

# ソフトウェア工学実習

# Software Engineering Practice

## (第07回)



SEP07-007 【実習編】NetBeans-SimpleCalc

拡張(3): エラーメッセージを表示しよう

こんにちは。  
この授業は、  
ソフトウェ  
ア工学実習  
です

慶應義塾大学・理工学部・管理工学科  
飯島 正

iijima@ae.keio.ac.jp



# エラーメッセージ用の参照変数を追加

The screenshot shows the NetBeans IDE interface with the project **P0401\_SimpleCalc** open. The **SimpleCalc.java** file is being edited. A red box highlights the line `private String errorMessage = null;`. Another red box highlights the declaration of this variable in the class definition. A large red box at the bottom highlights the same line again. A speech bubble on the right contains three dots (...).

```
21  /**
22   * 計算結果
23   */
24  private int result = 0;
25  /**
26   * エラーメッセージ
27   */
28  private String errorMessage = null;
29
30  /**
31   * 【コンストラクタ】
32   */
33  public SimpleCalc() {
34      x = 0;
35      y = 0;
36  }
37
38  /**
39   * 結果を表示する
40   */
41  public void outv(int v) {
42      result = v;
43  }
44
45  /**
46   * 加算する
47   */
48  public void addv(int v) {
49      x = v;
50  }
51
52  /**
53   * 減算する
54   */
55  public void divv(int v) {
56      y = v;
57  }
58
59  /**
60   * 乗算する
61   */
62  public void multv(int v) {
63      result = v;
64  }
65
66  /**
67   * 商を表示する
68   */
69  public void outd() {
70      System.out.println(result);
71  }
72
73  /**
74   * 商を表示する
75   */
76  public void outr() {
77      System.out.println(result);
78  }
79
80  /**
81   * 商を表示する
82   */
83  public void outr() {
84      System.out.println(result);
85  }
86
87  /**
88   * 商を表示する
89   */
90  public void outm() {
91      System.out.println(result);
92  }
93
94  /**
95   * 商を表示する
96   */
97  public void outv() {
98      System.out.println(result);
99  }
100 }
```

# コンストラクタで初期化(nullは正常終了も表す)

The screenshot shows the NetBeans IDE interface with the project P0401\_SimpleCalc selected. The main editor window displays the SimpleCalc.java file, which contains the following code:

```
11  /*
12   * SimpleCalc.java
13   */
14  public class SimpleCalc {
15      /**
16      * 第一パラメータ
17      */
18      private int x = 0;
19      /**
20      * 第二パラメータ
21      */
22      private int y = 0;
23      /**
24      * 計算結果
25      */
26      private int result = 0;
27      /**
28      * エラーメッセージ
29      */
30      private String errorMessage = null;
31      /**
32      * 【コンストラクタ】
33      */
34      public SimpleCalc() {
35          x = 0;
36          y = 0;
37          result = 0;
38          errorMessage = null;
39      }
40      /**
41      * 【setter】 基本データ型属性の設定
42      * @param x 第一パラメータ
43      */
44      void setX(int x) {
45          this.x = x;
46      }
47      void setY(int y) {
48          this.y = y;
49      }
50      void setResult(int result) {
51          this.result = result;
52      }
53      void setErrorString(String errorMessage) {
54          this.errorMessage = errorMessage;
55      }
56      /**
57      * 【getter】 基本データ型属性の取得
58      */
59      int getX() {
60          return x;
61      }
62      int getY() {
63          return y;
64      }
65      int getResult() {
66          return result;
67      }
68      String getErrorMessage() {
69          return errorMessage;
70      }
71  }
```

The constructor definition at lines 32-38 is highlighted with a red rectangle. The line "errorMessage = null;" is also highlighted with a red rectangle. A large speech bubble points from this line towards the bottom right corner of the screen.



```
public String getErrorMessage() {  
    return( errorMessage );  
}
```

```
59  /**  
60  * 【getter】 文字列属性の取得  
61  * @return エラーメッセージ(正常終了の時はnull)  
62  */  
63  
64  public String getErrorMessage() {  
65      return( errorMessage );  
66  }
```

```
getErrorMessage - ナビゲータ ×  
メンバー メンバー <空>  
SimpleCalc  
- SimpleCalc()  
- add()  
- div()  
- getErrorMessage(): String  
- getResult(): int  
- mul()  
- setX(int x)  
- setY(int y)  
- sub()  
- errorMessage: String
```

```
64  /**  
65  * 【getter】 文字列属性の取得  
66  * @return エラーメッセージ(正常終了の時はnull)  
67  */  
68  public String getErrorMessage() {  
69      return( errorMessage );  
70  }
```

