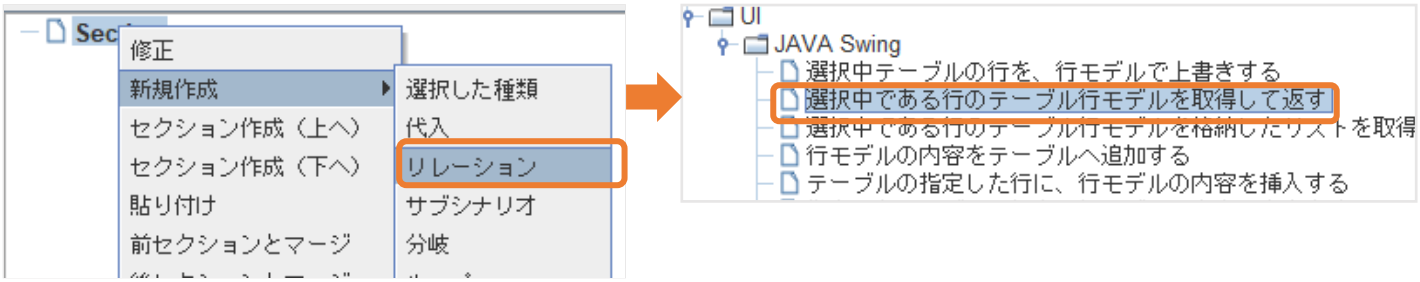
手順９：「value」から値を取得してテキストボックスに代入する

手順 10：テスト実行

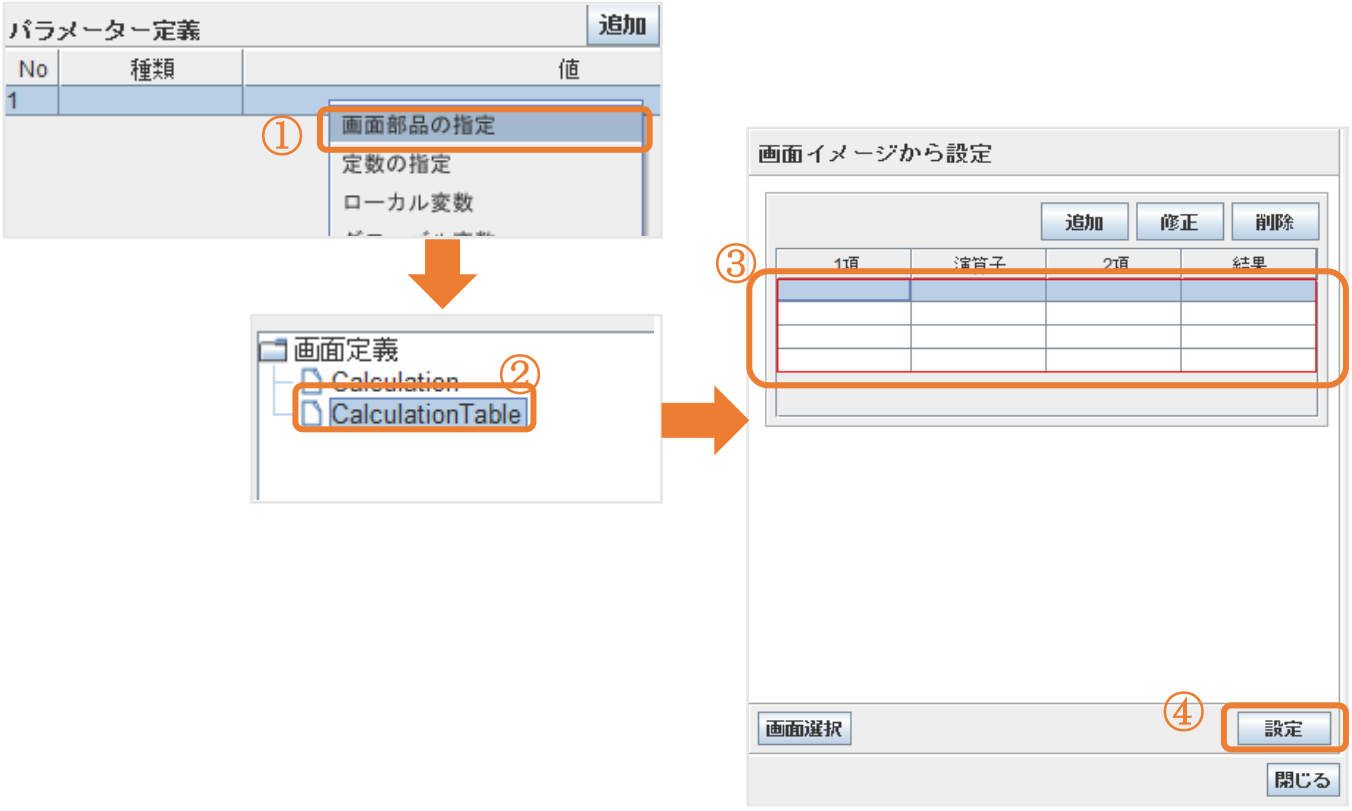
手順１：選択中である行のテーブル行モデルを取得

ローカル変数「edit 値」は未作成なので作成してください。

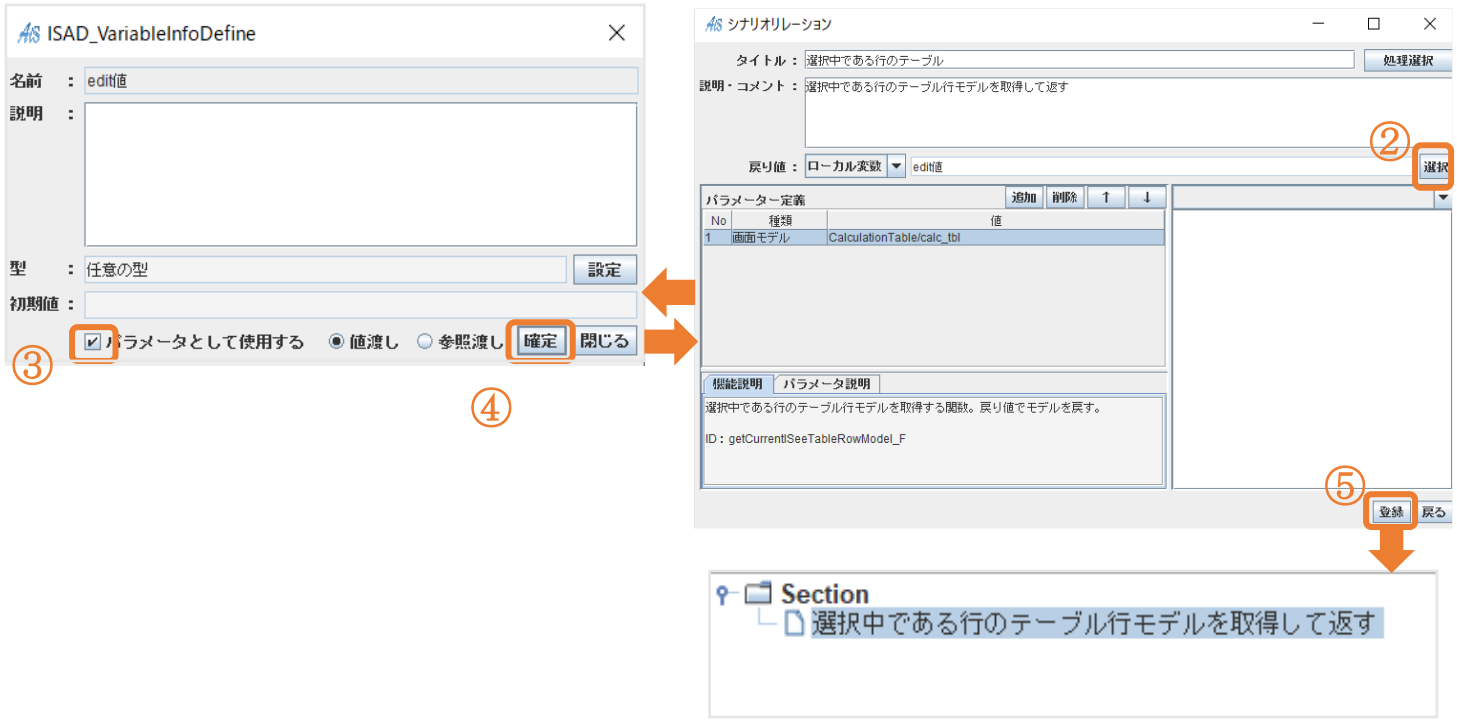
【図 53】



【図 54】



【図 55】

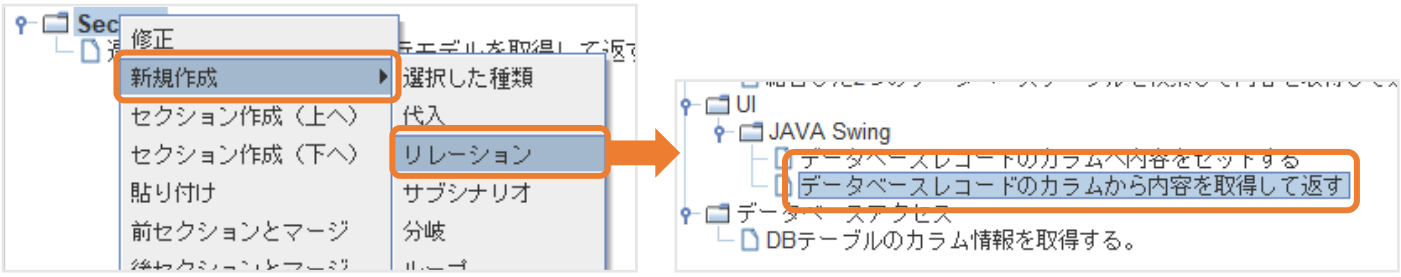


「パラメータとして使用する」にチェックを入れてください

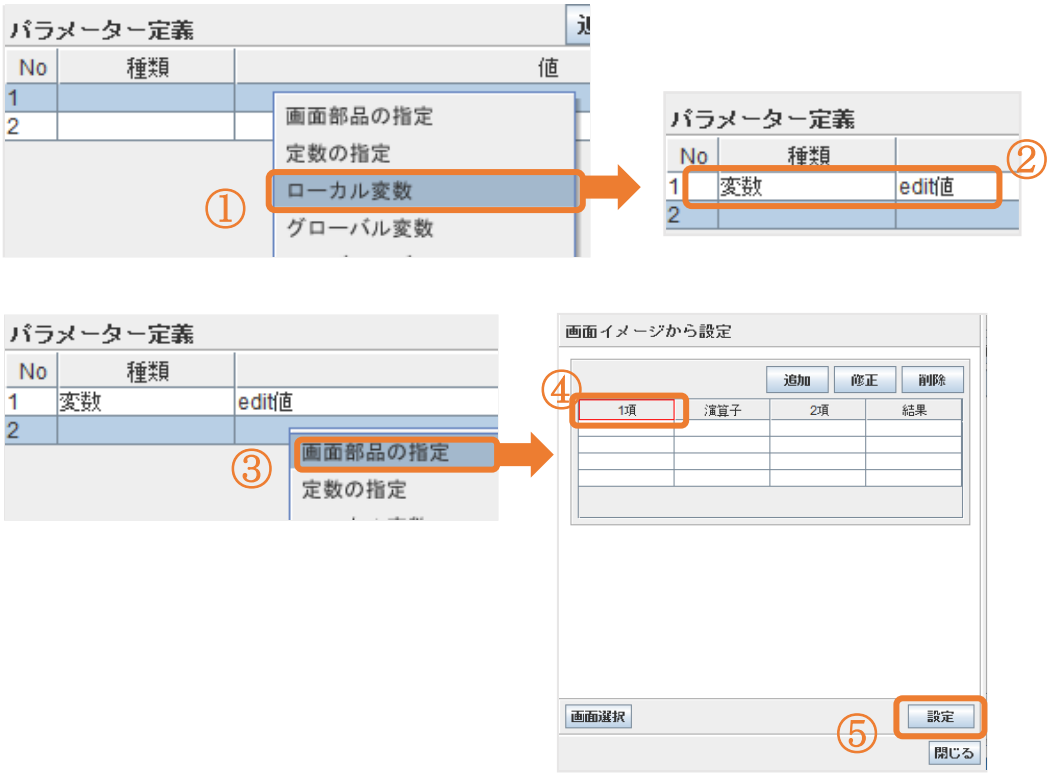
手順２：変数「edit 値」から「1 項」の値を取得する

ローカル変数「value」は未作成なので作成してください。

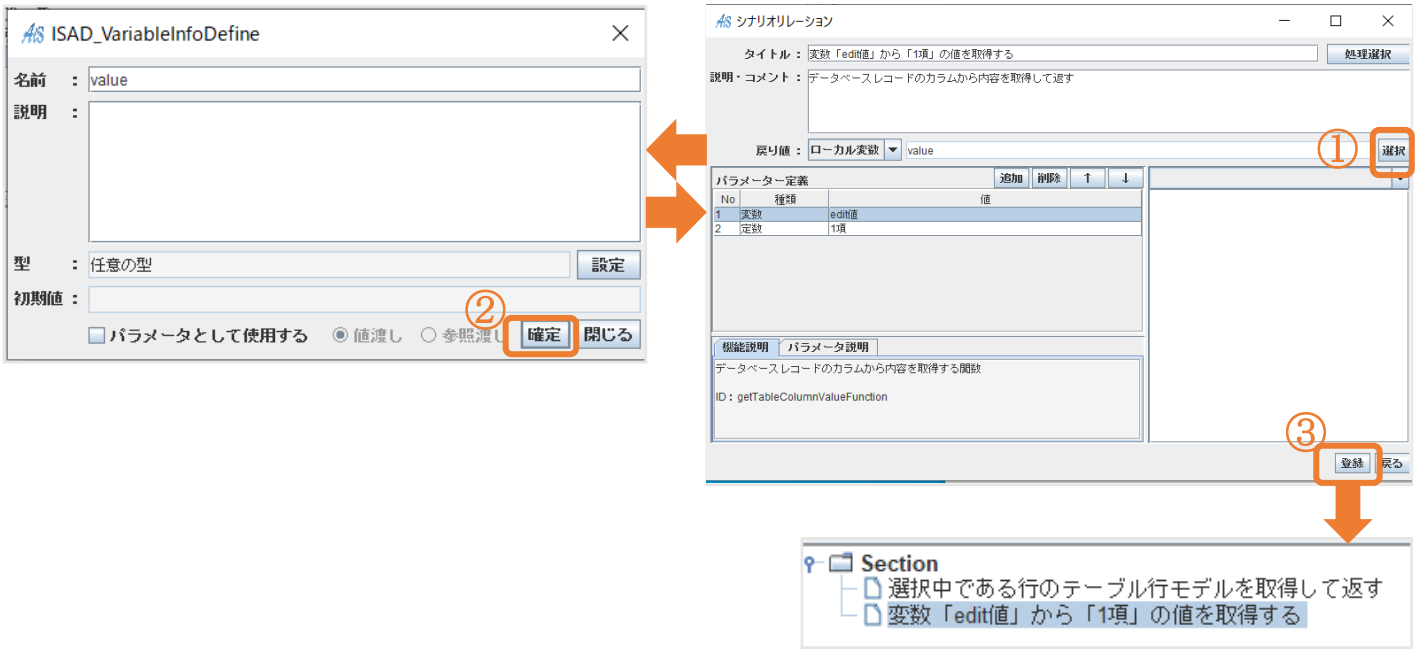
【図 56】



【図 57】

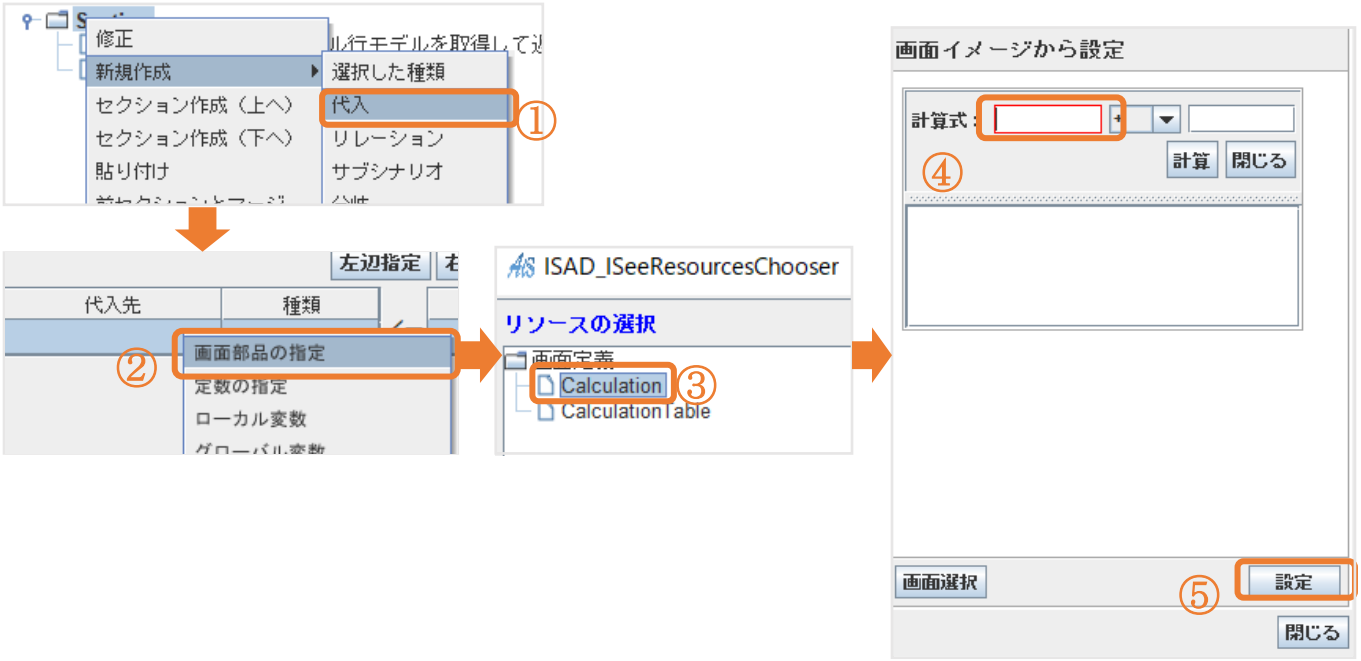

パラメーター定義
No
種類
値
1
2
画面部品の指定
定数の指定
ローカル変数
グローバル変数
①
パラメーター定義
No
種類
1
変数
edit値
2
②
パラメーター定義
No
種類
1
変数
edit値
2
③
画面部品の指定
定数の指定
画面イメージから設定
追加
修正
削除
④
1項
演算子
2項
結果
画面選択
設定
⑤
閉じる

【図 58】

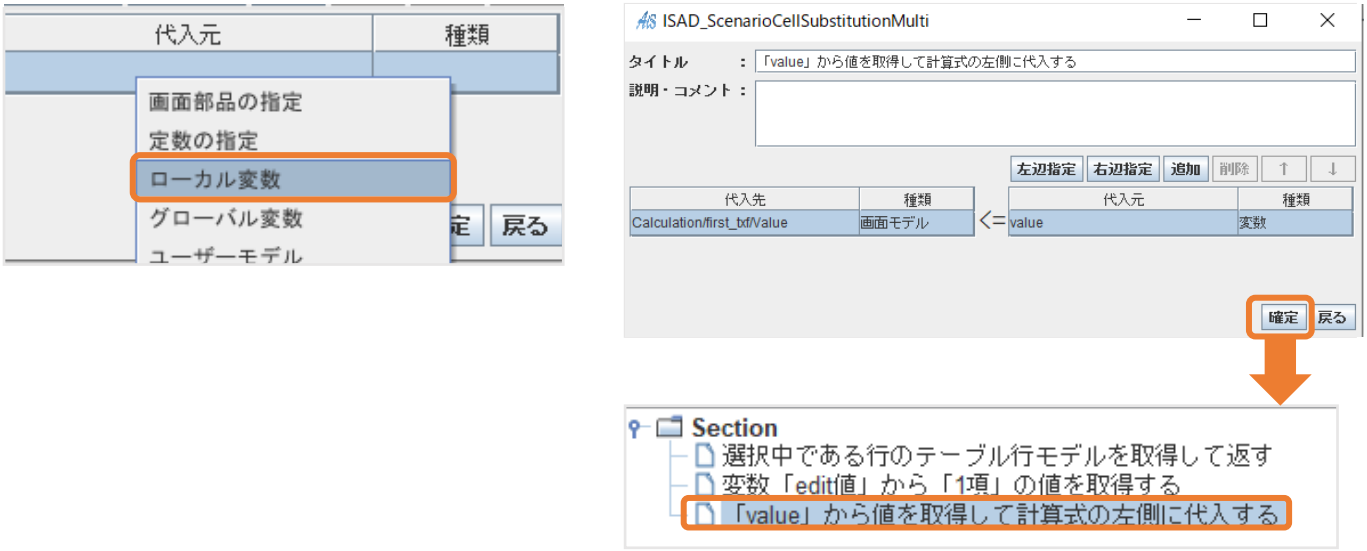

ISAD_VariableInfoDefine
名前 ：
value
説明 ：
型 ：
任意の型
設定
初期値 ：
パラメータとして使用する
値渡し
参照渡し
確定
閉じる
②
シナリオリレーション
タイトル ：
変数「edit値」から「1項」の値を取得する
処理選択
説明・コメント ：
データベースレコードのカラムから内容を取得して返す
戻り値 ：
ローカル変数
value
①
選択
パラメーター定義
追加
削除
No
種類
値
1
変数
edit値
2
定数
1項
機能説明
パラメータ説明
データベースレコードのカラムから内容を取得する関数
ID：getTableColumnValueFunction
③
登録
戻る
Section
選択中である行のテーブル行モデルを取得して返す
変数「edit値」から「1項」の値を取得する

手順３：「value」から値を取得して計算式の左側に代入する

【図 59】



【図 60】



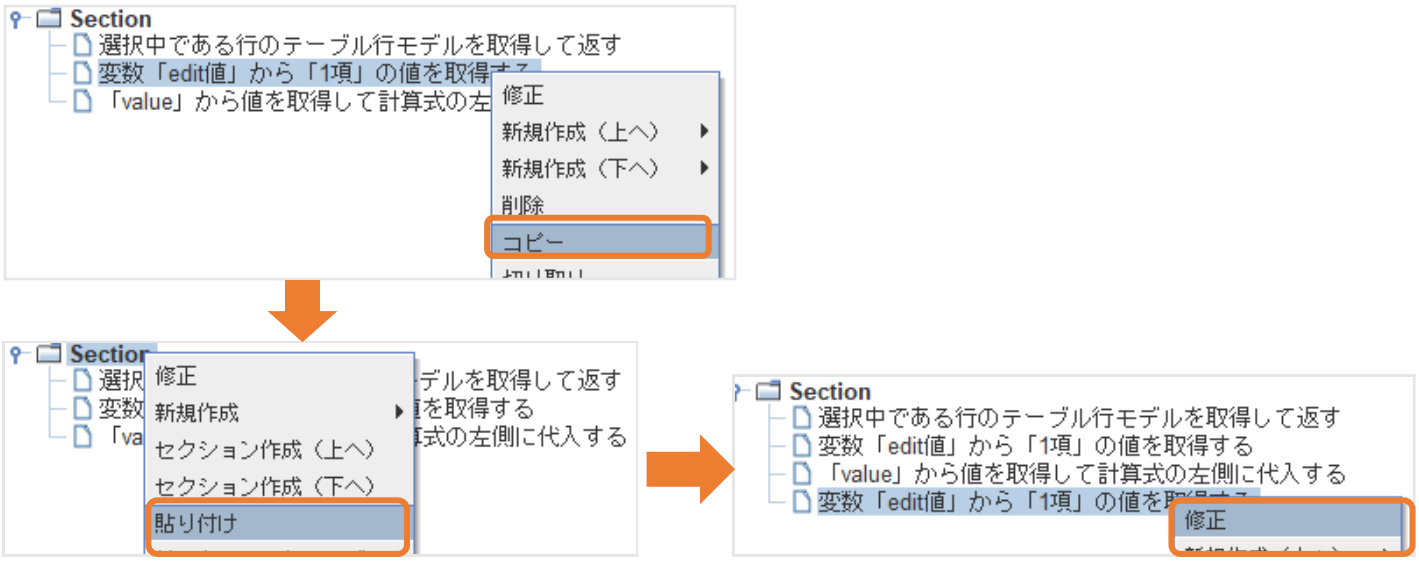
※図 57 に「ローカル変数」と「変数」がありますが、同じものです。

手順４ ：変数「edit 値」から「演算子」の値を取得する

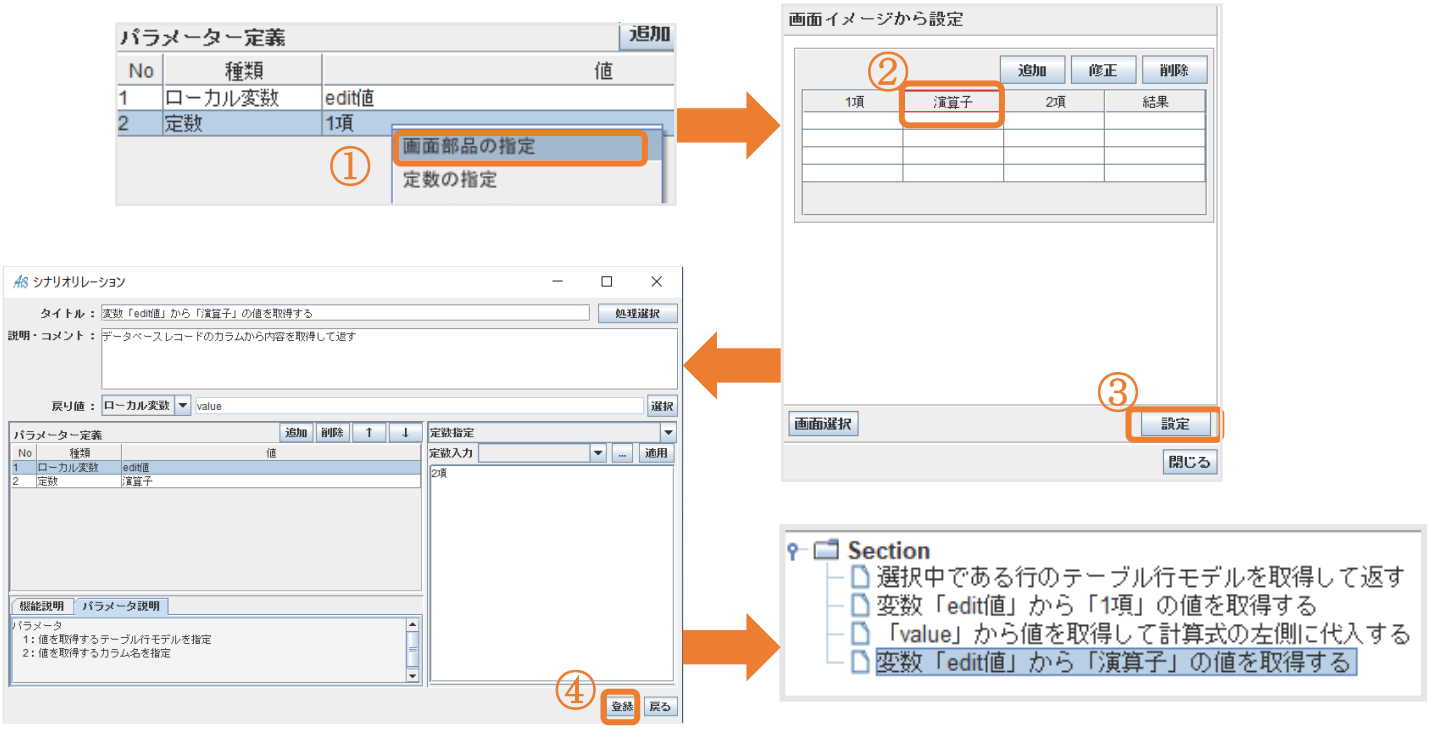
２ ：変数「edit 値」から「1 項」の値を取得するをコピーして修正をしましょう

違うポイントは「画面部品」で指定するカラムです。

【図 61】

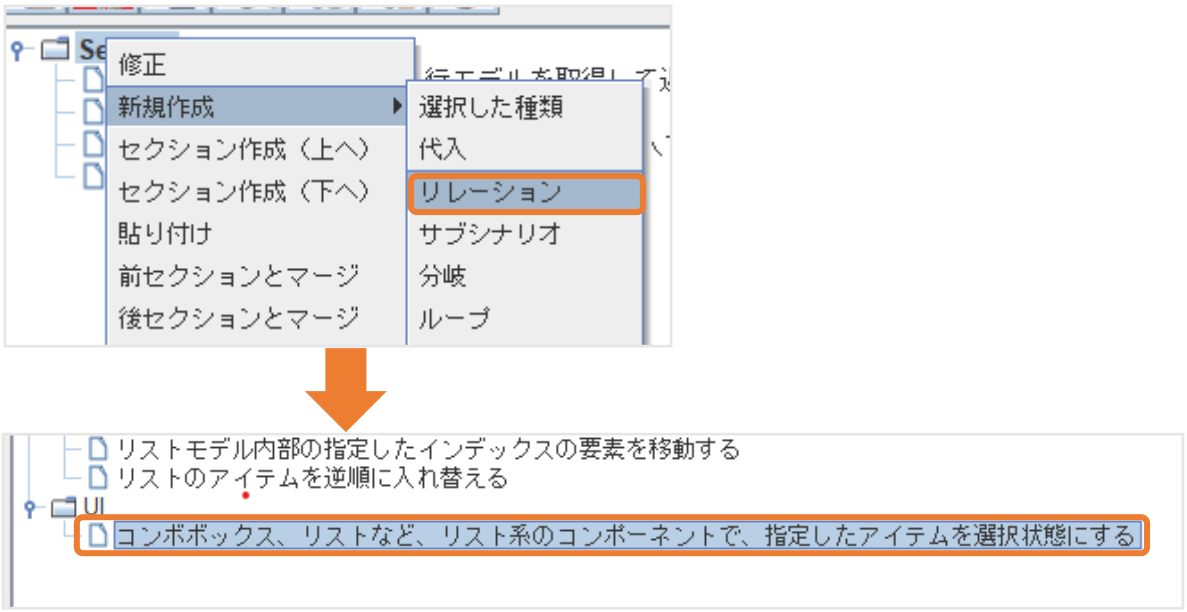


【図 62】

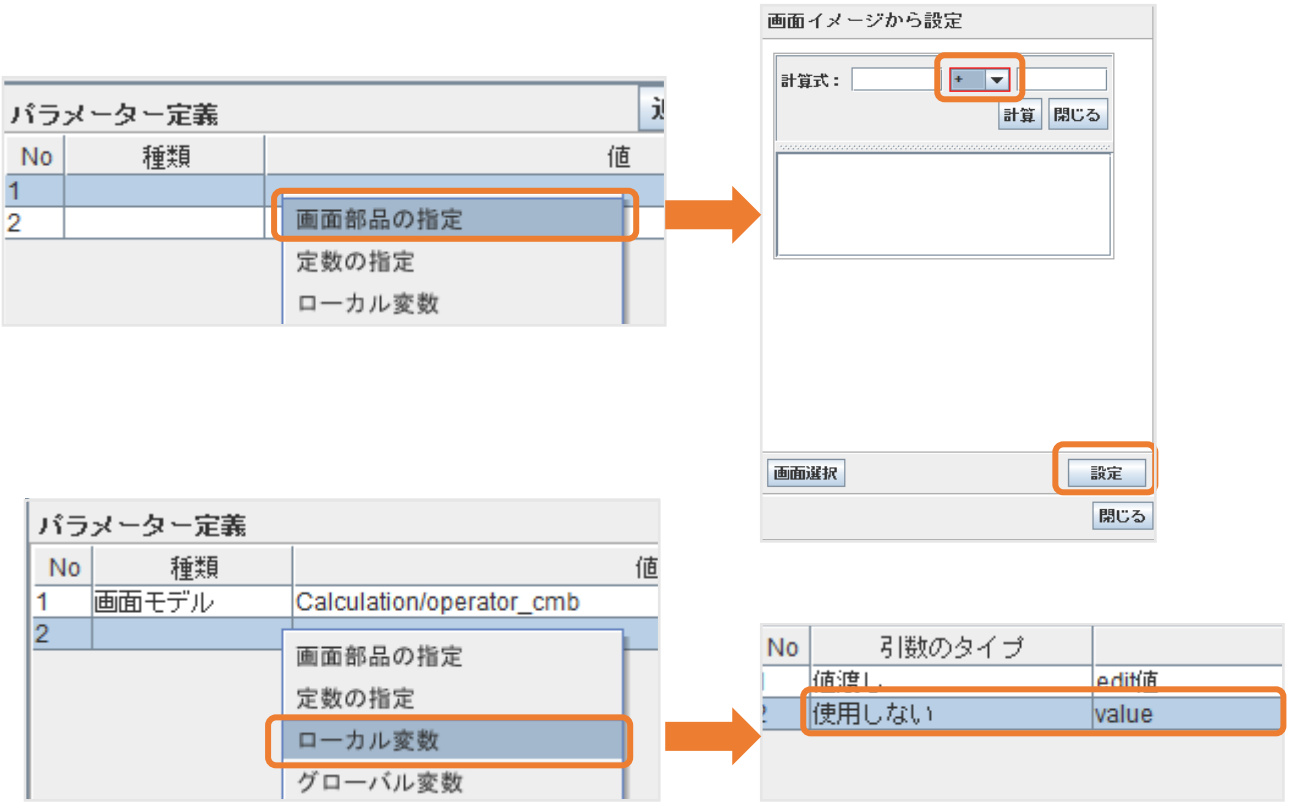

パラメーター定義
追加
No
種類
値
1
ローカル変数
edit値
2
定数
1項
①
画面部品の指定
定数の指定
画面イメージから設定
②
追加
修正
削除
1項
演算子
2項
結果
③
画面選択
設定
閉じる
シナリオリレーション
タイトル：
変数「edit値」から「演算子」の値を取得する
処理選択
説明・コメント：
データベースレコードのカラムから内容を取得して返す
戻り値：
ローカル変数
value
選択
パラメーター定義
追加
削除
定数指定
定数入力
適用
機能説明
パラメータ説明
パラメータ
1：値を取得するテーブル行モデルを指定
2：値を取得するカラム名を指定
④
登録
戻る
Section
選択中である行のテーブル行モデルを取得して返す
変数「edit値」から「1項」の値を取得する
「value」から値を取得して計算式の左側に代入する
変数「edit値」から「演算子」の値を取得する

手順５：コンボボックスを選択状態にする

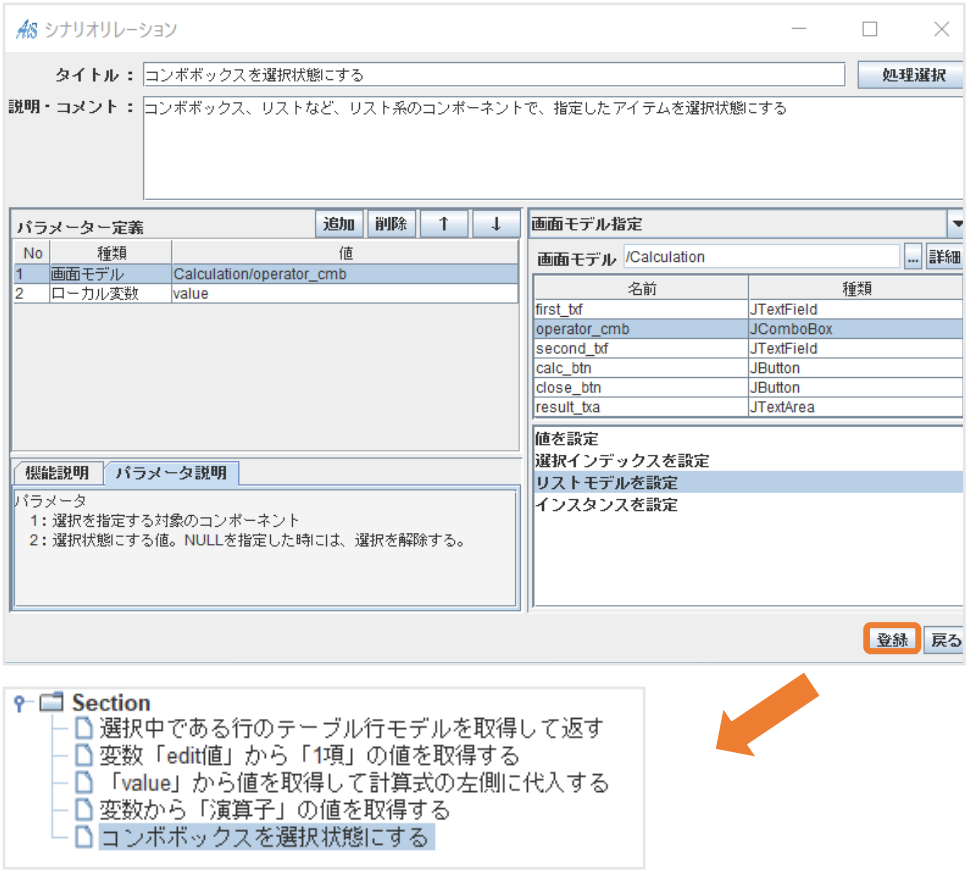
【図 63】



【図 64】



【図 65】


シナリオリレーション
タイトル：
コンボボックスを選択状態にする
処理選択
説明・コメント：
コンボボックス、リストなど、リスト系のコンポーネントで、指定したアイテムを選択状態にする
パラメーター定義
追加
削除
No
種類
値
1
画面モデル
Calculation/operator_cmb
2
ローカル変数
value
機能説明
パラメータ説明
パラメータ
1：選択を指定する対象のコンポーネント
2：選択状態にする値。NULLを指定した時には、選択を解除する。
画面モデル指定
画面モデル
/Calculation
詳細
名前
種類
first_txf
JTextField
operator_cmb
JComboBox
second_txf
JTextField
calc_btn
JButton
close_btn
JButton
result_txa
JTextArea
値を設定
選択インデックスを設定
リストモデルを設定
インスタンスを設定
登録
戻る

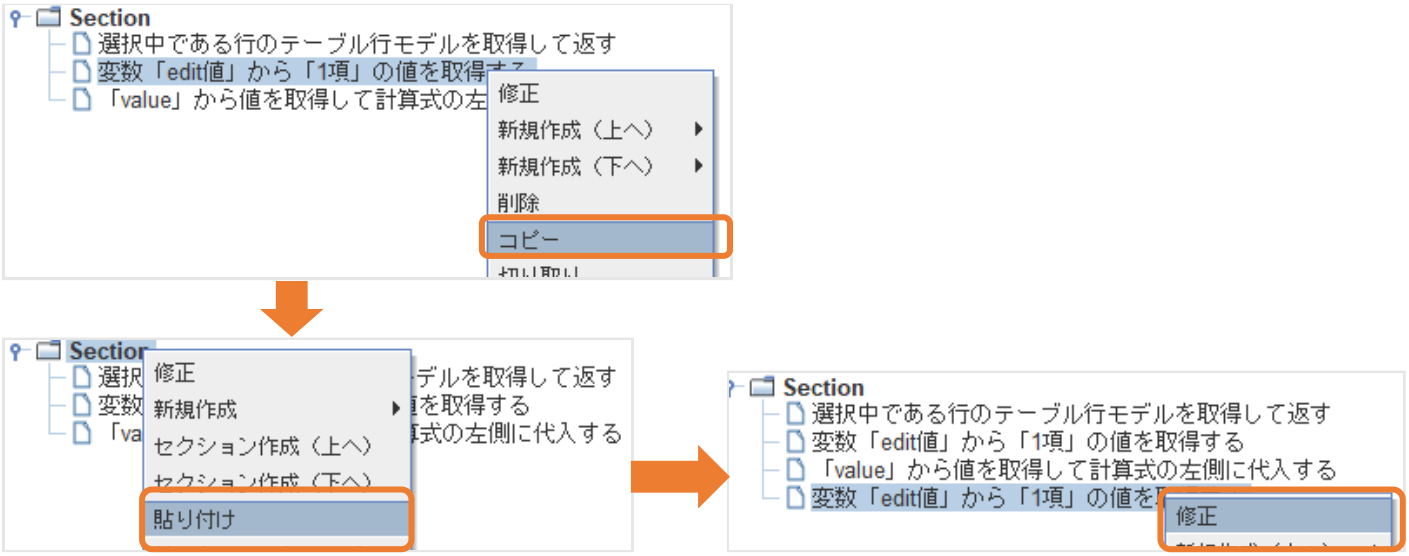
Section
選択中である行のテーブル行モデルを取得して返す
変数「edit値」から「1項」の値を取得する
「value」から値を取得して計算式の左側に代入する
変数から「演算子」の値を取得する
コンボボックスを選択状態にする

手順６ ：変数「edit 値」から「２項」の値を取得する

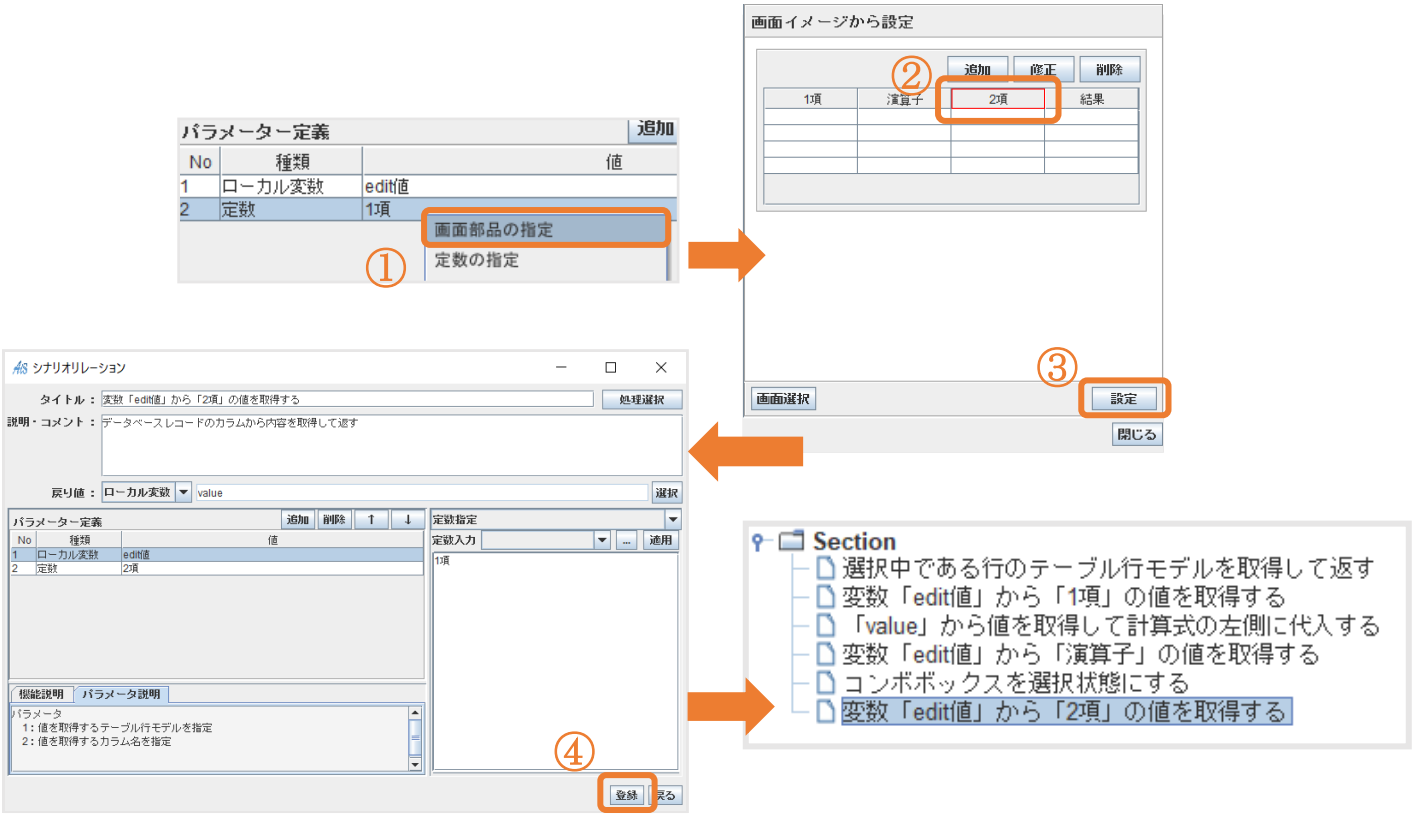
２ ：変数「edit 値」から「1 項」の値を取得するをコピーして修正をしましょう

