繰り返し制御(WHILE)

1. 概要

アプリケーションを作成するとき、何らかの処理を繰り返し実行させたいことがあります。ある条件の間繰り返すという処理を記述する場合には、『繰り返し制御(WHILE)』コンポーネントを用いることでそれを実現できます。このコンポーネントは Java における while 文や do-while 文に相当します。

繰り返し制御(WHILE)をビルダーで追加する際のメニューは[コンポーネント追加]-[処理部品]-[条件制御]-[繰り返し制御(WHILE)]です。

2. 用途

繰り返し制御(WHILE)の用途は以下の通りです。

- 条件を満たしている間処理を繰り返したい場合に用いる
- ユーザがボタンを押すまで処理を繰り返したい場合に用いる

3. ここで使用するイベントとメソッド

繰り返し制御(WHILE)に関して、この文書内で使用されるイベントとメソッドの一覧を示します。ここに記す以外にもメソッドがありますが、それらの情報が必要な場合はリファレンスや Javadoc ドキュメントを参照してください。

- アクションイベント

イベント発生条件	内包データ	イベント番号
一回のループ実行ごと	-	0

- 処理完了イベント

イベント発生条件	内包データ	イベント番号
一回のループ実行ごと	-	0

- メソッド一覧

メソッド名	機能
ループ処理を実行する(詳細指定)(int, int)	ループ処理を開始し、第 1 引数で指定した初期値から開始して 1 回の実行ごとに第 2 引数で指定した増分値で数字をカウントする
ループカウンタの現在値を取得する()	ループ実行中のカウンタの現在値を返す
ループ続行可否を設定する(boolean)	ループを続行するかどうかを引数で設定する

4. コンポーネント使用例

4.1. サンプルアプリケーションの概要と操作方法

繰り返し制御(WHILE)のサンプルアプリケーションは"AP_DATA¥Sample¥繰り返し制御(WHILE).mzax"にあります。このサンプルは、2 つの使用例を含んでいます。その画面イメージを図 1 に示します。

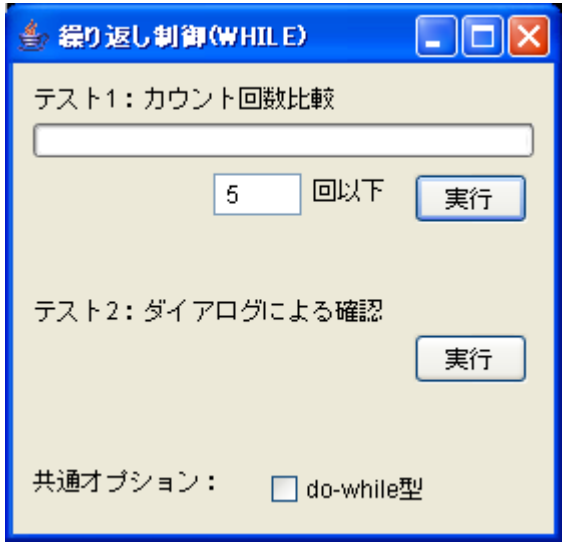

図 1　繰り返し制御(WHILE)サンプルアプリケーションの画面イメージ

1つ目の例（テスト1：カウント回数比較）は、指定した条件を満たす間ループを続行するというもので、ループのカウント回数が指定した値以下のときはループを続行する例になっています。数値入力フィールドに適当な数値を入れて「実行」ボタンを押すと、カウント数がダイアログで表示されて進捗ダイアログのバーがカウントごとに延びていきます。（ダイアログの「了解」ボタンを押すまではループの実行が停止します。）そして、カウント数が条件の数値を超えると処理が終了します。

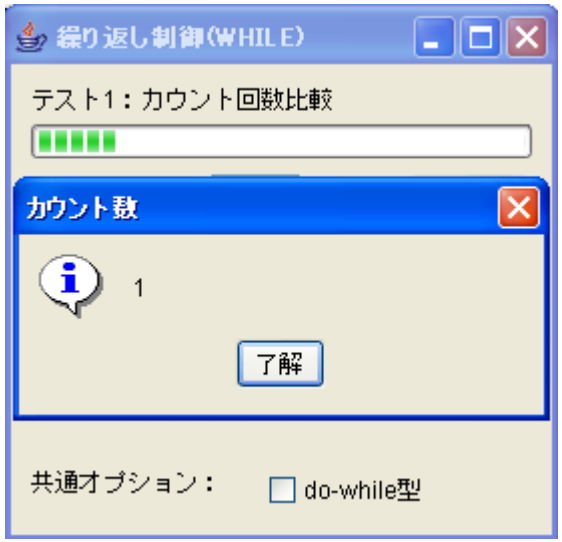

図 2　「テスト1：カウント回数比較」の実行画面

2つ目の例（テスト2：ダイアログによる確認）は、ループを実行するごとにダイアログによって続行確認を行うというものです。「実行」ボタンを押すとループが開始され、1回のループごとに確認ダイアログが表示されます。このダイアログで「はい」ボタンを押すとループが終了し、「いいえ」ボタンを押すと1回のループを実行してカウント数を表示します。続いて次のループに移り、また確認ダイアログが表示されます。