65:31 INS



# 加算は正常終了するので、エラーメッセージにnull

```
67  /**
68  * 演算(加算)
69  */
70 public void add() {
71     result = x + y;
72 }
```

```
P0401_SimpleCalc - NetBeans IDE 8.0.2
ソース(S) リファクタリング(A) 実行(R) ティザイグ(D) プロファイル(P) チーム(M) ツール(T) ウィンドウ(W) ヘルプ(H)
ツール構成> フォルト構成> リソース C:\Users\iijima\OneDrive\桌面\Java\NetBeans\p0401_simplecalc\src\jp\keio\iijima\SimpleCalc.java
70 errorMessage = null;
```

```
67  /**
68  * 演算(加算)
69  */
70 public void add() {
71     result = x + y;
72     errorMessage = null;
73 }
```

```
SimpleCalc.java
SimpleCalcFrame.java
ライブラリ

getErrorMessage - ナビゲータ ×
メンバーメンバー <空> フィルター
SimpleCalc
- SimpleCalc()
- add()
- div()
- getErrorMessage(): String
- getResult(): int
- mul()
- setX(int x)
- setY(int y)
- sub()
- errorMessage: String

出力 - P0401_SimpleCalc (clean.jar) ×
>> 65:32 | INS
```



# 加算は正常終了するので、エラーメッセージにnull

```
P0401_SimpleCalc - NetBeans IDE 8.0.2  
67  /**  
68   * 演算(加算)  
69   */  
70  public void add() {  
71      result = x + y;  
72  }  
73  /*  
74   * 演算(減算)  
75   */  
76  public void sub() {  
77      result = x - y;  
78  }  
79  /*  
80   * 演算(乗算)  
81   */  
82  public void mul() {  
83      result = x * y;  
84  }  
85  /*  
86   * 演算(除算)  
87   */  
88  public void div() {  
89      if (y == 0) {  
90          errorMessage = "除数が0です";  
91      } else {  
92          result = x / y;  
93      }  
94  }  
95  /**  
96   * 文字列属性の取得  
97   * @return エラーメッセージ(正常終了の時はnull)  
98   */  
99  public String getErrorMessage() {  
100     return( errorMessage );  
101 }  
102 /*  
103  * 演算(加算)  
104  */  
105 public void add() {  
106     result = x + y;  
107     errorMessage = null;  
108 }  
109 /*  
110  * 演算(減算)  
111  */  
112 public void sub() {  
113     result = x - y;  
114 }  
115 /*  
116  * 演算(乗算)  
117  */  
118 public void mul() {  
119     result = x * y;  
120 }  
121 /*  
122  * 演算(除算)  
123  */  
124 public void div() {  
125     if (y == 0) {  
126         errorMessage = "除数が0です";  
127     } else {  
128         result = x / y;  
129     }  
130 }  
131 /*  
132  * 文字列属性の設定  
133  */  
134 public void setErrorMessage(String errorMessage) {  
135     this.errorMessage = errorMessage;  
136 }  
137 /*  
138  * 結果を取得  
139  */  
140 public int getResult() {  
141     return result;  
142 }  
143 /*  
144  * Xを設定  
145  */  
146 public void setX(int x) {  
147     this.x = x;  
148 }  
149 /*  
150  * Yを設定  
151  */  
152 public void setY(int y) {  
153     this.y = y;  
154 }  
155 /*  
156  * エラーメッセージ  
157  */  
158 public String getErrorMessage() {  
159     return errorMessage;  
160 }
```

# 減算も同様

The screenshot shows the NetBeans IDE interface with two code editors and a central navigation pane.

**Left Editor:** Shows the original code for the `sub()` method:74 75 76 77 78 79  
74   /\*\*  
75    \* 演算(減算)  
76    \*/  
77  public void sub() {  
78    result = x - y;  
79 }

**Right Editor:** Shows the refactored code where the `errorMessage` variable is initialized to `null` at the top of the method:74 75 76 77 78 79 80  
74   /\*\*  
75    \* 演算(減算)  
76    \*/  
77  public void sub() {  
78    result = x - y;  
79    errorMessage = null;  
80 }

**Central Navigation:** The navigation pane shows the `SimpleCalc` class structure, including methods like `add()`, `div()`, `getErrorMessage()`, `getResult()`, `mul()`, `setX(int x)`, `setY(int y)`, and `sub()`. The `sub()` method is highlighted in both the left and right panes.

A large yellow arrow points from the original code in the left editor to the refactored code in the right editor. A speech bubble on the right contains three dots (...).



# 乗算も同様

```
P0401_SimpleCalc - NetBeans IDE 8.0.2  
80  * 演算(乗算)  
81  */  
82  public void mul() {  
83      result = x * y;  
84  }  
85  
86  /**  
87   * 演算(乗算)  
88   */  
89  public void mul() {  
90      result = x * y;  
91      errorMessage = null;  
92  }  
93  
94  /**  
95   * 演算(除算)  
96   */  
97  public void div() {  
98      result = x / y;  
99  }  
100 }
```

The image shows the NetBeans IDE interface with the file P0401\_SimpleCalc.java open. The code defines a class SimpleCalc with four methods: add(), div(), mul(), and sub(). Each method sets the result variable and initializes errorMessage to null. The code is annotated with Japanese comments explaining the operations. A yellow arrow highlights the assignment of errorMessage to null in the first mul() method. Another red box highlights the same line in the second mul() method. A large speech bubble is positioned to the right of the code area.