違うポイントは「画面部品」で指定するカラムです。

【図 66】

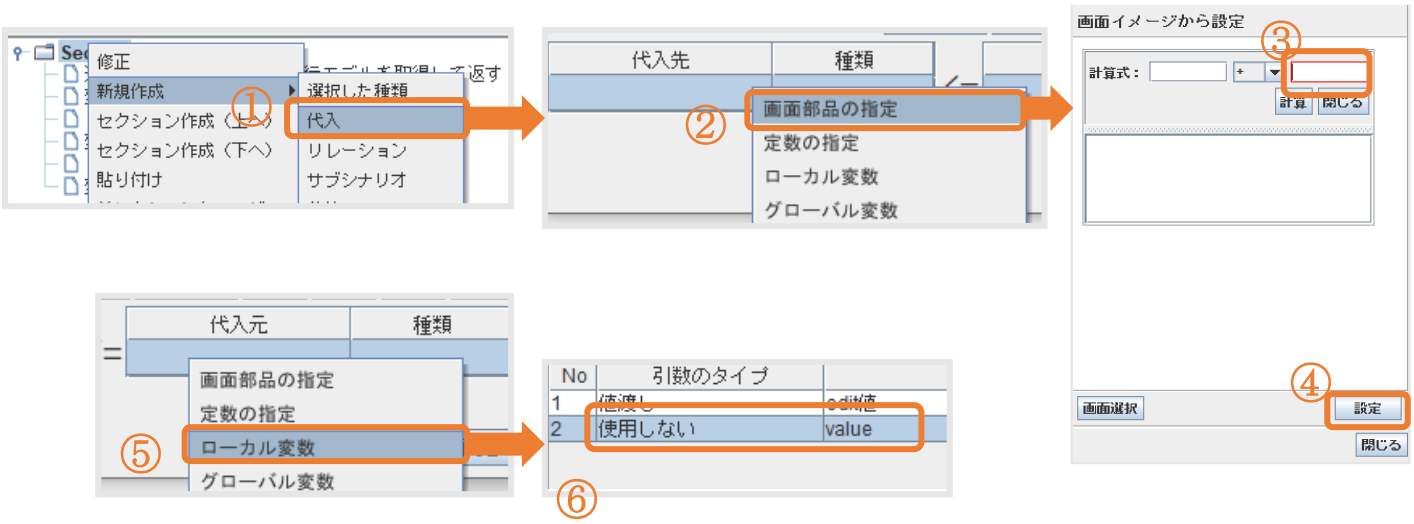


【図 67】

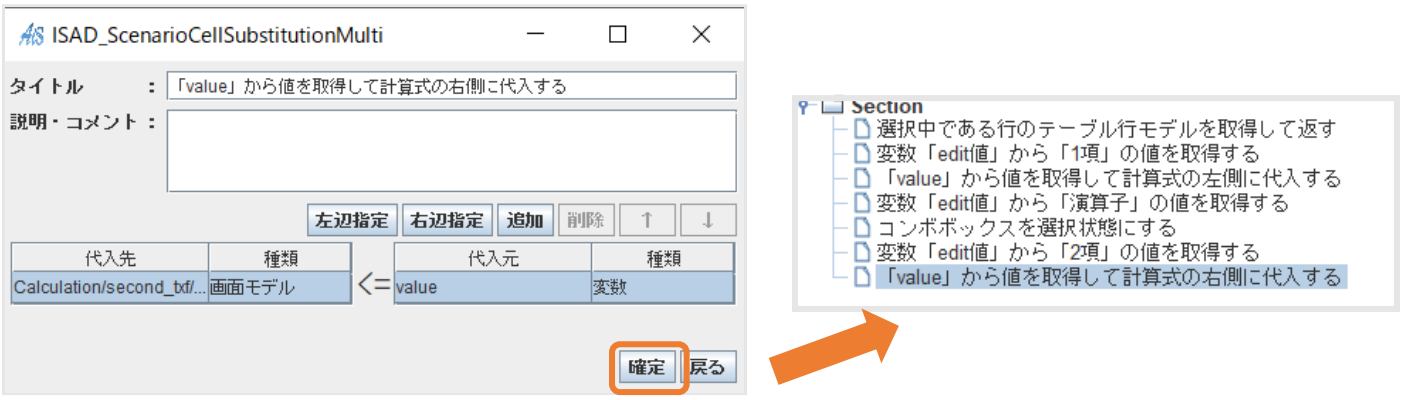

パラメーター定義
追加
No
種類
値
1
ローカル変数
edit値
2
定数
1項
画面部品の指定
定数の指定
①
画面イメージから設定
②
追加
修正
削除
1項
演算子
2項
結果
③
画面選択
設定
閉じる
シナリオリレーション
タイトル：
変数「edit値」から「2項」の値を取得する
処理選択
説明・コメント：
データベースレコードのカラムから内容を取得して返す
戻り値：
ローカル変数
value
選択
パラメーター定義
追加
削除
定数指定
定数入力
適用
機能説明
パラメータ説明
パラメータ
1：値を取得するテーブル行モデルを指定
2：値を取得するカラム名を指定
④
登録
Section
選択中である行のテーブル行モデルを取得して返す
変数「edit値」から「1項」の値を取得する
「value」から値を取得して計算式の左側に代入する
変数「edit値」から「演算子」の値を取得する
コンボボックスを選択状態にする
変数「edit値」から「2項」の値を取得する

手順７：「value」から値を取得して計算式の左側に代入する

【図 68】

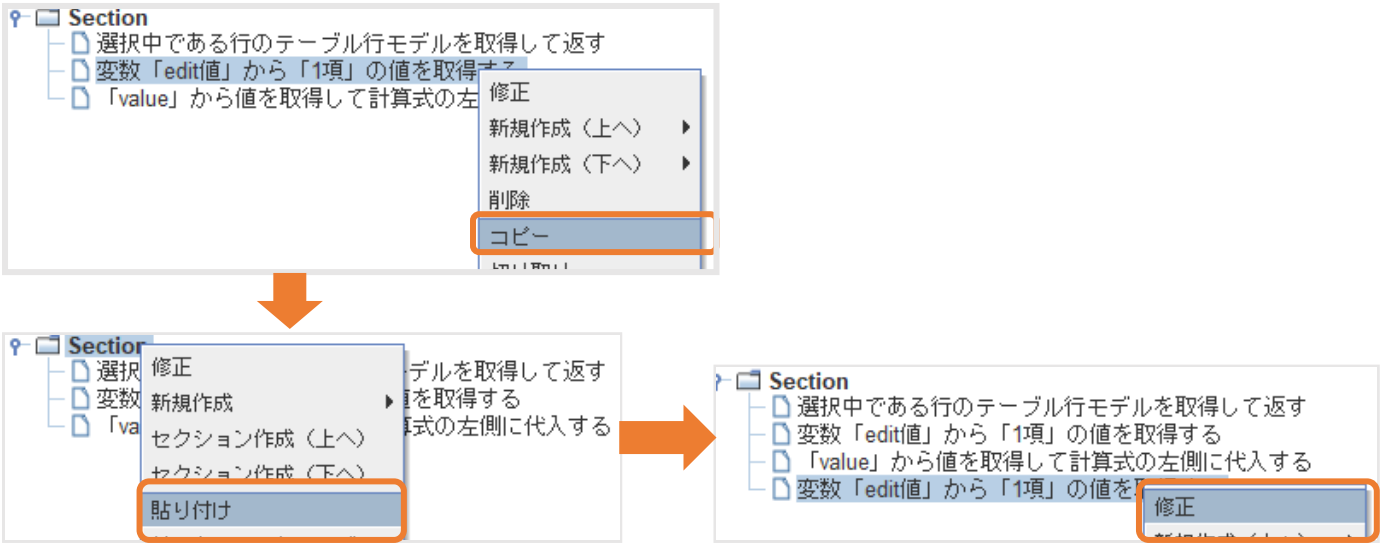


【図 69】

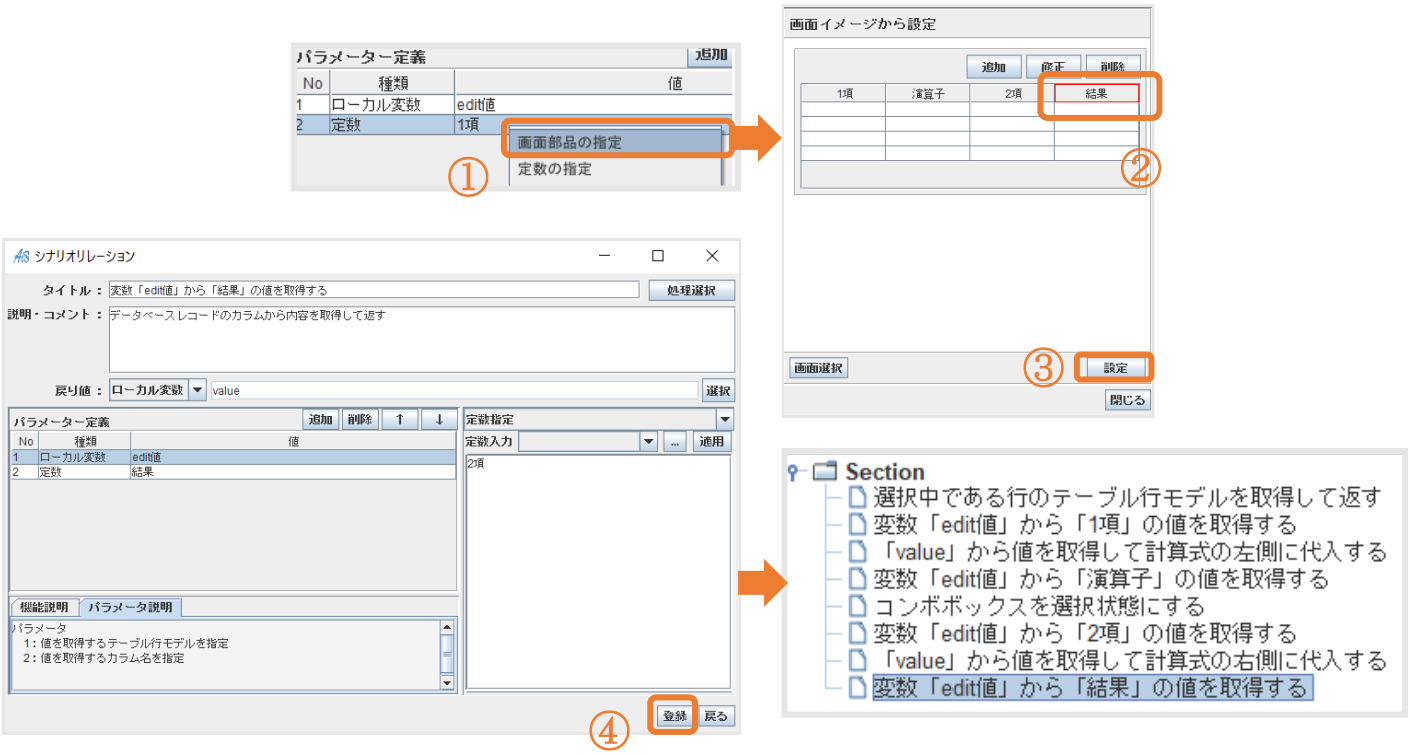


手順８：変数から「結果」の値を取得する

【図 70】

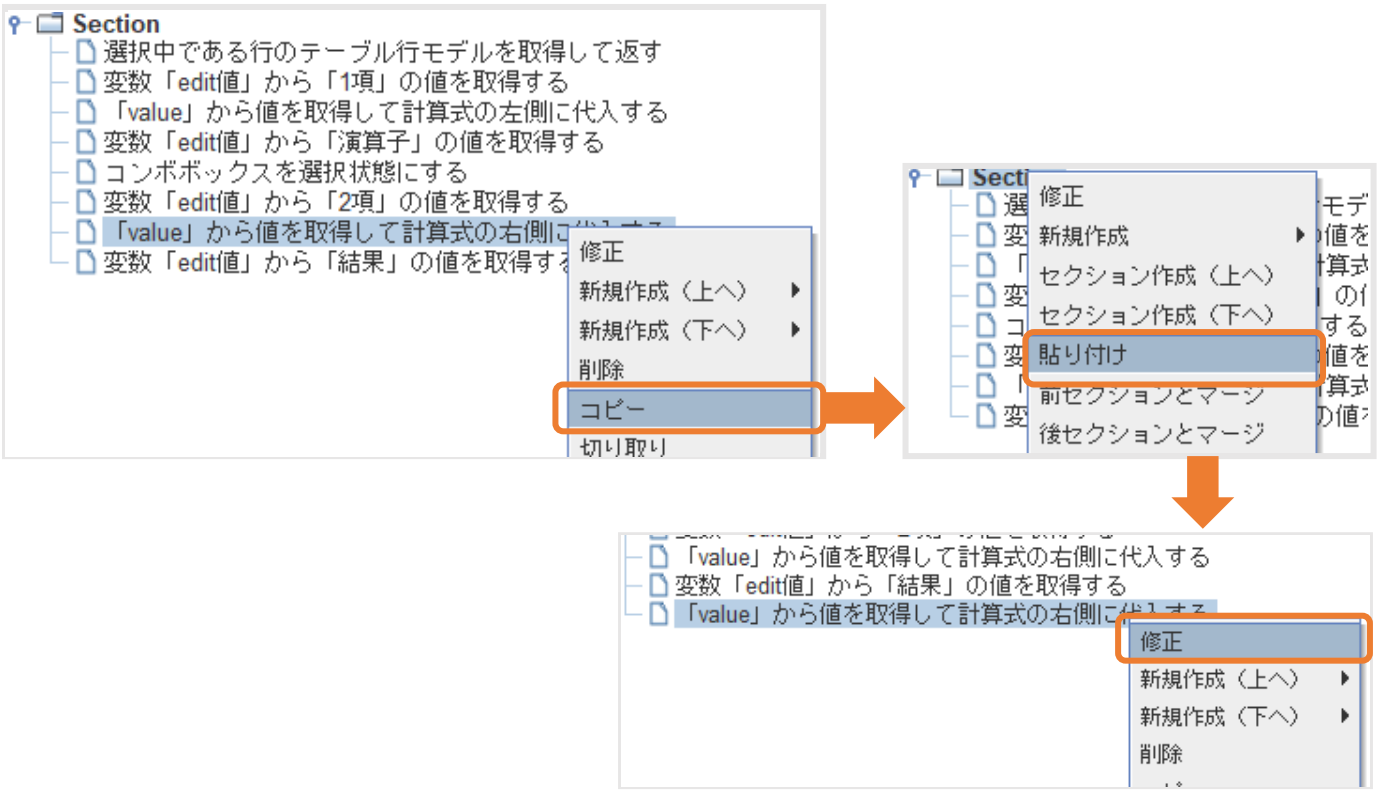


【図 71】

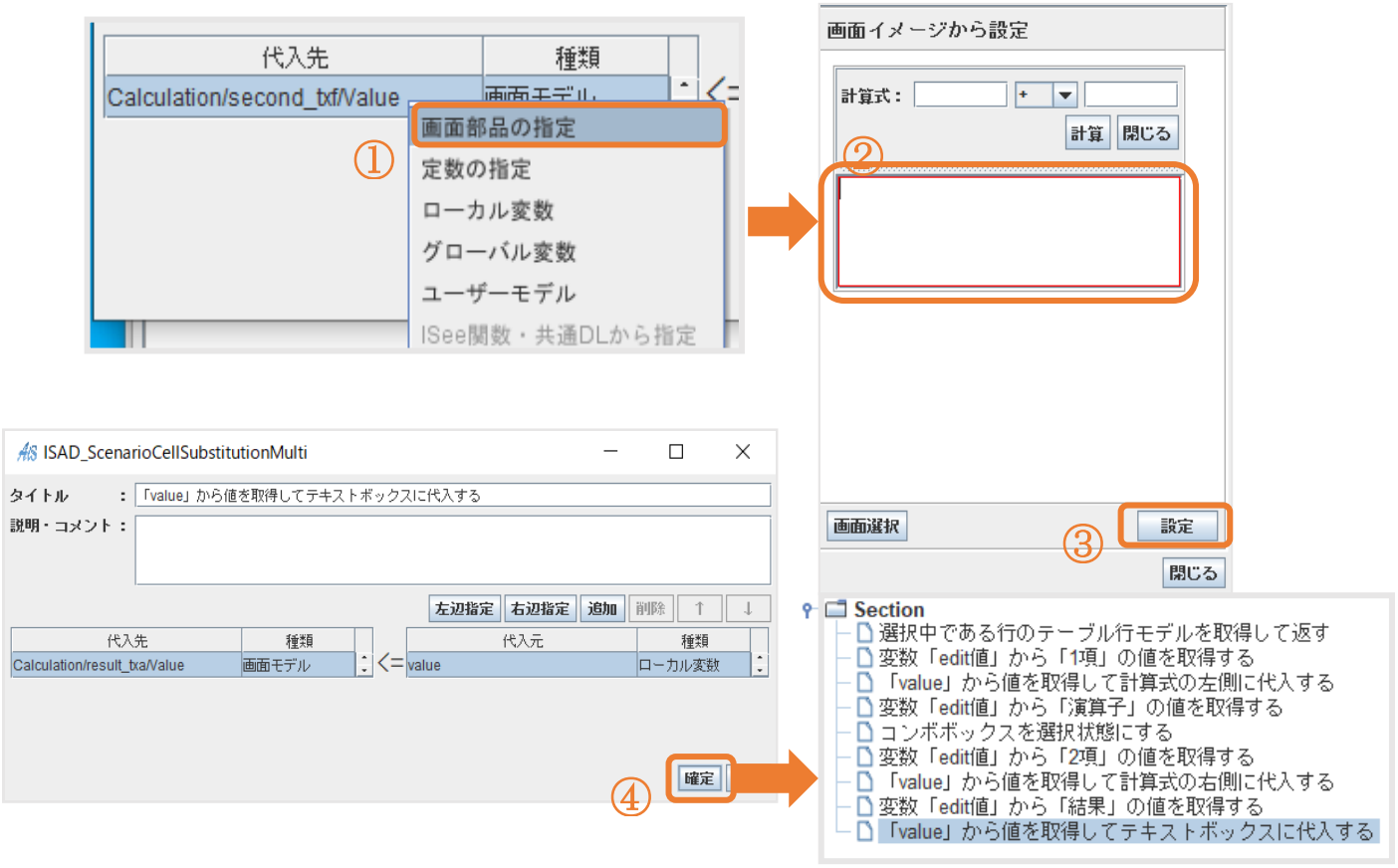


手順９：「value」から値を取得してテキストボックスに代入する

【図 72】

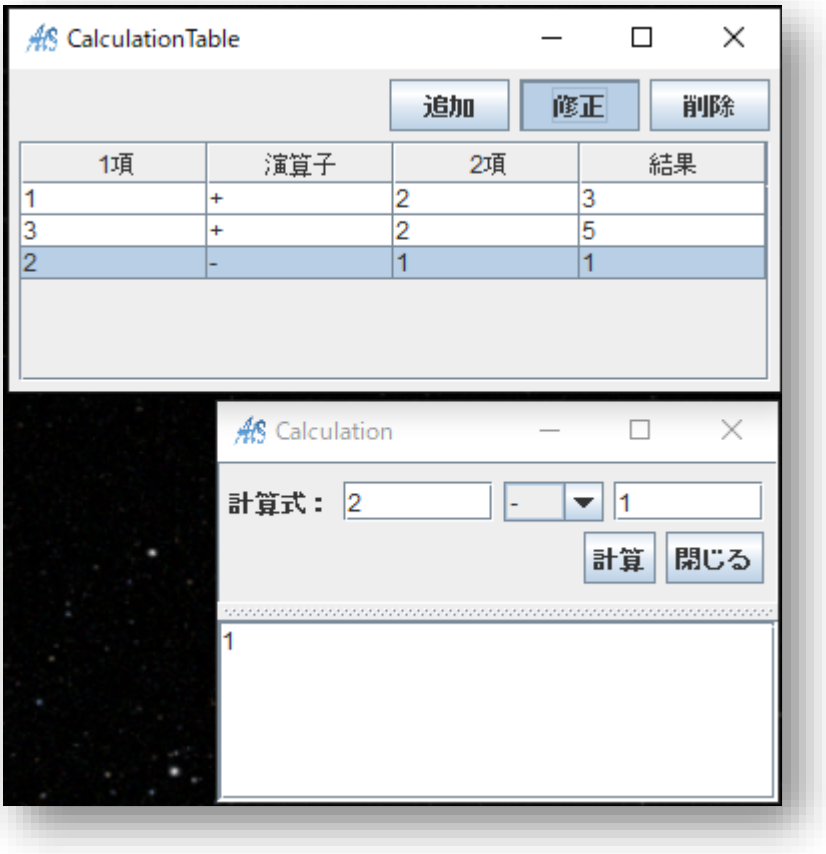


【図 73】


代入先
種類
Calculation/second_txf/Value
画面部品の指定
定数の指定
ローカル変数
グローバル変数
ユーザーモデル
ISee関数・共通DLから指定
①
画面イメージから設定
計算式：
計算
閉じる
②
画面選択
設定
③
閉じる
ISAD_ScenarioCellSubstitutionMulti
タイトル
：
「value」から値を取得してテキストボックスに代入する
説明・コメント：
左辺指定
右辺指定
追加
削除
代入先
種類
代入元
種類
Calculation/result_txa/Value
画面モデル
value
ローカル変数
確定
④
Section
選択中である行のテーブル行モデルを取得して返す
変数「edit値」から「1項」の値を取得する
「value」から値を取得して計算式の左側に代入する
変数「edit値」から「演算子」の値を取得する
コンボボックスを選択状態にする
変数「edit値」から「2項」の値を取得する
「value」から値を取得して計算式の右側に代入する
変数「edit値」から「結果」の値を取得する
「value」から値を取得してテキストボックスに代入する

ここで F5 キーを押し、**テスト実行**を行いましょう。

問題がなければ「CalculationTable」の行を選択して「修正」を押下すると「Calculation」に選択した行の計算内容が反映されます。



5.3.8 【共通】合計処理

「CulculationTable」で「メニュー」画面から「合計」を押下すると「結果」カラムの合計数値が表示される実装

手順１：シナリオ定義

手順２：「メニュー」画面の作成

手順３：「CalculationTable」で右クリックをすると「メニュー」が出てくる実装

手順４：テーブルの全ての行を取得する

手順５：最終合計値に「0」を代入する

手順６：すべての行に対して、下記の処理を繰り返し実行

手順７：行の「結果」カラムから値を取り出す

手順８：取り出した値を合計値に足す

手順９：計算された合計値を返す

手順 10：テスト実行

手順１：シナリオ定義

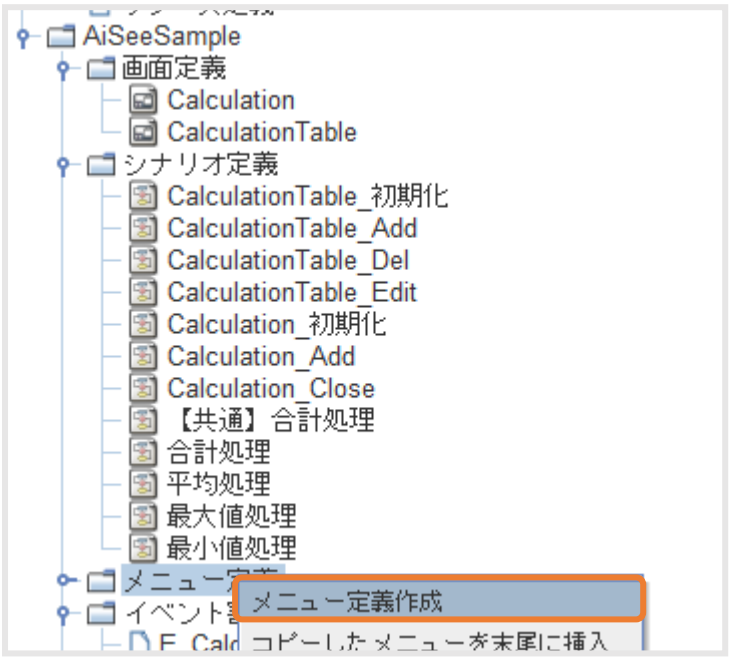
今回実装する「合計処理」のシナリオ定義を作成しましょう。

その後実装する「平均」「最大値」「最小値」「【共通】合計処理」のシナリオ定義作成も同時に行います。

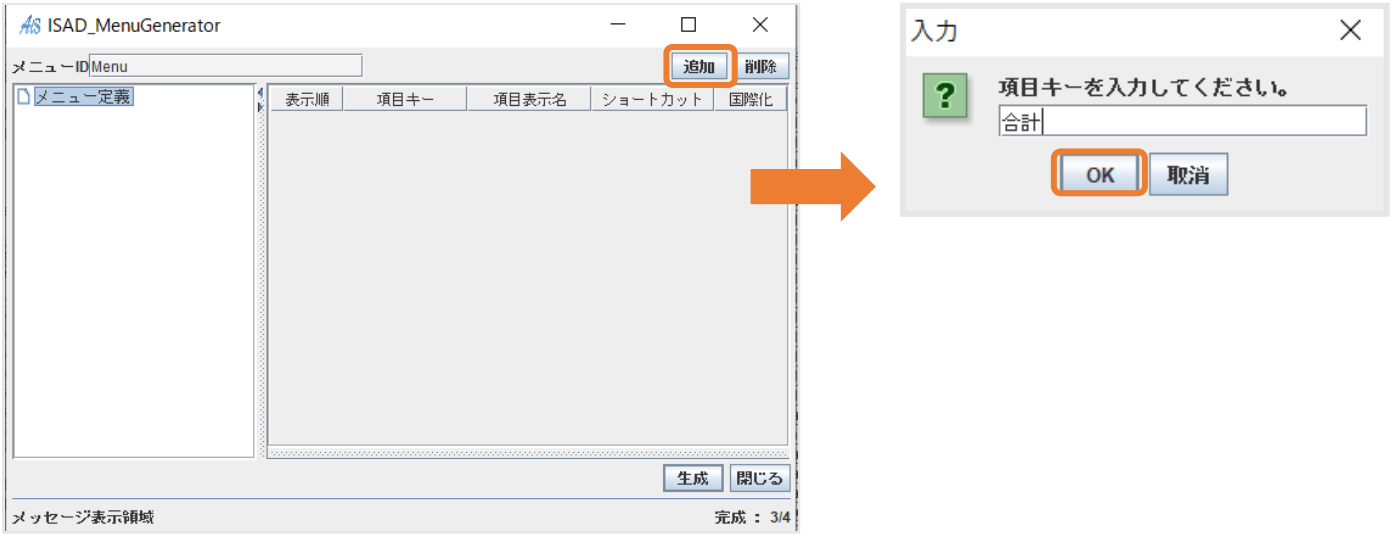
作り方がわからなくなってしまった方はこちら。

手順２：「メニュー」画面の作成

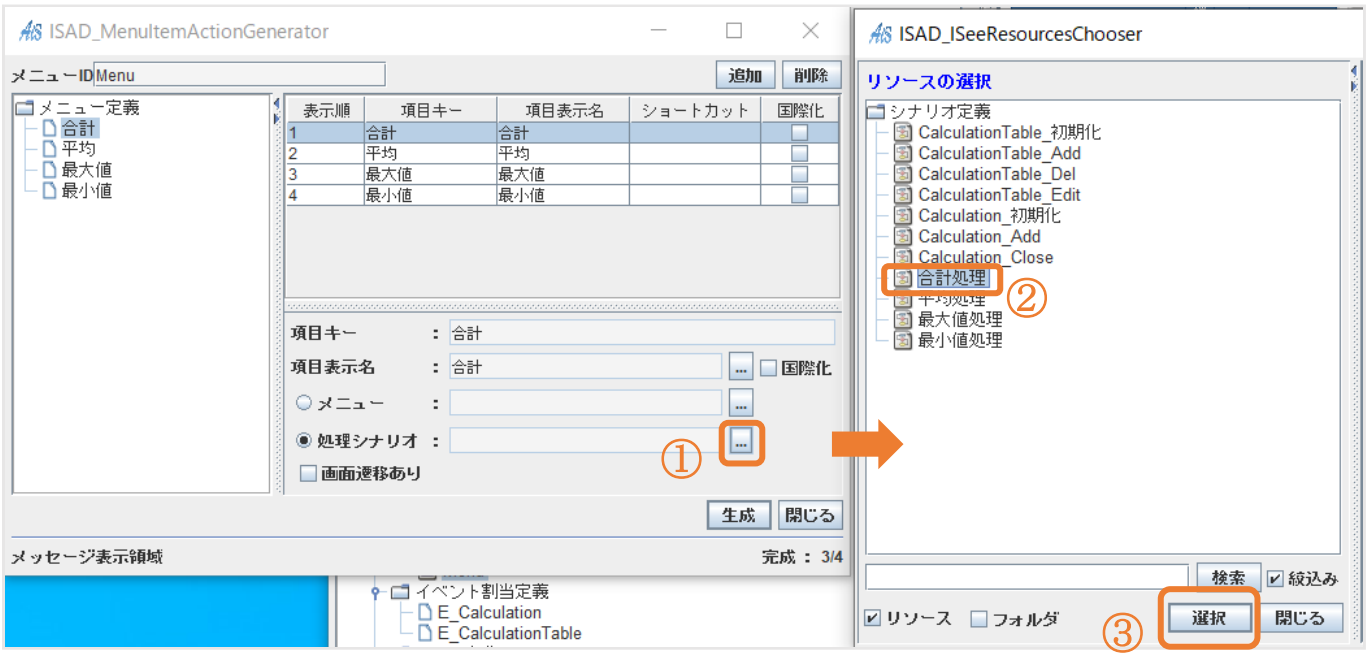
【図 74】



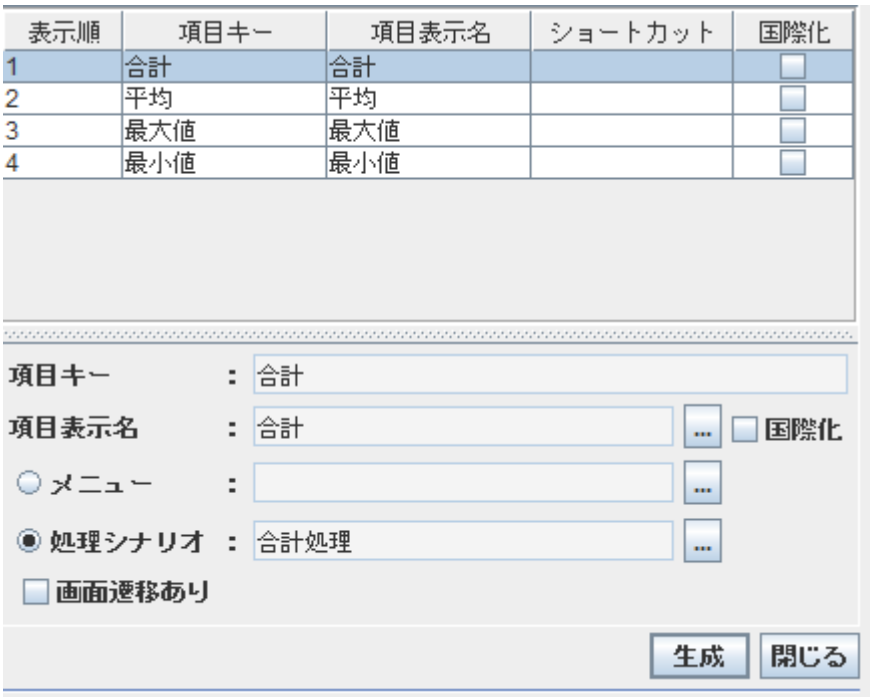
【図 75】


ISAD_MenuGenerator
メニューID Menu
追加
削除
メニュー定義
表示順
項目キー
項目表示名
ショートカット
国際化
生成
閉じる
メッセージ表示領域
完成 : 3/4
入力
項目キーを入力してください。
合計
OK
取消

【図 76】

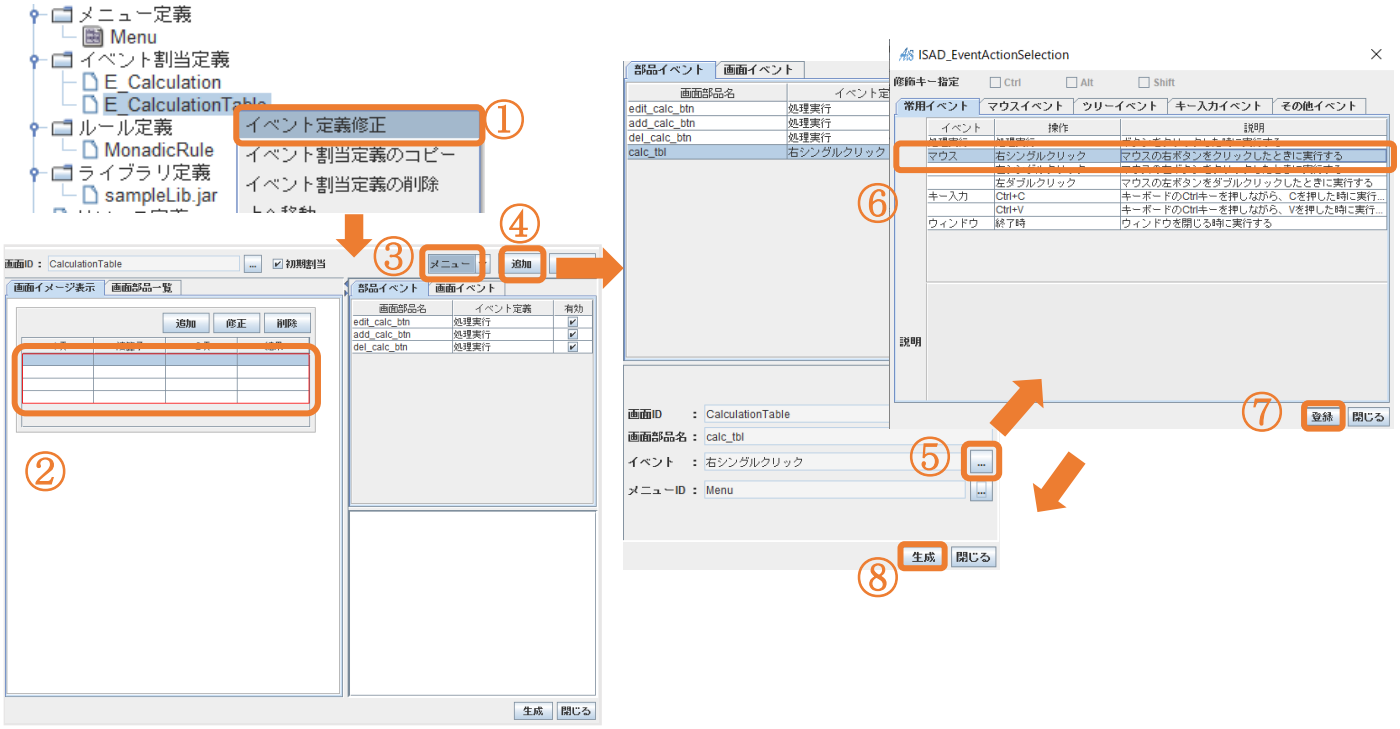

ISAD_MenuItemActionGenerator
メニューID Menu
追加
削除
メニュー定義
合計
平均
最大値
最小値
表示順
項目キー
項目表示名
ショートカット
国際化
1 合計 合計
2 平均 平均
3 最大値 最大値
4 最小値 最小値
項目キー : 合計
項目表示名 : 合計
国際化
メニュー :
処理シナリオ :
画面遷移あり
①
生成
閉じる
メッセージ表示領域
完成 : 3/4
イベント割当定義
E_Calculation
E_CalculationTable
ISAD_ISeeResourcesChooser
リソースの選択
シナリオ定義
CalculationTable_初期化
CalculationTable_Add
CalculationTable_Del
CalculationTable_Edit
Calculation_初期化
Calculation_Add
Calculation_Close
合計処理
②
最大値処理
最小値処理
検索
絞込み
リソース
フォルダ
③
選択
閉じる

【図 77】


表示順
項目キー
項目表示名
ショートカット
国際化
1
合計
合計
2
平均
平均
3
最大値
最大値
4
最小値
最小値
項目キー ： 合計
項目表示名 ： 合計
国際化
メニュー ：
処理シナリオ ： 合計処理
画面遷移あり
生成
閉じる

手順３：「CalculationTable」で右クリックをすると「メニュー」が出てくる実装

【図 78】

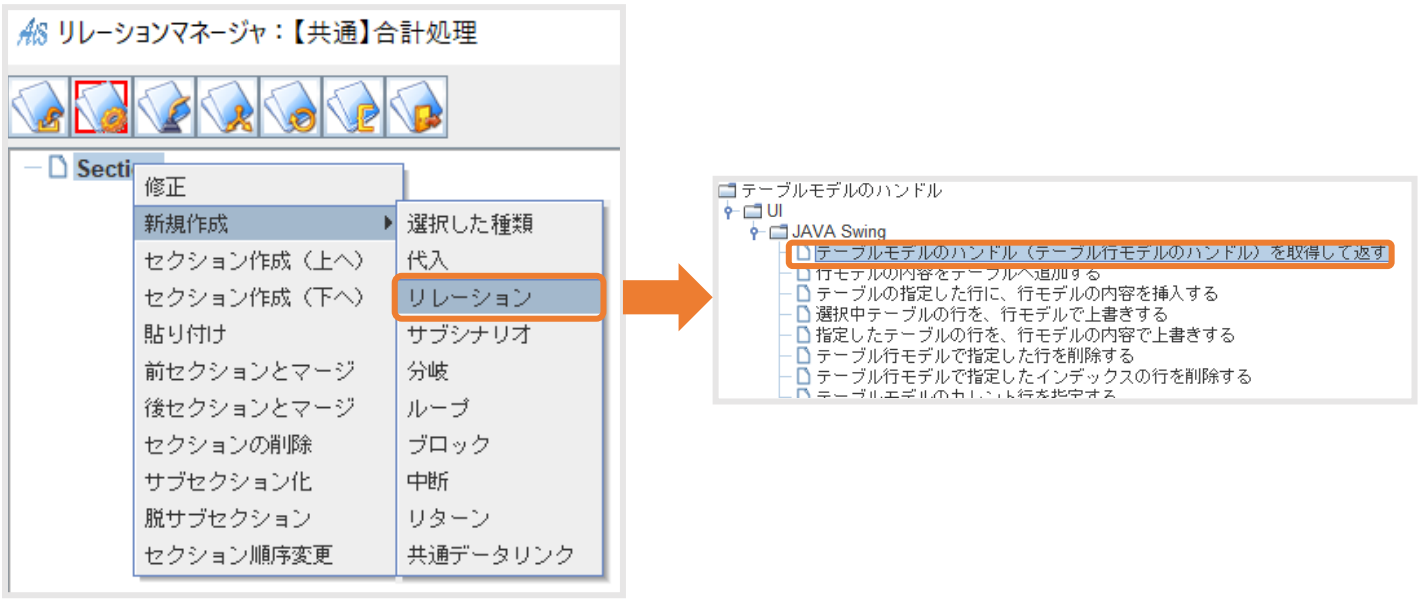


ここから「【共通】合計処理」のリレーション作成に移ります。

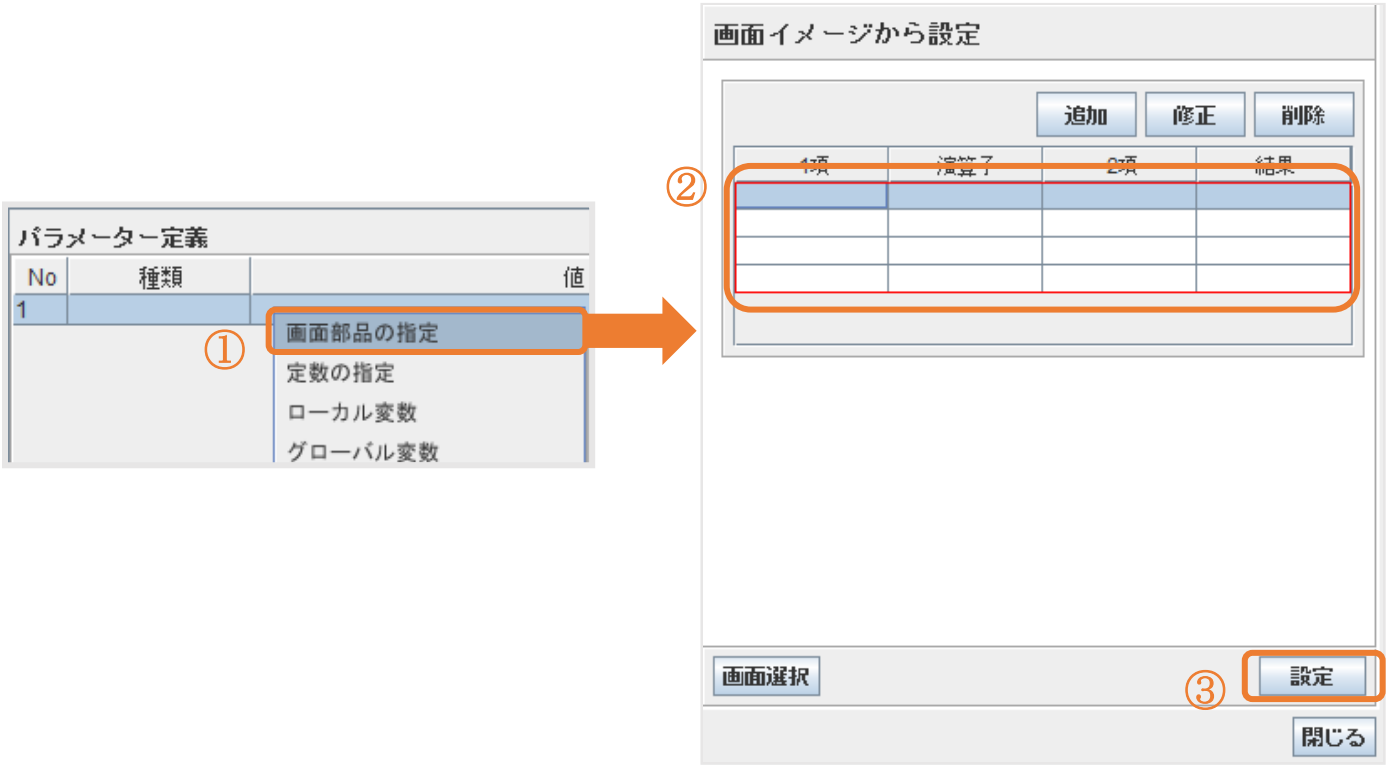
手順４ ：テーブルの全ての行を取得する

ローカル変数「すべての行」は未作成なので作成してください。

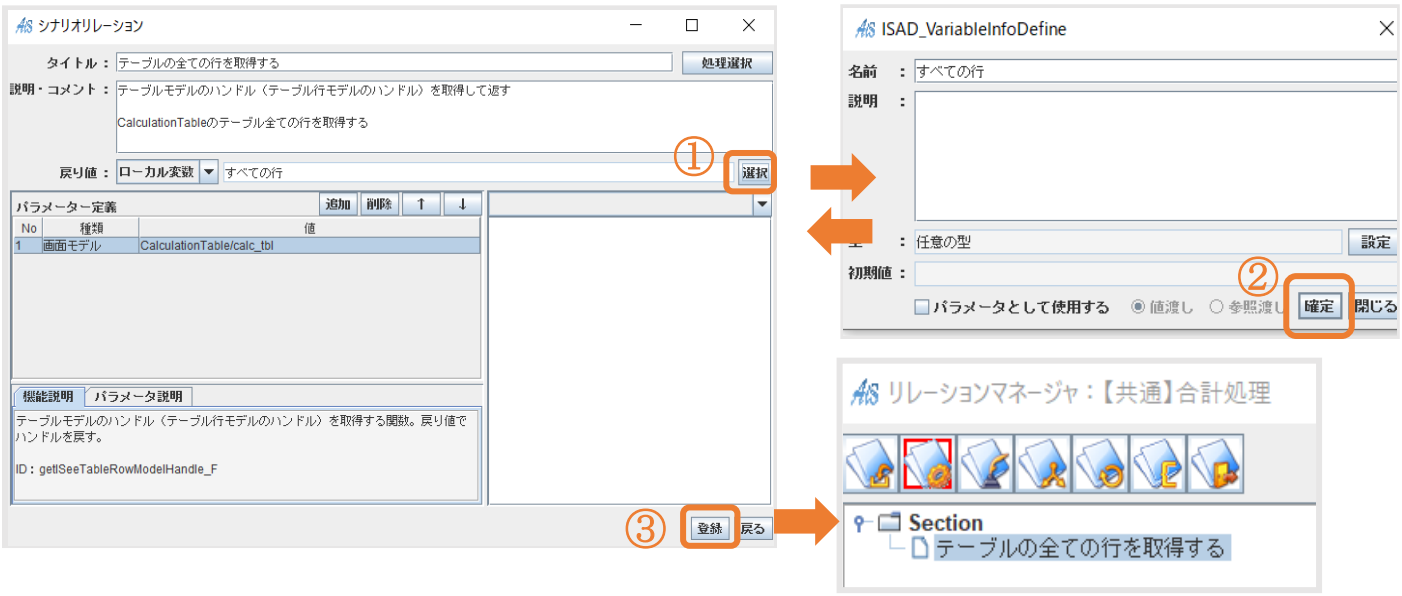
【図 79】



【図 80】


パラメーター定義
No
種類
値
1
①
画面部品の指定
定数の指定
ローカル変数
グローバル変数
画面イメージから設定
追加
修正
削除
②
画面選択
③
設定
閉じる