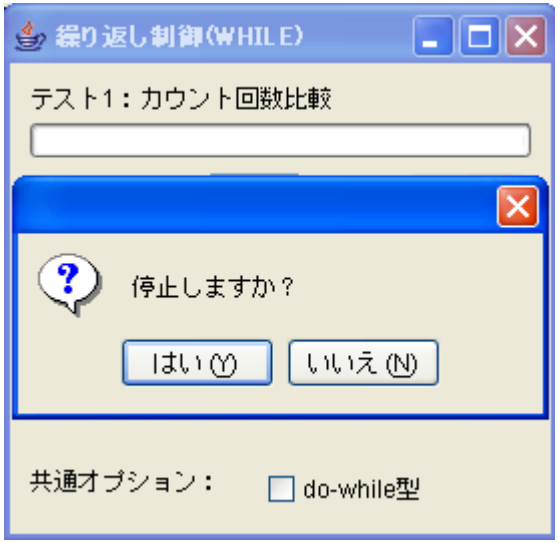

図 3 「テスト 2：ダイアログによる確認」の実行画面

フレームの下部にある「共通オプション：do-while 型」のチェックボックスをチェックすると、繰り返し制御(WHILE)が while 文ではなく do-while 文にあたる動作をします。つまり、ループを実行する前に条件判定する方式から最初にループを実行してから条件判定をする方式になります。1 つ目の例では動作に違いが表れませんが、2 つ目の例ではまずカウント数を示すダイアログが表示され、続いてループの停止を尋ねる確認ダイアログが表示されます。

4.2. サンプルアプリケーションに含まれる使用方法

サンプルに含まれる 2 つ目の例を使って、繰り返し制御(WHILE)の使い方について簡単に説明します。この繰り返し制御(WHILE)は、使用法を誤ると無限ループを引き起こすなどの問題が生じますので、お使いの場合は充分ご注意ください。

注意事項：繰り返し制御(WHILE)以外のコンポーネント（例えばボタンや確認ダイアログ）に関しては、使用法がわかっているものとして説明を省きます。それらに関しては別途チュートリアルやリファレンス、Javadoc ドキュメントを参照してください。