# 除算の場合は、「ゼロ除算エラー」があり得ます

```
88  /**
89  * 演算(除算)
90  */
91 public void div() {
92     if (y != 0) {
93         result = x / y;
94         errorMessage = null;
95     } else {
96         result = 0;
97         errorMessage = "エラー:ゼロ除算";
98     }
99 }
100 }
```

```
88 /**
89 * 演算(除算)
90 */
91 public void div() {
92     if (y != 0) {
93         result = x / y;
94         errorMessage = null;
95     } else {
96         result = 0;
97         errorMessage = "エラー:ゼロ除算";
98     }
99 }
100 }
```

... (in a large speech bubble)



# GUI部品の参照変数を確認しておきます。

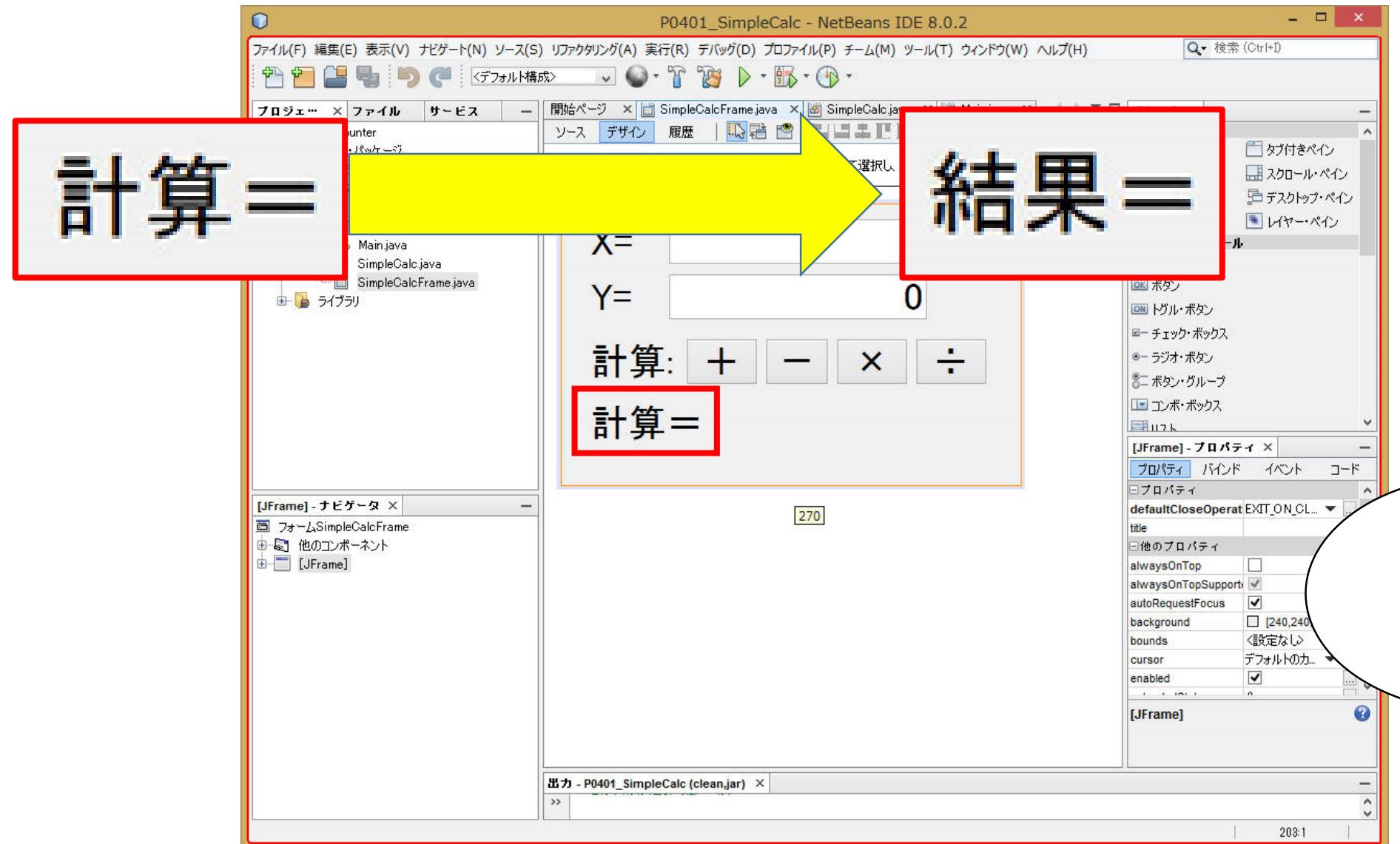
```
// Variables declaration - do not modify
private javax.swing.JButton addButton;
private javax.swing.JButton divButton;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JButton mulButton;