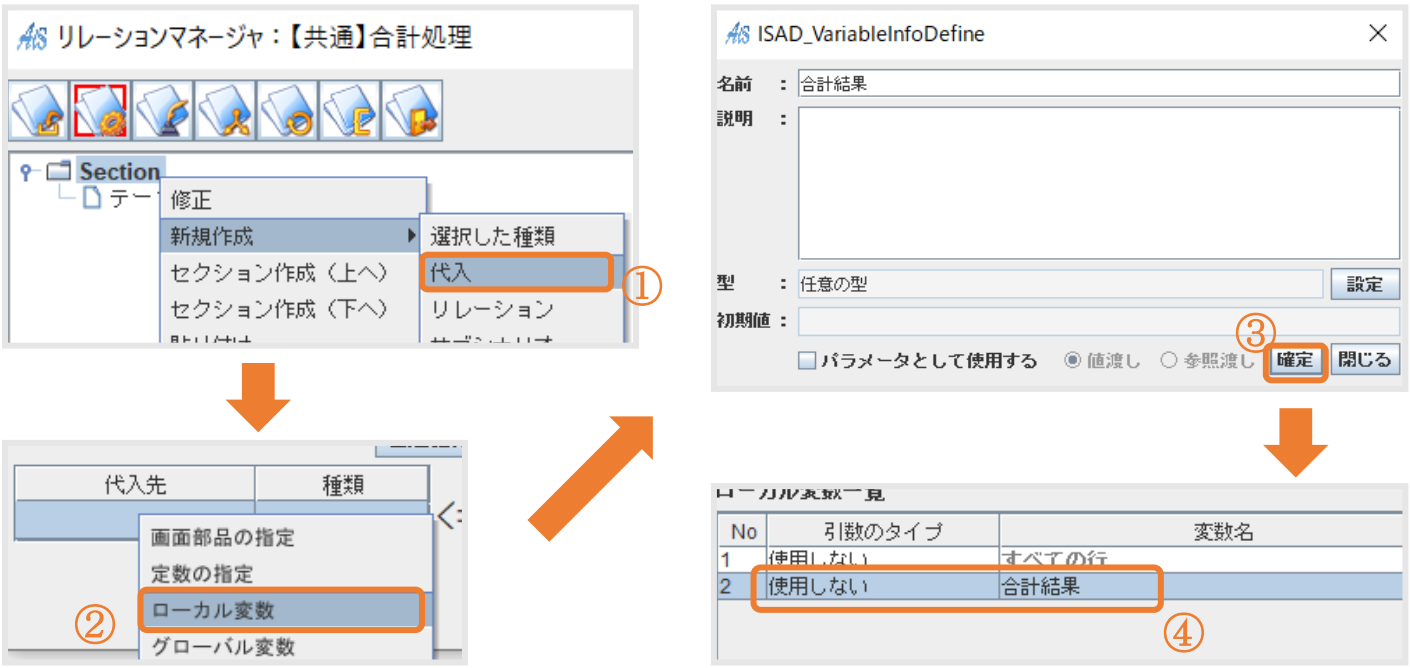
【図 81】


シナリオリレーション
タイトル：
テーブルの全ての行を取得する
処理選択
説明・コメント：
テーブルモデルのハンドル（テーブル行モデルのハンドル）を取得して返す
CalculationTableのテーブル全ての行を取得する
①
戻り値：
ローカル変数
すべての行
選択
パラメーター定義
追加
削除
No
種類
値
1
画面モデル
CalculationTable/calc_tbl
機能説明
パラメータ説明
テーブルモデルのハンドル（テーブル行モデルのハンドル）を取得する関数。戻り値でハンドルを戻す。
ID：getISeeTableRowModelHandle_F
③
登録
戻る
ISAD_VariableInfoDefine
名前
：
すべての行
説明
：
：
任意の型
設定
初期値
：
②
パラメータとして使用する
値渡し
参照渡し
確定
閉じる
リレーションマネージャ：【共通】合計処理
Section
テーブルの全ての行を取得する

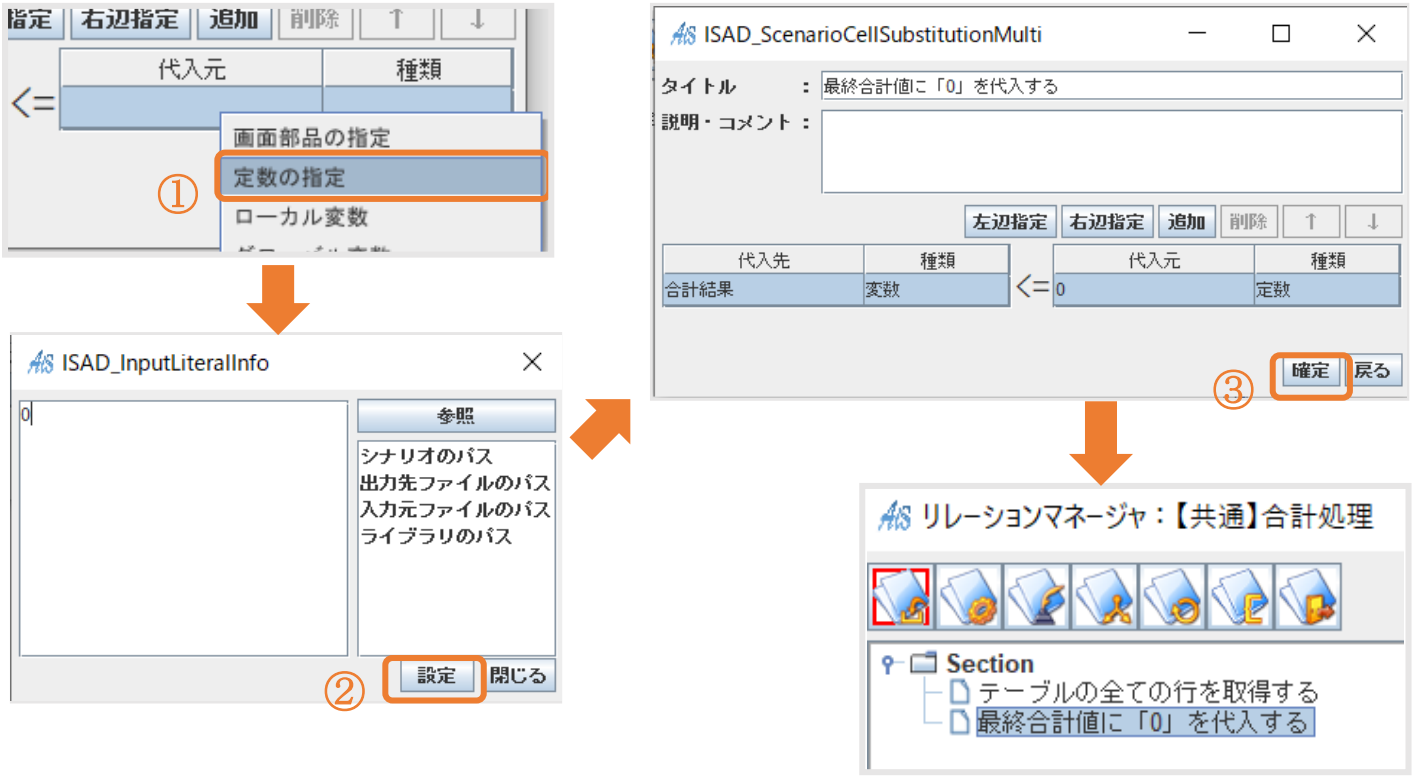
手順５：最終合計値に「0」を代入する

ローカル変数「合計結果」は未作成なので作成してください。

【図 82】



【図 83】

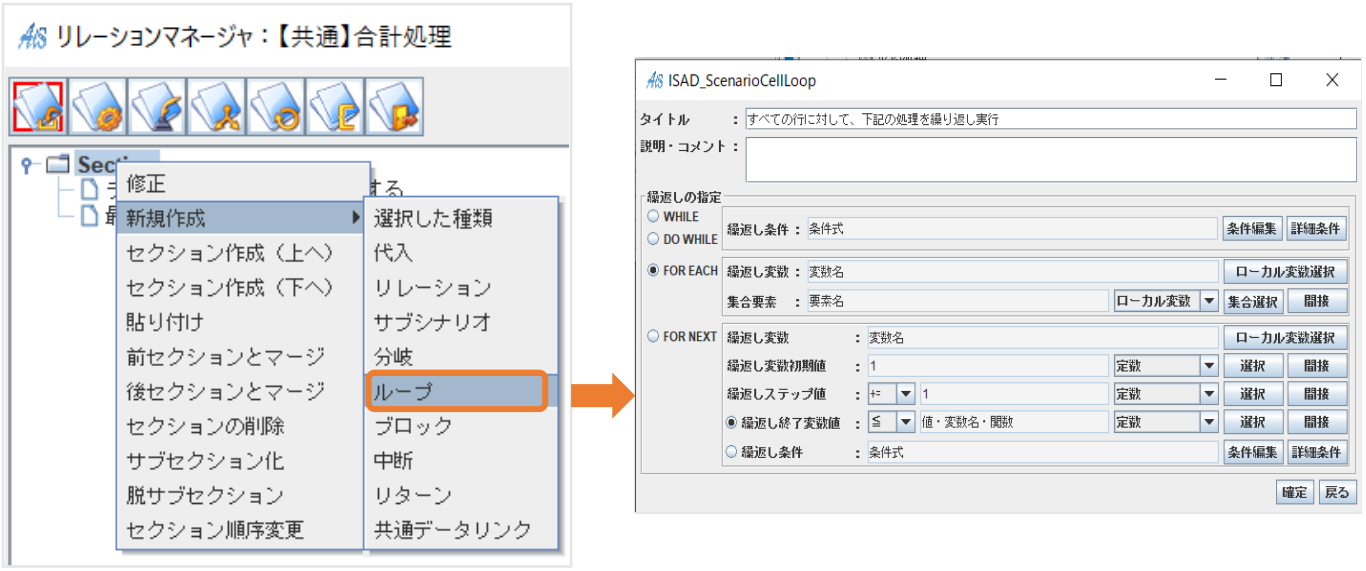

右辺指定
追加
削除
代入元
種類
<=
画面部品の指定
定数の指定
ローカル変数
①
ISAD_InputLiteralInfo
0
参照
シナリオのパス
出力先ファイルのパス
入力元ファイルのパス
ライブラリのパス
設定
閉じる
②
ISAD_ScenarioCellSubstitutionMulti
タイトル ： 最終合計値に「0」を代入する
説明・コメント ：
左辺指定
右辺指定
追加
削除
代入先
種類
合計結果
変数
<=
代入元
種類
0
定数
確定
戻る
③
リレーションマネージャ：【共通】合計処理
Section
テーブルの全ての行を取得する
最終合計値に「0」を代入する

手順６：すべての行に対して、下記の処理を繰り返し実行

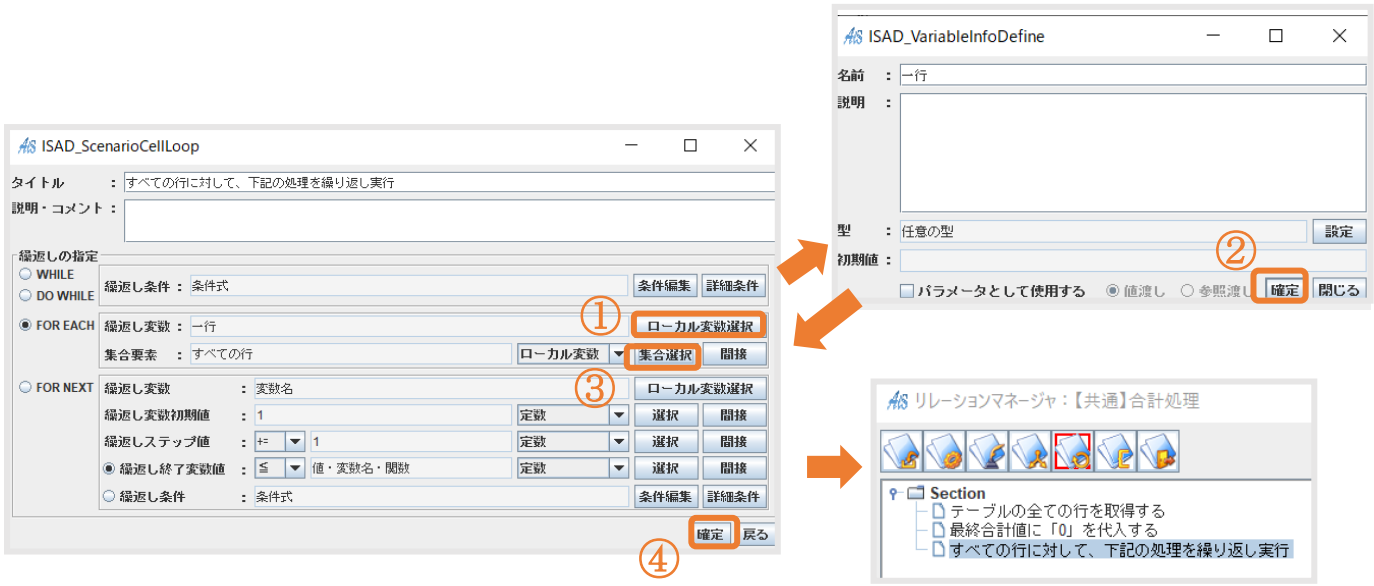
ローカル変数「一行」は未作成なので作成してください。

※ループ処理についての考え方(フローチャート図)は[こちらを参照]

【図 84】



【図 85】


ISAD_ScenarioCellLoop
タイトル ： すべての行に対して、下記の処理を繰り返し実行
説明・コメント ：
繰返しの指定
WHILE
DO WHILE
繰返し条件 ： 条件式
条件編集
詳細条件
FOR EACH
繰返し変数 ： 一行
集合要素 ： すべての行
ローカル変数
集合選択
間接
ローカル変数選択
①
FOR NEXT
繰返し変数 ： 変数名
繰返し変数初期値 ： 1
繰返しステップ値 ： += 1
繰返し終了変数値 ： ≦ 値・変数名・関数
繰返し条件 ： 条件式
定数
選択
③
確定
戻る
④
ISAD_VariableInfoDefine
名前 ： 一行
説明 ：
型 ： 任意の型
設定
初期値 ：
パラメータとして使用する
値渡し
参照渡し
確定
閉じる
②
リレーションマネージャ：【共通】合計処理
Section
テーブルの全ての行を取得する
最終合計値に「0」を代入する
すべての行に対して、下記の処理を繰り返し実行

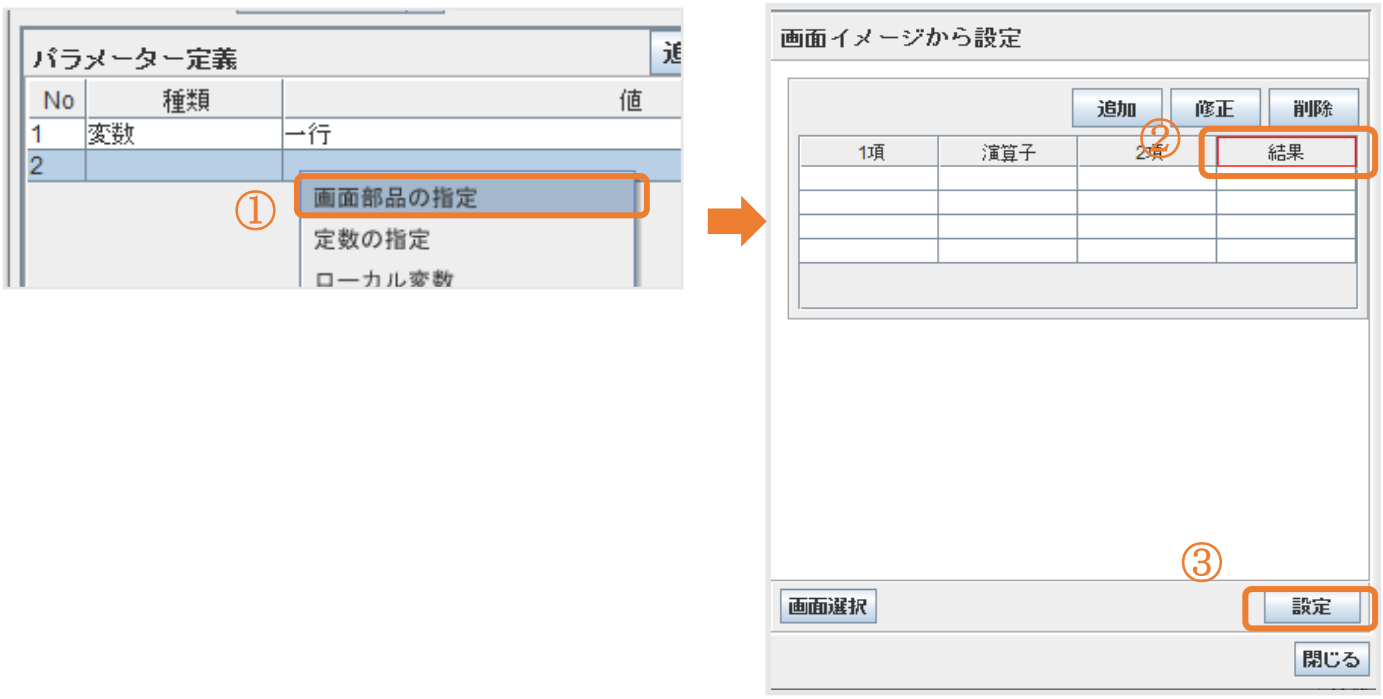
手順７：行の「結果」カラムから値を取り出す

これは「すべての行に対して、下記の処理を繰り返し実行」の直下に作成したいので、「すべての行に対して、下記の処理を繰り返し実行」を右クリックで「新規作成」→「リレーション」→「データベースレコードのカラムから内容を取得して返す」を選択

ローカル変数「結果カラムの値」は未作成なので作成してください。

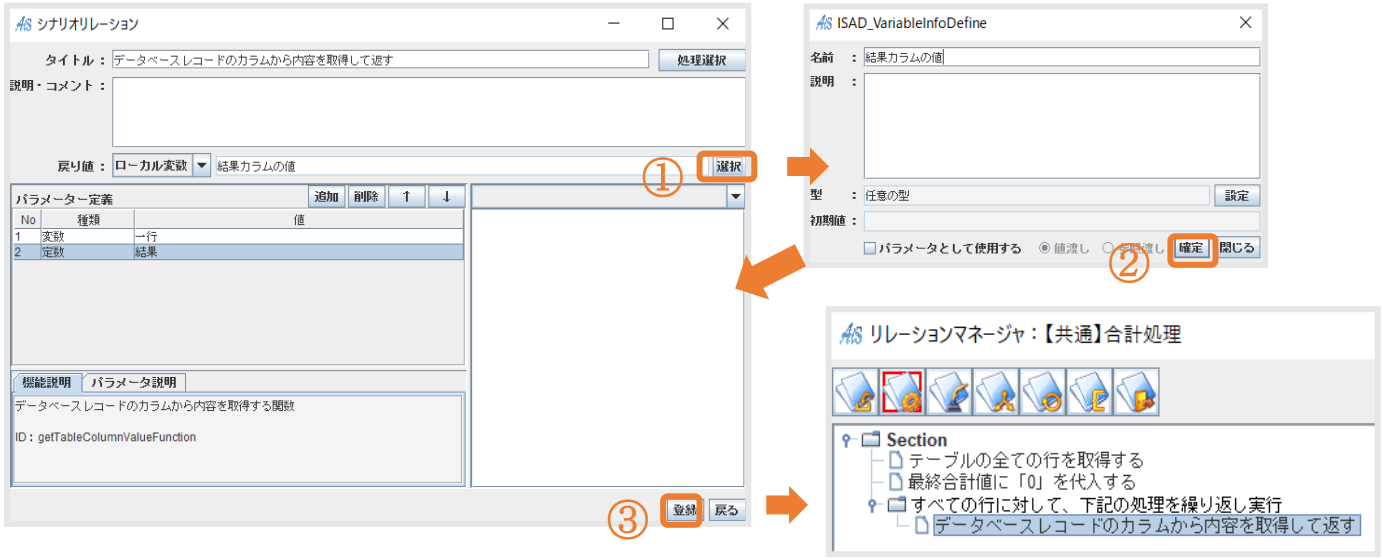
【図 86】

【図 87】



パラメータ定義の No.1 はローカル変数「一行」を選択してください。

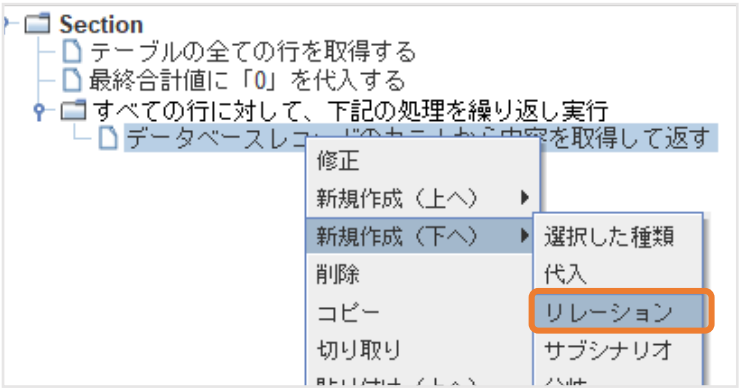
【図 88】

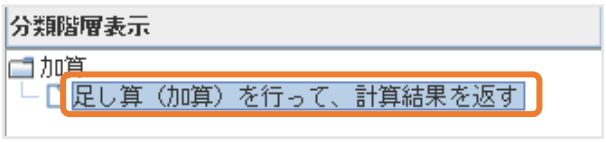


手順８ ：取り出した値を合計値に足す

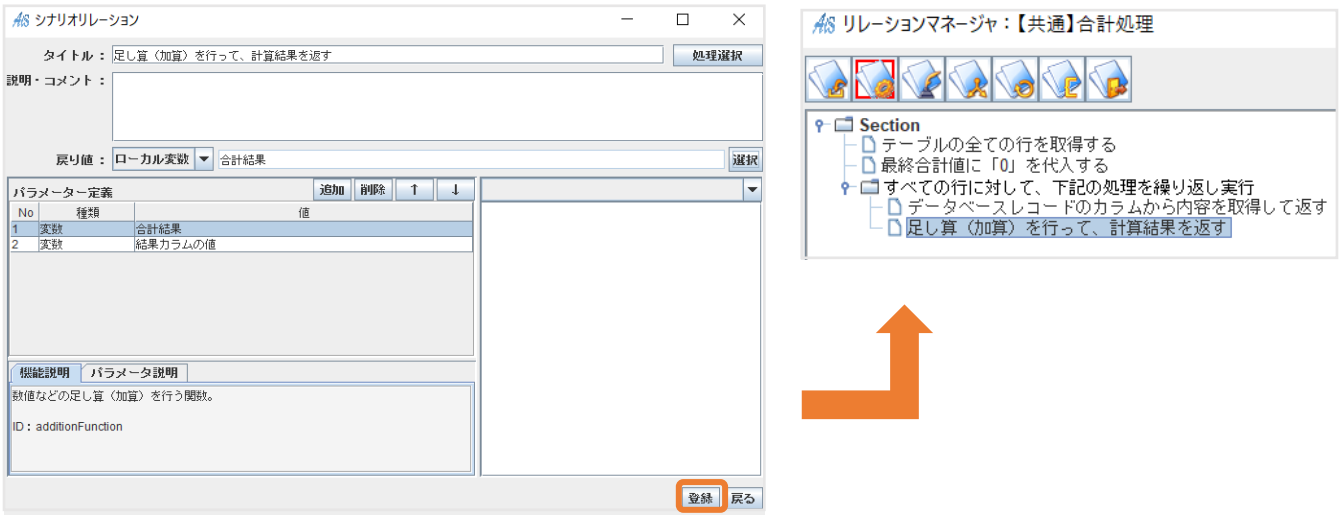
これも「すべての行に対して、下記の処理を繰り返し実行」の直下に作成したいので、「すべての行に対して、下記の処理を繰り返し実行」を右クリックで「新規作成」→「リレーション」→「足し算(加算)を行って、計算結果を返す」を選択

【図 89】



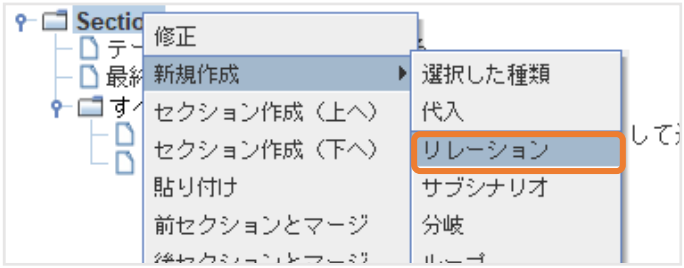


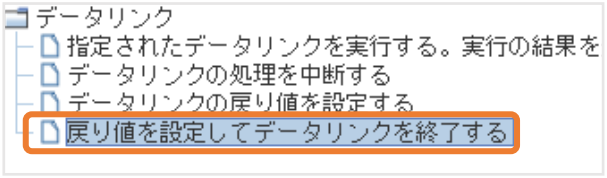
【図 90】


シナリオリレーション
タイトル：
足し算（加算）を行って、計算結果を返す
処理選択
説明・コメント：
戻り値：
ローカル変数
合計結果
選択
パラメーター定義
追加
削除
No
種類
値
1
変数
合計結果
2
変数
結果カラムの値
機能説明
パラメータ説明
数値などの足し算（加算）を行う関数。
ID：additionFunction
登録
戻る
リレーションマネージャ：【共通】合計処理
Section
テーブルの全ての行を取得する
最終合計値に「0」を代入する
すべての行に対して、下記の処理を繰り返し実行
データベースレコードのカラムから内容を取得して返す
足し算（加算）を行って、計算結果を返す

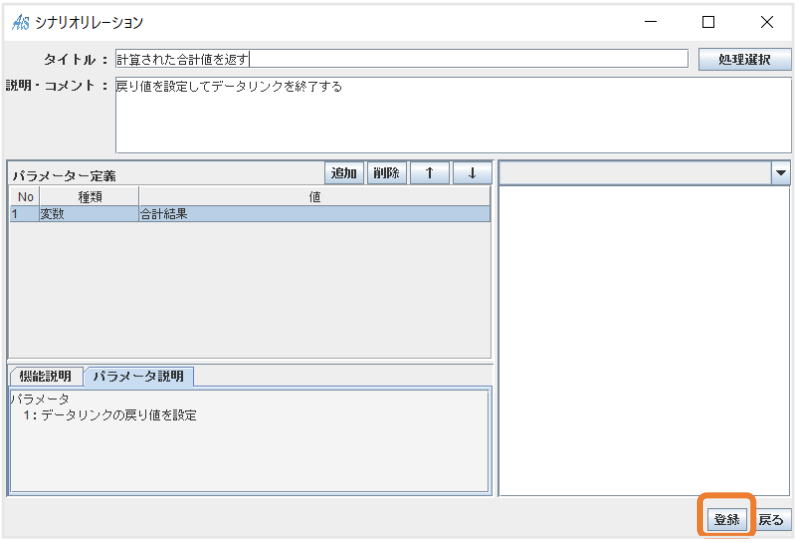
手順９：計算された合計値を返す

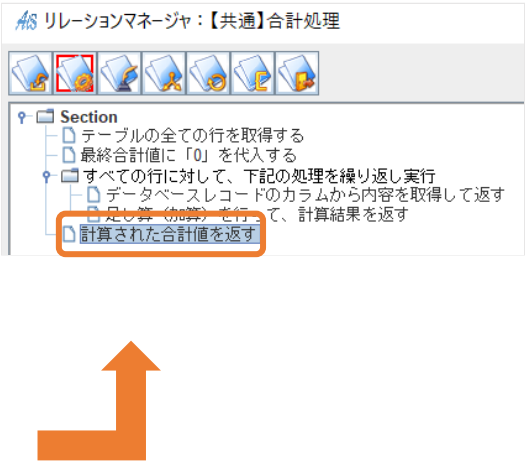
【図 91】





【図 92】





ここで F5 キーを押し、**テスト実行**を行いましょう。

問題がなければ「CalculationTable」内で右クリックをすると「メニュー」が表示され「合計」を押下すると「結果」カラムの合計値がメッセージとして表示されます。

5.3.9　合計処理

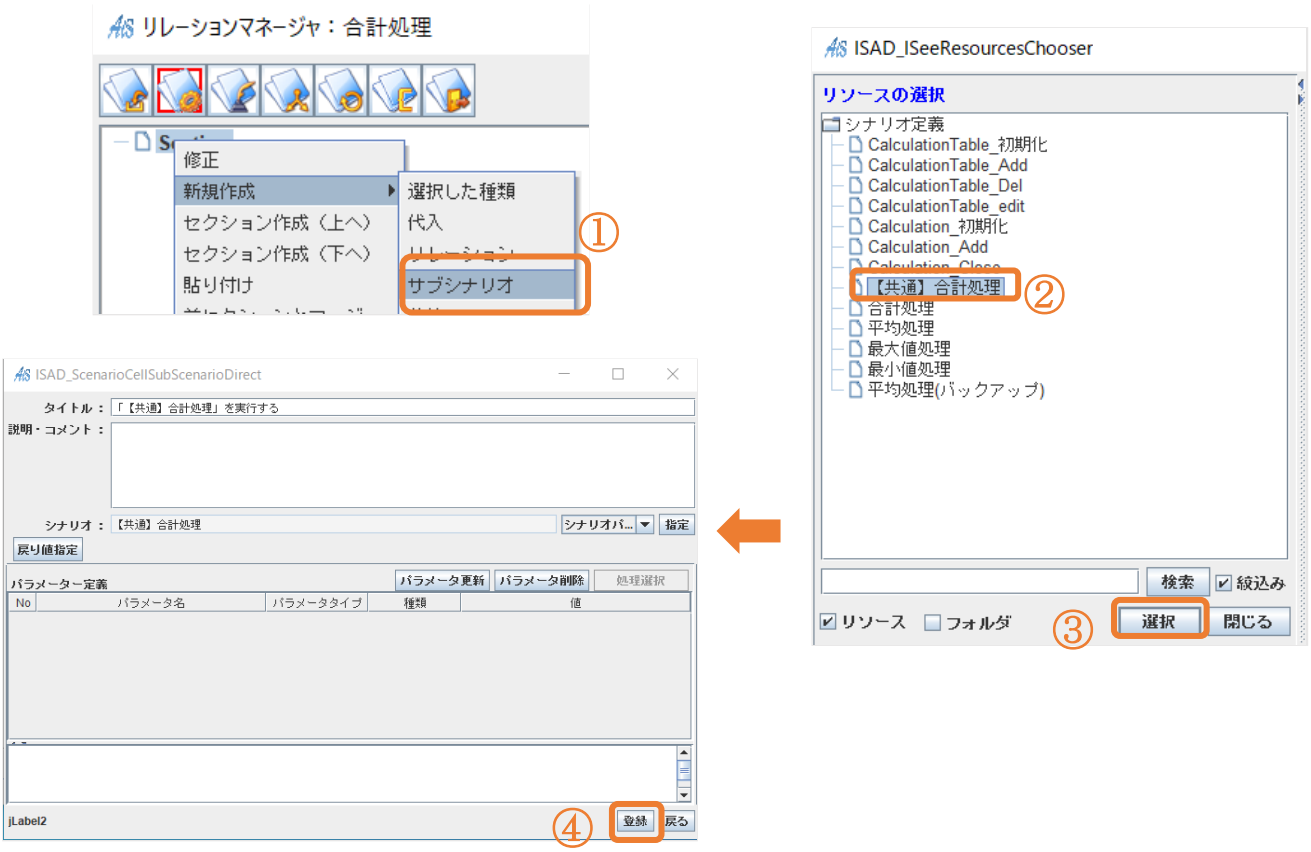
手順１：「【共通】合計処理」を実行する

手順２：合計値の計算結果を表示する

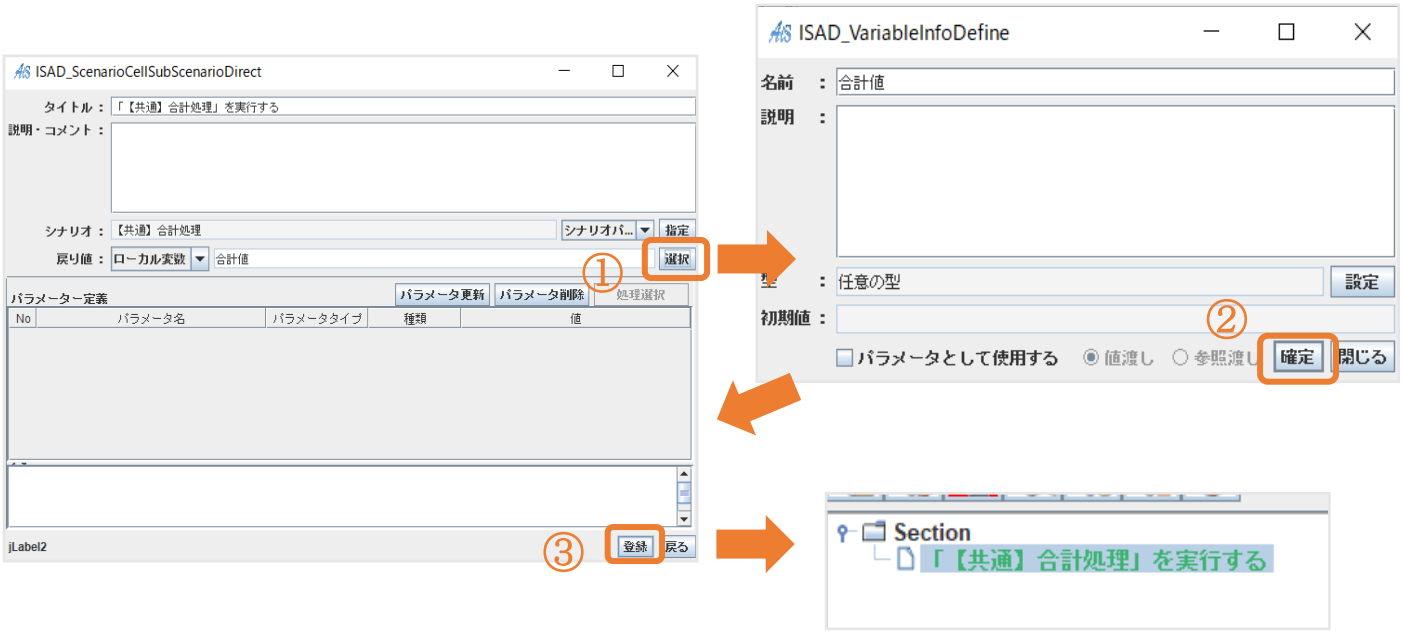
手順１：「【共通】合計処理」を実行する

ローカル変数「合計値」は未作成なので作成してください。

【図 93】

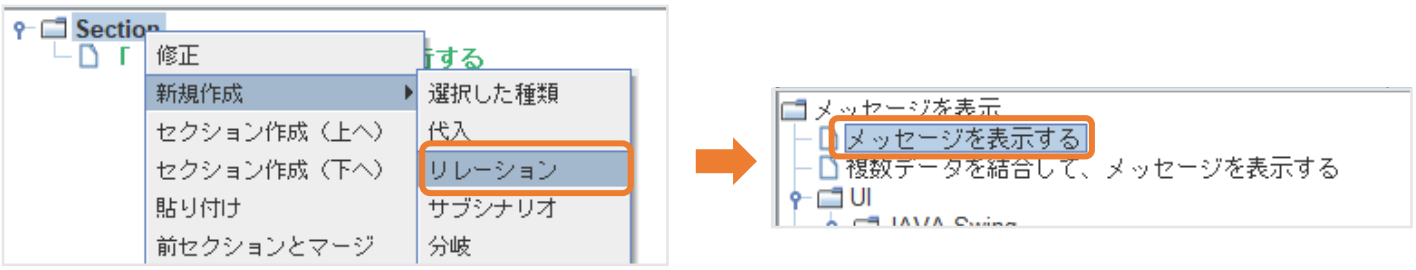


【図 94】

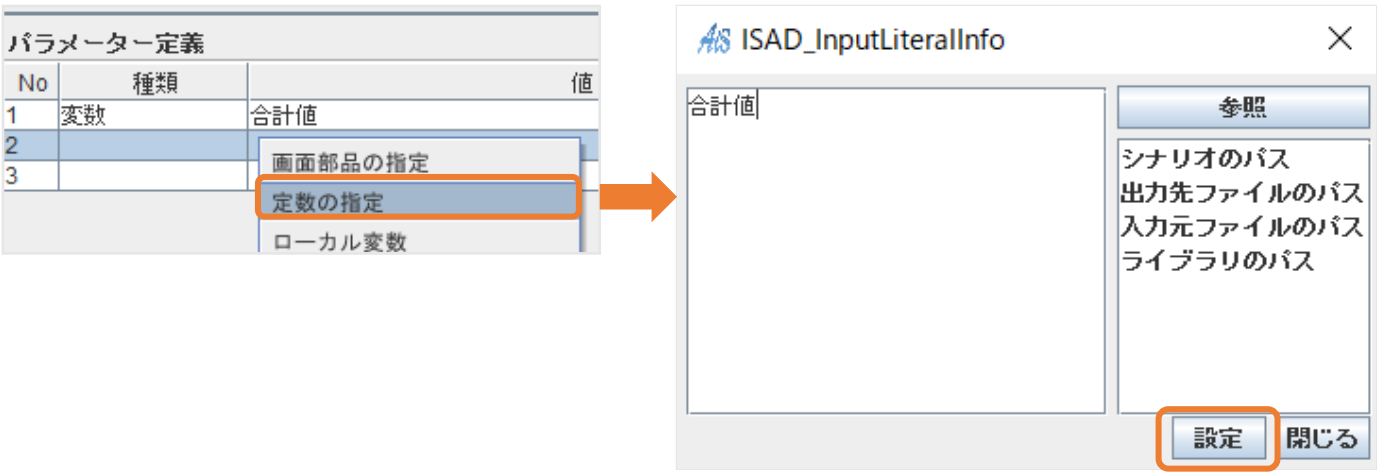

ISAD_ScenarioCellSubScenarioDirect
タイトル：
「【共通】合計処理」を実行する
説明・コメント：
シナリオ：
【共通】合計処理
シナリオパ...
指定
戻り値：
ローカル変数
合計値
選択
①
パラメーター定義
パラメータ更新
パラメータ削除
処理選択
No
パラメータ名
パラメータタイプ
種類
値
jLabel2
③
登録
戻る
ISAD_VariableInfoDefine
名前
：
合計値
説明
：
：
任意の型
設定
初期値
：
②
パラメータとして使用する
値渡し
参照渡し
確定
閉じる
Section
「【共通】合計処理」を実行する