図 4 は、前述の 2 つ目の例に関連する主なコンポーネントとそれに連なるイベント処理を示したものです。

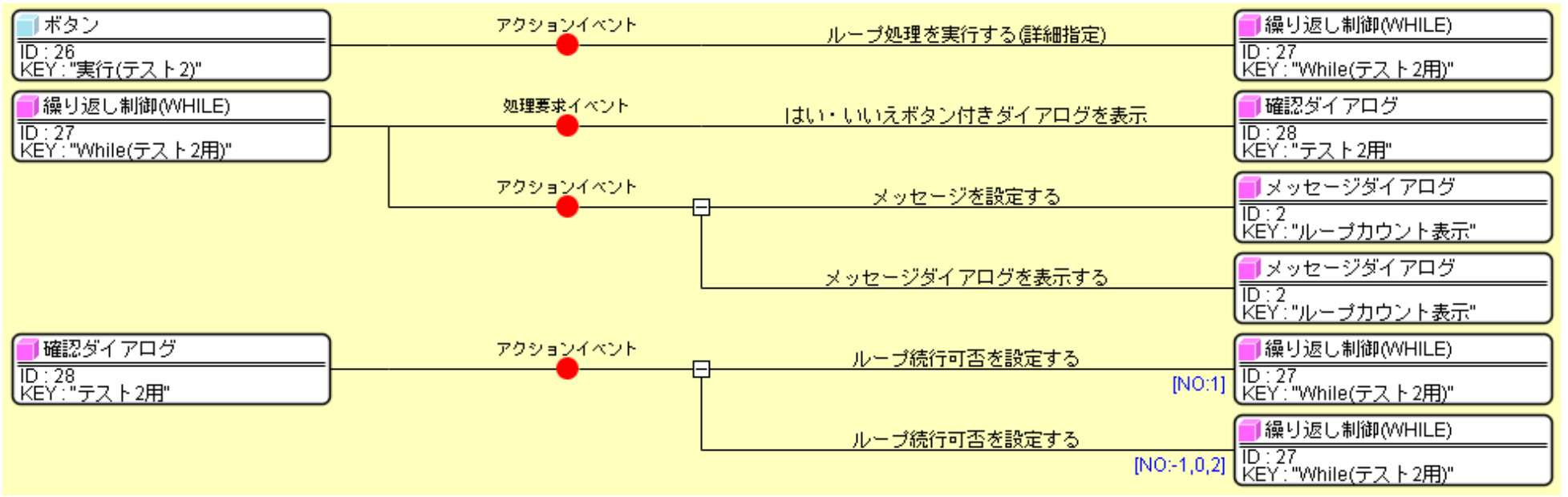

図 4　繰り返し制御(WHILE)の使用例

ここでは、「実行」ボタンを押したときに繰り返し制御(WHILE)に対して「ループ処理を実行する(詳細指定)」メソッドを起動しています。これによって、ループ処理が開始されます。繰り返し制御(WHILE)では、ループの実行時にループカウンタの値を取得することができるようになっており、通常の実行では初期値 0 から開始して 1 ずつ増えていきます。ここで使用した詳細指定のメソッドでは、そのループカウンタの初期値と増分値を指定できます。ここで示した例では、初期値 1 から開始して 1 ずつ増えるように指定しています。

繰り返し制御(WHILE)では、ループ処理が開始されるとループごとにコンポーネントから 2 種類のイベントが発生します。「処理要求イベント」はループを続行するための条件評価のタイミングで発生し、「アクションイベント」は 1 回のループ処理実行ごとに発生します。どちらのイベントが先に発生するかは、繰り返し制御(WHILE)の属性「DoWhileType」に依存します。この属性は、while 文のように処理するか、do-while 文のように処理するか指定するために用います。前者の場合は値を false とし、後者の場合は値を true とします。初期値は false です。

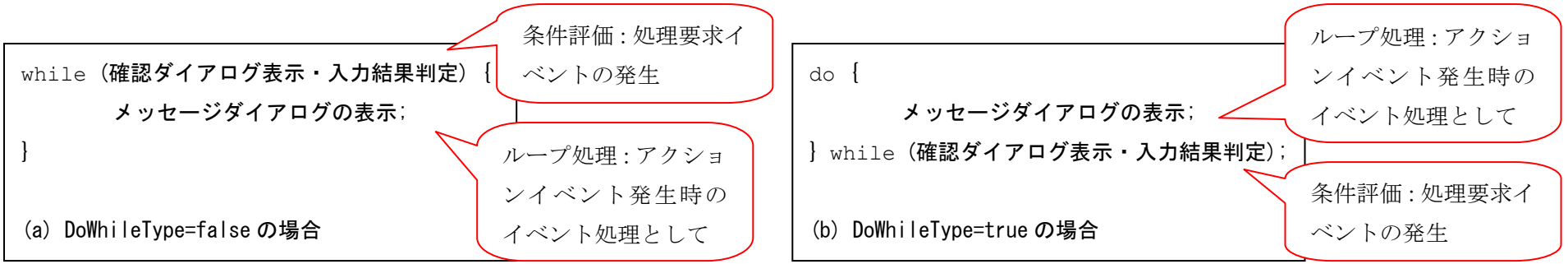

図 5 while 文と do-while 文の比較

上の図は、ここで対象としているサンプルの処理を Java の文法に従った仮想的なコードで記述したものです。属性の変化によって２つのイベントの発生する順番はこのように変わります。ループ処理はアクションイベントのイベント処理として記述され、この例ではメッセージダイアログにループカウンタの値を表示しています。

ループを続行するための条件評価では処理要求イベントが発生しますが、その際の評価方法に 2 種類の方法を用意しています。1 つは処理要求イベントのイベント処理の最後に起動するメソッドの戻り値が true の場合にループを続行するという方法で、もう 1 つは処理要求イベント発生後に繰り返し制御(WHILE)の属性「Continued」が true の場合に続行するという方法です。これらの値が両方設定されてかつ値が相反する場合には、メソッドの戻り値が優先されます。また、メソッドの戻り値が明示的に true や false を示さない場合には属性の値が優先されます。ここで紹介したサンプルアプリケーションの 1 つ目の例ではメソッドの戻り値を用いた方法を採用していて、2 つ目の例では属性を用いた方法を採用しています。

上に示す２つ目の例の処理を具体的に説明すると、条件評価のタイミングで処理要求イベントが発生したとき、確認ダイアログが表示されます。確認ダイアログには「はい」・「いいえ」ボタンがあり、これらのボタンが押されるまでループは条件評価のところで停止した状態になります。いずれかのボタンを押すと、イベント番号で区別されたアクションイベントが発生します。この確認ダイアログでは「停止しますか？」と質問しており、「はい」ボタンを押した場合(イベント番号１)に繰り返し制御(WHILE)のメソッド「ループ続行可否を設定する」を起動し、値を false にしてループを終了します。「いいえ」ボタンを押した場合は「ループ続行可否を設定する」で値を true にしてループを続行します。