private javax.swing.JLabel resultLabel;
private javax.swing.JButton subButton;
private javax.swing.JTextField xField;
private javax.swing.JTextField yField;
// End of variables declaration

private void divButtonActionPerformed(java.awt.event.ActionEvent evt) {
    setParameters();
    aCalc.div();
    updateView();
}

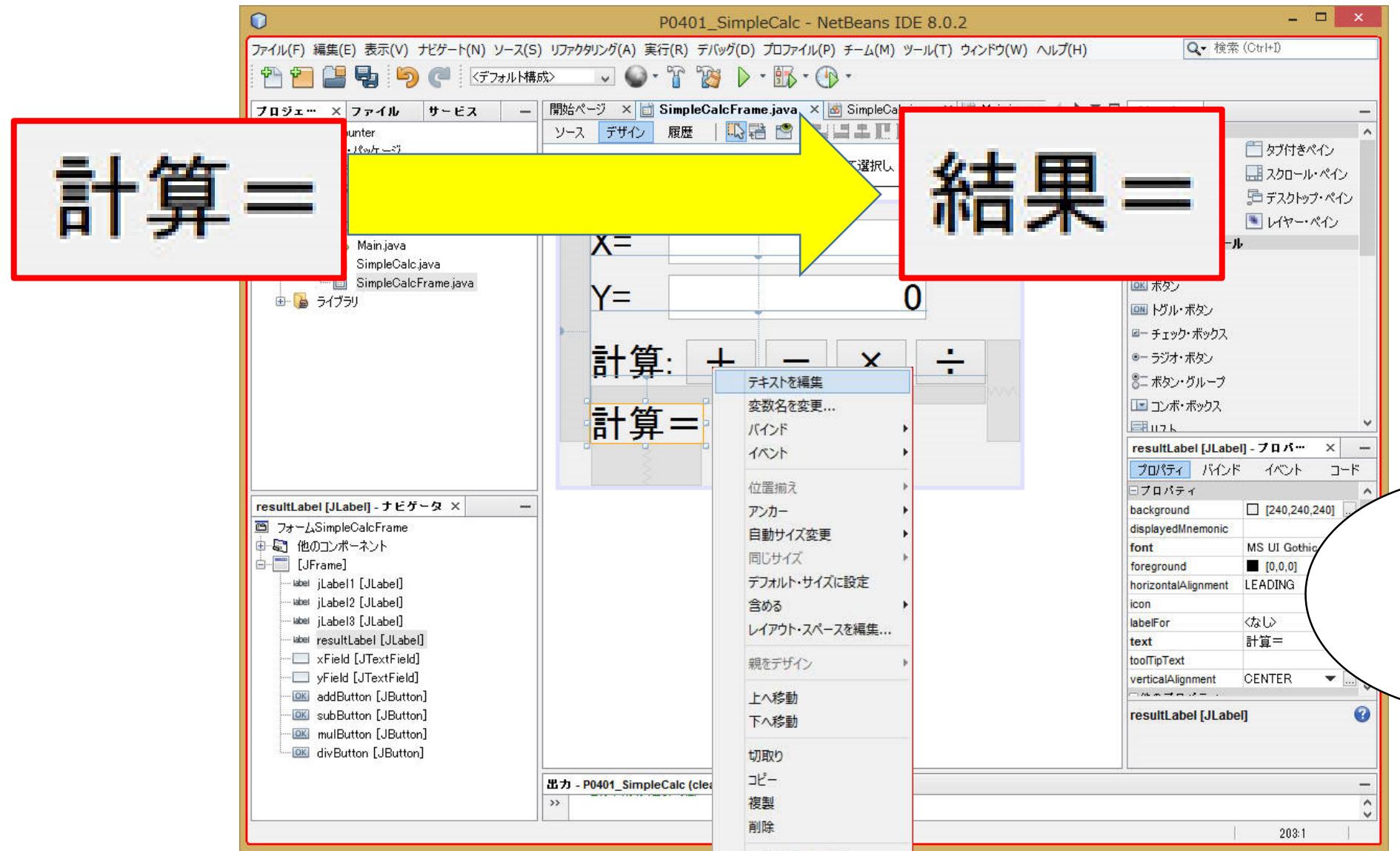
// Variables declaration - do not modify
private javax.swing.JButton addButton;
private javax.swing.JButton divButton;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JButton mulButton;
private javax.swing.JLabel resultLabel;
private javax.swing.JButton subButton;
private javax.swing.JTextField xField;
private javax.swing.JTextField yField;
// End of variables declaration
```