手順２：合計値の計算結果を表示する

【図 95】



【図 96】



【図 97】

5.3.10 平均処理

手順１：「【共通】合計処理を実行する」

手順２：テーブルモデルの行数を取得して返す

手順３：合計÷行数

手順４：合計値の計算結果を表示する

手順５：テスト実行

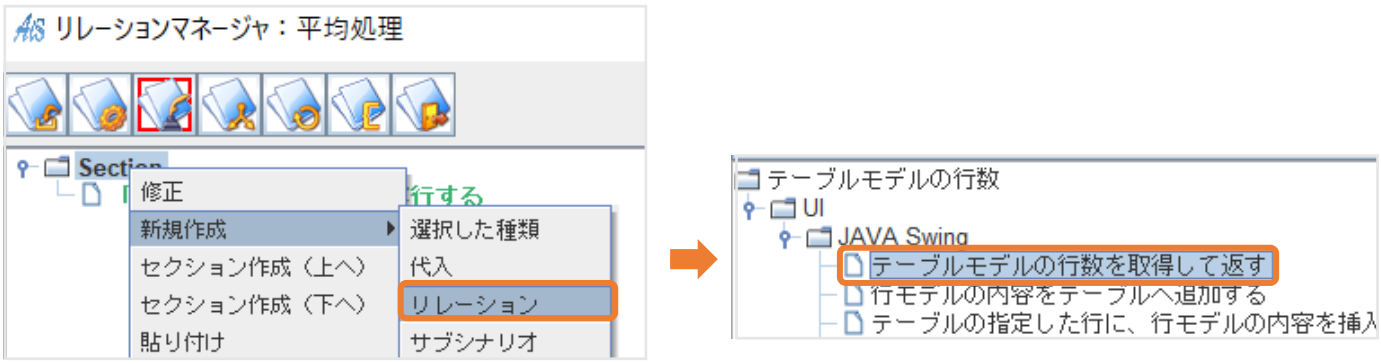
手順１：「【共通】合計処理を実行する」

合計処理の手順１と同じです

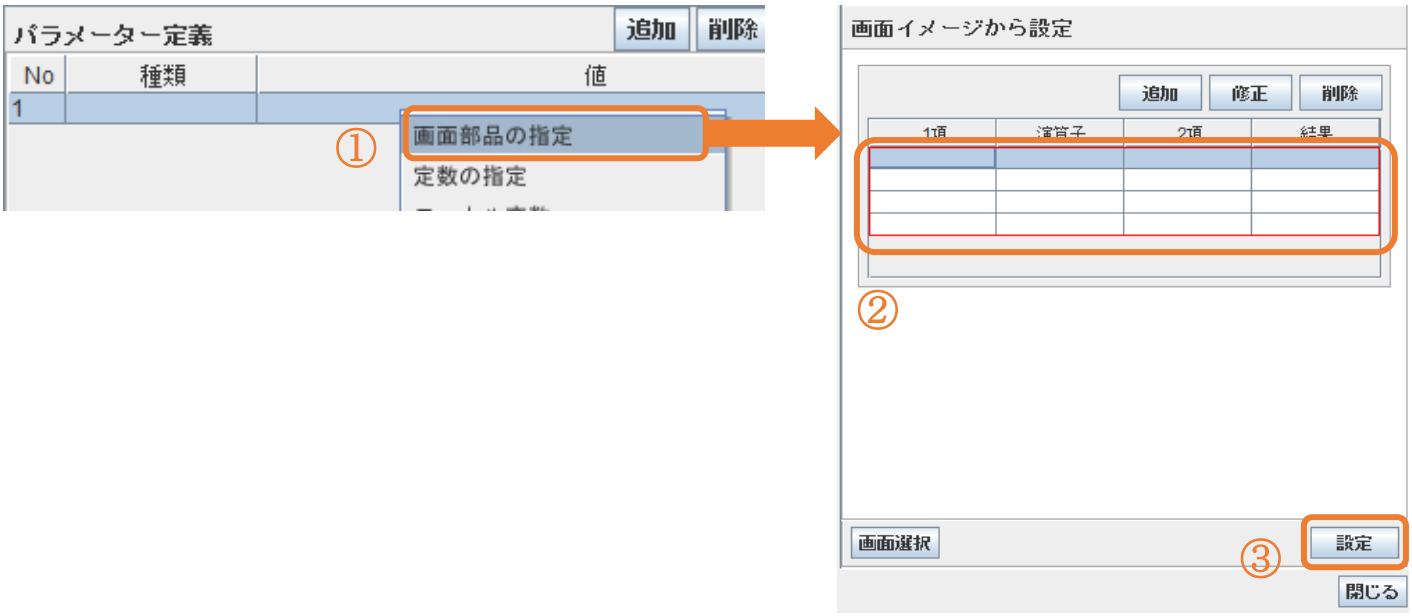
手順２：テーブルモデルの行数を取得して返す

ローカル変数「行数」は未作成なので作成してください。

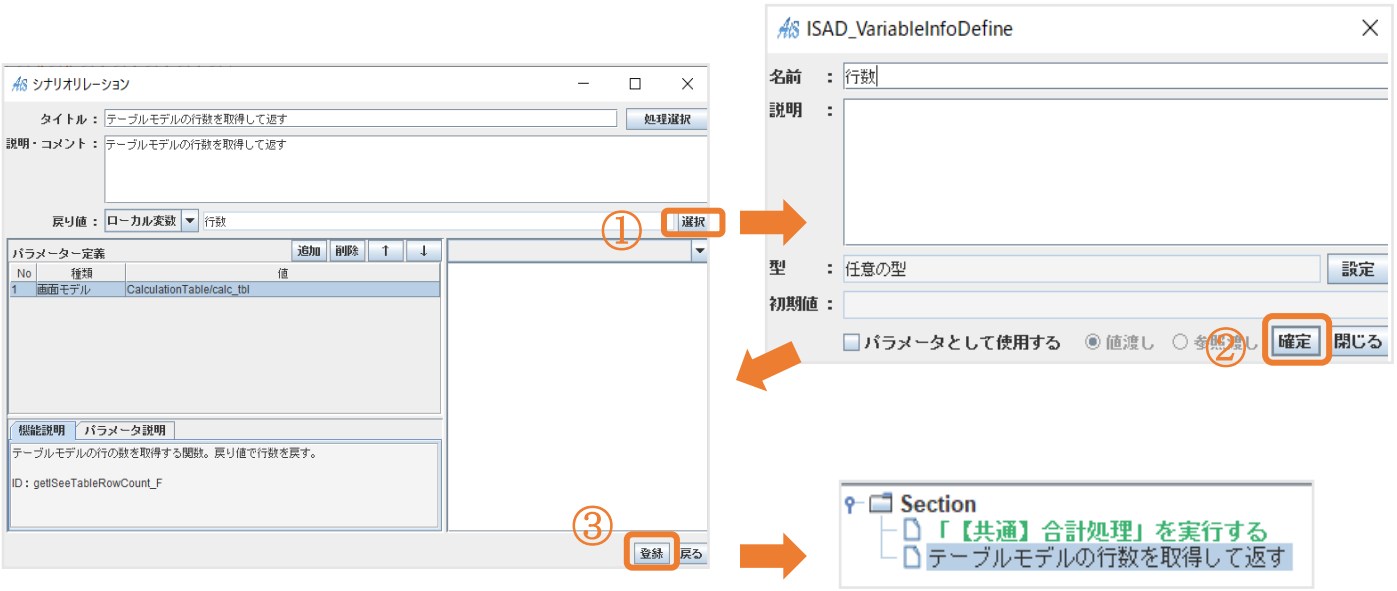
【図 98】



【図 99】

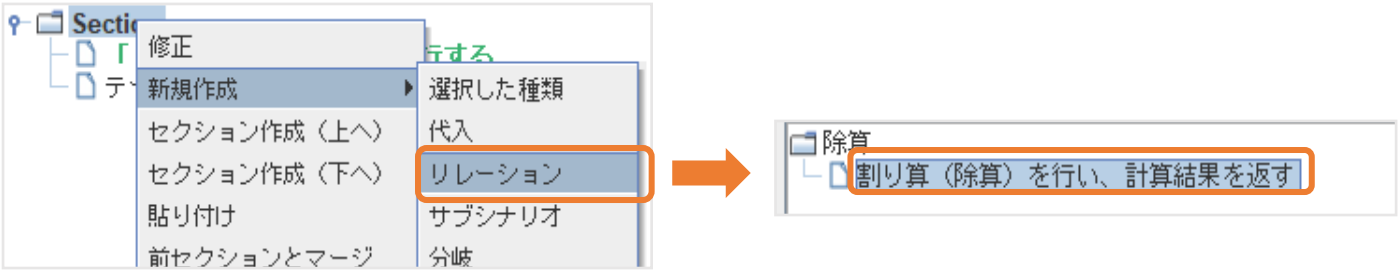

パラメーター定義
追加
削除
No
種類
値
1
①
画面部品の指定
定数の指定
画面イメージから設定
追加
修正
削除
1項
演算子
2項
結果
②
画面選択
③
設定
閉じる

【図 100】

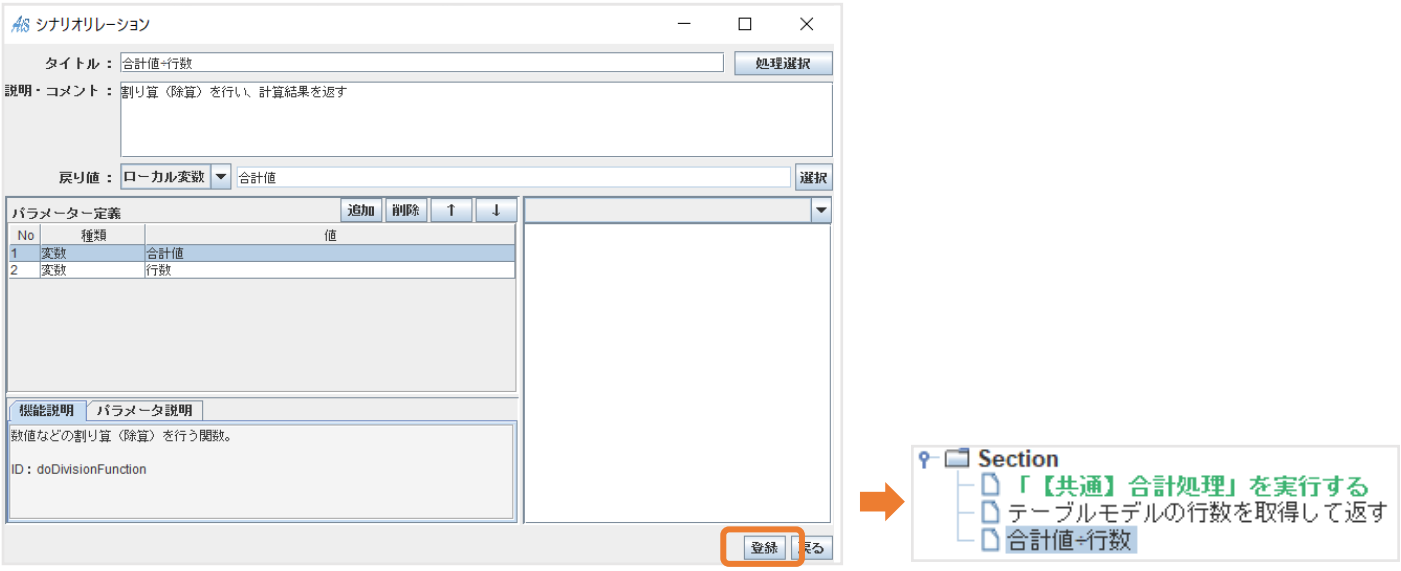

シナリオリレーション
タイトル：
テーブルモデルの行数を取得して返す
処理選択
説明・コメント：
テーブルモデルの行数を取得して返す
戻り値：
ローカル変数
行数
①
選択
パラメーター定義
追加
削除
No
種類
値
1
画面モデル
CalculationTable/calc_tbl
機能説明
パラメータ説明
テーブルモデルの行の数を取得する関数。戻り値で行数を戻す。
ID：getISeeTableRowCount_F
③
登録
戻る
ISAD_VariableInfoDefine
名前
：
行数
説明
：
型
：
任意の型
設定
初期値
：
パラメータとして使用する
値渡し
参照渡し
②
確定
閉じる
Section
「【共通】合計処理」を実行する
テーブルモデルの行数を取得して返す

手順 3：合計÷行数

【図 101】

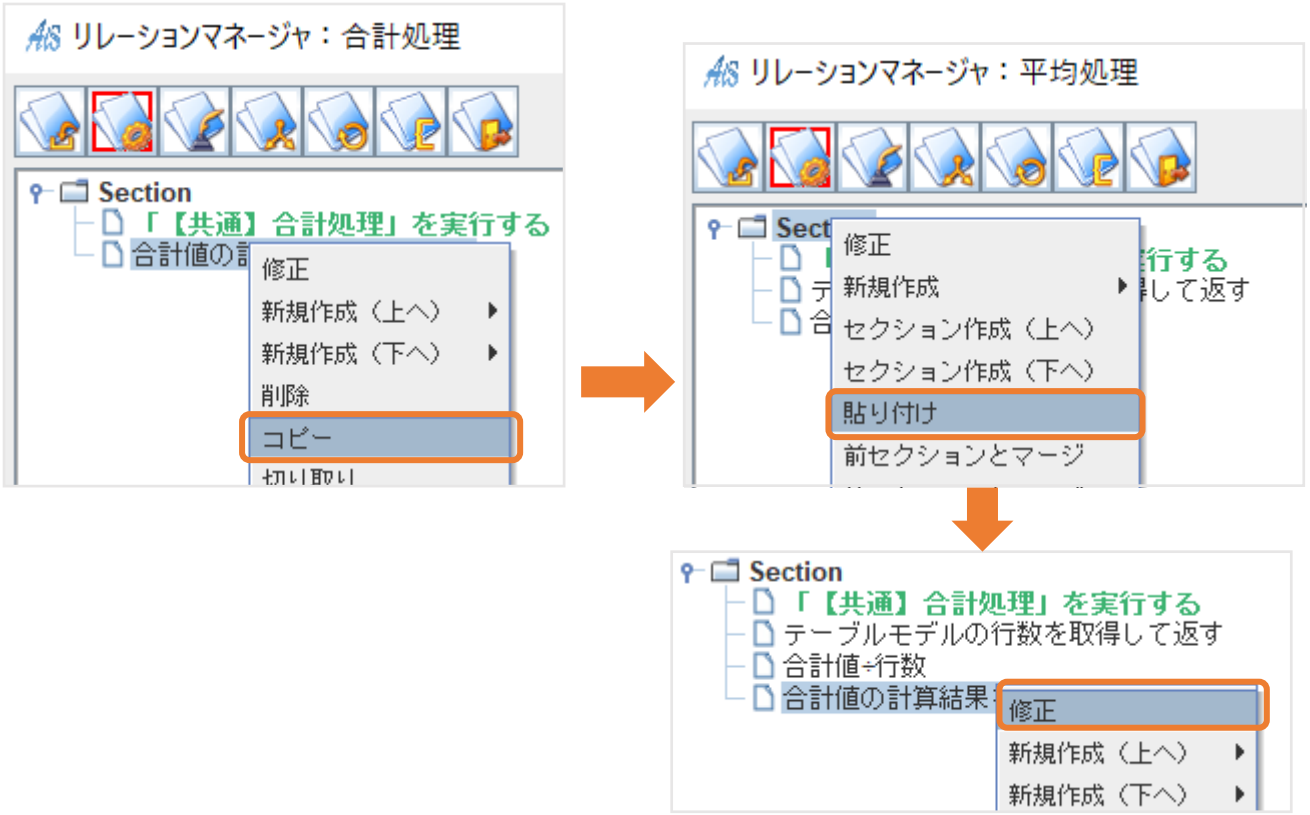


【図 102】

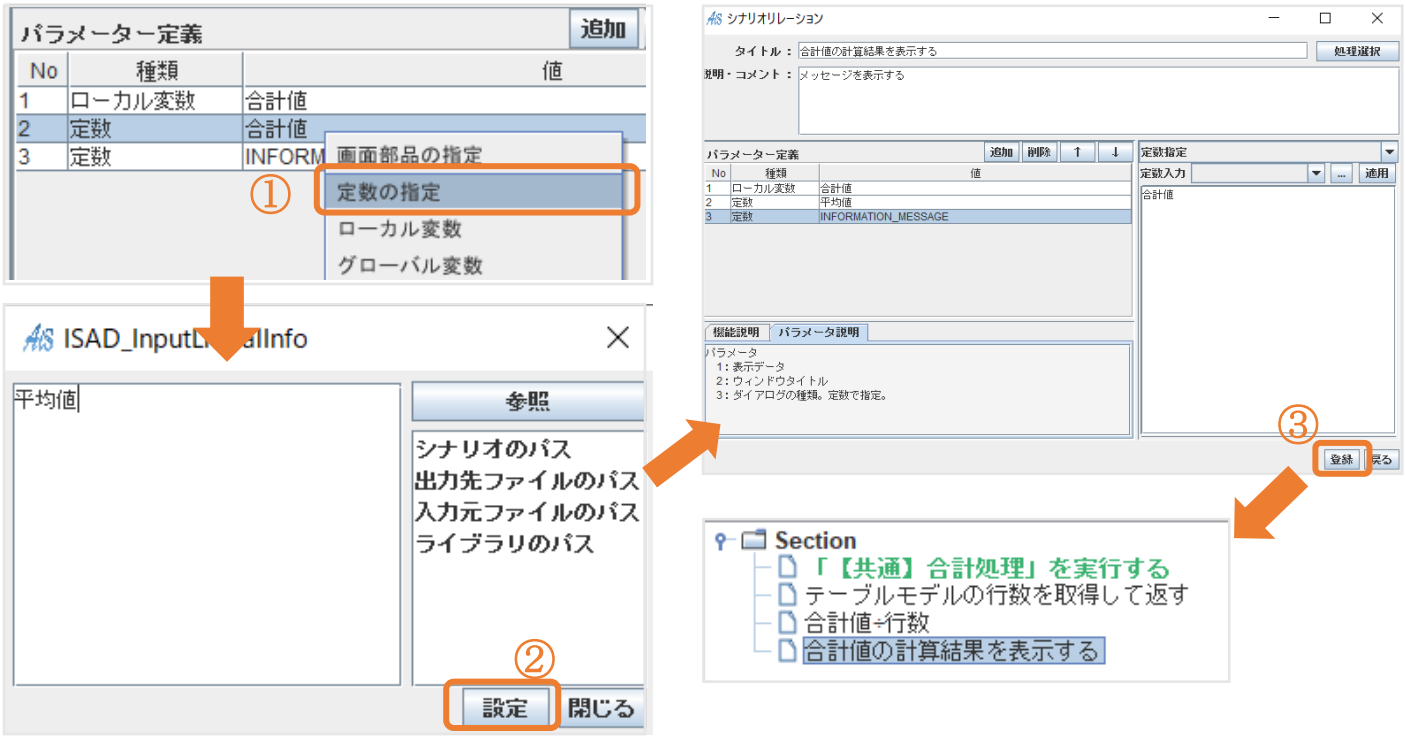


手順４：合計値の計算結果を表示する

【図 103】



【図 104】



ここで F5 キーを押し、**テスト実行**を行いましょう。

問題がなければ「CalculationTable」内で右クリックをすると「メニュー」が表示され「平均」を押下すると「結果」カラムの平均値がメッセージとして表示されます。(図)


CalculationTable
追加
修正
削除
1項
演算子
2項
結果
1
+
2
3
3
+
2
5
2
-
1
1
平均値の計算結果
3
OK

5.3.11 最大値処理

「CulculationTable」で「メニュー」画面から「最大値」を押下すると「結果」カラムの最大数値が表示される実装

手順１： CalculationTable のテーブル全ての行を取得する

手順２：「最大値」に比較する固定の値を代入する

手順３：繰り返し処理を行う

手順４：「一行」から「結果」の値を取得する

手順５：分岐処理を行う

手順６：「結果の値」を「最大値」へ代入

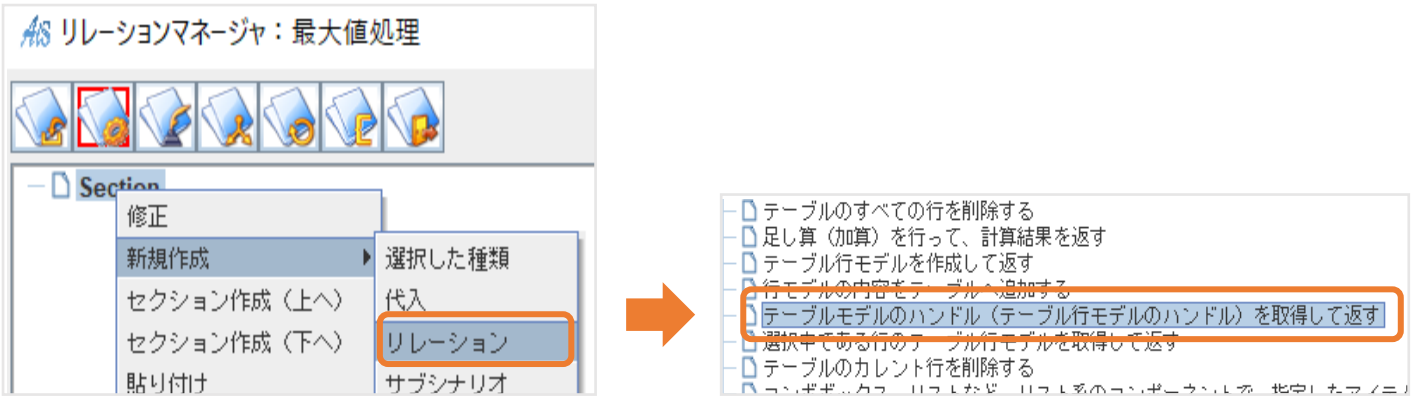
手順７：「最大値」を表示する

手順８：テスト実行

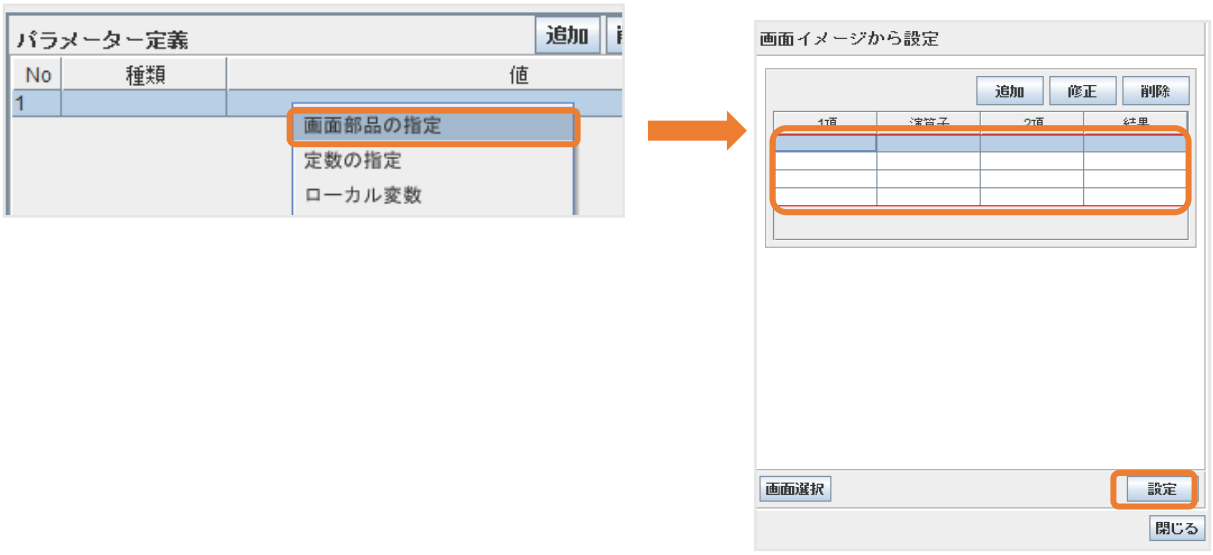
手順１：CalculationTable のテーブル全ての行を取得する

ローカル変数「すべての行」は未作成なので作成してください。

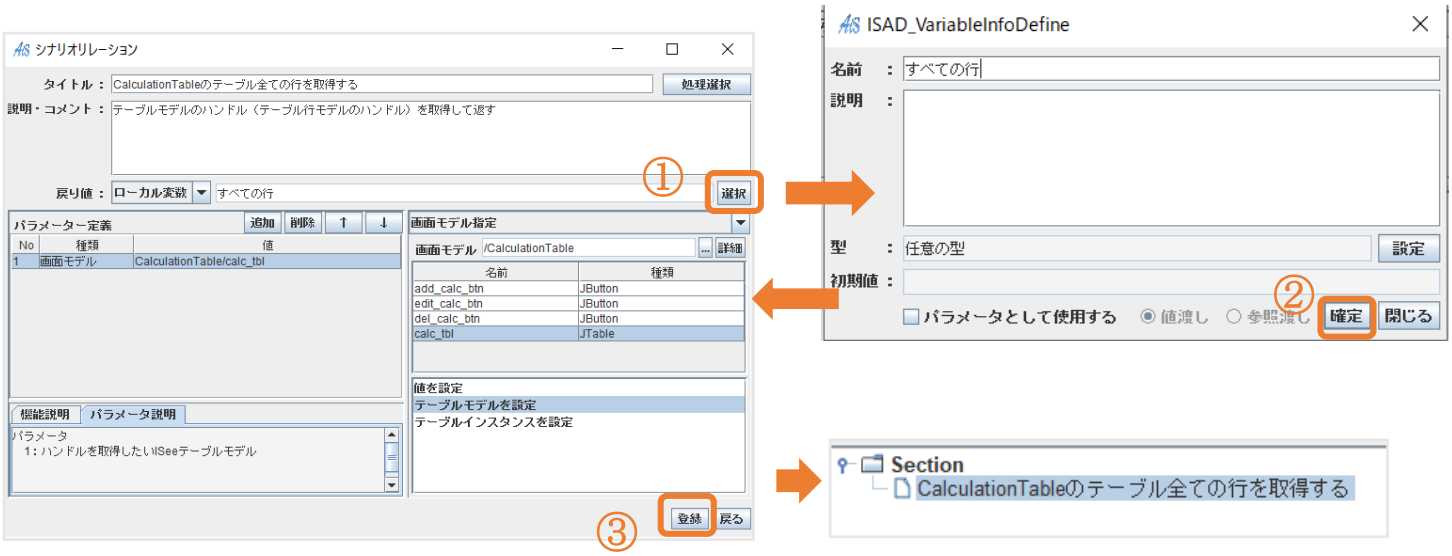
【図 105】



【図 106】


パラメーター定義
追加
No
種類
値
1
画面部品の指定
定数の指定
ローカル変数
画面イメージから設定
追加
修正
削除
画面選択
設定
閉じる

【図 107】

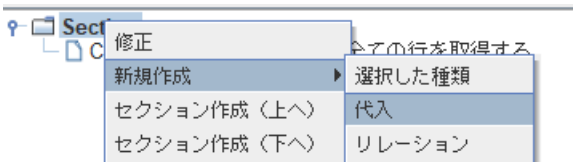

シナリオリレーション
タイトル：
CalculationTableのテーブル全ての行を取得する
処理選択
説明・コメント：
テーブルモデルのハンドル（テーブル行モデルのハンドル）を取得して返す
戻り値：
ローカル変数
すべての行
選択
①
パラメーター定義
追加
削除
No
種類
値
1
画面モデル
CalculationTable/calc_tbl
画面モデル指定
画面モデル
/CalculationTable
詳細
名前
種類
add_calc_btn
JButton
edit_calc_btn
JButton
del_calc_btn
JButton
calc_tbl
JTable
値を設定
テーブルモデルを設定
テーブルインスタンスを設定
機能説明
パラメータ説明
パラメータ
1：ハンドルを取得したいISeeテーブルモデル
③
登録
戻る
ISAD_VariableInfoDefine
名前
：
すべての行
説明
：
型
：
任意の型
設定
初期値
：
②
パラメータとして使用する
値渡し
参照渡し
確定
閉じる
Section
CalculationTableのテーブル全ての行を取得する

手順２：「最大値」に比較する固定の値を代入する

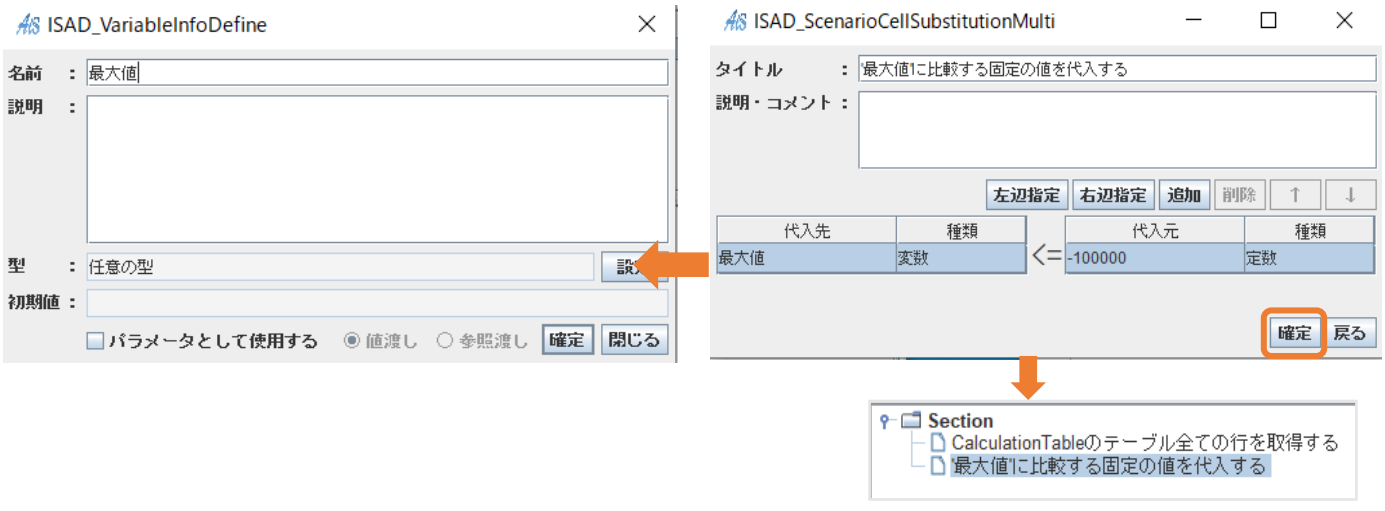
ローカル変数「最大値」は未作成なので作成してください。

※ここでは、テーブルに入る値と比較して充分に小さい任意の値（-100000）を設定しています。
AiSee で設定できる最小値は「-9223372036854775808」

【図 108】



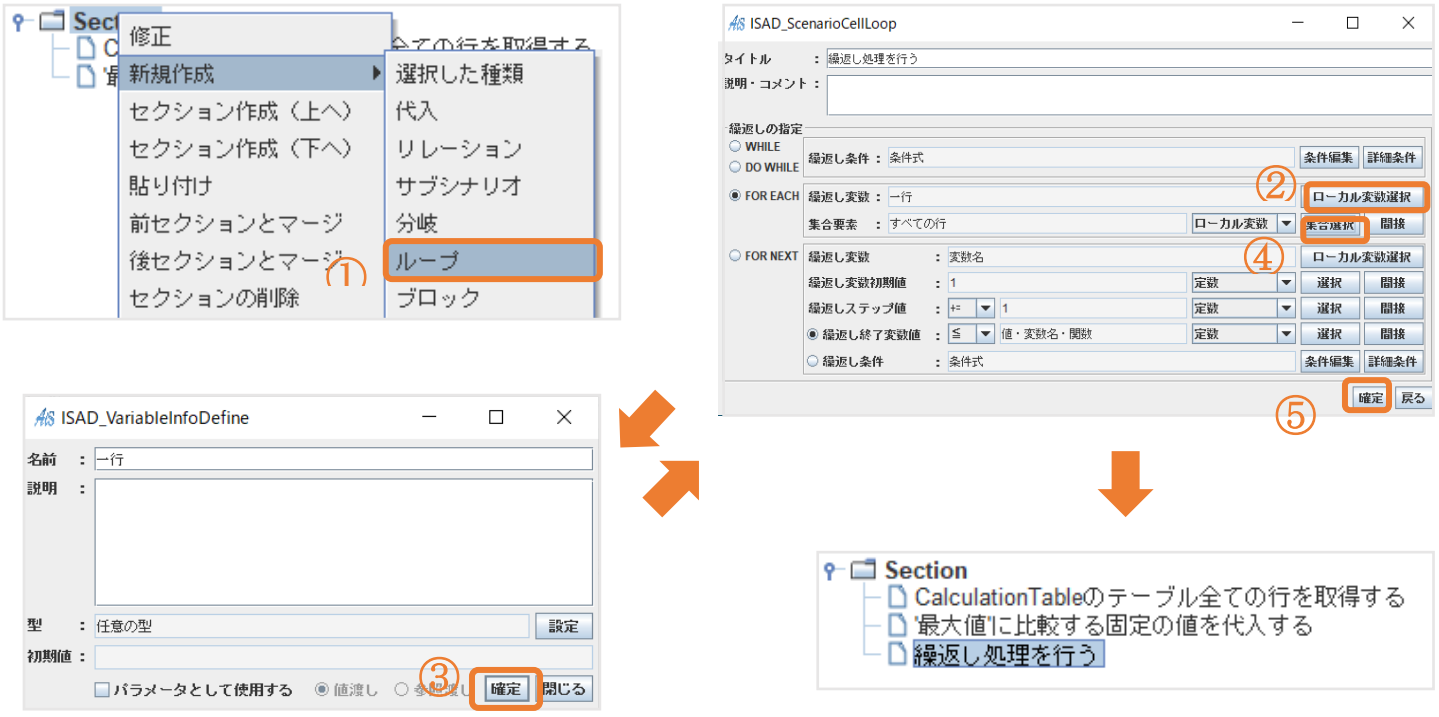
【図 109】



手順３：繰り返し処理を行う

ローカル変数「一行」は未作成なので作成してください。

【図 110】



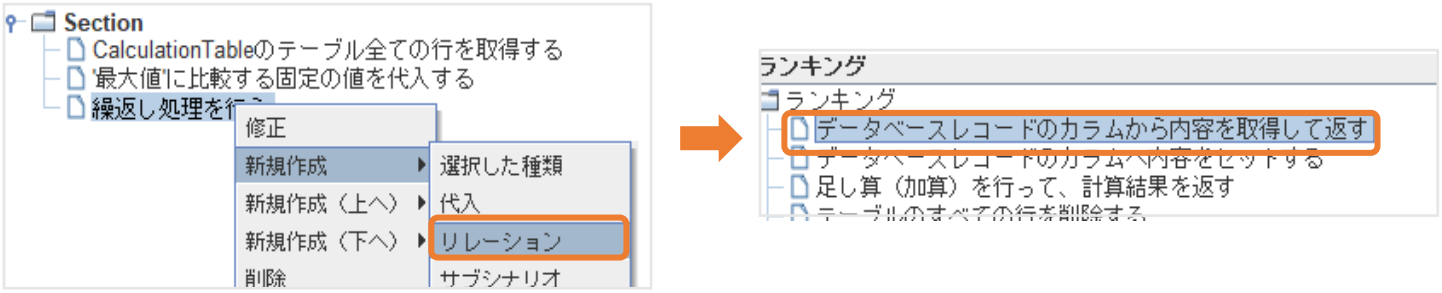
手順４：「一行」から「結果」の値を取得する

これは「繰り返し処理を行う」の直下に作成したいので、「繰り返し処理を行う」を右クリックで「新規作成」→「リレーション」→

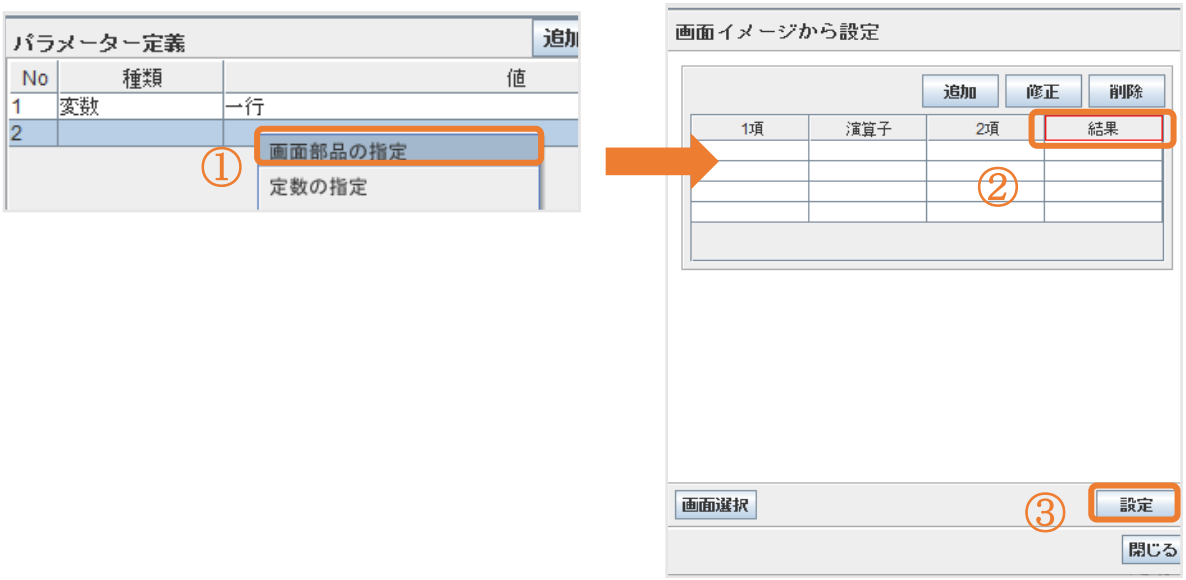
「データベースレコードのカラムから内容を取得して返す」を選択

ローカル変数「結果の値」は未作成なので作成してください。

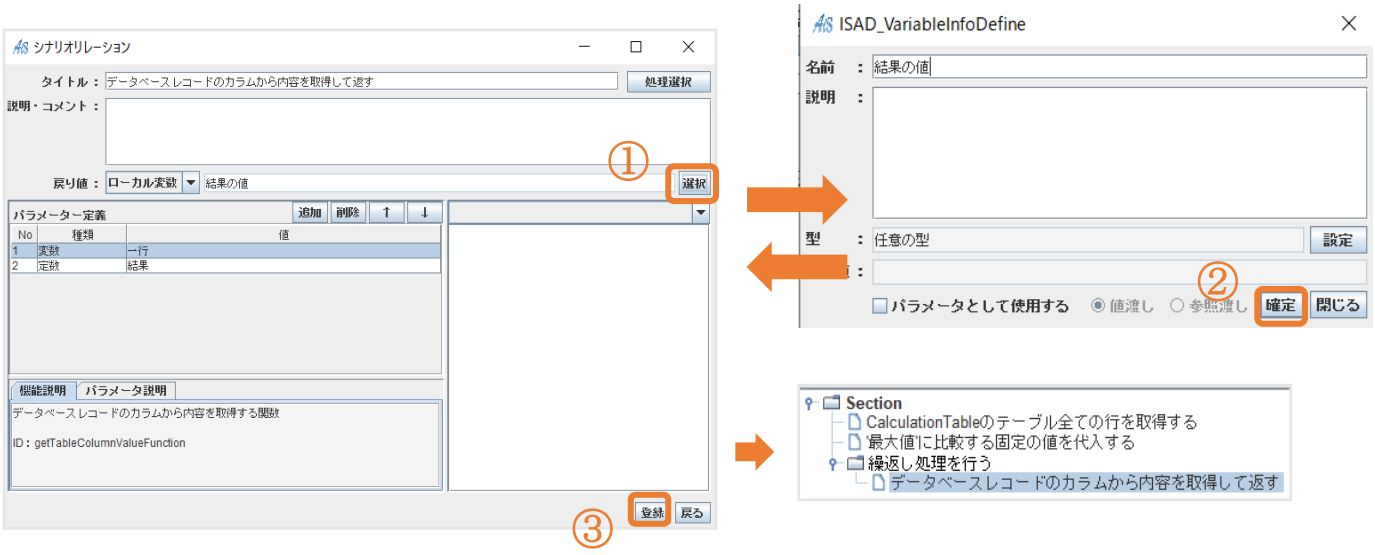
【図 111】



【図 112】



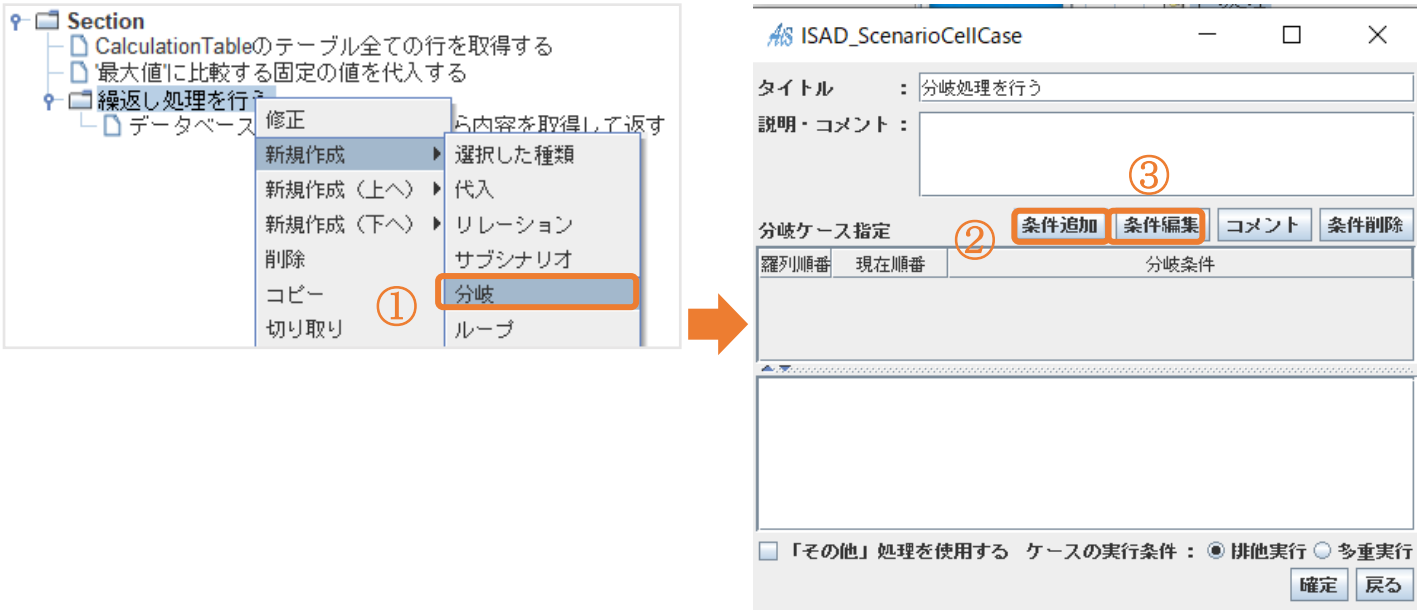
【図 113】


シナリオリレーション
タイトル：
データベースレコードのカラムから内容を取得して返す
処理選択
説明・コメント：
戻り値：
ローカル変数
結果の値
選択
①
パラメーター定義
追加
削除
No
種類
値
1
変数
一行
2
定数
結果
機能説明
パラメータ説明
データベースレコードのカラムから内容を取得する関数
ID：getTableColumnValueFunction
登録
戻る
③
ISAD_VariableInfoDefine
名前
結果の値
説明
型
任意の型
設定
パラメータとして使用する
値渡し
参照渡し
②
確定
閉じる
Section
CalculationTableのテーブル全ての行を取得する
'最大値'に比較する固定の値を代入する
繰返し処理を行う
データベースレコードのカラムから内容を取得して返す

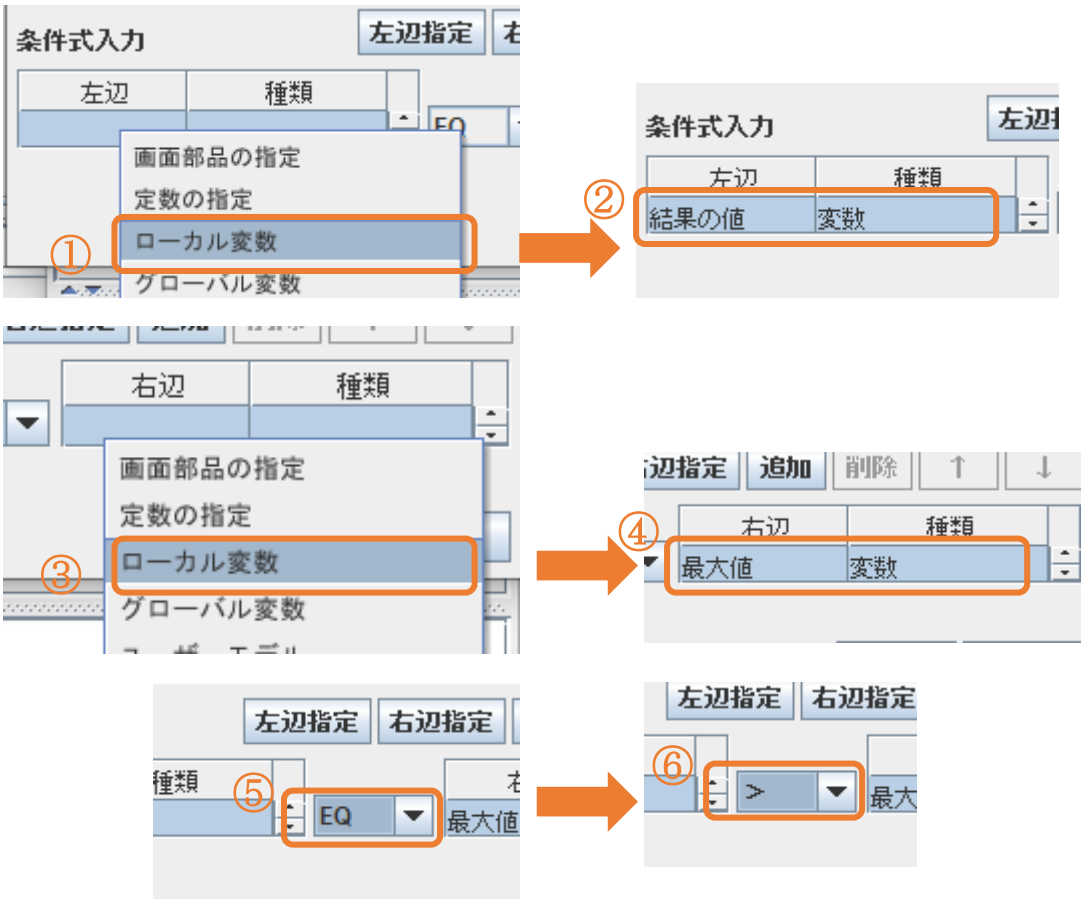
手順５：分岐処理を行う

これも「繰り返し処理を行う」の直下に作成したいので、「繰り返し処理を行う」を右クリックで「新規作成」→「分岐」を選択

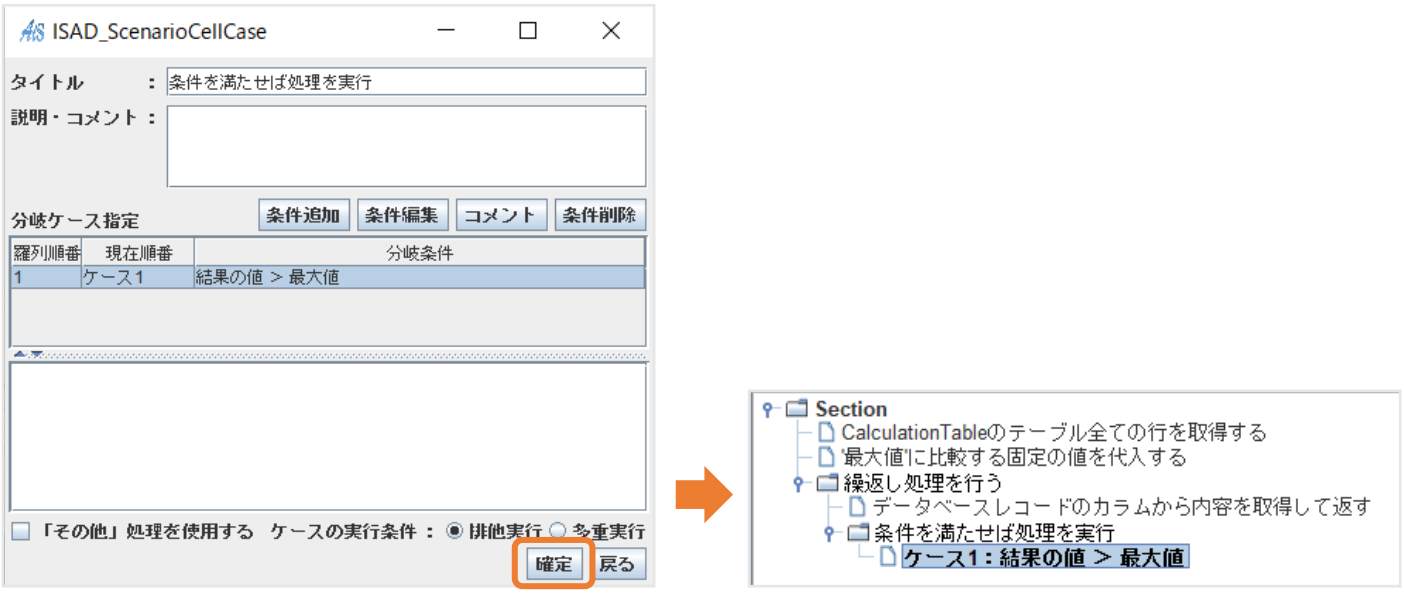
【図 114】



【図 115】


条件式入力
左辺指定
左辺
種類
画面部品の指定
定数の指定
ローカル変数
グローバル変数
①
②
結果の値
変数
右辺
③
追加
削除
④
最大値
⑤
EQ
⑥
>
最大