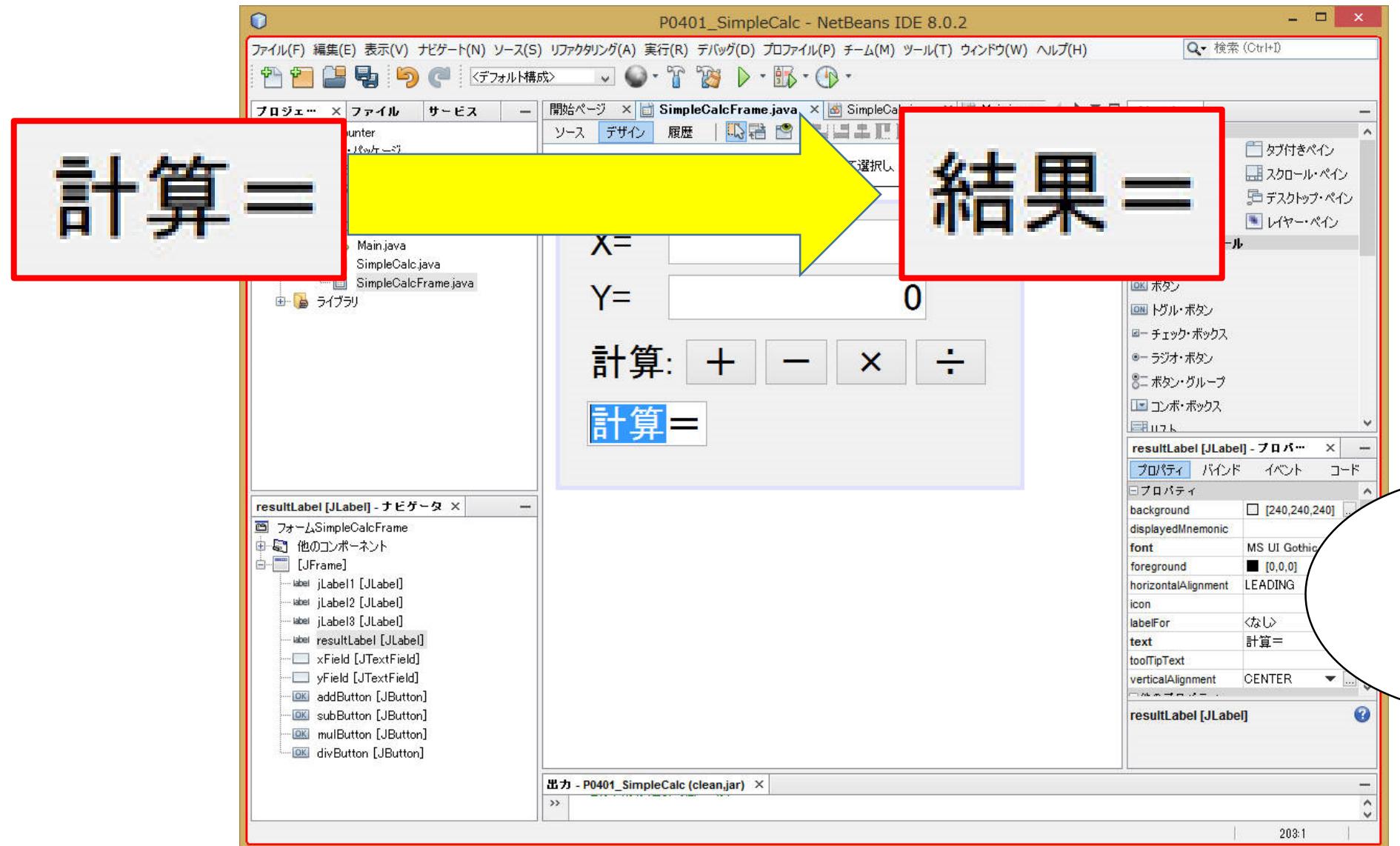
# 必要に応じてエラーメッセージを表示させるようにします



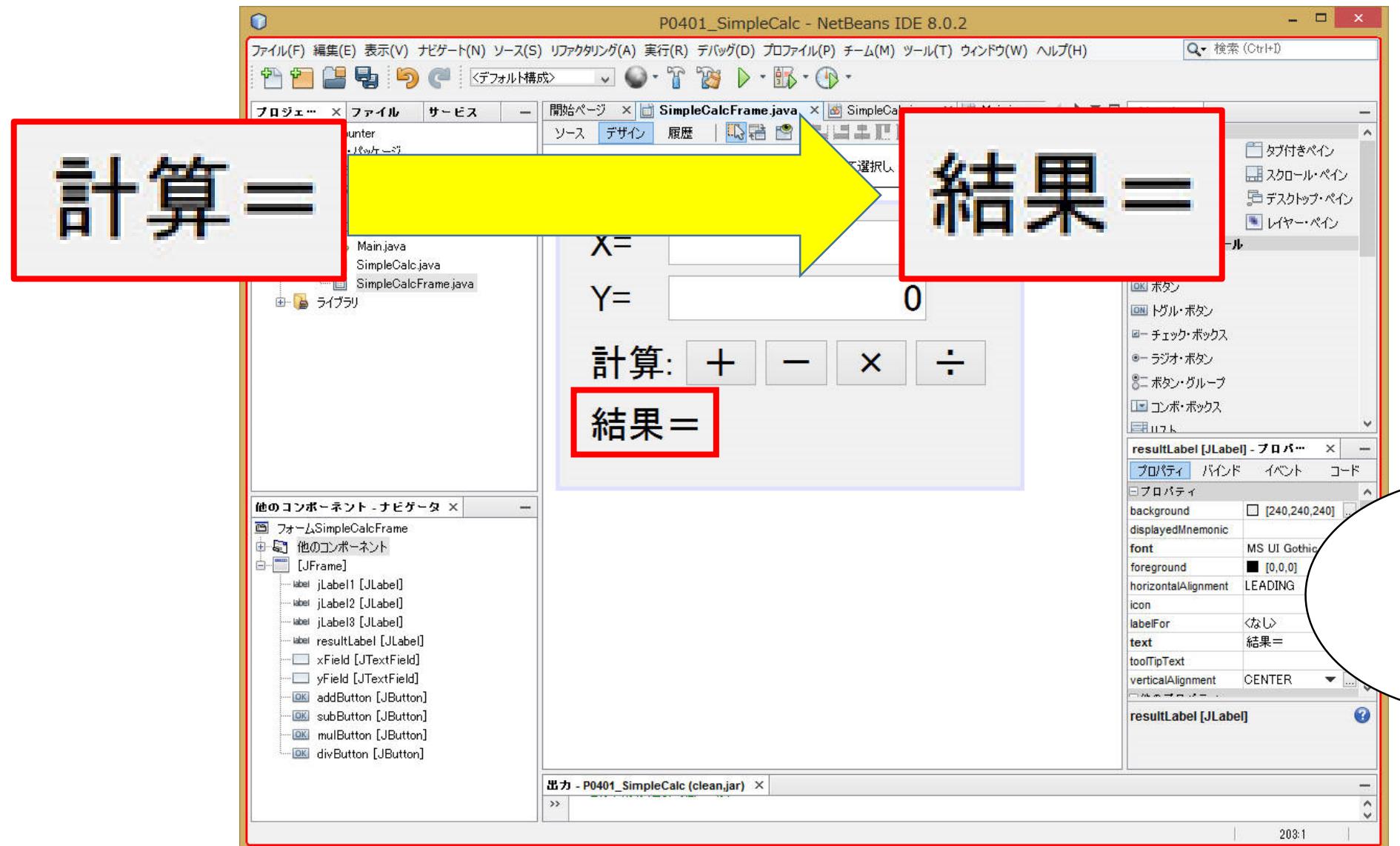
# その前に一つ、ラベルを修正しましょう



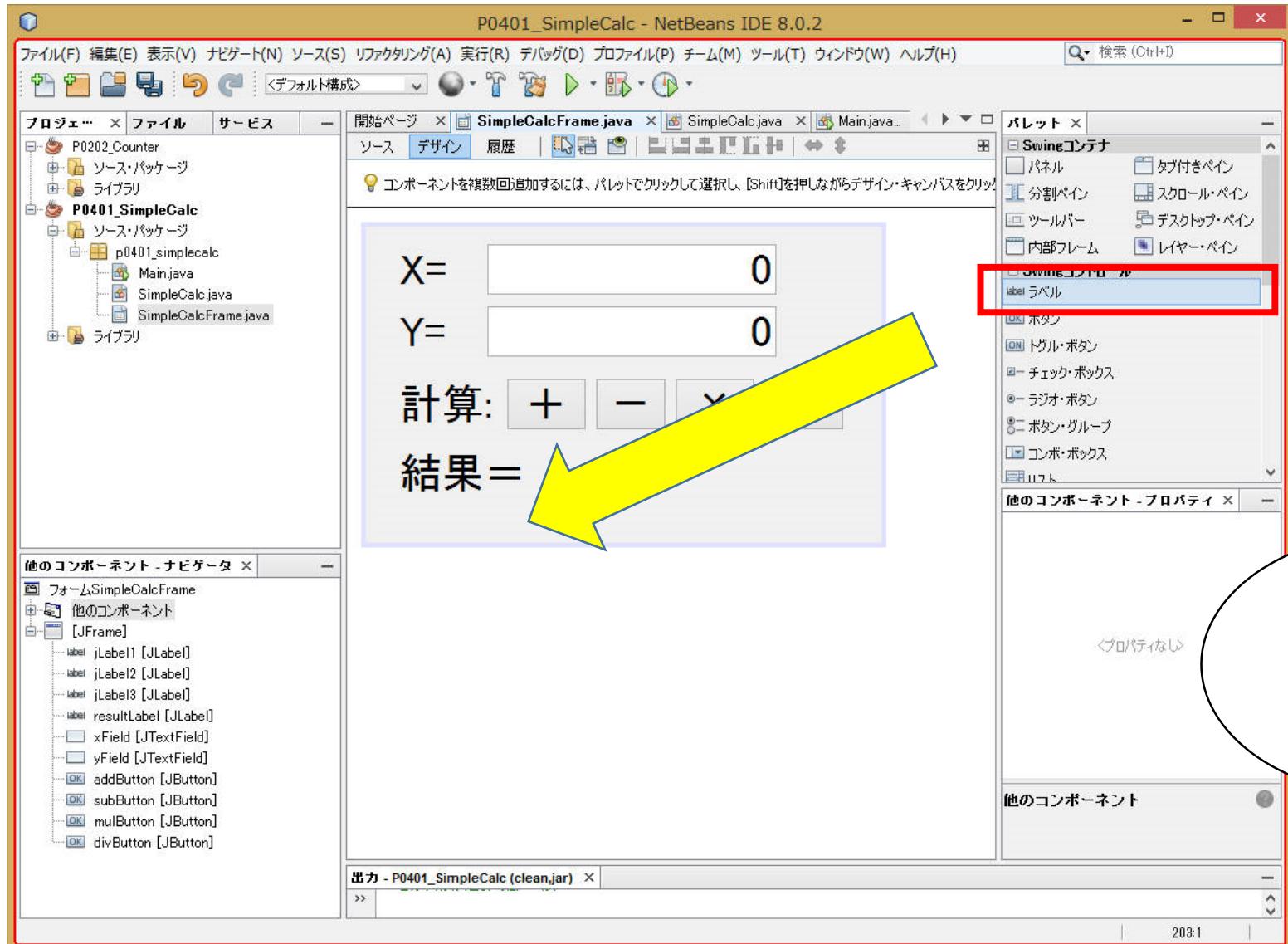