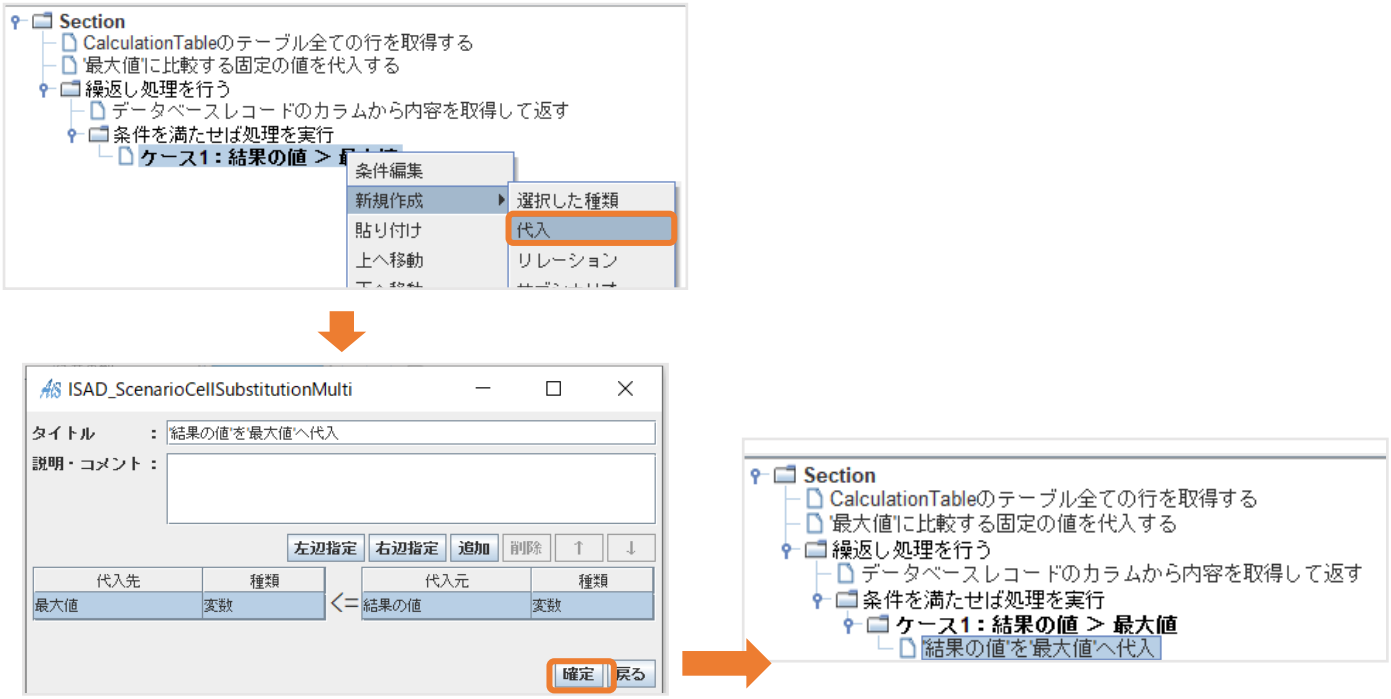
【図 116】


ISAD_ScenarioCellCase
タイトル ： 条件を満たせば処理を実行
説明・コメント ：
分岐ケース指定
条件追加
条件編集
コメント
条件削除
羅列順番
現在順番
分岐条件
1
ケース1
結果の値 > 最大値
「その他」処理を使用する
ケースの実行条件 ： 排他実行 多重実行
確定
戻る
Section
CalculationTableのテーブル全ての行を取得する
'最大値'に比較する固定の値を代入する
繰返し処理を行う
データベースレコードのカラムから内容を取得して返す
条件を満たせば処理を実行
ケース1：結果の値 > 最大値

手順６：「結果の値」を「最大値」へ代入

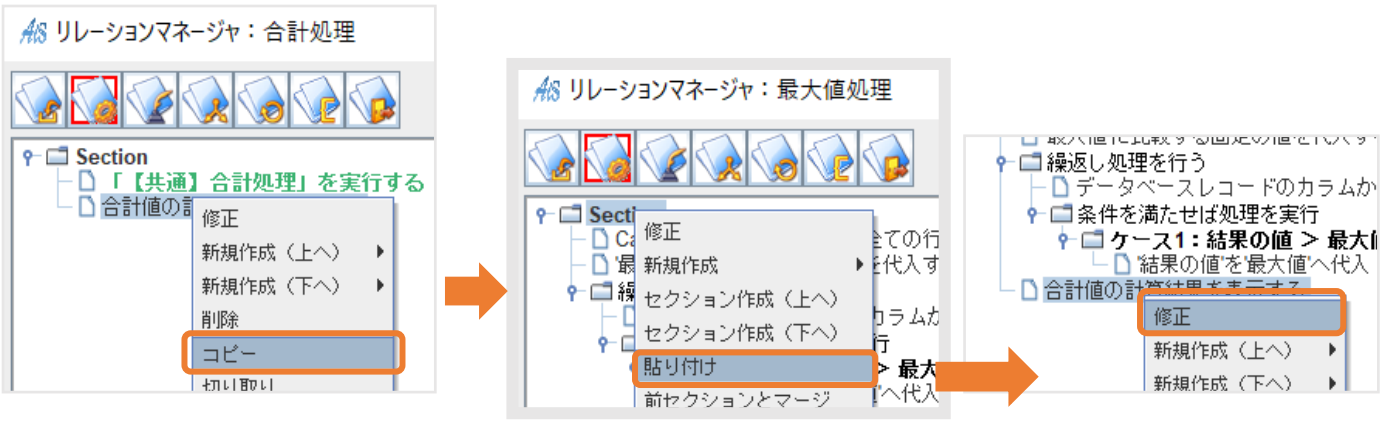
これは「ケース１：結果の値＞最大値」の直下に作成したいので、「ケース１：結果の値＞最大値」を右クリックで「新規作成」→「代入」を選択

【図 117】

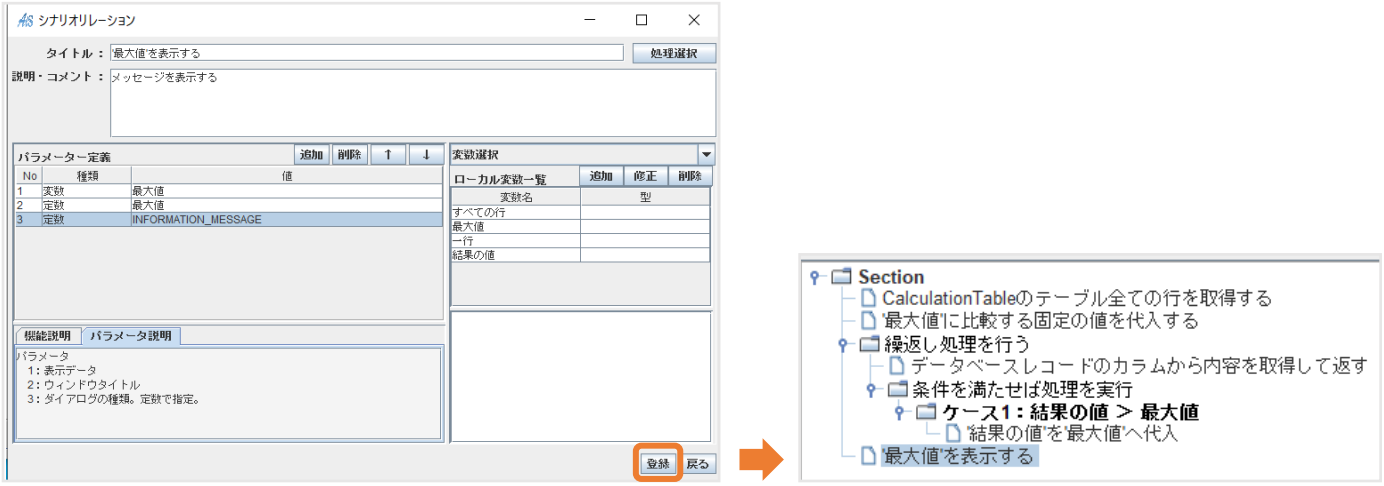


手順７：「最大値」を表示する

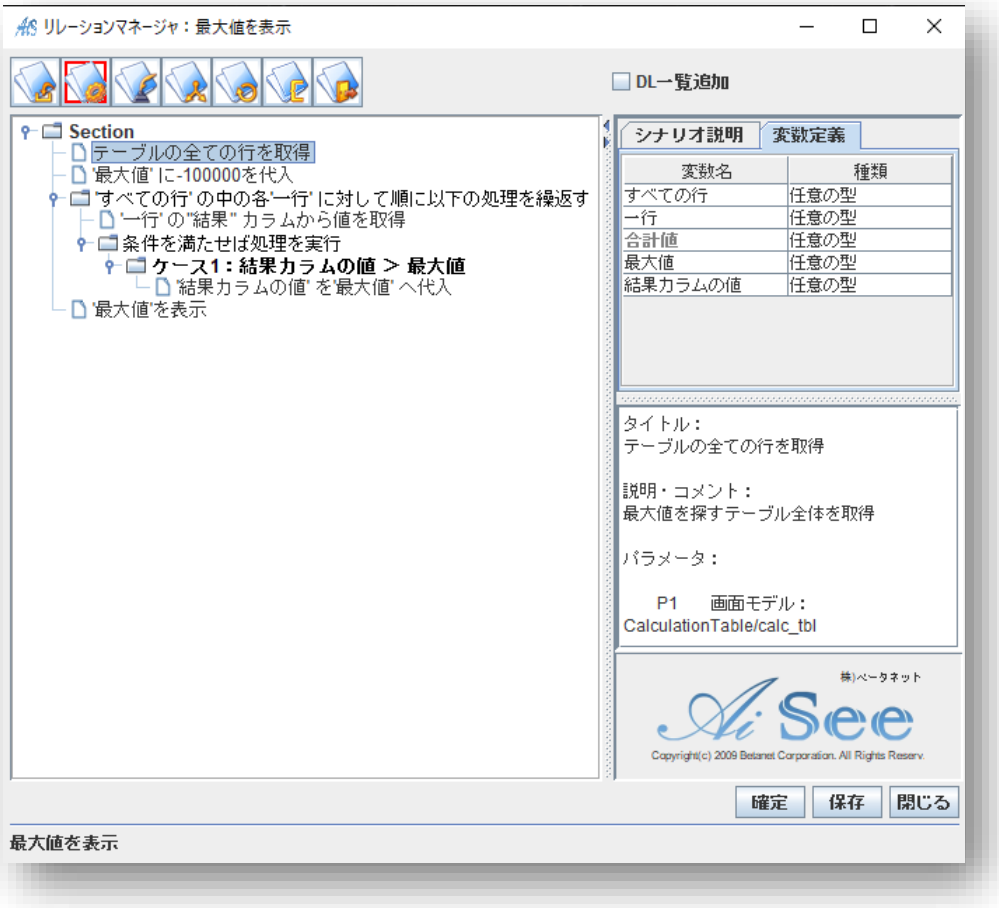
【図 118】



【図 119】



＜最大値処理の全体像＞



※'最大値' の初期値設定は Java 言語の long 型で扱える最小の値(-9223372036854775808)までであれば代入が可能

ここで**テスト実行**を行いましょう。

問題がなければ「CalculationTable」内で右クリックをすると「メニュー」が表示され「最大値」を押下すると「結果」カラムの

最大値がメッセージとして表示されます。(図)

5.3.12 最小値処理

「CulculationTable」で「メニュー」画面から「最小値」を押下すると「結果」カラムの最小数値が表示される実装

手順１：最大値処理からリレーションをコピーしてくる

手順２：「最小値」に比較する固定の値を代入する

手順３：条件を満たせば処理を実行

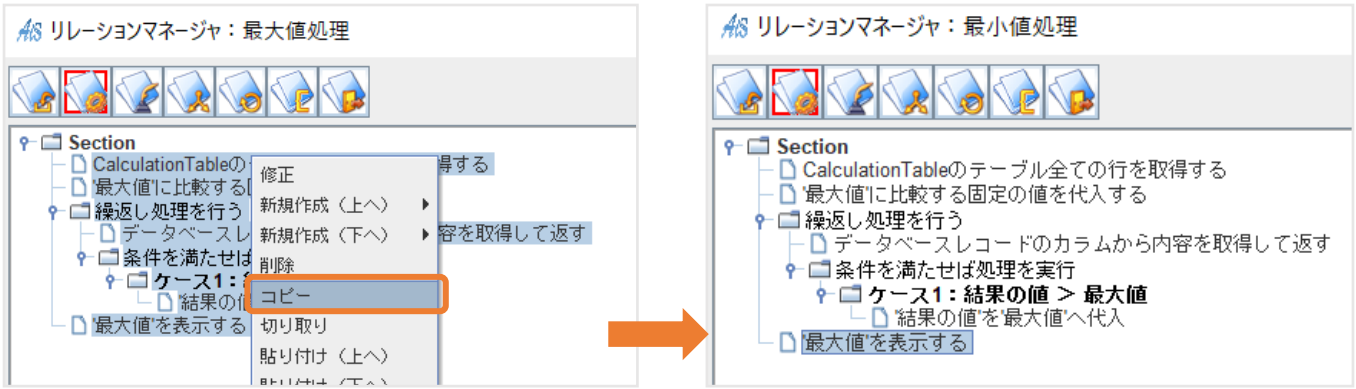
手順４：「結果の値」を「最小値」へ代入

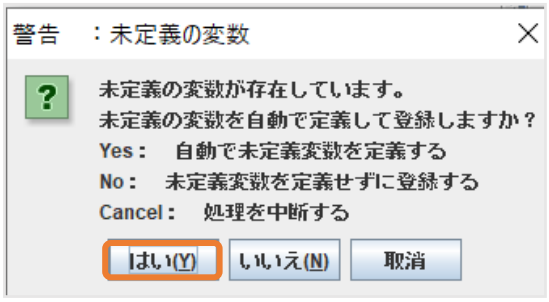
手順５：「最小値」を表示する

手順６：テスト実行

手順１：最大値処理からリレーションをコピーしてくる

【図 120】





「最小値処理」に貼り付けをした後に右下の「確定」を押してください。

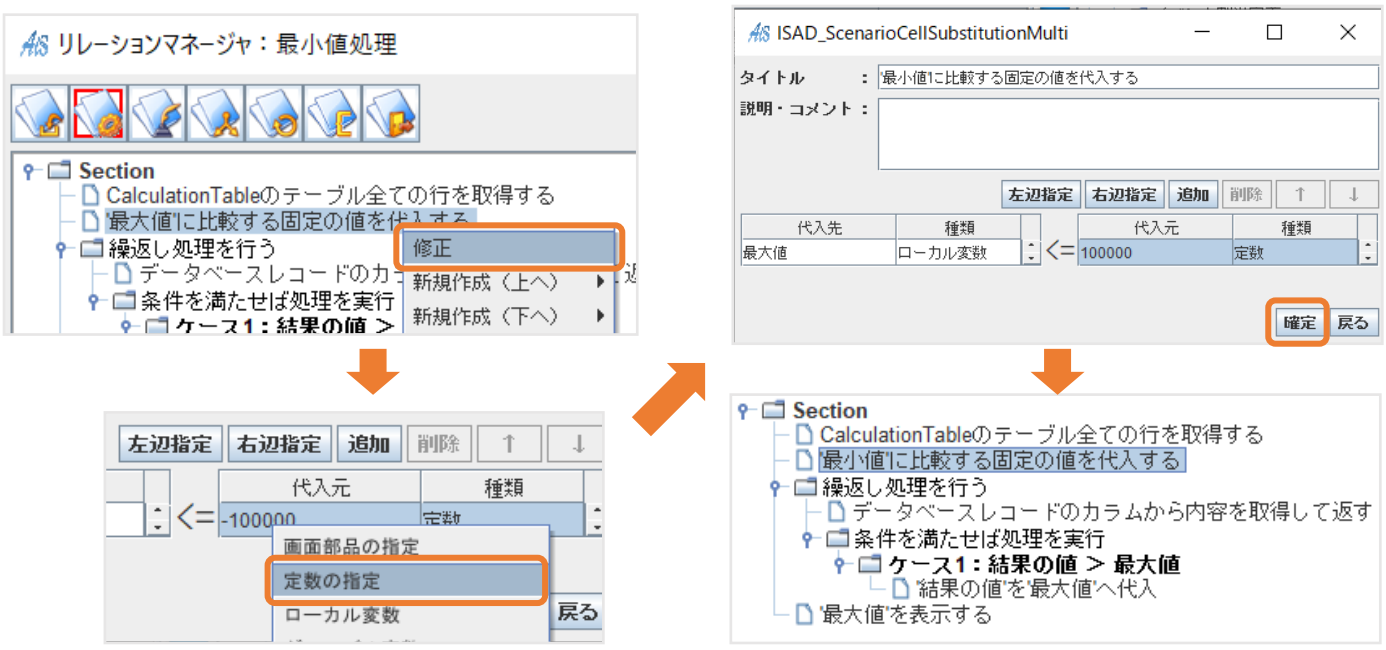
警告文が出てきますが、「はい」で一度確定をして未定義の変数を自動で定義しましょう。

手順２：「最小値」に比較する固定の値を代入する

※ここでは、テーブルに入る値と比較して充分に大きい任意の値（100000）を設定しています。

AiSee で設定できる最大値は「9223372036854775807」

【図 121】

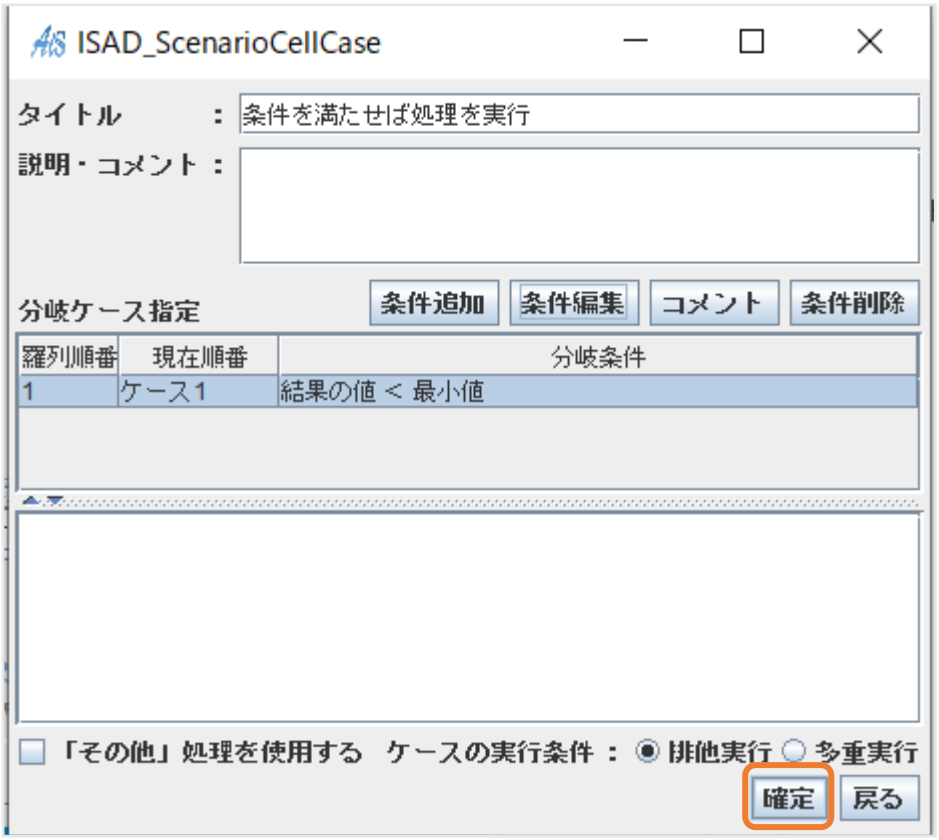


手順 3：条件を満たせば処理を実行

これも「繰り返し処理を行う」の直下に作成したいので、「繰り返し処理を行う」を右クリックで「新規作成」→「分岐」を選択

【図 122】

【図 123】


ISAD_ScenarioCellCase
タイトル ： 条件を満たせば処理を実行
説明・コメント ：
分岐ケース指定
条件追加
条件編集
コメント
条件削除
羅列順番
現在順番
分岐条件
1
ケース1
結果の値 < 最小値
「その他」処理を使用する
ケースの実行条件 ：
排他実行
多重実行
確定
戻る

手順４：「結果の値」を「最小値」へ代入

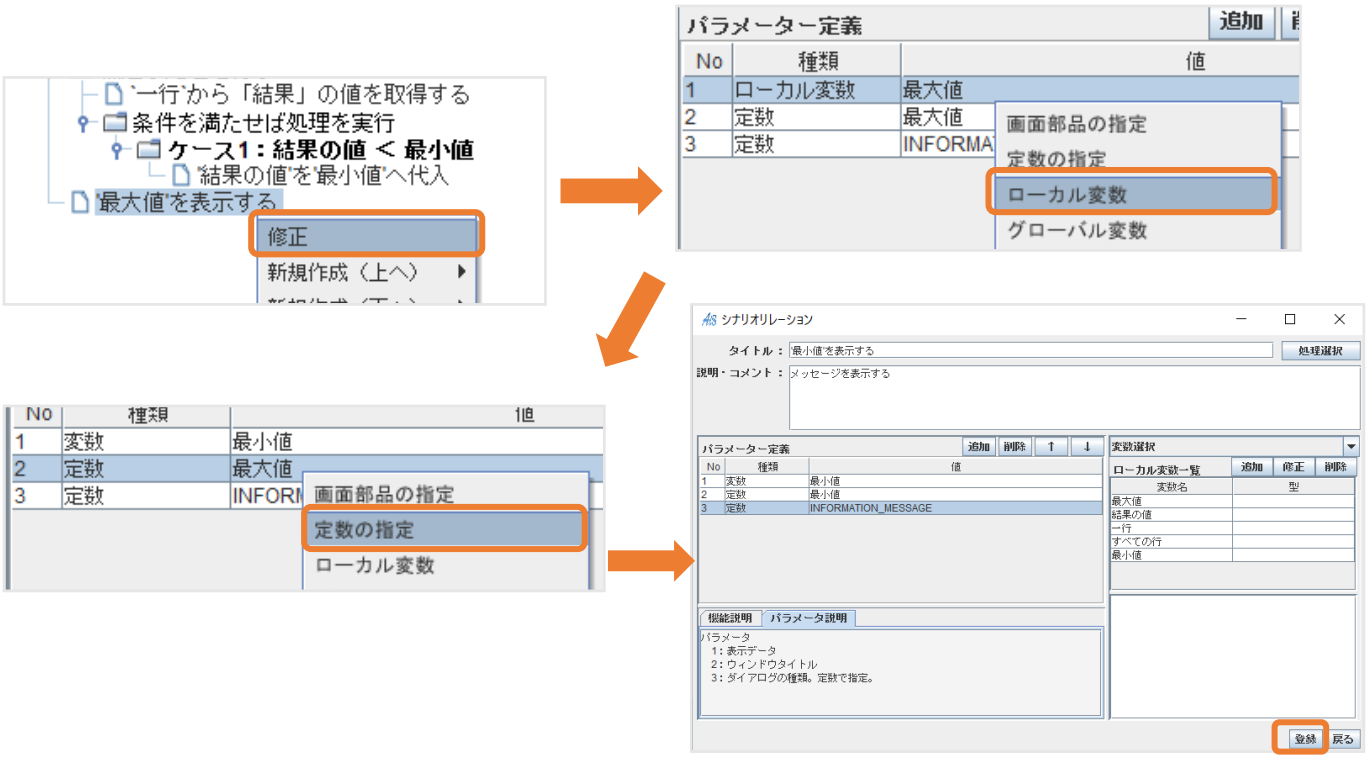
これは「ケース１：結果の値＜最小値」の直下に作成したいので、「ケース１：結果の値＜最小値」右クリックで「新規作成」→「代入」を選択

ローカル変数「最小値」は未作成なので作成してください。

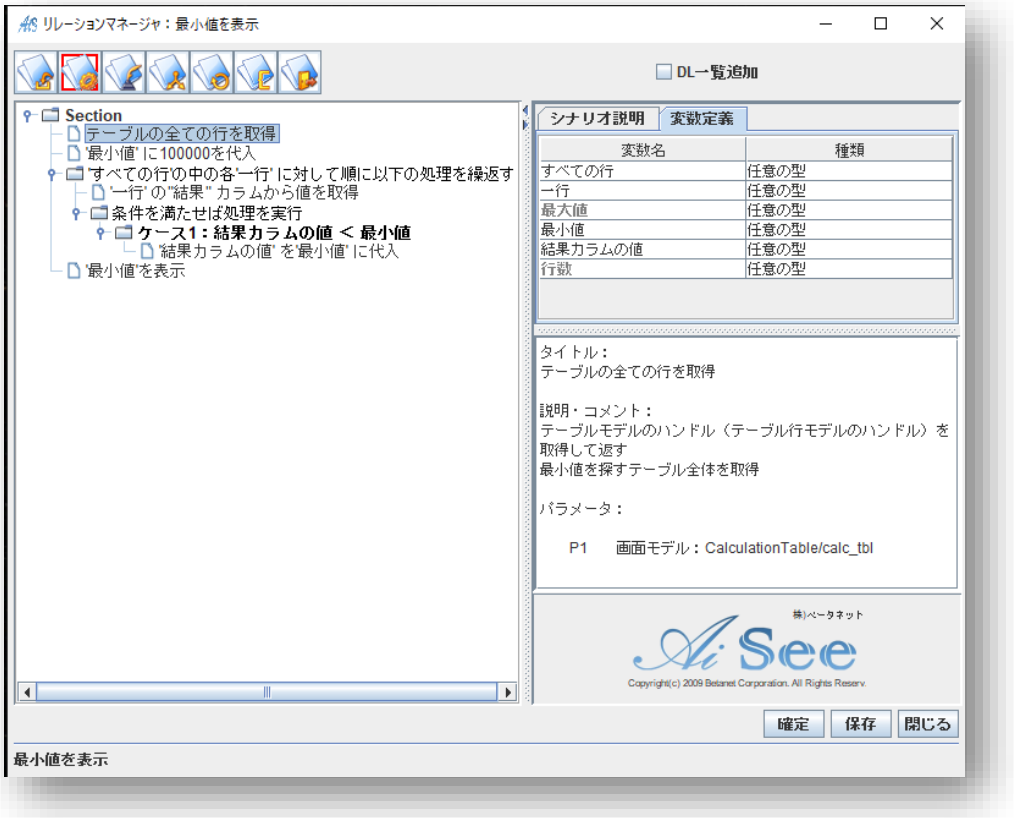
【図 124】

手順５：「最小値」を表示する

【図 125】



＜最小値処理の全体像＞

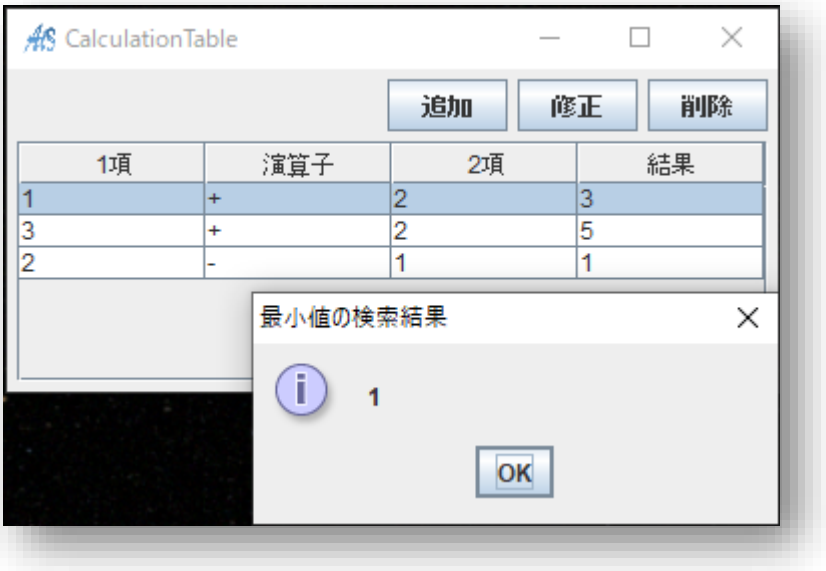


※'最小値' の初期値設定は Java 言語の long 型で扱える最大の値(9223372036854775807)までであれば代入が可能

ここで**テスト実行**を行いましょう。

問題がなければ「CalculationTable」内で右クリックをすると「メニュー」が表示され「最小値」を押下すると「結果」カラムの

最小値がメッセージとして表示されます。



5.4 データベースについて

データベース(以下 DB)とは…

決まった形式（データモデル）で整理されたデータの集まりのことです。

DB にも種類があり今回の例では「リレーショナルデータベース」を使用します。

リレーショナルデータベース(以下 RDB)とは…

Excel のような表の形式でデータを管理するタイプです。AiSee ではエクセルでいうシートを「テーブル」といい、列を「フィールド/カラム」、行を「レコード」と呼びます。

カラムには項目が入り、レコードには項目ごとに該当するデータが入ります。

<リレーショナルデータベースのイメージ図>

↓テーブル

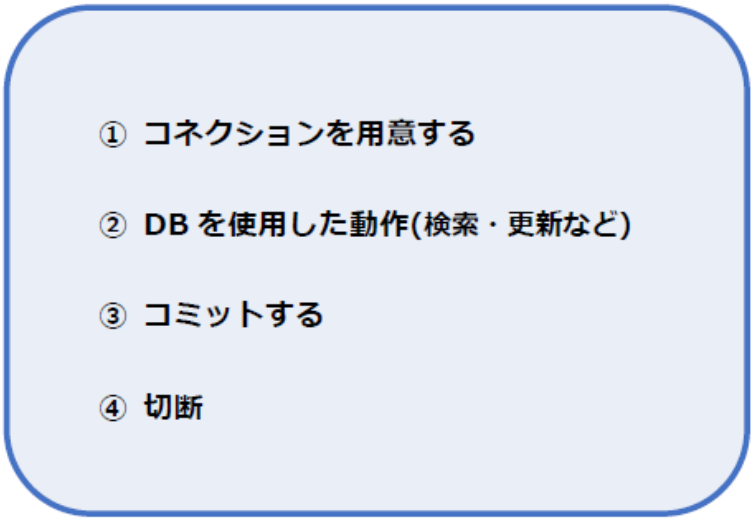
ユーザーID	名前	PASSWORD
0001	山田太郎	111
0002	田中花子	222
0003	鈴木次郎	333

←レコード

↑フィールド/カラム

5.4.1 AiSee における DB の使用方法

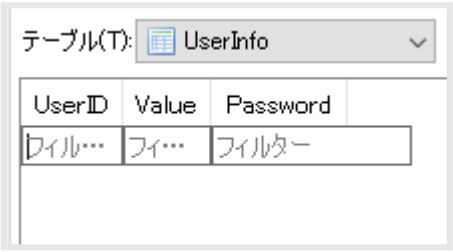
基本的な DB を使用した大まかなフローは以下の通りです。



以降の章では具体的な例を見ながら実際に作成していきましょう。

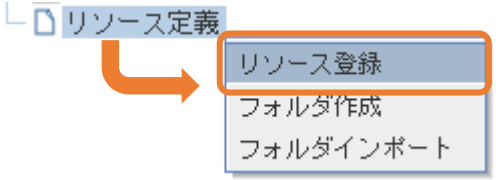
5.4.2 DB を使ったシナリオ作成

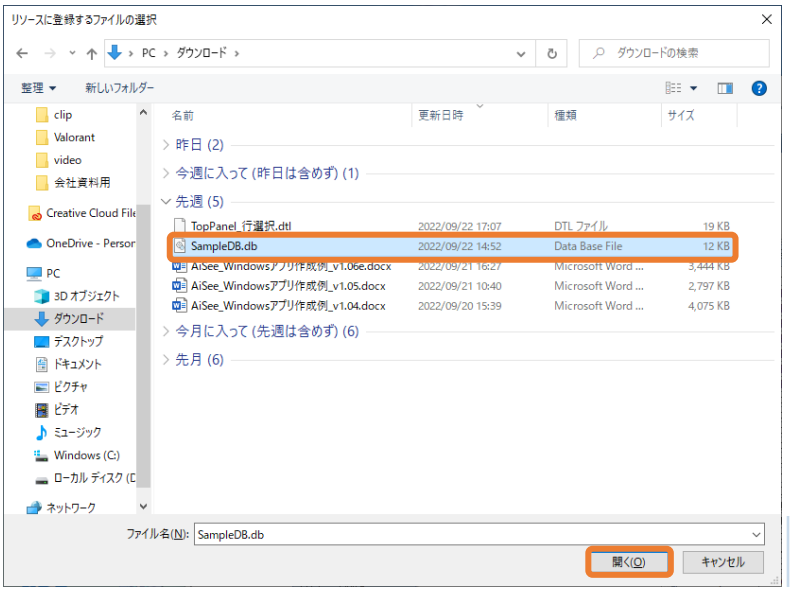
今回は、以下のような DB のテーブル（SampleDB.db）を使用し、このテーブルに AiSee アプリからデータを書き込んだり、データの検索や削除ができるようにシナリオを作成します。
テーブルの名前が「UserInfo」で、「UserID」「Value」「Password」の 3 つのカラムを作成済です。



1.データを DB へ追加する

(1)リソース定義に、作成済の DB ファイルを登録



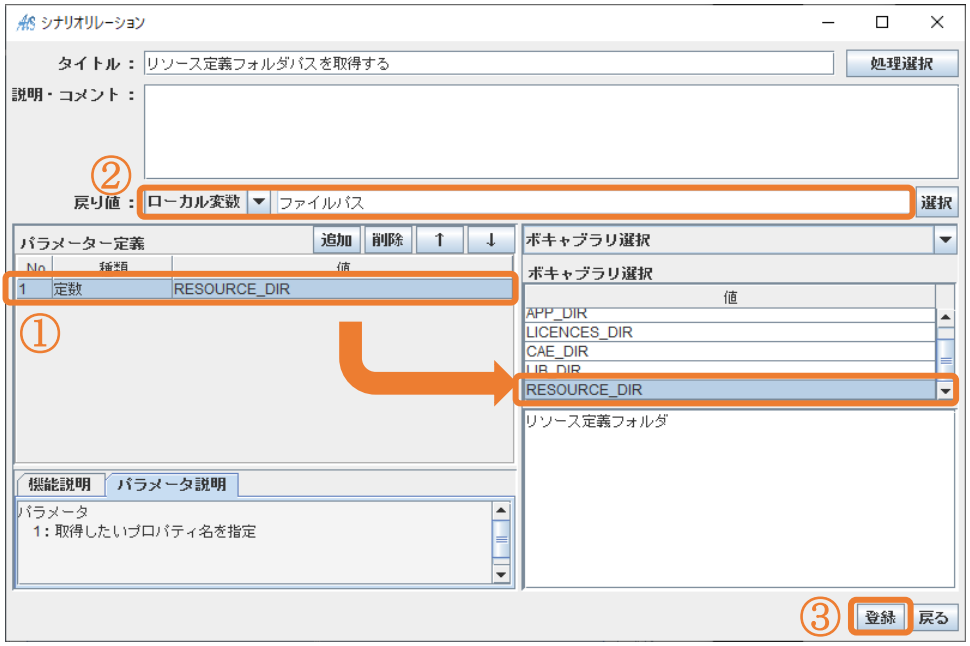


①リソース定義を右クリック
②リソースを登録をクリック
③事前にダウンロードした「 SampleDB.db 」を選択→開くをクリック

(2)シナリオ作成
手順 1:リソース定義フォルダパスを取得する
手順 2:リソース定義フォルダパスに DB ファイル名を結合して DB ファイルパスを作成する
手順 3:データベースへのコネクションを作成して返す
手順 4:挿入マップを作成する
手順 5:データベースを挿入する
手順 6:データベースをコミットする
手順 7:データベースのコネクションを切断する

※詳しい手順はリンク先を参照

手順 1:リソース定義フォルダパスを取得する

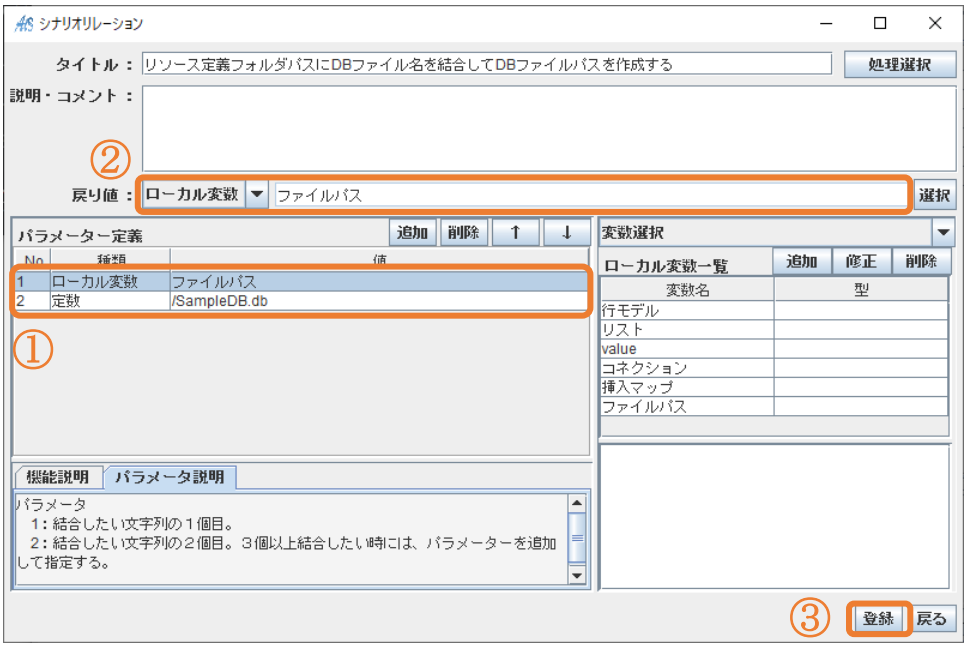


①パラメータをセットする

②戻り値の指定(変数を作成してない場合はここで作成する)

③登録

手順 2:リソース定義フォルダパスに DB ファイル名を結合して DB ファイルパスを作成する



①パラメータをセットする

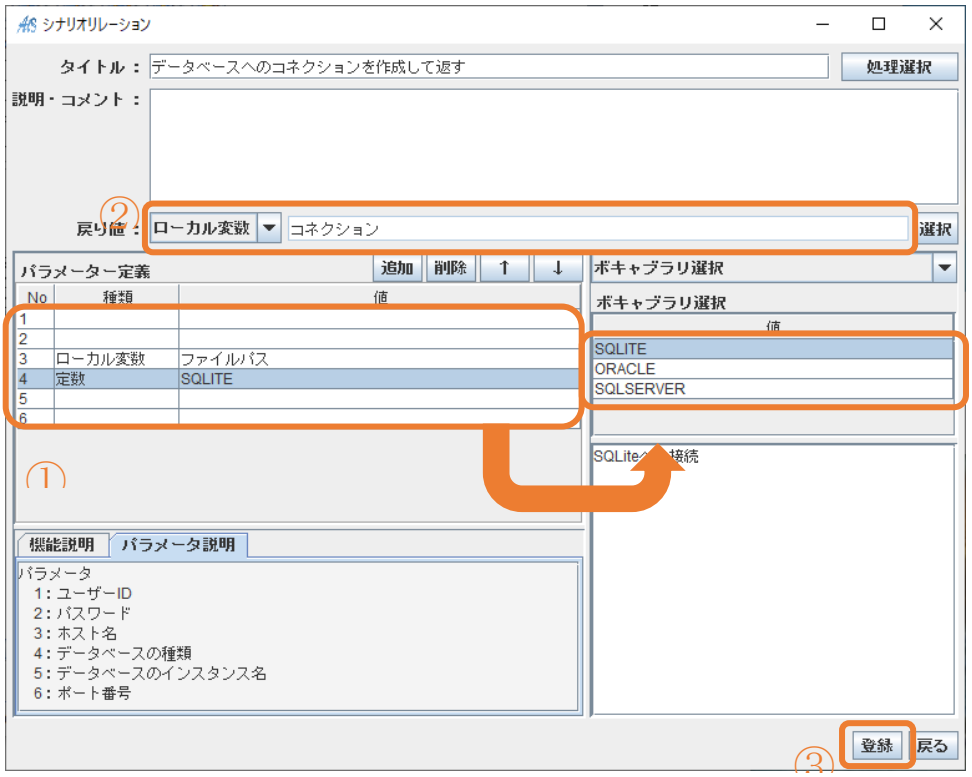
No.2 のところに定数「/SampleDB.db」と入力します

※任意で用意した場合はそのファイル名を入力してください

②戻り値の指定(変数を作成してない場合はここで作成する)

③登録

手順 3:データベースへのコネクションを作成して返す



①パラメータをセットする

No4 のところを左クリック→右の欄から「SQLITE」を選択

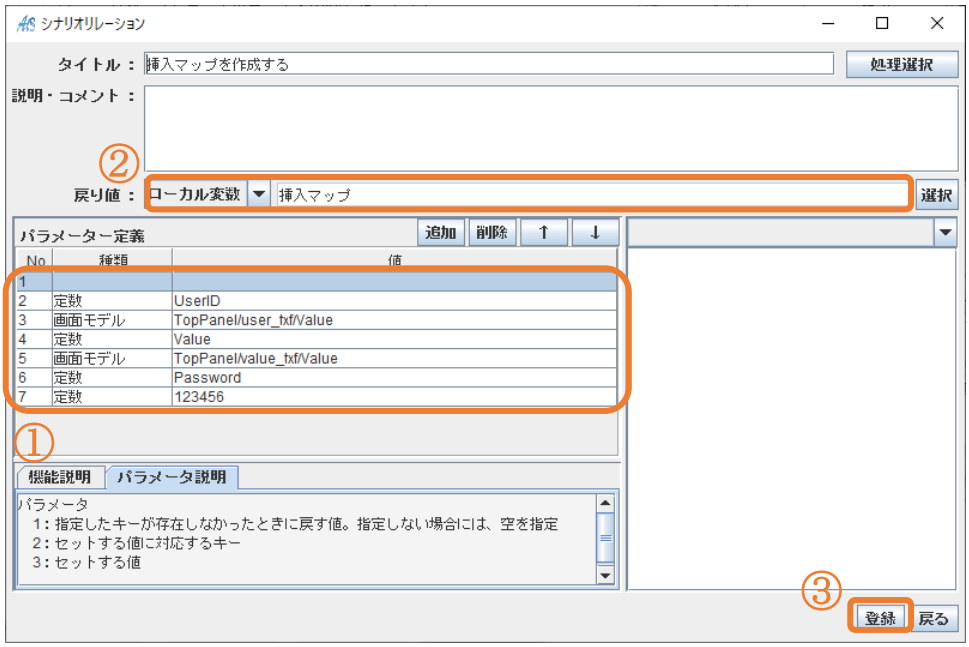
②戻り値の指定(変数を作成してない場合はここで作成する)

③登録

※コネクションとはデータベースにアクセスするチャンネルのようなもの。

パラメータ定義の No3 と No4 にのみ上記変数を挿入することで OK

手順 4:挿入マップを作成する



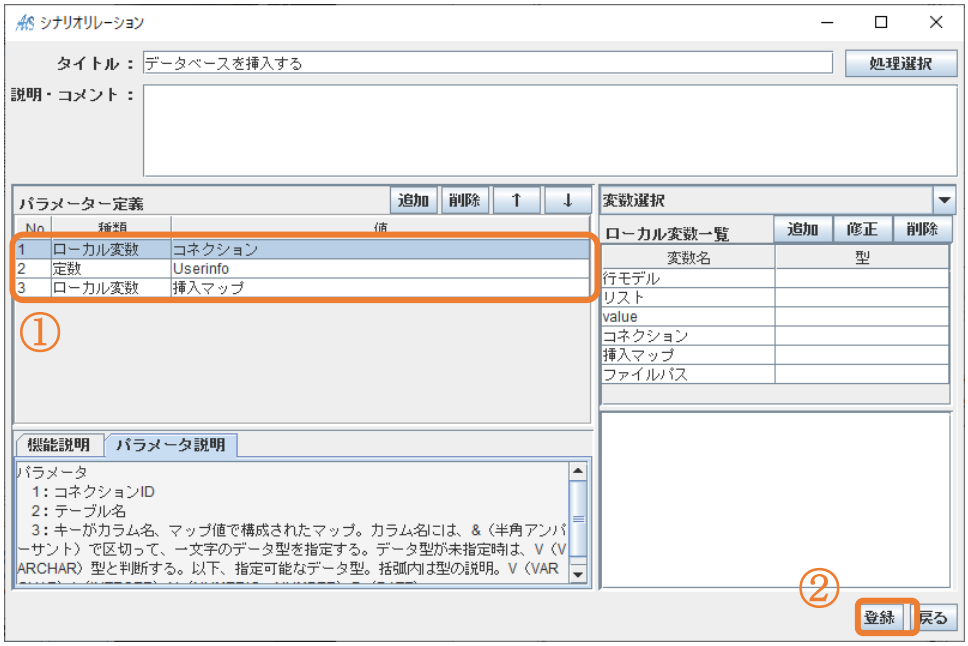
①パラメータをセットする

②戻り値の指定(変数を作成してない場合はここで作成する)

③登録

※挿入マップを作成することでデータベースとシナリオを紐づけることができる。

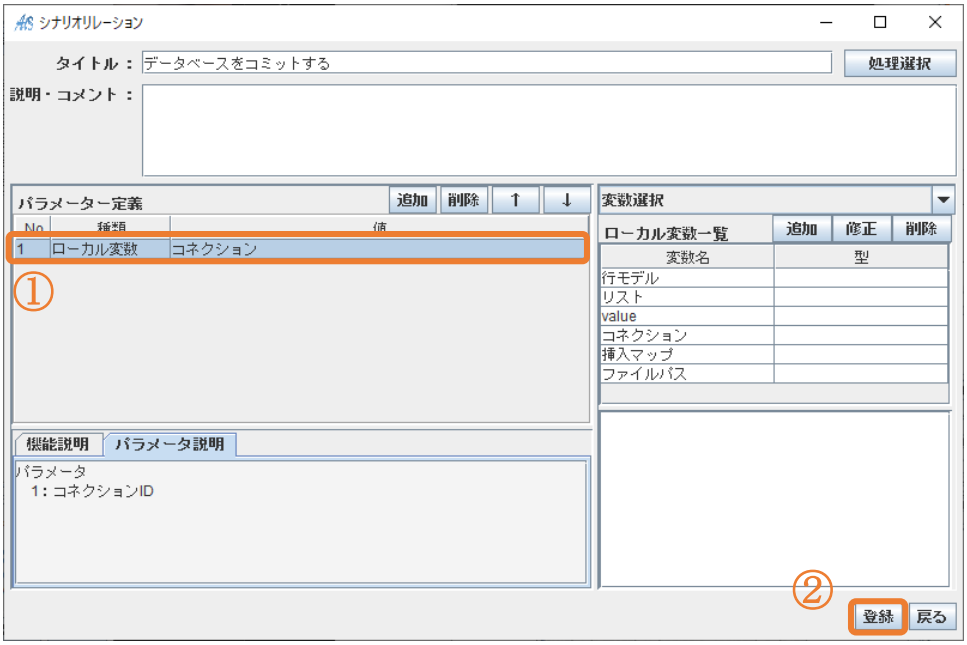
手順 5:データベースを挿入する



①パラメータをセットする

②登録

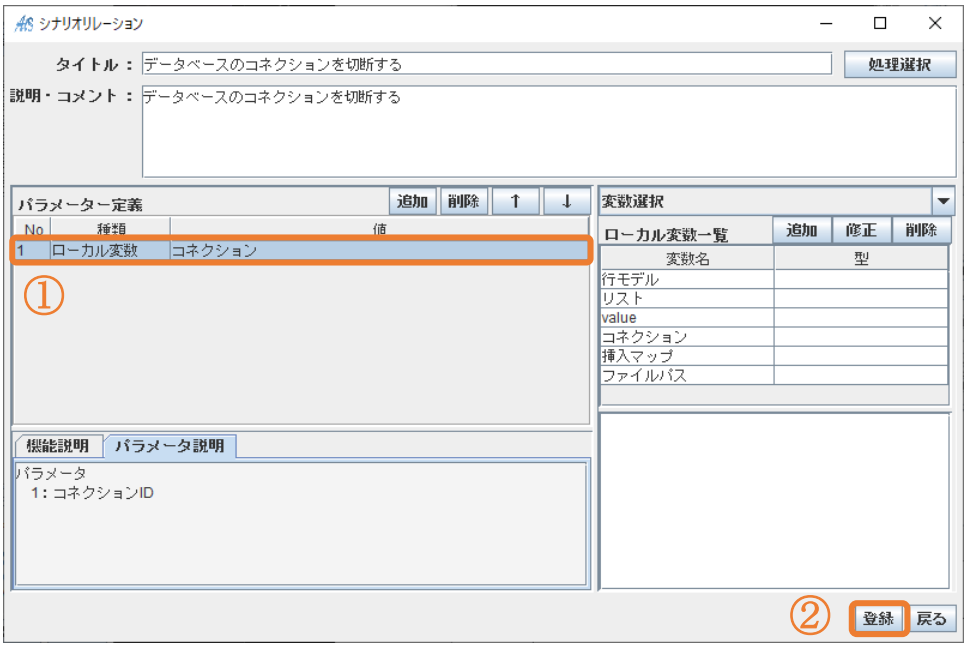
手順 6:データベースをコミットする



①パラメータをセットする

②登録

手順 7:データベースのコネクションを切断する



①パラメータをセットする

②登録

<追加のリレーション全体像>

2.DB を使用してデータを削除する

(1)リソース定義に DB ファイルを登録※すでに登録している場合は省略

(2)シナリオ作成

手順 1:リソース定義フォルダパスを取得する

手順 2:リソース定義フォルダパスに DB ファイル名を結合して DB ファイルパスを作成する

手順 3:データベースへのコネクションを作成して返す

手順 4:条件マップを作成する

手順 5:データベースを削除する

手順 6:データベースをコミットする

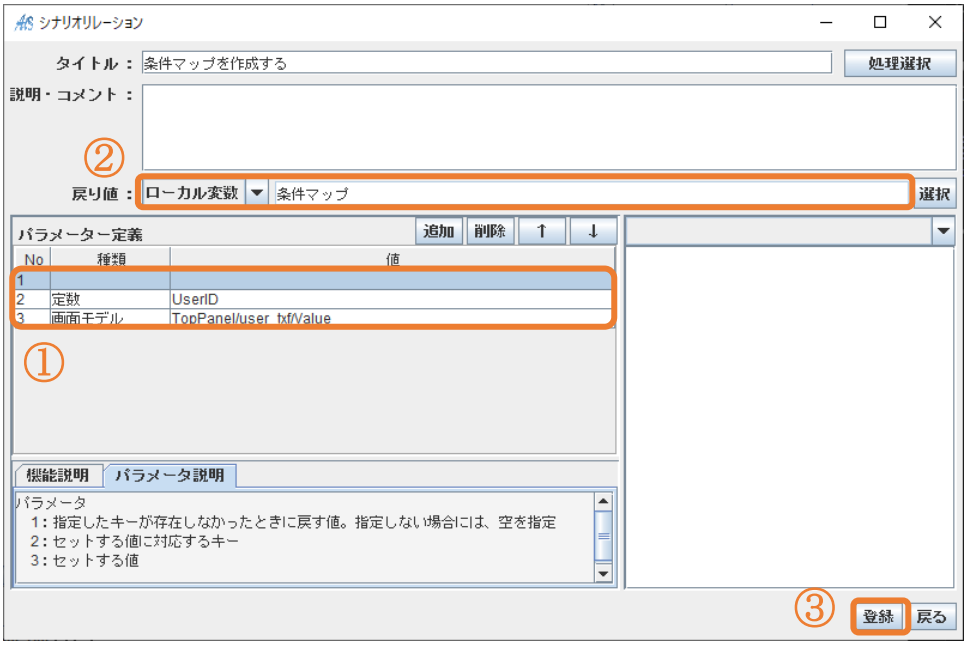
手順 7:データベースのコネクションを切断する

手順 8:テーブルのカレント行を削除する

※手順 1~3・6・7 に関しては「追加」と一緒なのでこの章では省略

その他の詳しい手順はリンク先を参照

手順 4:条件マップを作成する

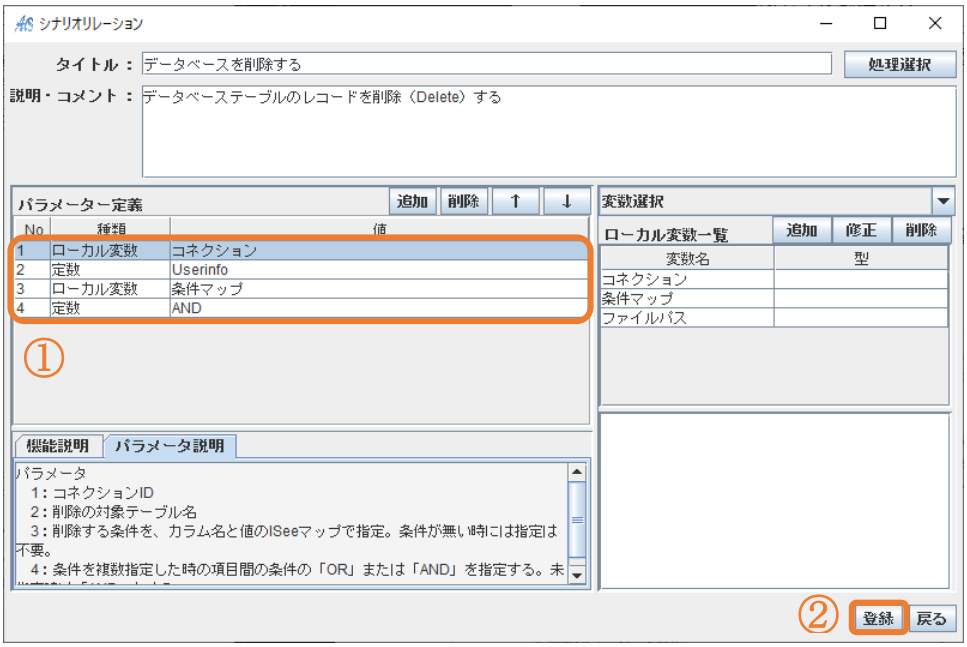


①パラメータをセットする

②戻り値の指定(変数を作成してない場合はここで作成する)

③登録

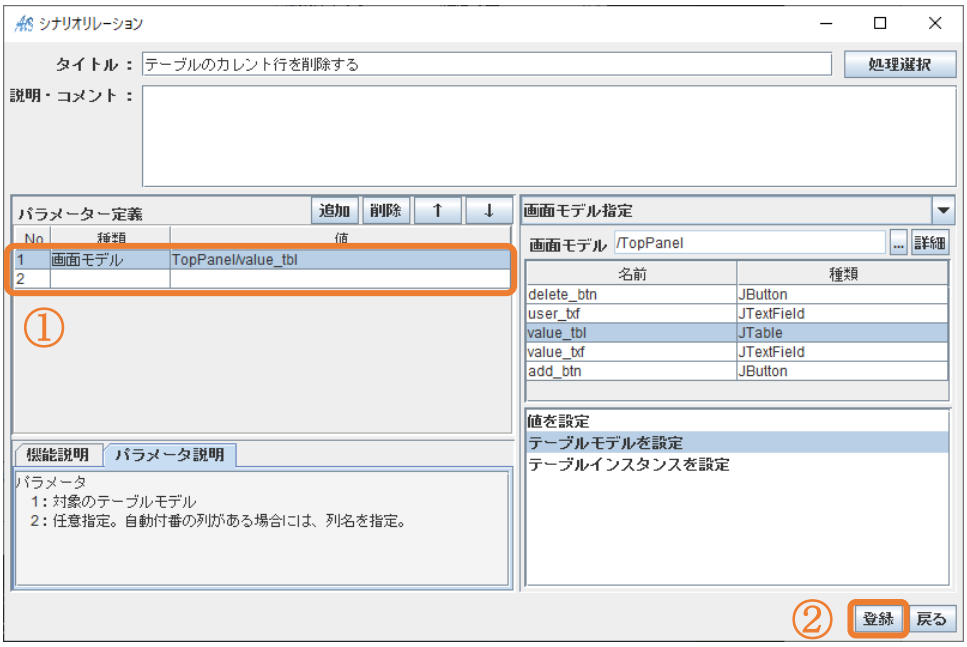
手順 5:データベースを削除する



①パラメータをセットする

②登録

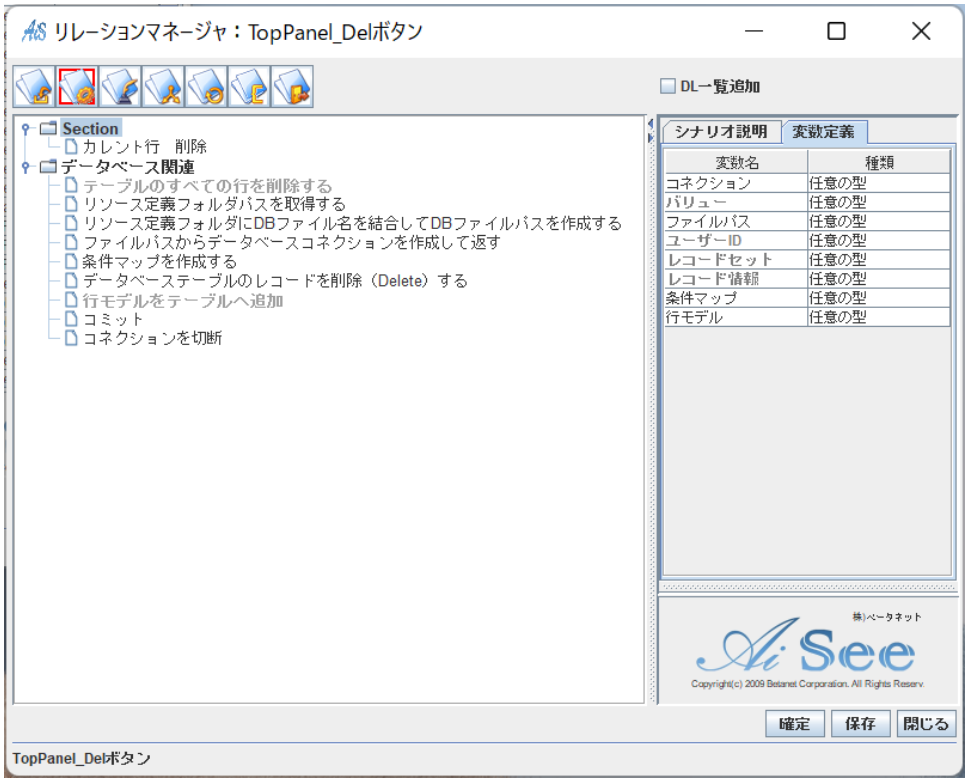
手順 8:テーブルのカレント行を削除する



①パラメータをセットする

②登録

＜削除のリレーション全体像＞



3.DB を使用してデータを検索する

(1)リソース定義に DB ファイルを登録※すでに登録している場合は省略

(2)シナリオ作成

手順 1:画面テーブルを削除する

手順 2:リソース定義フォルダパスを取得する

手順 3:リソース定義フォルダパスに DB ファイル名を結合して DB ファイルパスを作成する

手順 4:データベースへのコネクションを作成して返す

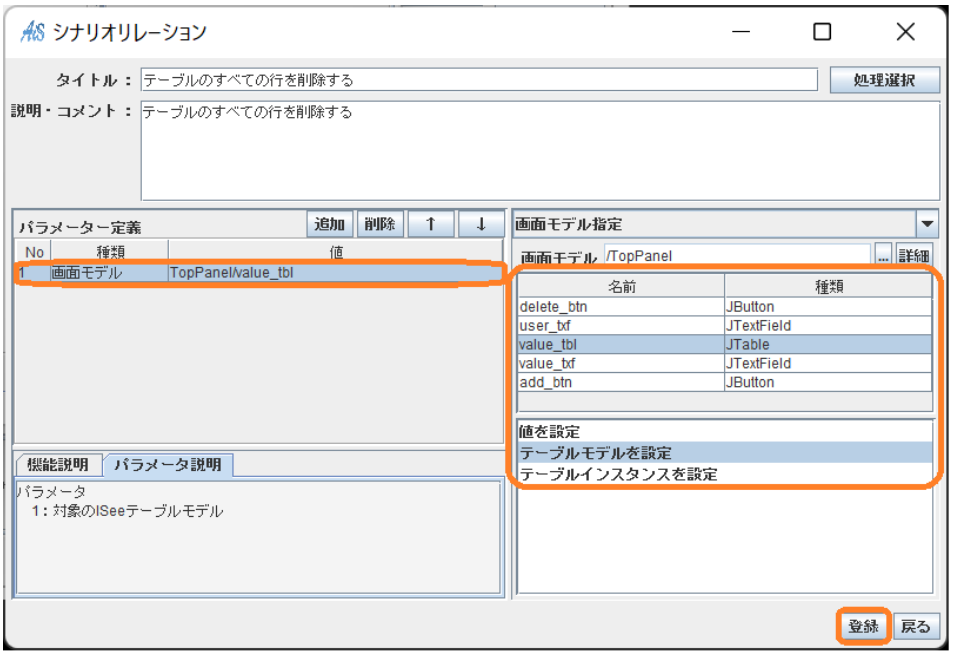
手順 5:挿入マップを作成する

手順 6:データベースを検索する

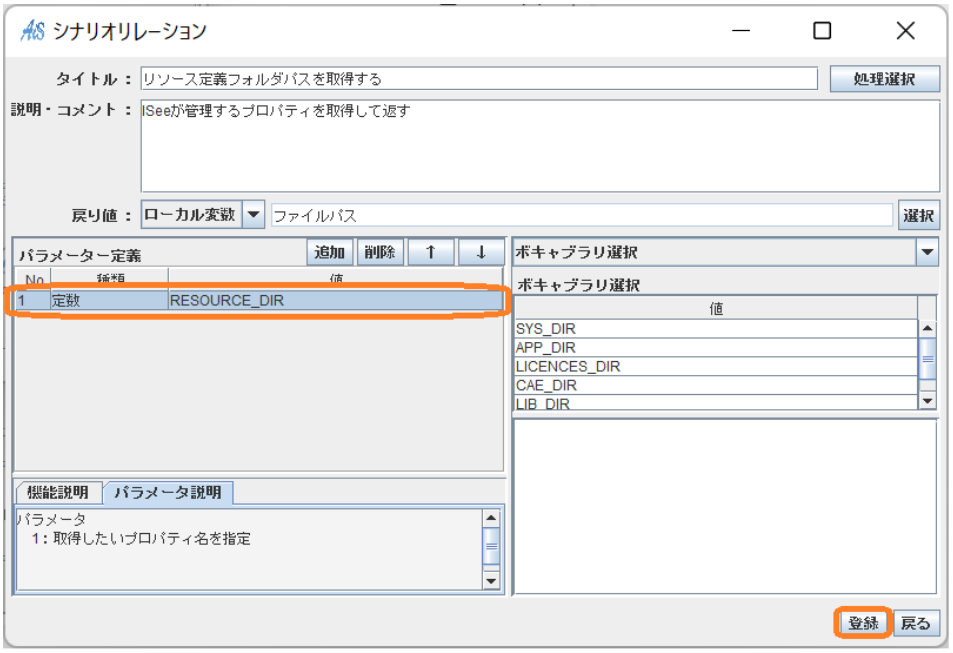
手順 7:レコードセットで繰り返し処理を行う

手順 8:データベースのコネクションを切断する

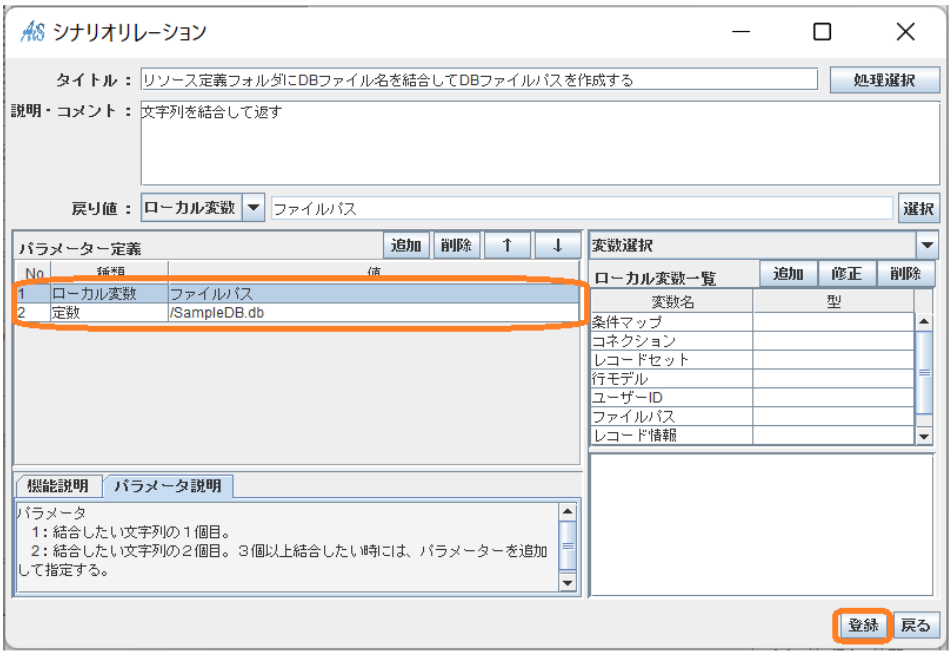
手順 1:画面テーブルを削除する



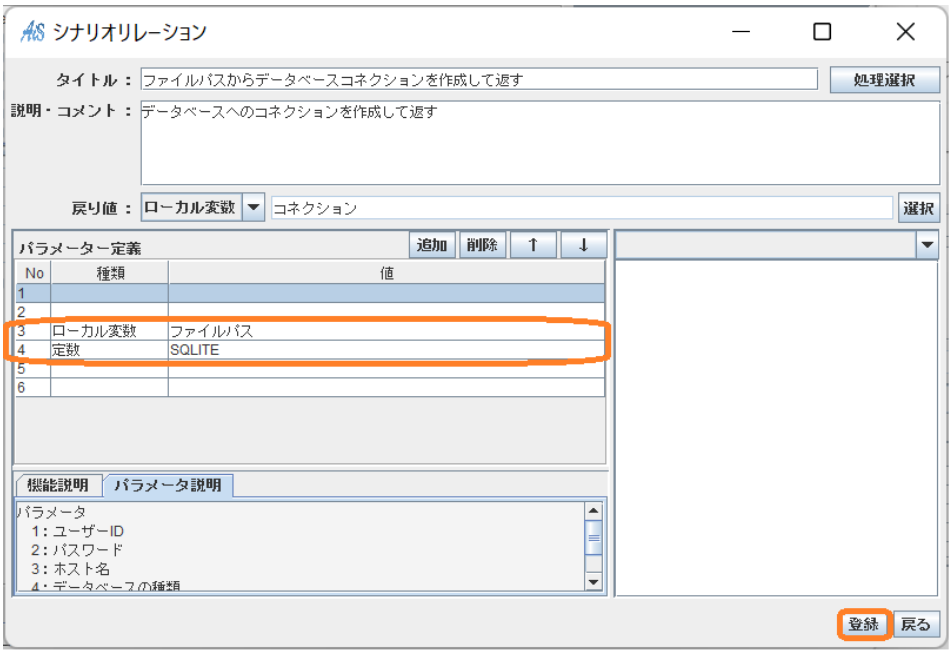
手順 2:リソース定義フォルダパスを取得する



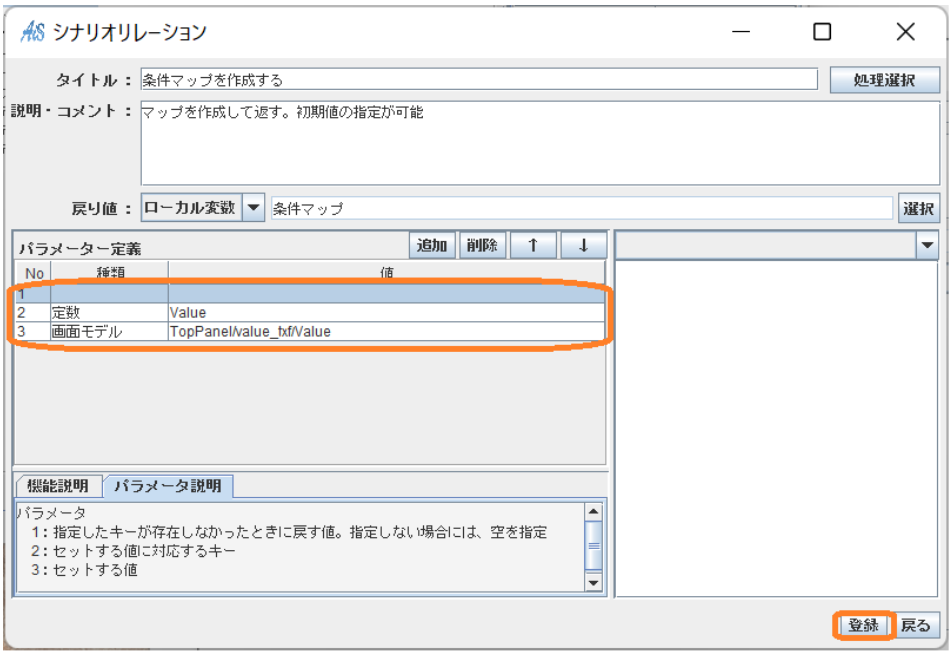
手順 3:リソース定義フォルダパスに DB ファイル名を結合して DB ファイルパスを作成する



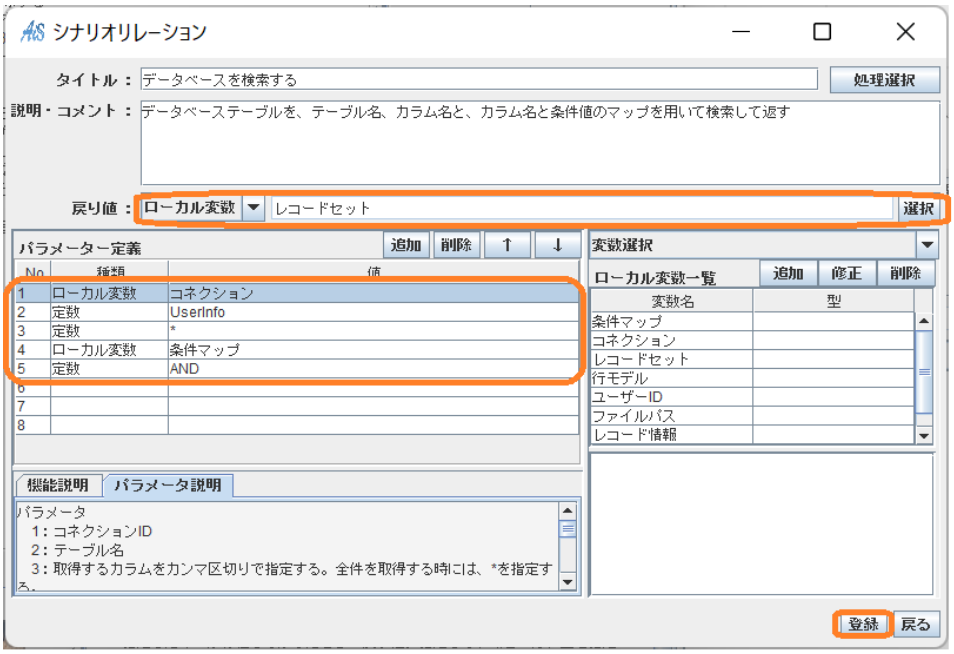
手順 4:データベースへのコネクションを作成して返す



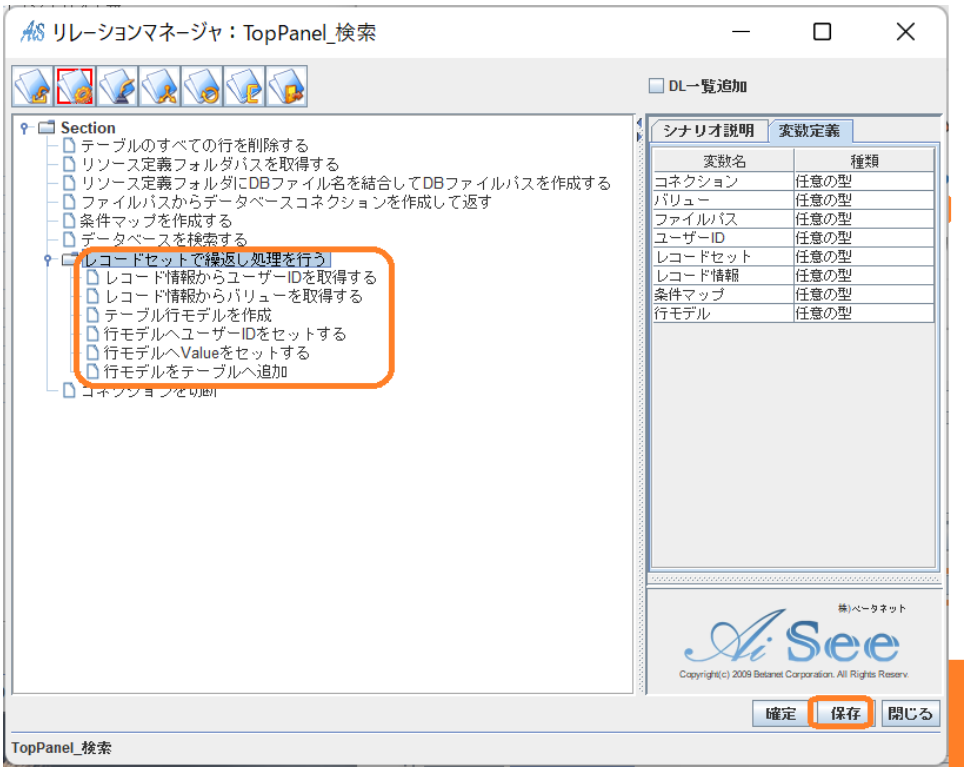
手順 5:挿入マップを作成する



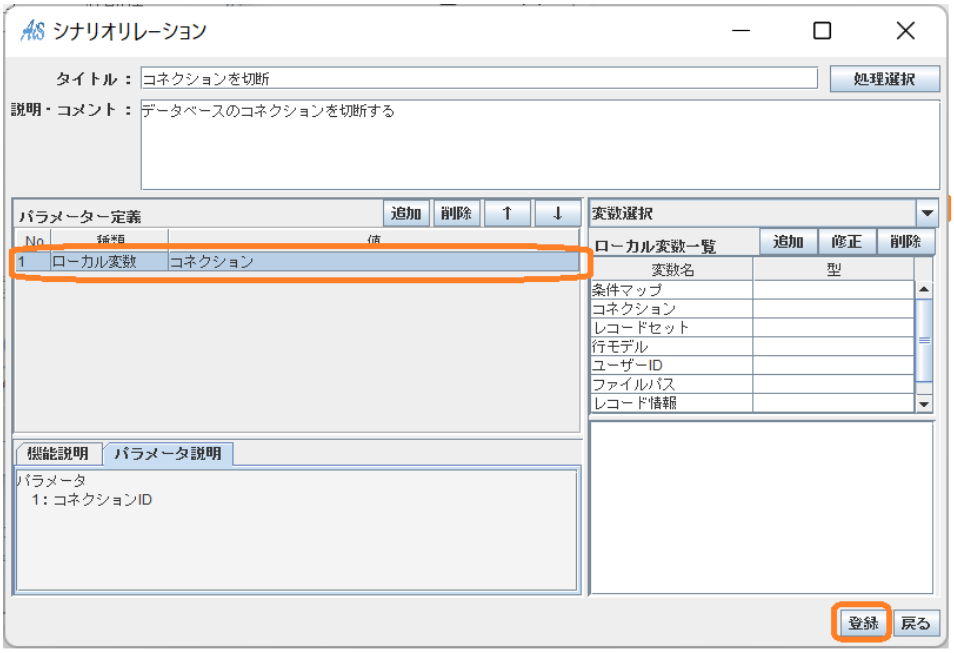
手順 6:データベースを検索する



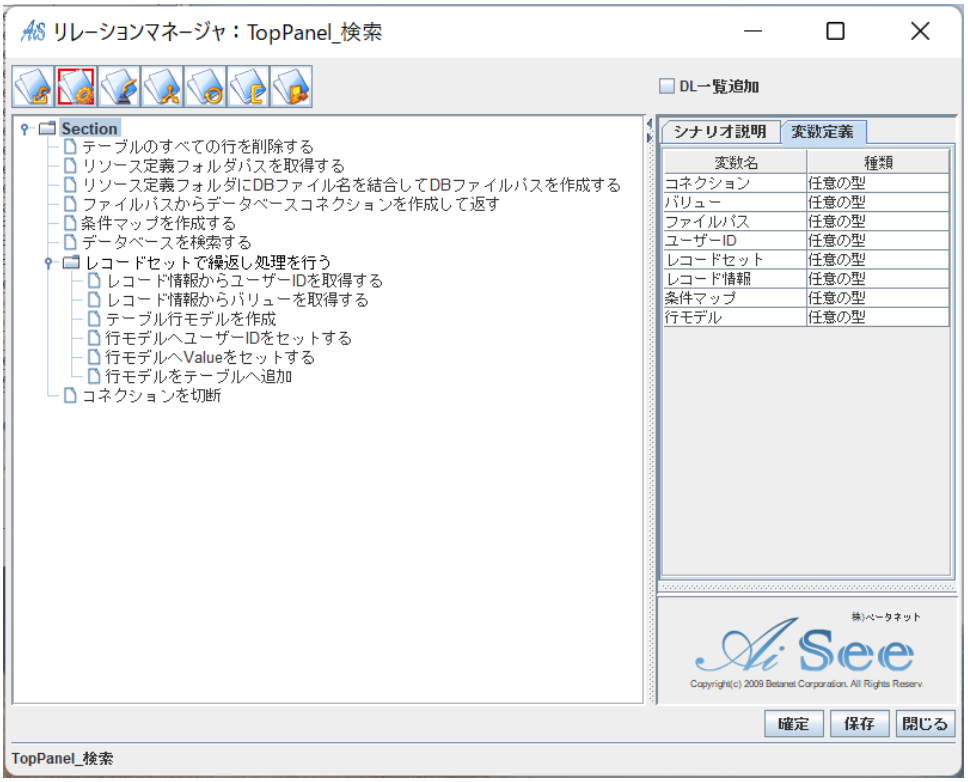
手順 7:レコードセットで繰り返し処理を行う



手順 8:データベースのコネクションを切断する



＜検索のリレーション全体像＞



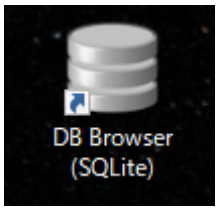
5.4.3 DB を作成したい場合（DB Browser）

上記で作成したアプリでは、事前に DB の作成を「SQLite」を用いて行っています。
ここでは簡単に、SQLite のデータが作成、編集できる「DB Browser」について紹介します。

DB を直接作成・編集したい場合には、以下の 2 つをインストールしてください。

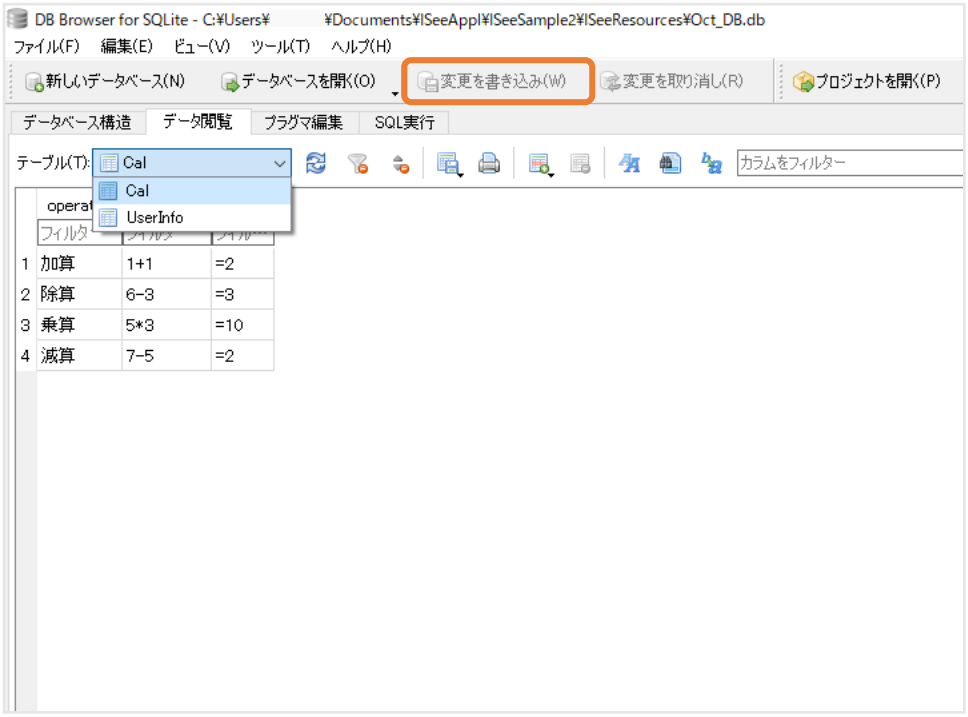
SQLite　https://www.sqlite.org/download.html/
DB Browser　https://sqlitebrowser.org/blog/version-3-12-2-released/

DB Browser をインストールする際に、デスクトップにショートカットを作成するオプションを選択すると、このようなアイコンが表示されます。
※DB Browser(SQL Cipher)　はインストール不要です。DB Browser(SQLite)のみ使用します。



下図は DB Browser の画面です。作成済みの「〇〇〇.db」ファイルを開いた場合、「データ閲覧」のタブから、データベースのテーブルの内容が以下のように表示されます。
（「〇〇〇.db」ファイルをドラッグ＆ドロップで開けます）