# その前に一つ、ラベルを修正しましょう



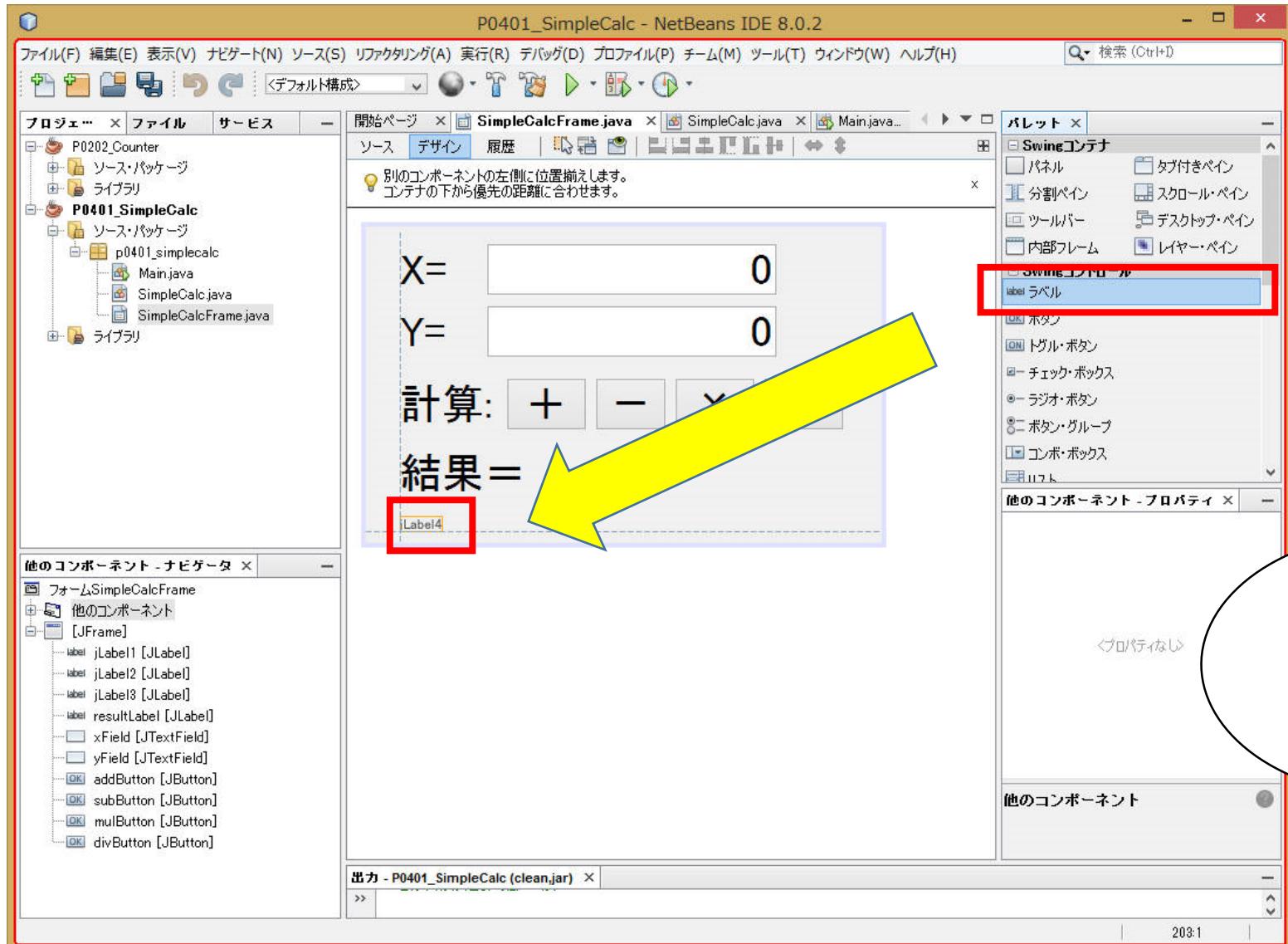
# その前に一つ、ラベルを修正しましょう