このページ上でテーブルのデータを直接編集することが可能です。
変更した後は「変更を書き込み」ボタンを押して内容を更新してください。



DB を新規作成したい場合には、現在開いている DB があれば一旦接続解除してから、画面左上の「新しいデータベース」ボタンをクリックし作成してください。

6　NetBeans でアプリの画面を作成する方法

ここでは、NetBeans を用いて、アプリの画面の作成に必要な Jar ファイルを作成します。

最初に Java1.8(JDK8)と、NetBeans IDE15 をインストールしてください。

＜ Java1.8(JDK8) ＞http://www.betanet.co.jp/download/jdk-8u192-windows-x64.exe

＜ NetBeans IDE15＞https://netbeans.apache.org/download/nb15/

上記をインストール後、以下の手順で作成していきます。

手順 1：NetBeans を起動

手順 2：パッケージとクラスを作成

手順 3：パネルの name にクラス名を入力

手順 4：画面に部品を並べる

手順 5：AiSee で操作する全てのコンポーネント(ボタン等の部品)の名前を設定

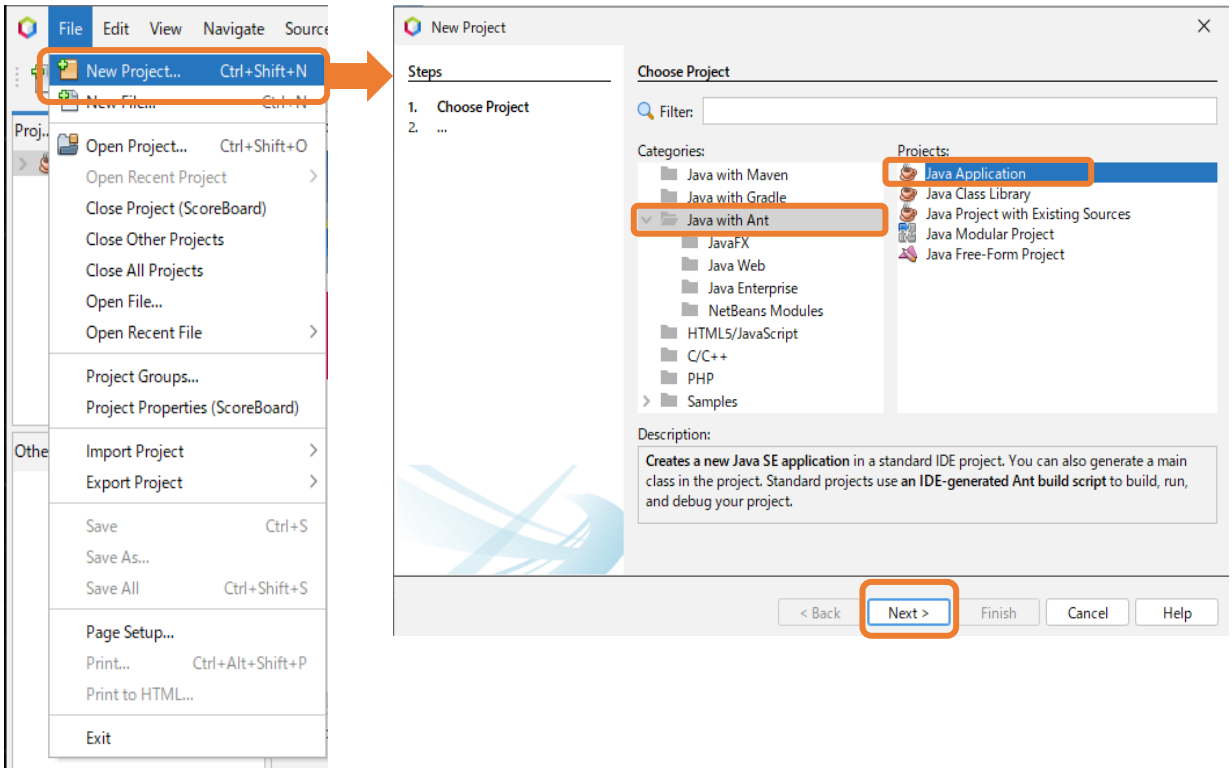
手順 6：JDK の設定

手順 7：プロジェクトをビルドする

手順 8：Jar ファイルが作成できたことを確認

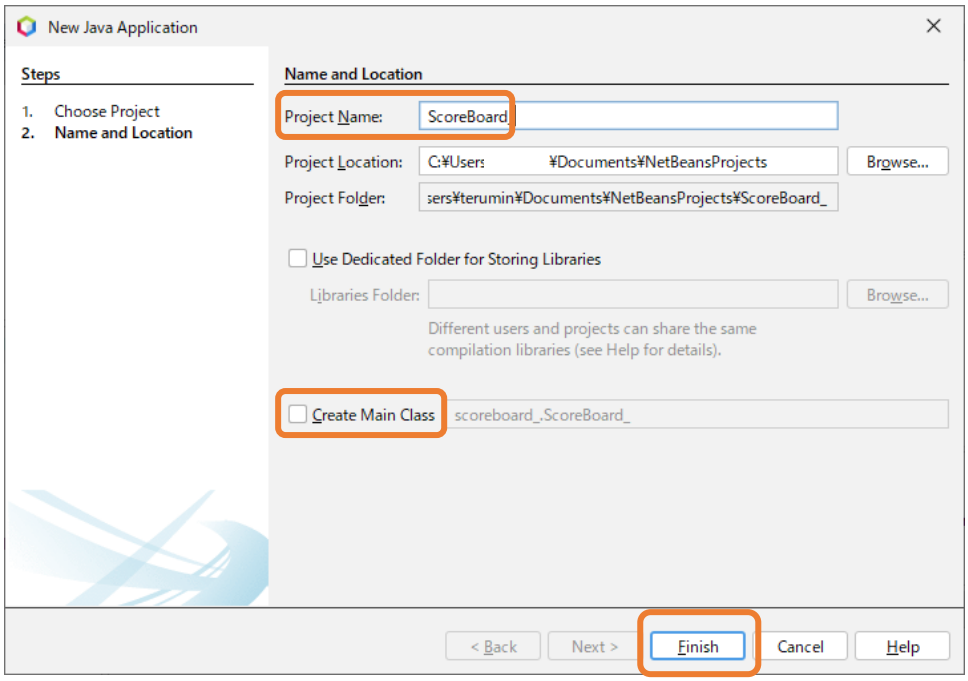
手順 1： NetBeans を起動

「File」→「New Project」 で新しいプロジェクトを作成します。



プロジェクト名(ここでは ScoreBoard)を入力、Create Main Class のチェックは外し

Finish を押下します。

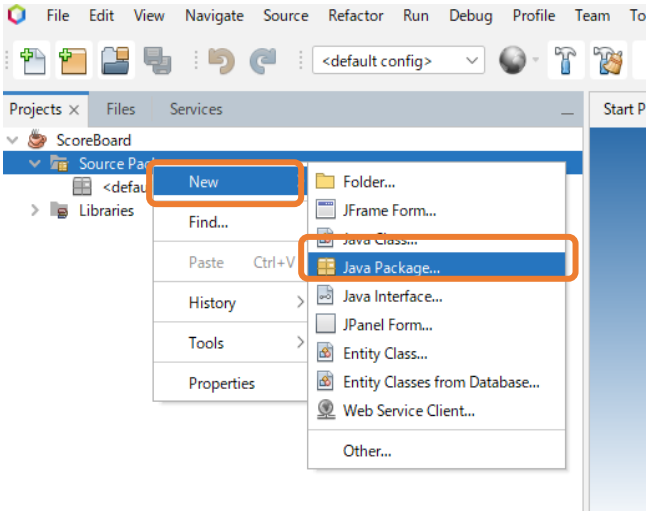


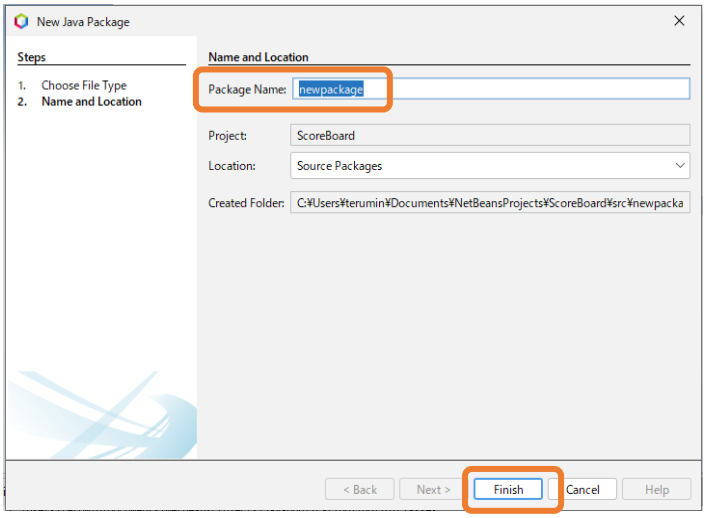
手順 2：パッケージとクラスを作成

下図の通り「Source Package」の上で右クリック →「New」→「Java Package」を作成します。

Package 名は任意の名前で作成が可能です。

（Package は複数作る事ができますが、ここではまず 1 つ作成します）

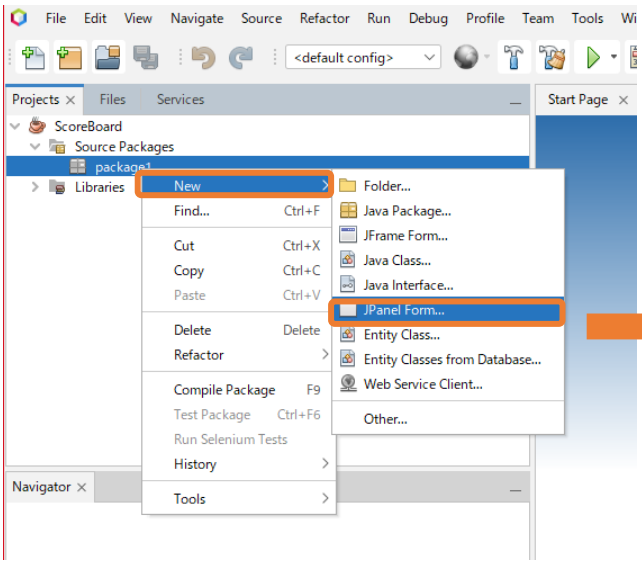


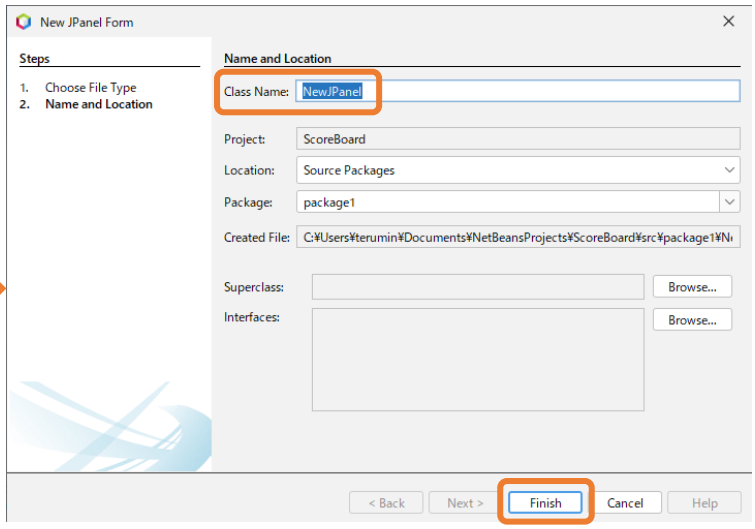


次に、名前を付けたパッケージを右クリックして、「New」→「JPanel Form」をクリック。

分かりやすい Class Name（クラス名）を決め、Finish を押下し JPanel クラスを作成します。

ここで決めたクラス名は、以下手順 3 でも入力します。





手順 3：パネルの name にクラス名を入力

下図の左側の四角は、初めから作成されているパネルで、この上に部品を並べていきます。

まずはこのパネルを選択した状態で、JPanel の「プロパティ」内、「name」欄にクラス名を入力します。

任意の名前ではなく、手順 2 で決めたクラス名と一字一句同じように入力してください。

（ここでは「TopPanel」と入力しています。）

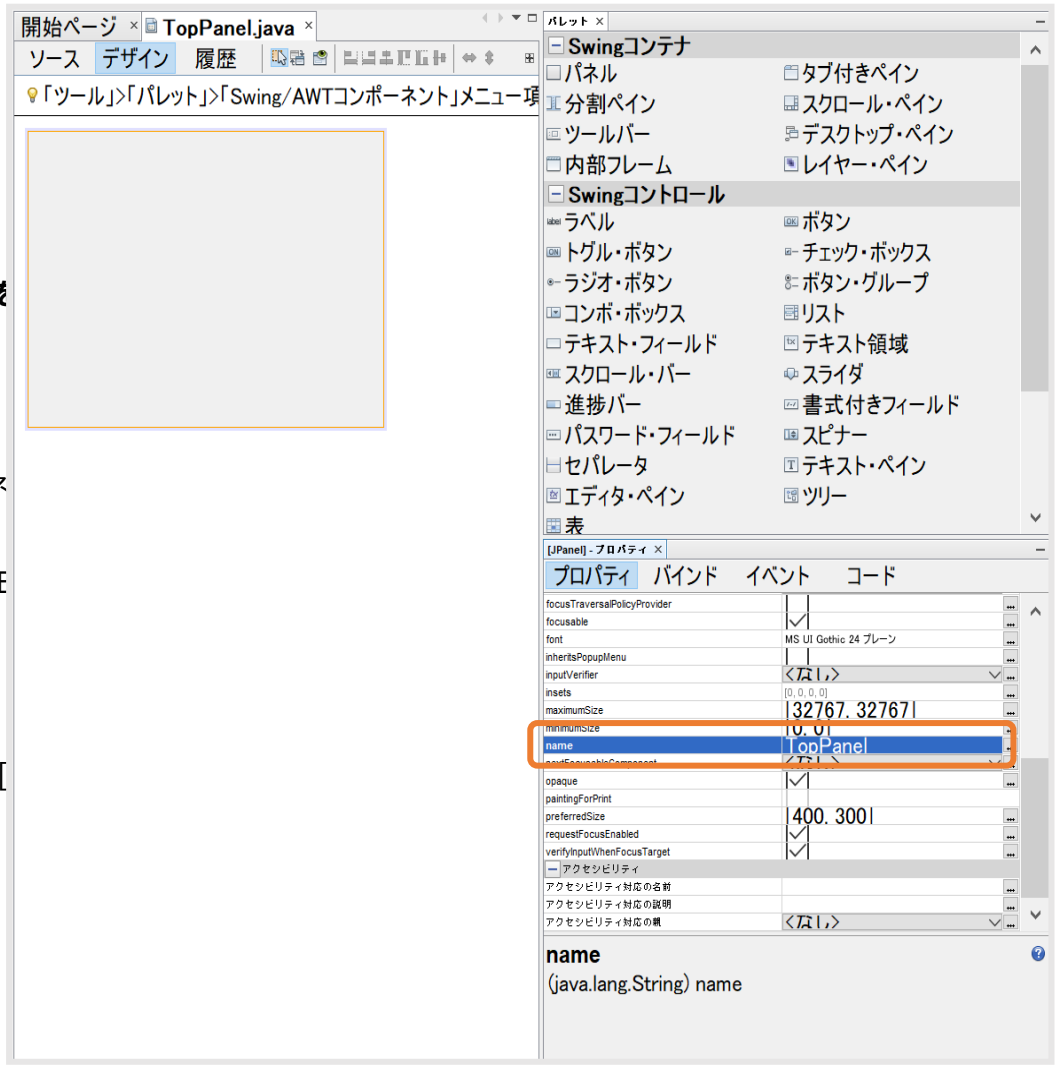
※下図では事前に NetBeans の日本語化を行っております。

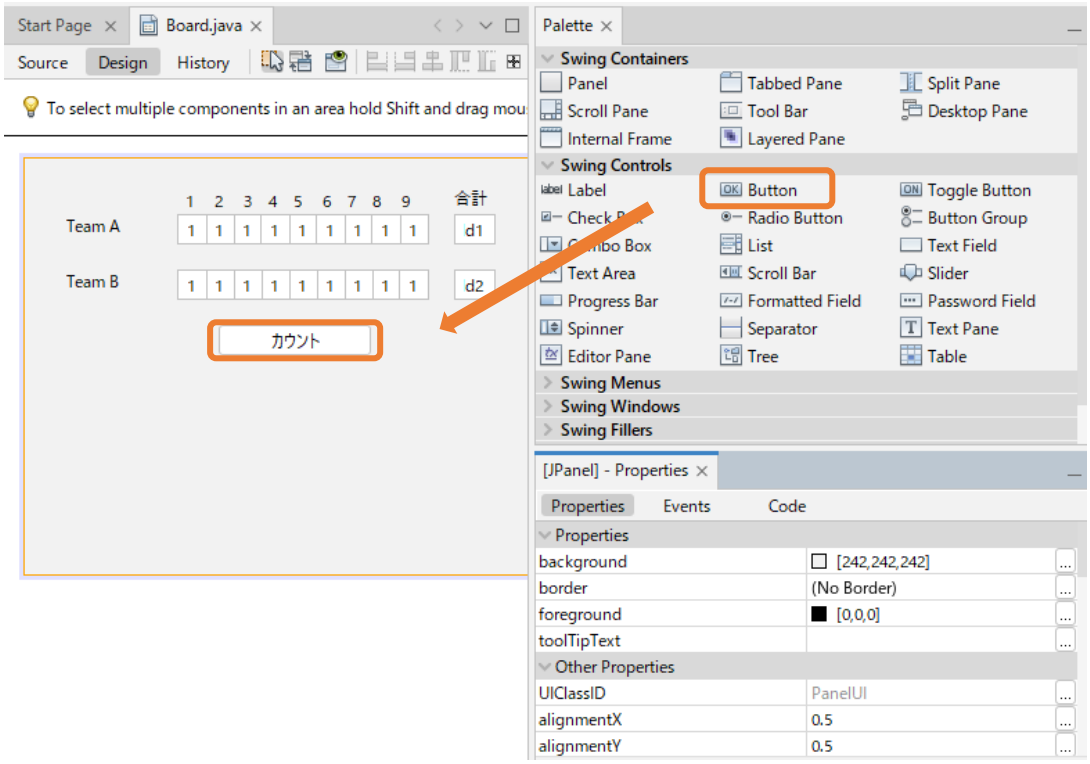
手順 4：画面に部品を

画面左のパネ

下の例では「E

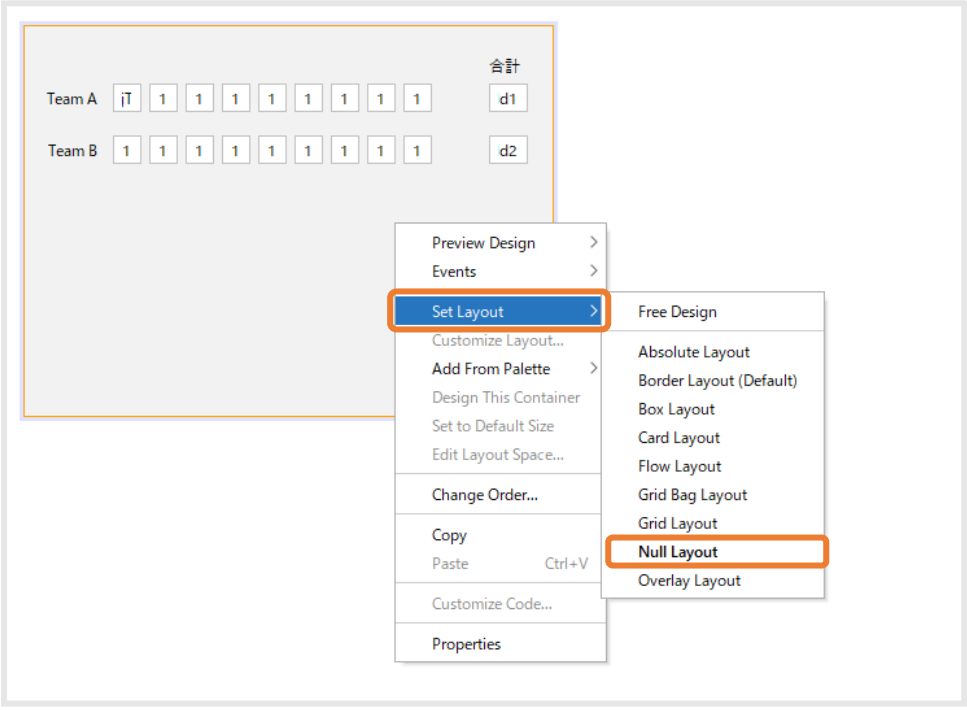
また、右下の[



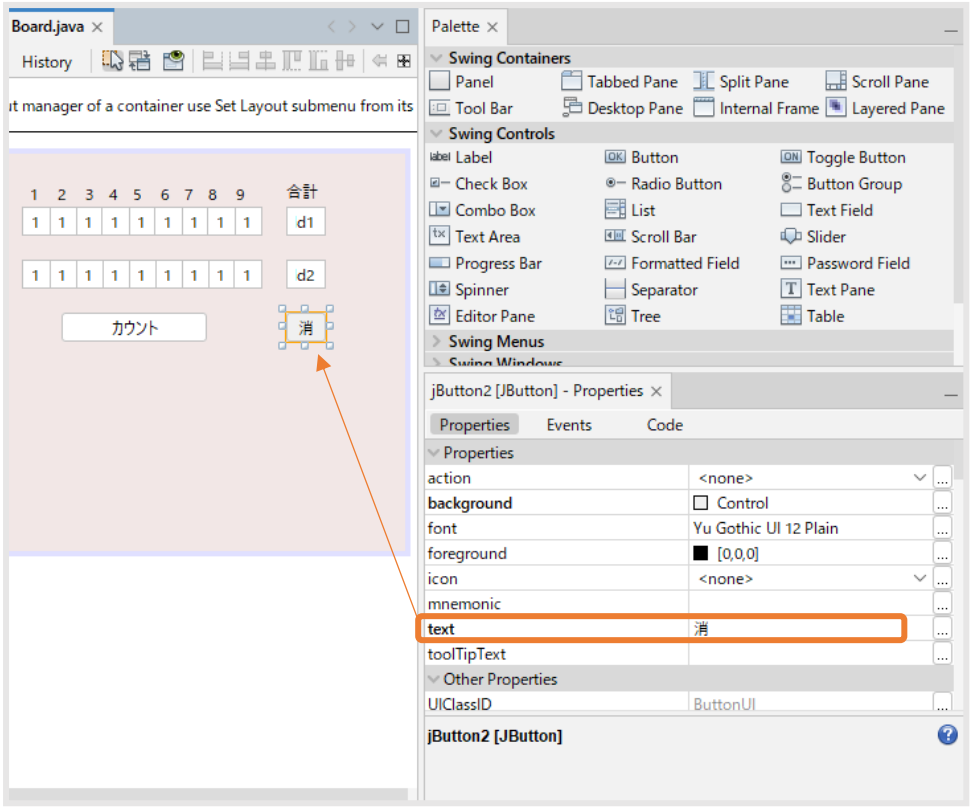


※レイアウト形式について、今回は「Null Layout」で作成します。

画面左のパネル部分を右クリック→「Set Layout」 →「Null Layout」 を選択します。



表示されるボタン上の文字は「Properties」の「text」欄へ入力すると変更可能です。



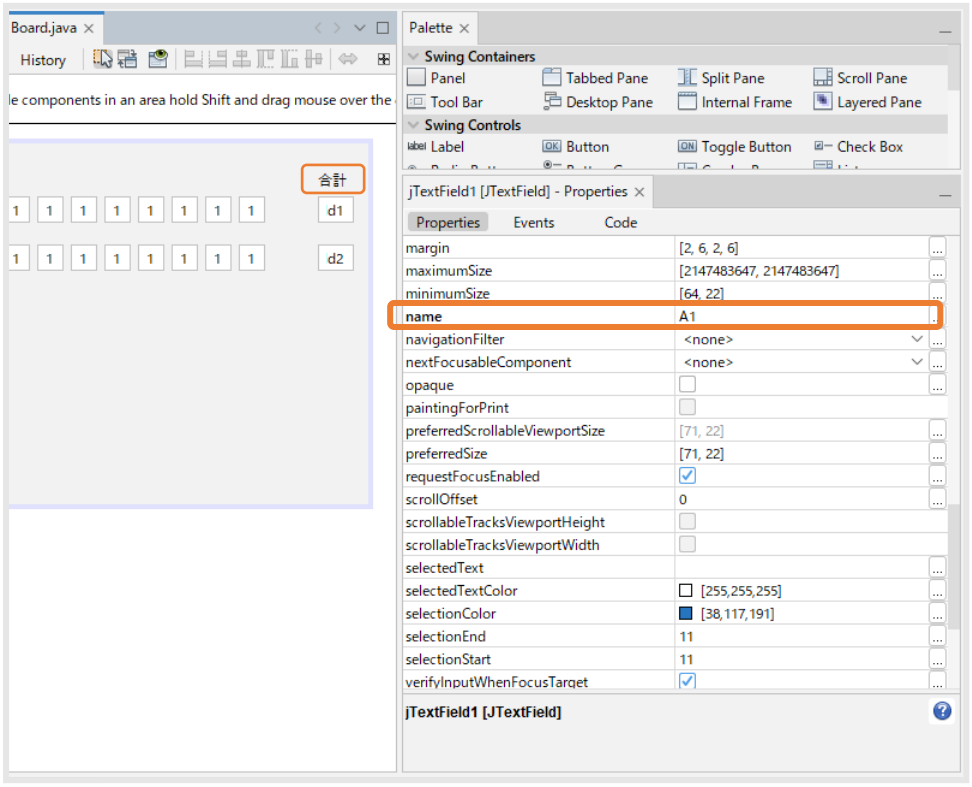
手順 5：AiSee で操作する全てのコンポーネント(ボタン等の部品)の名前を設定

部品を並べ終わったら、コンポーネントをひとつずつ選択して「name」欄に名称を入力します。

例えばボタンなど、後でシナリオを付ける部品は必ず「name」の入力が必要となります。

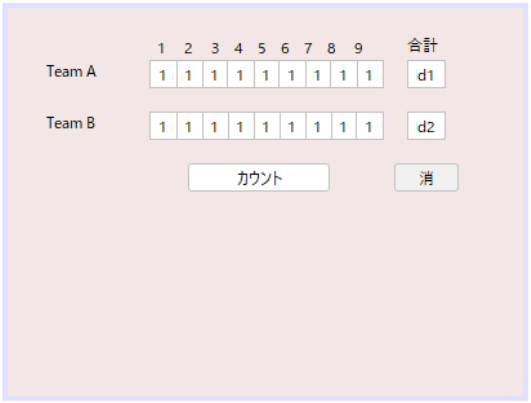
図左の「合計」のように、表示するだけで他に機能がない部品については「name」の入力は任意です。

下図では部品に「A1」という名前を付けています。入力後、Enter を押して変更を確定させてください。



（左図）NetBeans 上のアプリ画面作成例

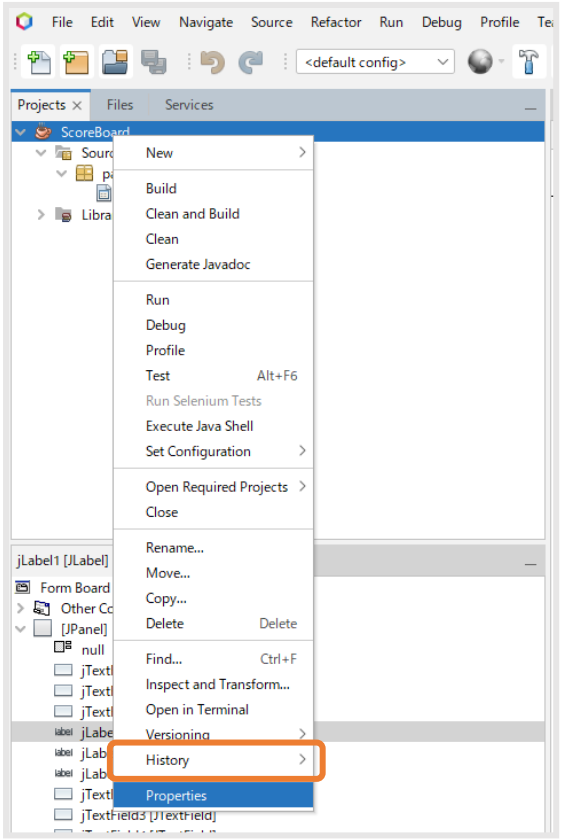
（右図）出力した Jar ファイルを取り込み、AiSee 上で表示させたもの



手順６：JDK の設定

次に JDK 設定を行います。

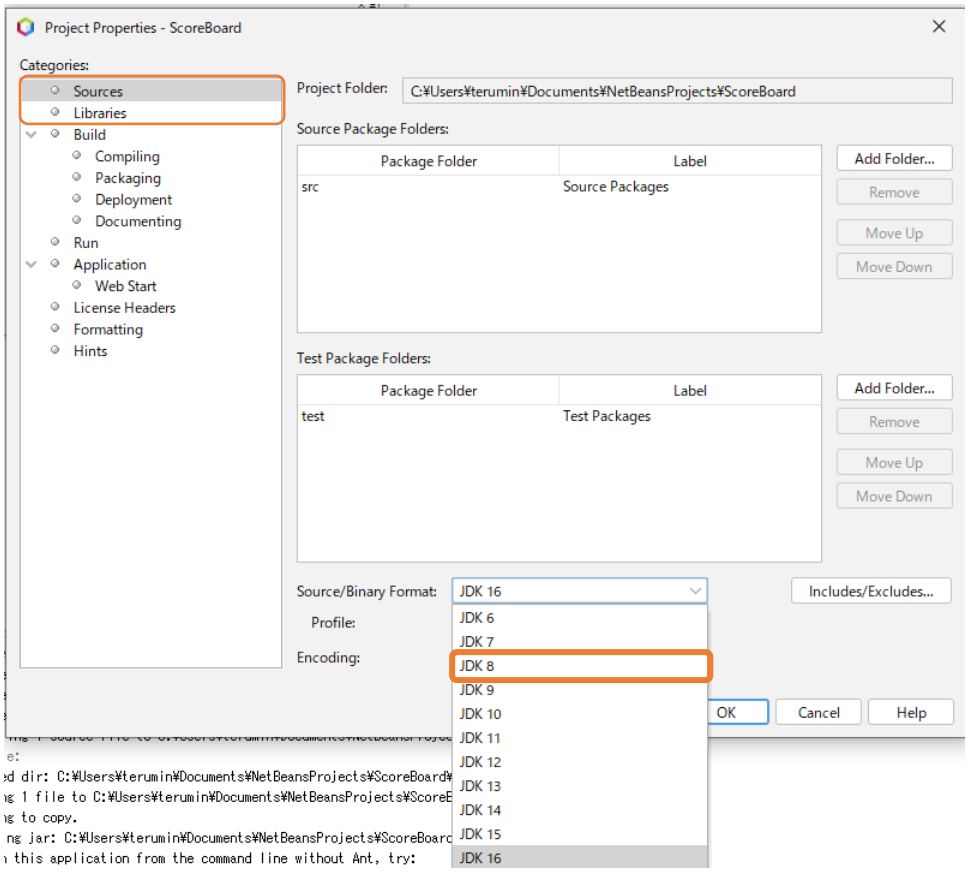
プロジェクト名の上で右クリック → プロパティ（Properties）を選択



下の画面左の「Sources」と「Libraries」 2 項目で設定が必要になります。

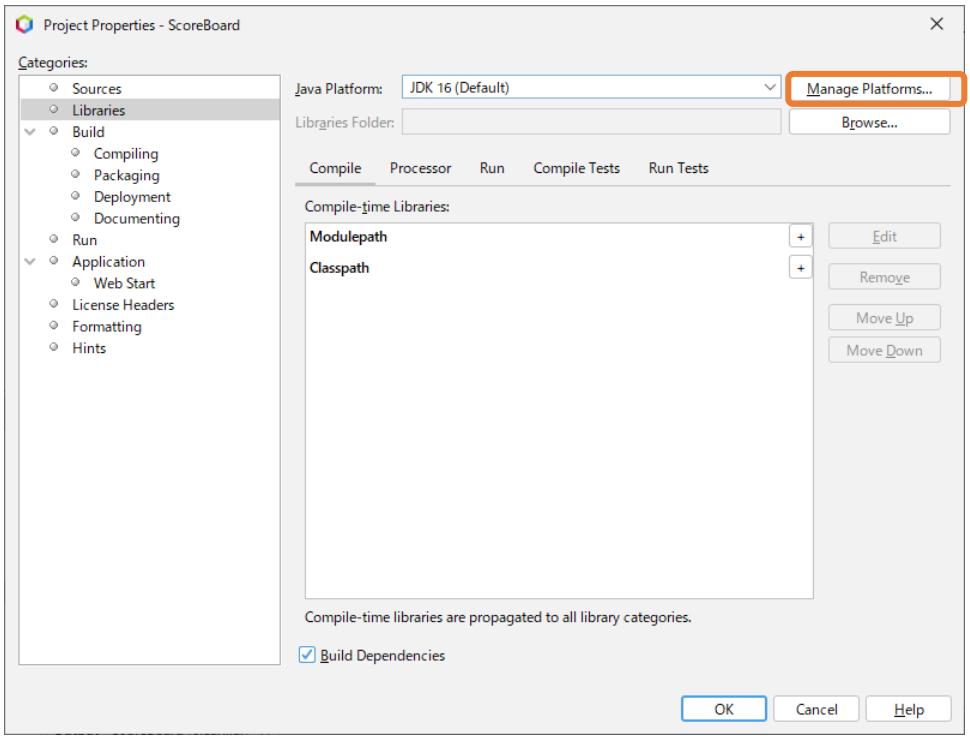
まずは「Sources」の中の「Source/Binary Format」のプルダウンで JDK8 を選択します。

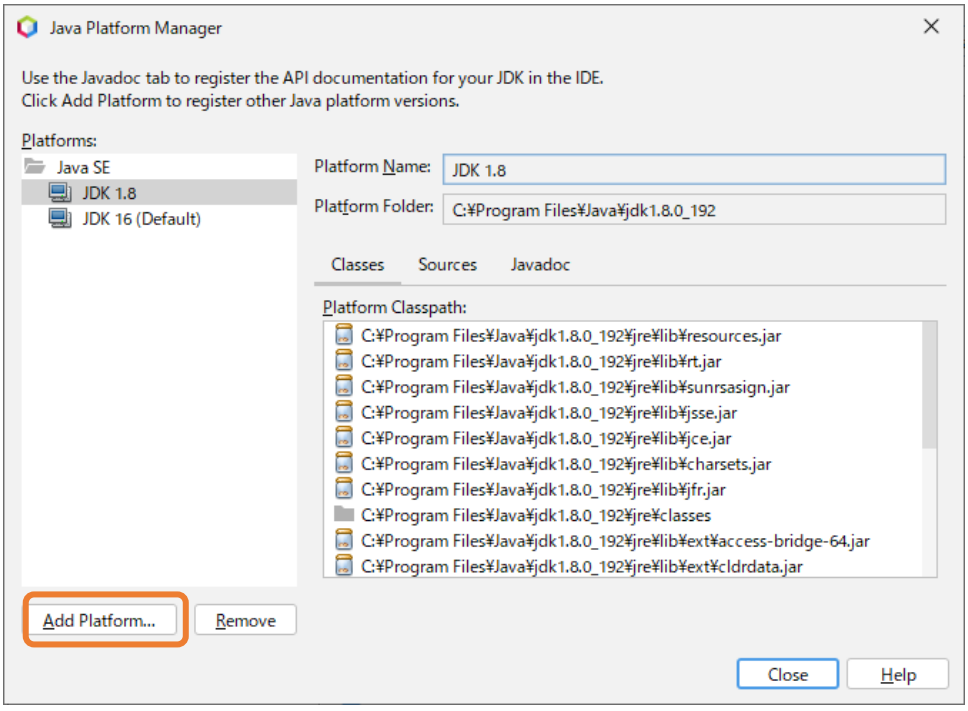
※元々JDK8 が選択されている場合には手順 6 の操作は不要です。手順 7 に進んでください。



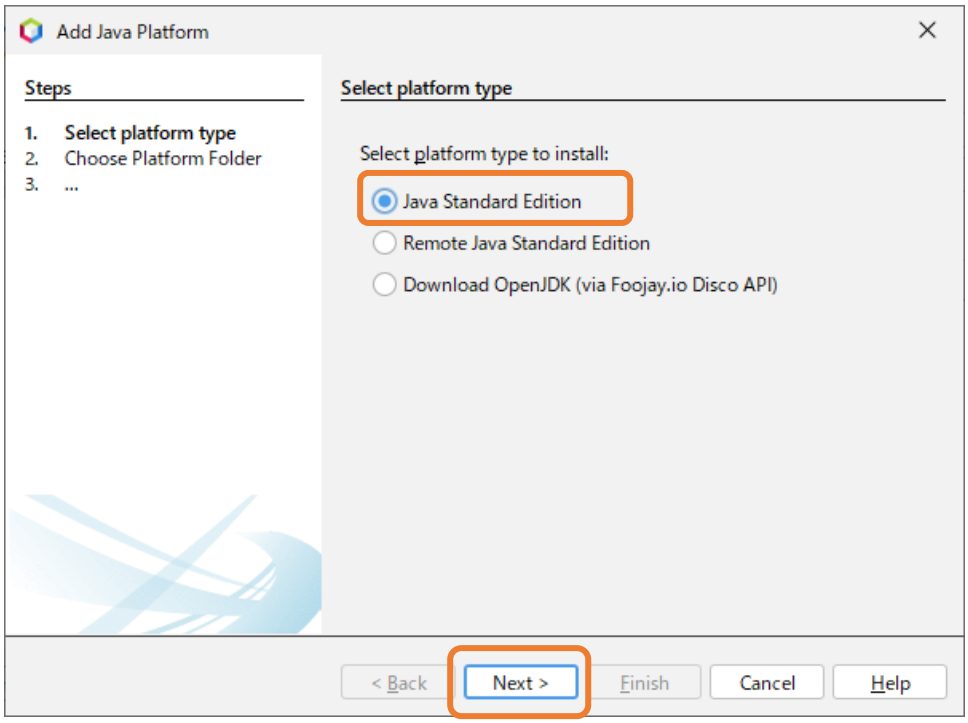
次に「Libraries」を選択し「Manage Platformes...（プラットフォームの管理）」をクリック、

次画面で「Add Platform」をクリックします。

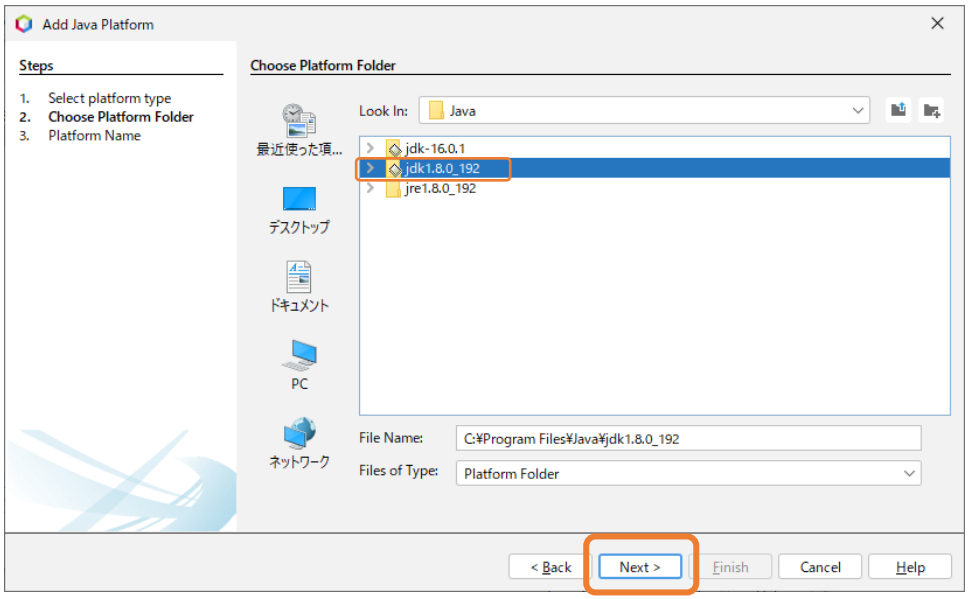




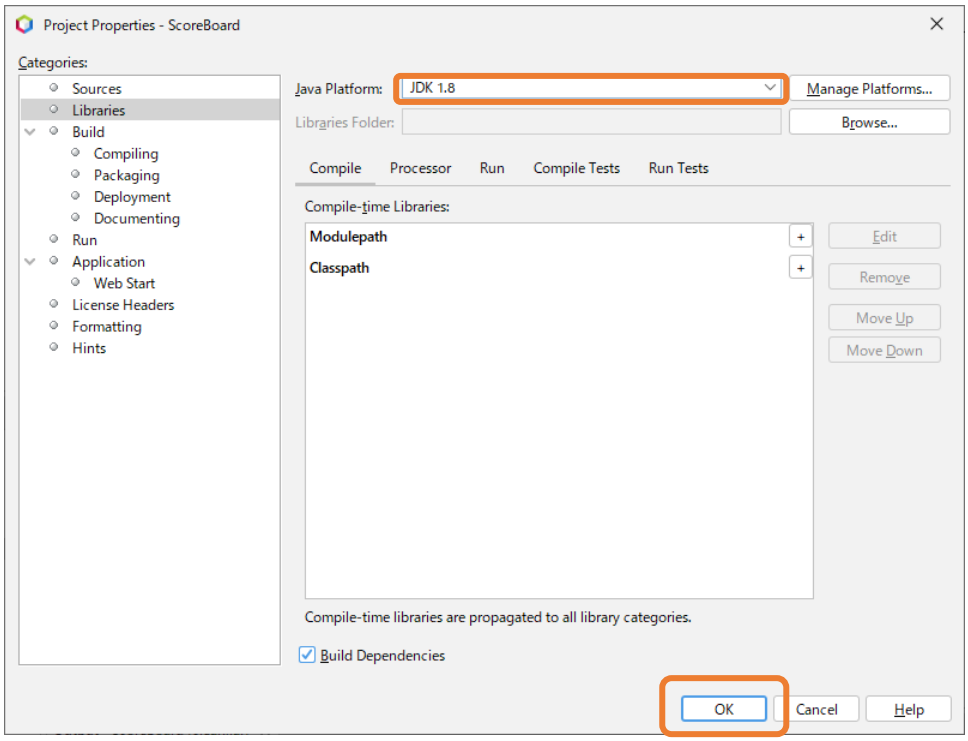
「Java Standard Edition」を選択し Next をクリックします。



「jdk1.8.0_192」を選び、Next → 次の画面で Finish を押します。



「Libraries」画面に戻ると、Java Plattforme　で「JDK1.8」が選べるようになっているので、「JDK1.8」を選択し OK を押します。JDK の設定は終了です。※JDK8 と 1.8 は同じものです。

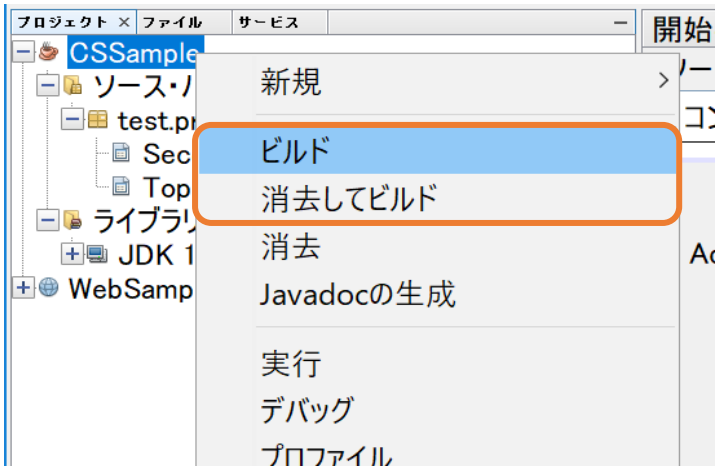


手順 7：プロジェクトをビルドする

画面左上のプロジェクト名の上で右クリックし「ビルド(Build)」を押します。

ビルドが成功したことを確認してください。

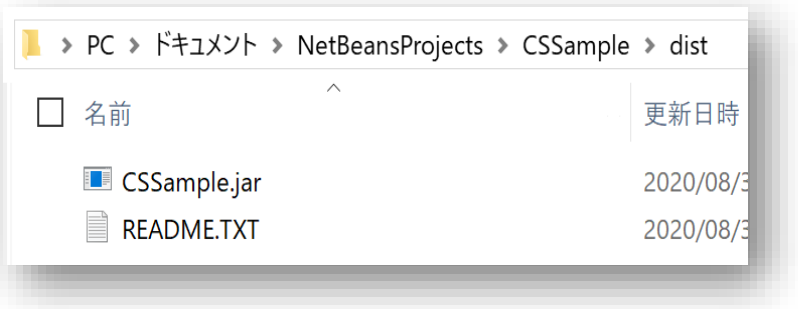
※既に 1 度でもビルドしたことがある場合は「消去してビルド(Clean and Build)」を推奨します。



手順 8：Jar ファイルが作成できたことを確認

エクスプローラー→「ドキュメント」→「NetBeansProjects」フォルダ→ 作成したプロジェクト下の「dist」フォルダ内に「〇〇〇.jar」ファイルが作成されていることを確認します。

この Jar ファイルを 3.3.1 章の【図 2】【図 3】の手順で、AiSee で使用できるように取り込みます。



補足：Jar ファイルを AiSee に取り込み、ボタン等にシナリオをセットした後に画面の修正やカスタマイズが必要となった場合には、AiSee で作成中のプロジェクトの「画面定義」下のファイルと、「ライブラリ定義」下の Jar ファイルの 2 点を削除し、改めて更新した Jar ファイルを取り込めば問題なく作成済みのシナリオが適用されます。

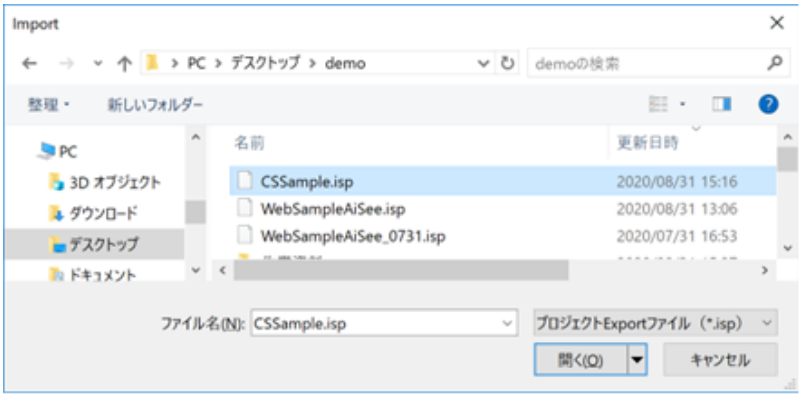
7　既存プロジェクトをインポートする方法

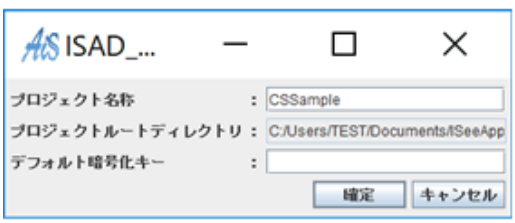
AiSee のプロジェクトファイルは、「〇〇〇.isp」という拡張子のファイルです。
この章では既存の「〇〇〇.isp」ファイルをインポートする方法を説明します。

1：「ISeeProjectInformation」を右クリックし、「プロジェクトのインポート」をクリックする。

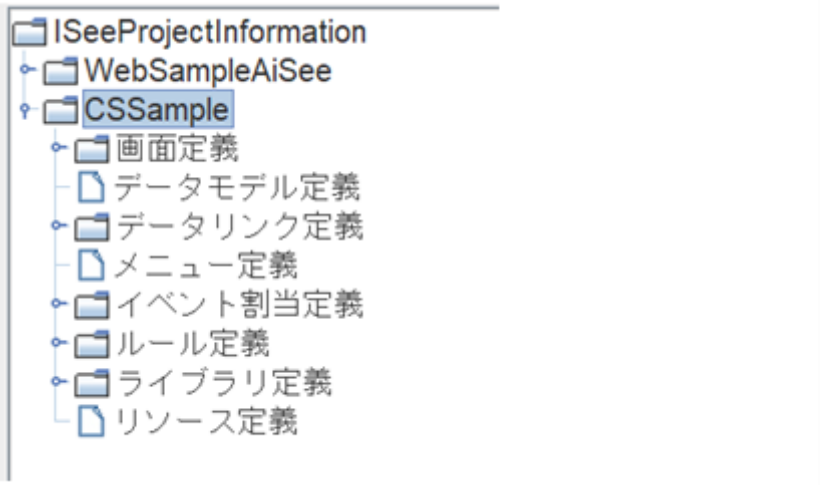
2：対象の Isp ファイルを選択する。

※本資料と同列の「AiSeeProject」フォルダ内にサンプルファイル（CSSample.isp）があります。





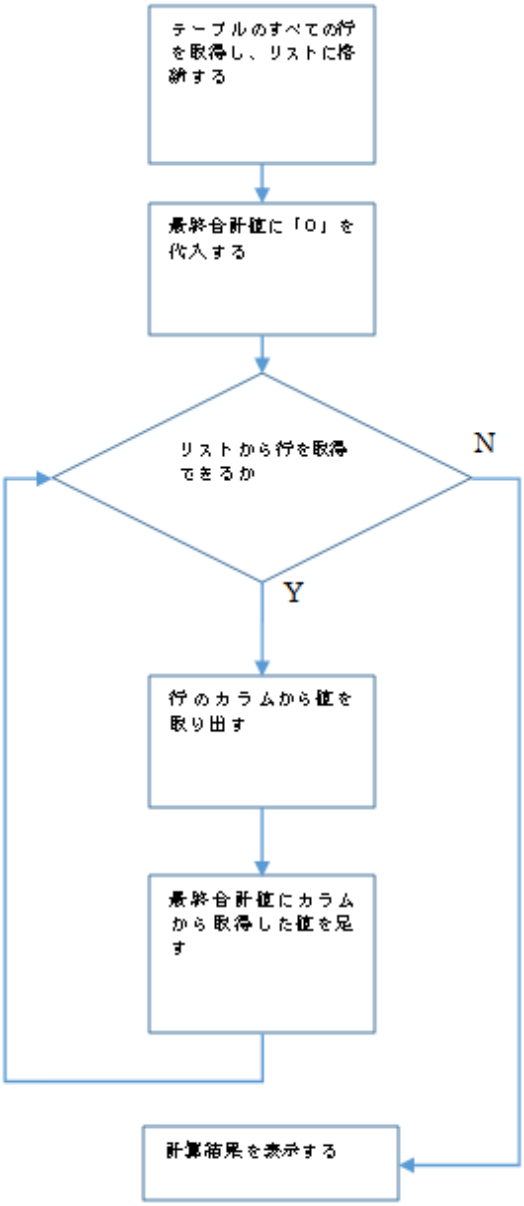
３：インポートされたことが確認できたら完了です。

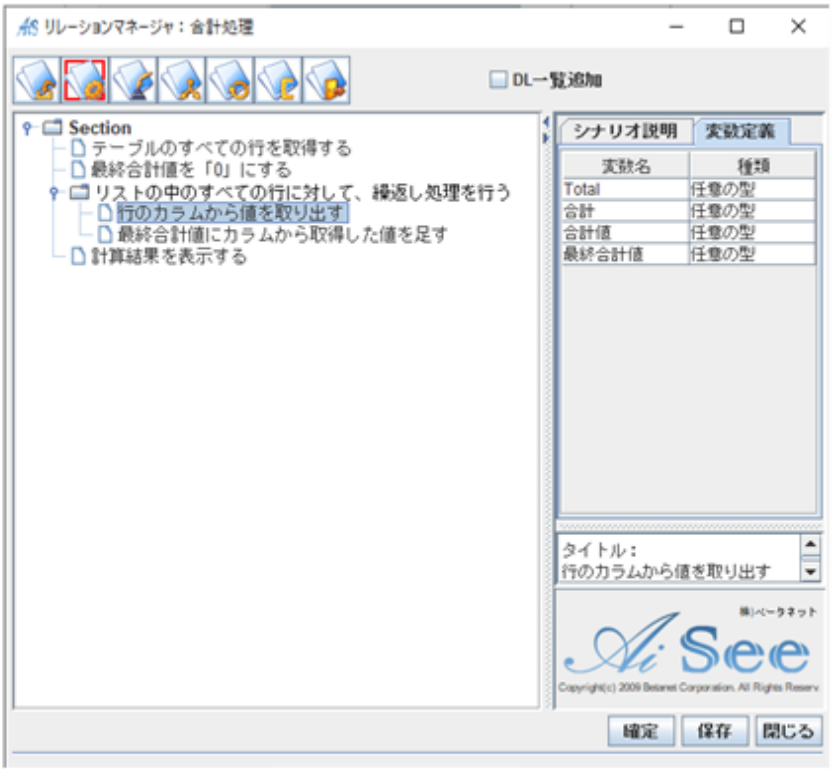


８　繰り返し(ループ)処理の説明

例１：

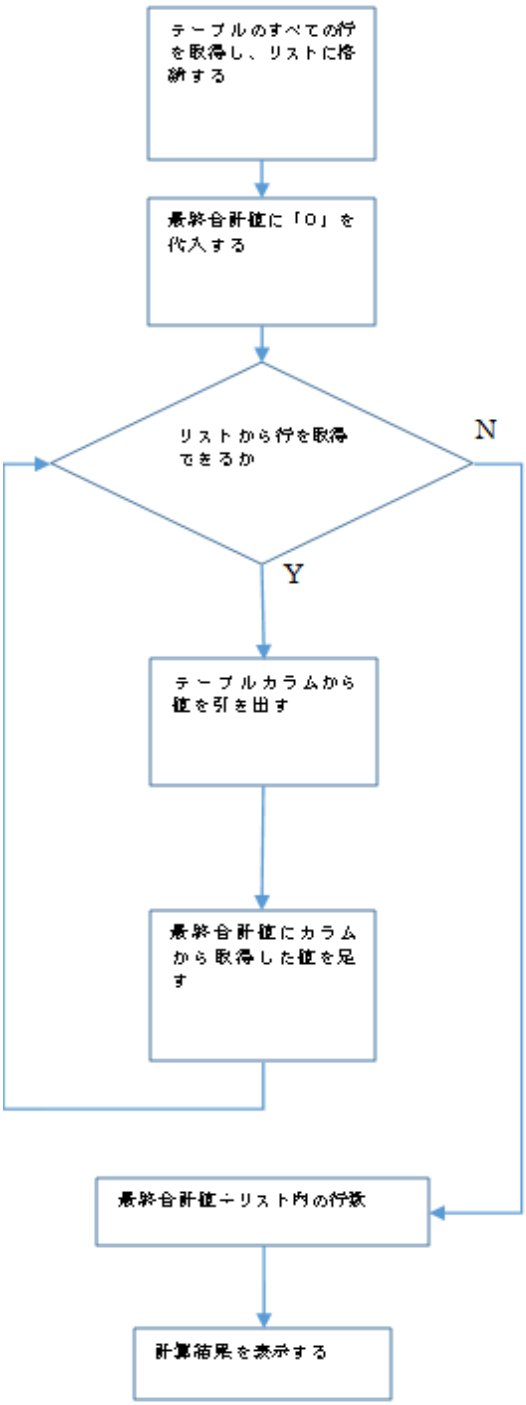
「結果」の合計を求めたい場合のループ処理を流れ図（フローチャート）で表すと以下のようになります。

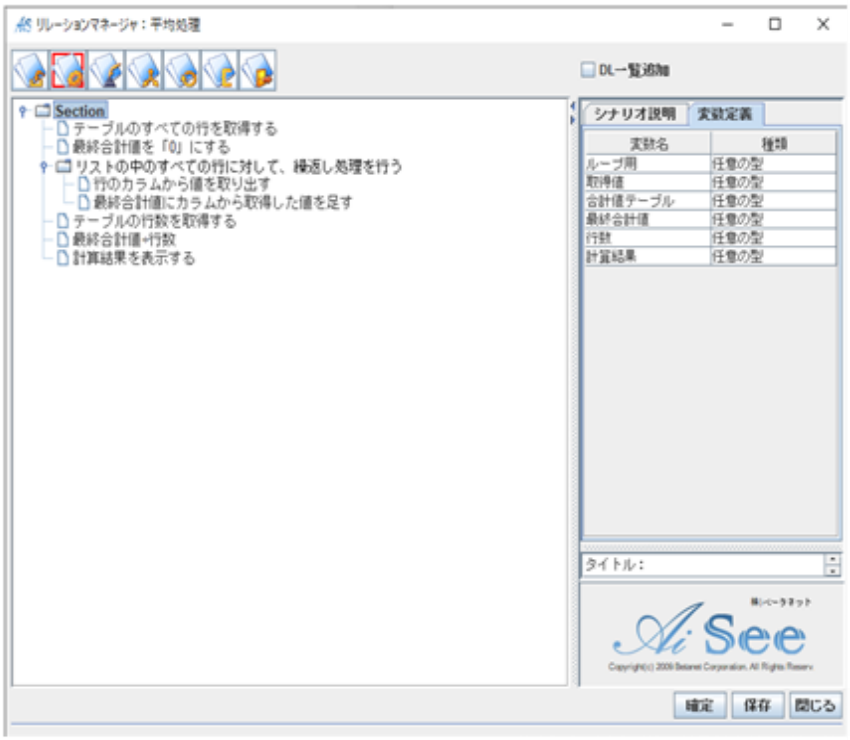

テーブルのすべての行を取得し、リストに格納する
最終合計値に「0」を代入する
リストから行を取得できるか
N
Y
行のカラムから値を取り出す
最終合計値にカラムから取得した値を足す
計算結果を表示する


リレーションマネージャ：合計処理
DL一覧追加
Section
テーブルのすべての行を取得する
最終合計値を「0」にする
リストの中のすべての行に対して、繰返し処理を行う
行のカラムから値を取り出す
最終合計値にカラムから取得した値を足す
計算結果を表示する
シナリオ説明
変数定義
変数名
種類
Total
任意の型
合計
任意の型
合計値
任意の型
最終合計値
任意の型
タイトル：
行のカラムから値を取り出す
AiSee
確定
保存
閉じる

例２：

「結果」の平均値を求めたい場合のループ処理を流れ図（フローチャート）で表すと以下のようになります。





例３：

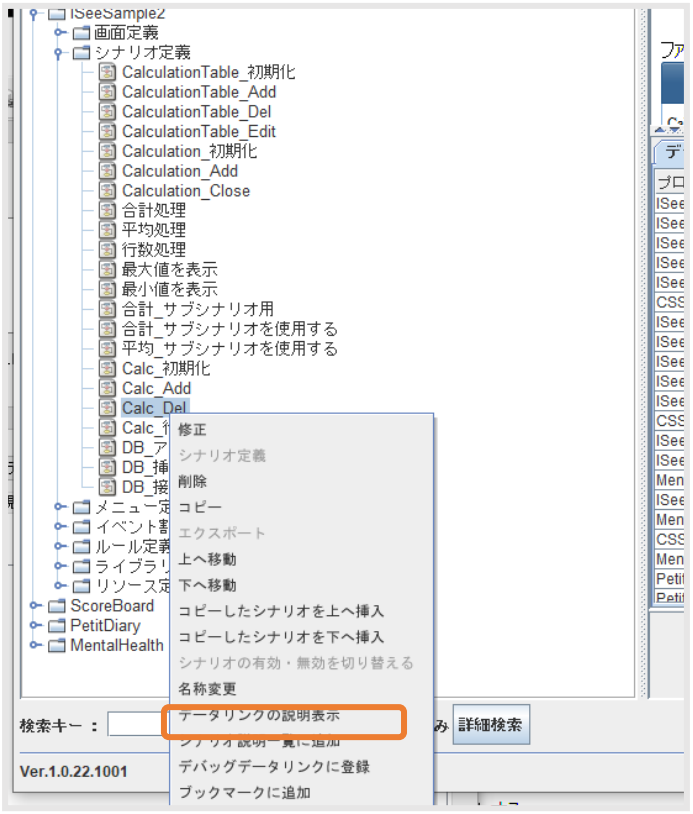
DB を使用して「検索」したい場合のループ処理を流れ図（フローチャート）で表すと以下のようになります。

9　便利な Tips

- **F1 キー**
  プロジェクトマネージャーを表示し、アクティブな状態にしてから「F1 キー」を押すとブラウザが起動しヘルプが表示されます。ショートカットキーの一覧などが確認できます。

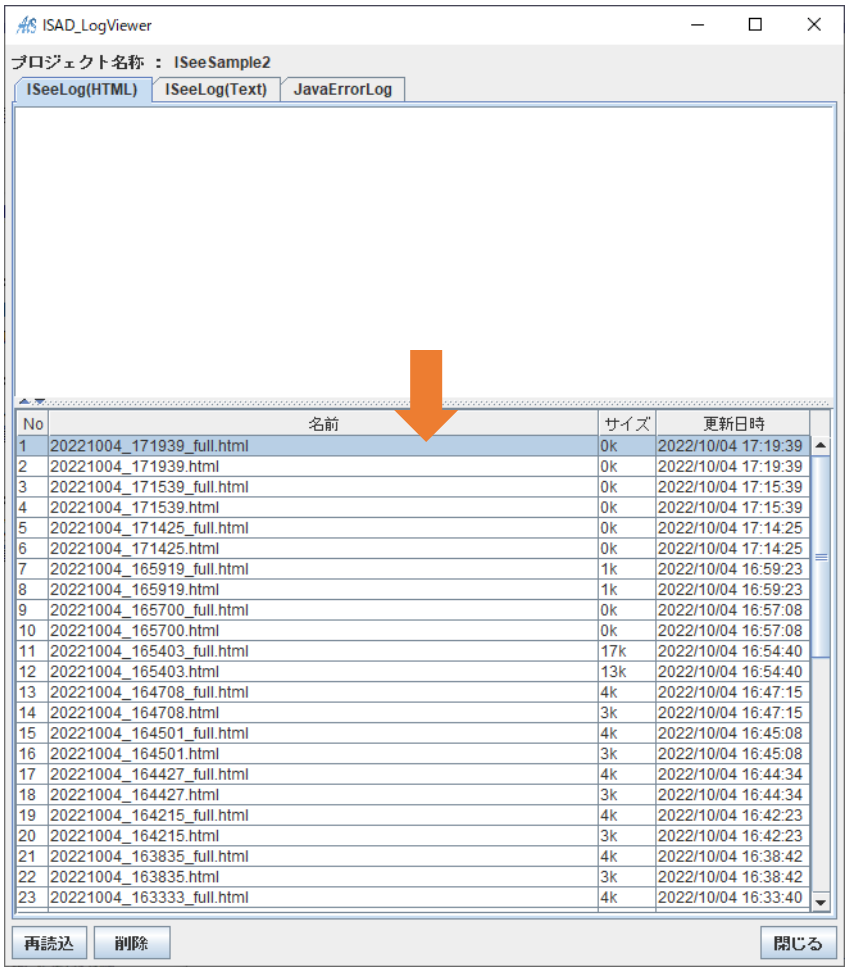
- **F5 キー**
  テストを実行する場合に使います。テスト実行したいプロジェクトのフォルダや、任意のシナリオを選択した状態で「F5 キー」を押すと、簡単にテスト実行ができるショートカットキーです。

- **F4 キー**
  デバックしたい場合に使います。
  まずは、デバックしたいシナリオ上で右クリック→「デバックデータリンクに登録」をクリックします。



次に、実際にテスト実行（F5 キー）してください。

テスト実行が終わったら、デバックしたいプロジェクトのフォルダや、任意のシナリオを
を選択した状態で「F4 キー」を押すと、以下のような画面が表示されます。
矢印の部分をクリックすると、テスト実行結果の細かい内容を確認することができます。
（上が最新のログ、下が古いログです）

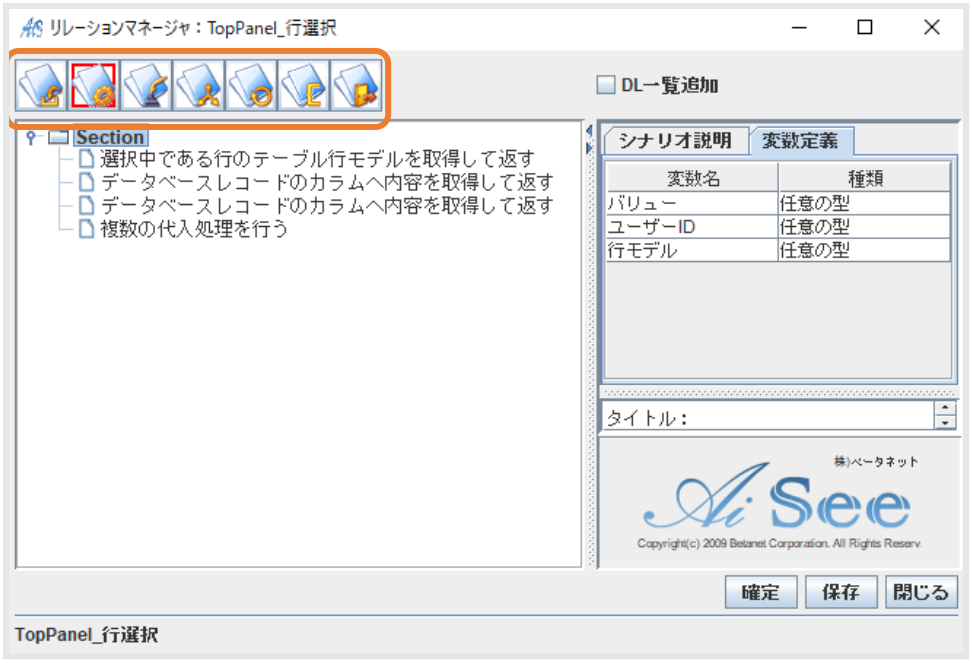


デバック登録したシナリオを解除したい場合や、どのシナリオを登録したか確認したい場合は「シナリオ定義」の上で右クリックし、「デバックデータリンク一覧表示」を開くと確認や編集が可能です。

※データリンク＝シナリオを指します。

- **シナリオ作成画面のアイコン**

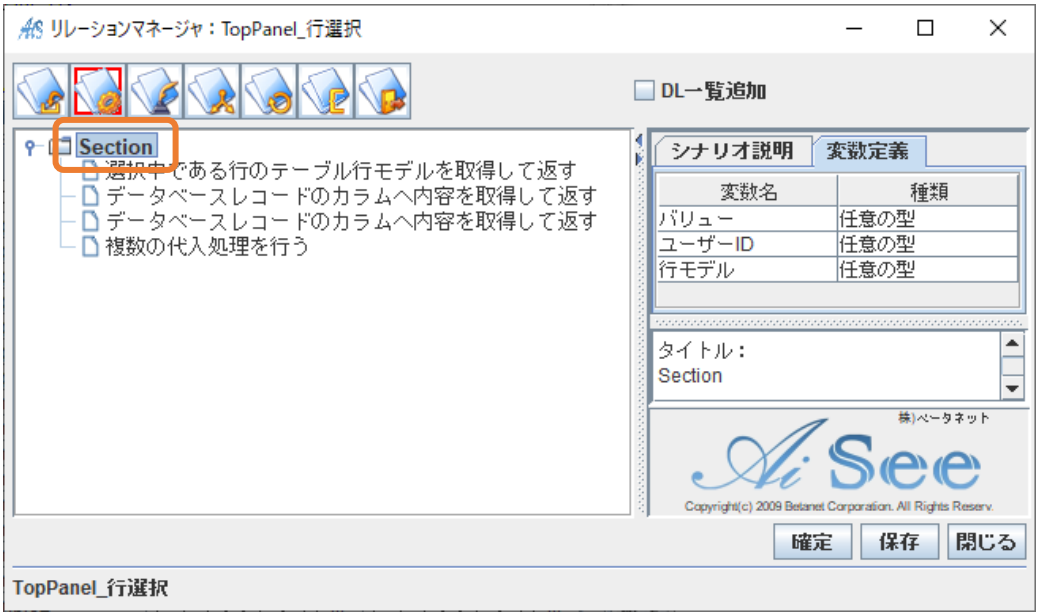
シナリオを作成する際に、通常は「Section」を右クリックして作成しますが、「Section」の上にあるアイコンからもリレーションやサブシナリオを新規作成する事ができます。 (下図)



10 Q&A

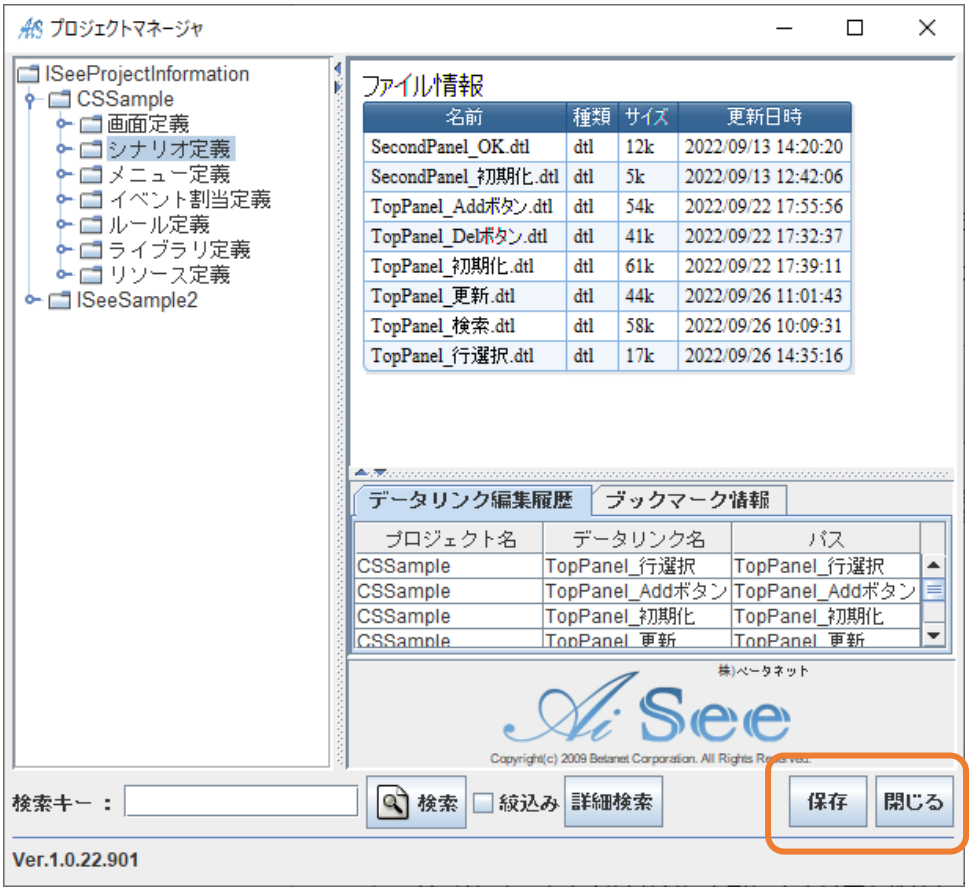
- **Q. シナリオ内にリレーションを作成したいが、何度「登録ボタン」を押しても反映されず、作成ができません。**

A. リレーション作成前に、下図の「Section」の部分を左クリックで選択してください。
Section の周囲に細い青枠が表示されたことを確認してから、リレーションの新規作成を行ってください。(下図)



♦ **Q.「Section」をクリックしてからシナリオを作成していますが、それでも作成できません。**

A. 念のため、AiSee の再起動をお試しください。リレーションマネージャ画面を閉じて、プロジェクトを保存してから、「閉じる」で終了し再起動してください。 (下図)

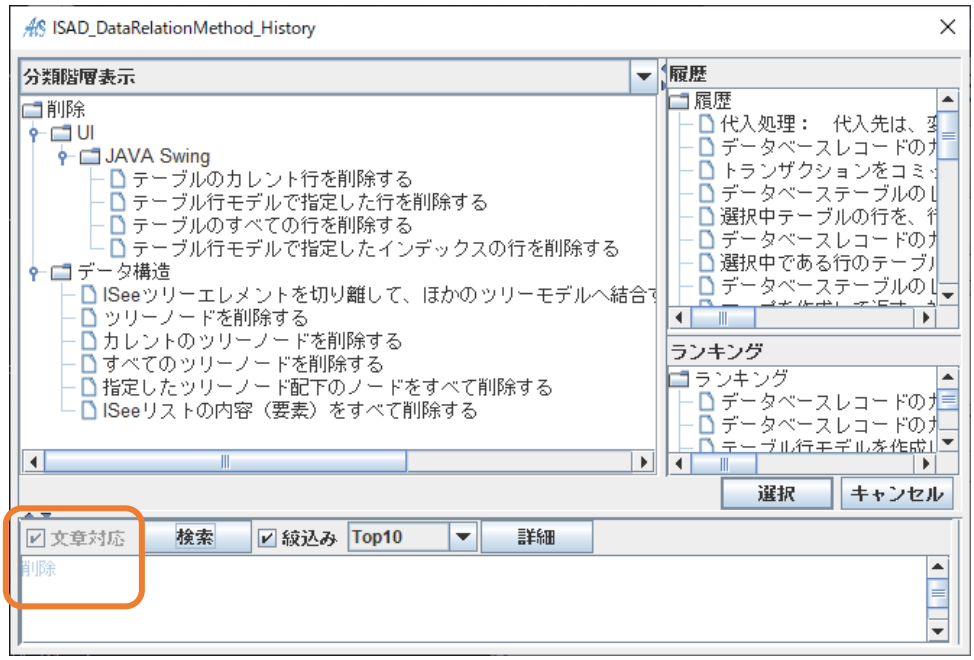


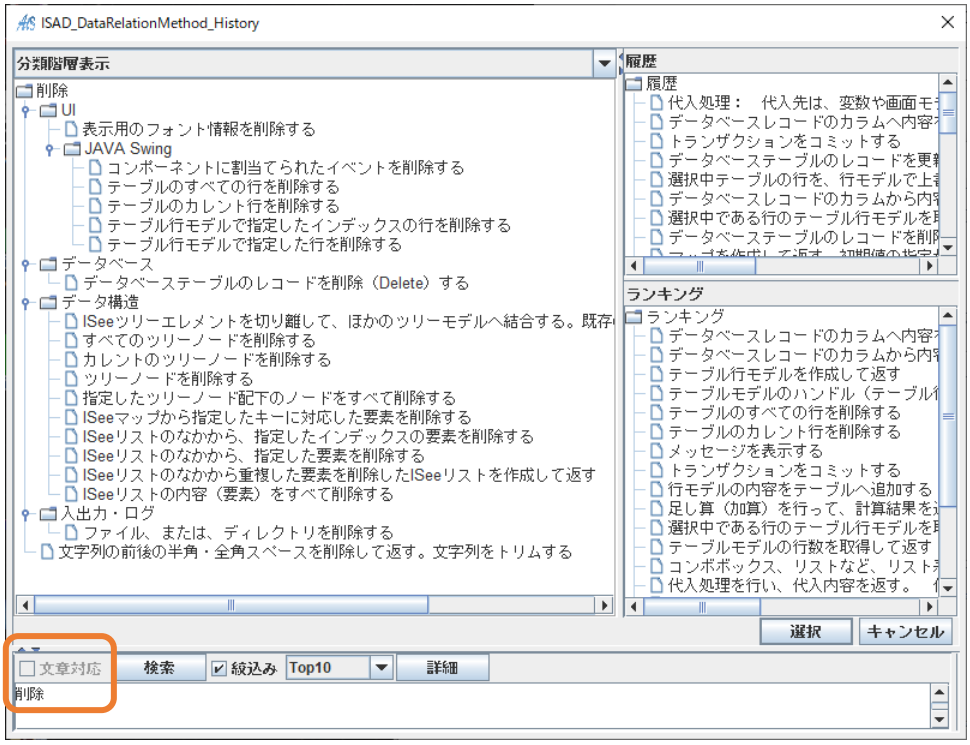
♦ **Q. シナリオの検索がうまくいきません。**

A.下図のように「文章対応」のチェックを外して検索をお試しください。
「文章対応」にチェックがあると、あいまい検索も可能であるため、基本的にはチェック有での検索を推奨いたしますが、チェックの有無で検索結果が異なる場合がございます。

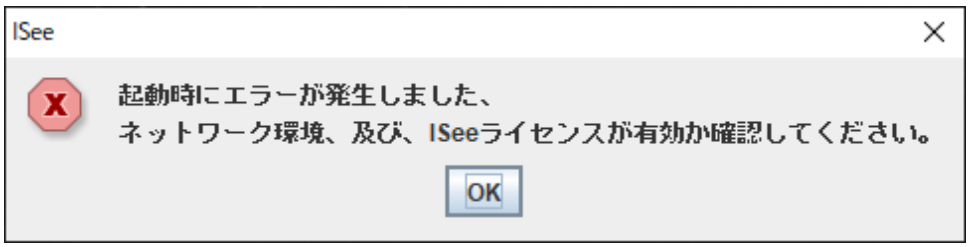
併せて、例えば「データを削除する」と検索されている場合には「データ　削除」など
キーワードのみでの検索もお試しください。 (下図)

また、複数回の検索を試されたい場合には、一度検索した後に「絞り込み」のチェックを外せば再度検索ができる状態になります。





♦ **Q.　テスト実行したら、このようなエラーが出てしまいました。**



A1. テスト実行前に、起動設定の見直しを行ってください。
特に、プロジェクトフォルダの上で右クリック→「起動設定」→ 画面中央の「画面 ID」の部分が
正しく設定されているかどうかご確認ください。5.3.1 章【図 17】【図 18】をご参照ください。

A2. もし上記の方法で解決できなかった場合は、一度プロジェクトをエクスポートして、新たなプロジェクトとしてインポートしてください。

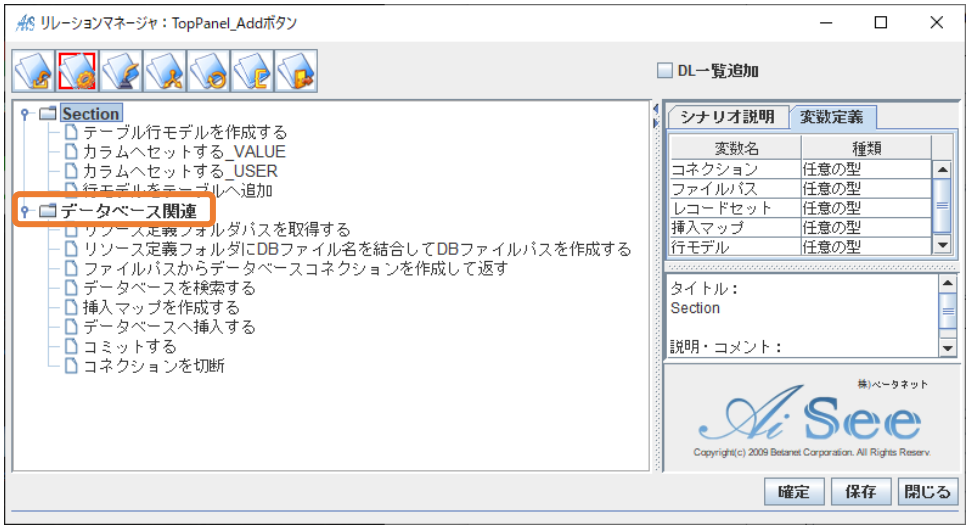
♦ **Q.　テスト実行（F5）を押しましたが、何の反応もありません。**

A.起動設定も正しいのに、テスト実行ができなくなってしまった場合、一度 AiSee を
終了し、念のため PC の再起動も行い、再度 AiSee を起動してテスト実行をお試しください。

改善する場合があります。

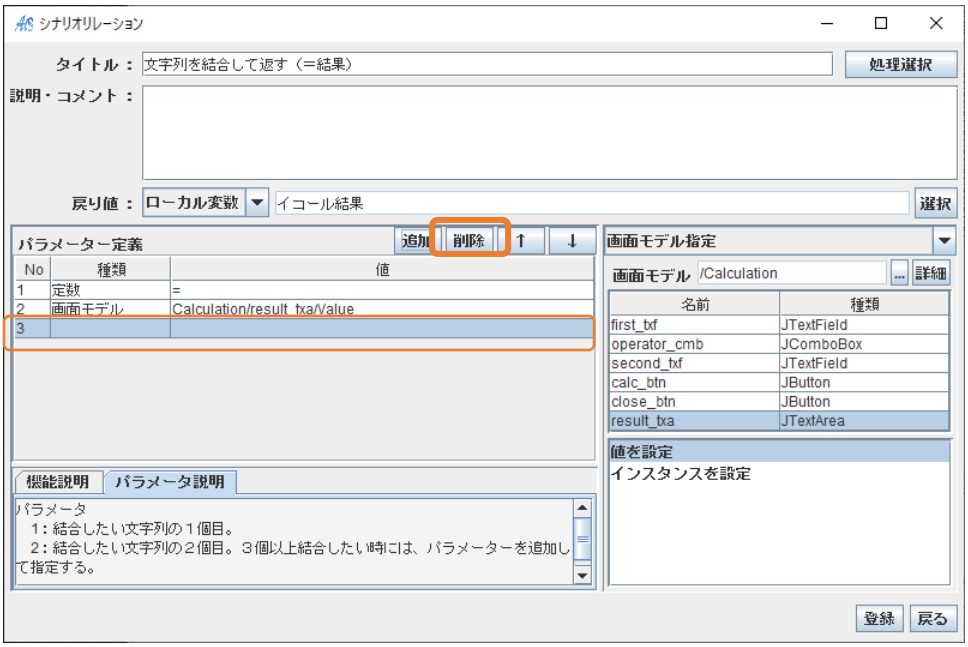
- **Q.　いくつかのシナリオをフォルダ分けする方法はありますか？**

A. シナリオを複数のセクションに分ける事ができます。
「Section」の上で右クリック→「セクションの作成（下へ）」で、新たなセクションが作成できます。
下図では、「データベース関連」というセクションを作成しました。
作成した新たなセクションの下にシナリオを追加していくと、新たなセクション名の頭にフォルダのマークが現れ、複数のセクションが見やすくなります。 (下図)



- **Q.文字列を結合するシナリオを作成しましたが、実行結果に「null」が余分に入ってしまいます。**

A.シナリオリレーション作成時、不要なパラメータが入ってしまった可能性があります。下図のパラメーター定義の No3 の部分を選択し、削除すれば、null が消えます。



♦ **Q.どこを見ても間違いがないように思うのですが、ほかに探すところはありますか？**

A.定数で不要な改行が入っていたり、漢字の「二」をカタカナの「ニ」などの書き方に間違いがないか一度確認をしてみましょう。

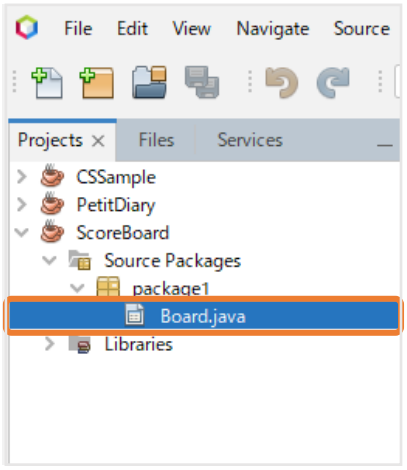
♦ **Q.NetBeans で作成したボタンにシナリオを付けたが、シナリオは正しいのに動かないです。**

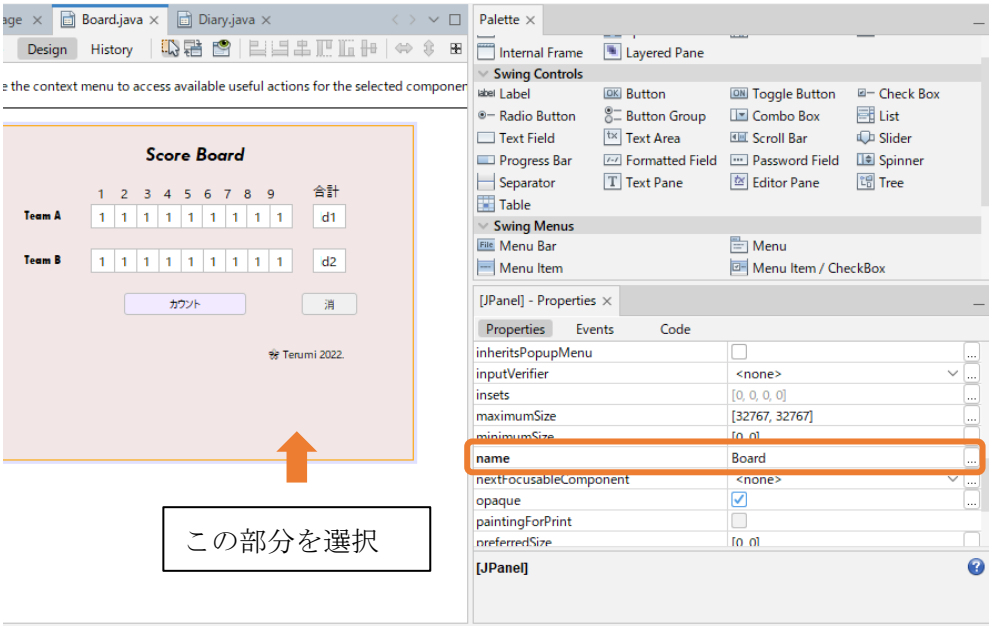
A.作成したボタンなどの部品の名称に、「.（ピリオド）」など使用できない文字が入っていないかご確認ください。ピリオドは消去するか、アンダーバーに変更してから再度お試しください。

♦ **Q.NetBeans で画面を作成し、その後画面を修正したが、AiSee 上で見ると修正が反映しないです。**

A. AiSee の中で、同じクラス名の画面がある場合に競合してしまう可能性があります。
そのため NetBeans 上で、画面の名前を別のものに変更して、上手くいくかお試しください。

下の例であれば、変更部分は Project 下の「Board.java」と画面右下の Properties 内の name にある「Board」です。変更後も、2 つが同じ名前になるよう変更をお願いします。





- **Q.データベースにデータを書き込んだはずだが、DB Browser を見ても反映していないです。**

A.DB Browser の最上部に、PC の中のどのデータベースを表示しているのか、場所が表示されています。もしこの表示が、

「 C:¥Users¥ユーザー名¥Documents¥ISeeAppl¥プロジェクト名¥ISeeResources 」（以下、Documents 下）

になっているようであれば、また別の、以下の場所のデータベースを確認し、更新時刻が編集した時刻になっているかご確認ください。

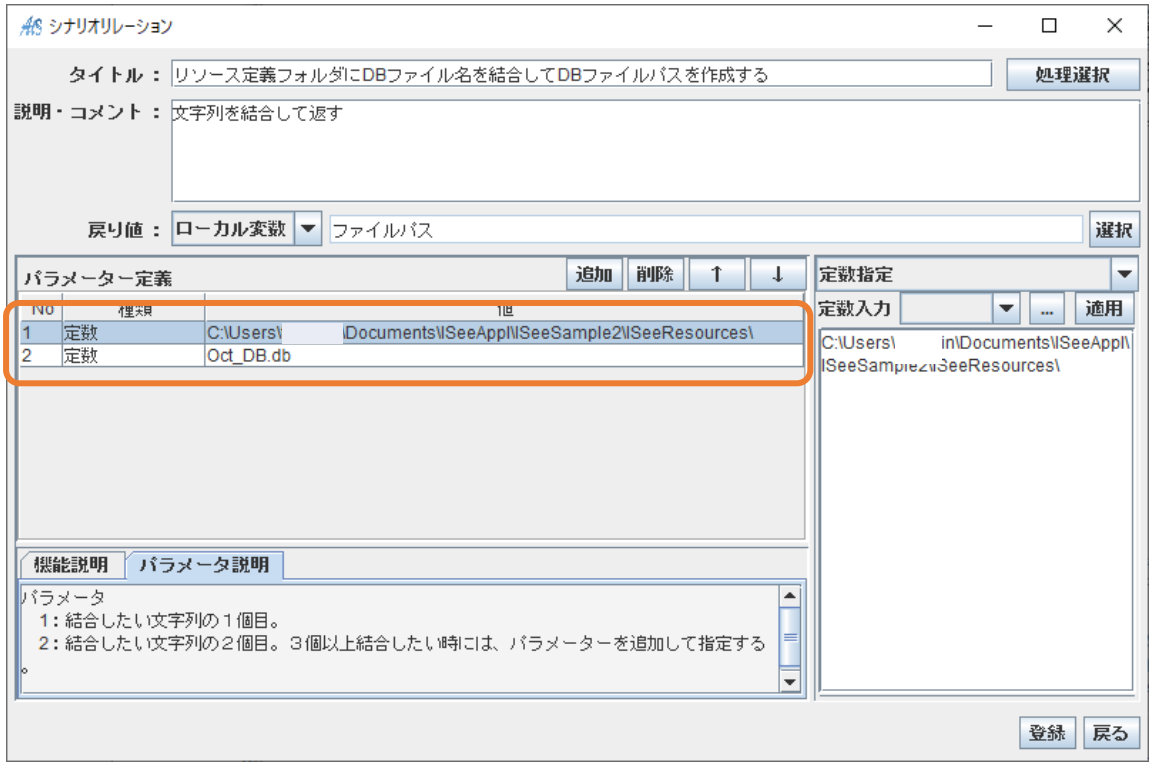
「 C:¥Users¥ユーザー名¥AppData¥Roaming¥ISeeIDE¥ISeeRun¥プロジェクト名 」

（以下、AppData 下）

※上記 URL は通常、編集が必要ない場所であるため隠しファイルになっています。まず Windows で隠しファイルを表示する設定にしてから、アクセスをお試しください。

【 AppData 下の DB ファイルが、最新の更新時刻となっている場合】
DB は正常に更新されていますので AiSee 上のパス指定は正しいと考えられます。
通常は、Documents 下の DB を DB Browser で参照すれば更新されているはずですが、もし更新されていなければ AppData 下の DB ファイルを DB Browser で開いて、DB が更新されているかご確認ください。

【 AppData 下の DB ファイルが、最新の更新時刻となっていない場合】
AiSee 上で DB の指定した DB のパスに誤りがないかご確認の上、正しいパスをご指定下さい。
パスが正しいように見えるのに更新が反映しない場合には、下図のように絶対パス指定で DB の場所の指定をお試しください。



AiSee の画面から DB ファイルを追加した際には、 エクスプローラーでは Documents 下に DB が追加されますが、更新や削除など実際の処理は、 AppData 下にコピーされた DB で実行されます。
ただし、AiSee 上で絶対パスで DB の場所を指定したい場合は、Documents 下の DB を指定してください。