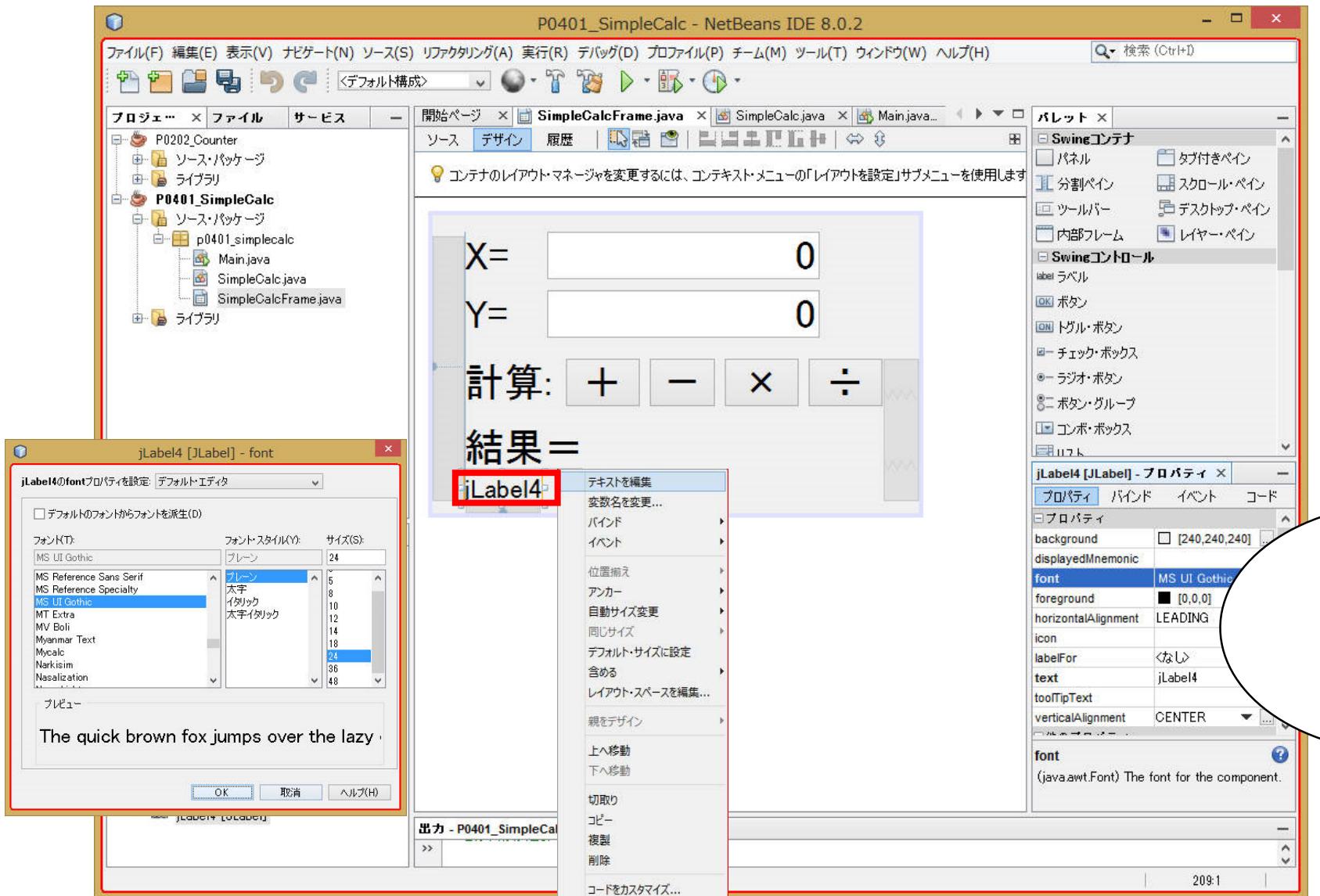
# Swingコントロールパレットからラベルをドラッグ&ドロップ



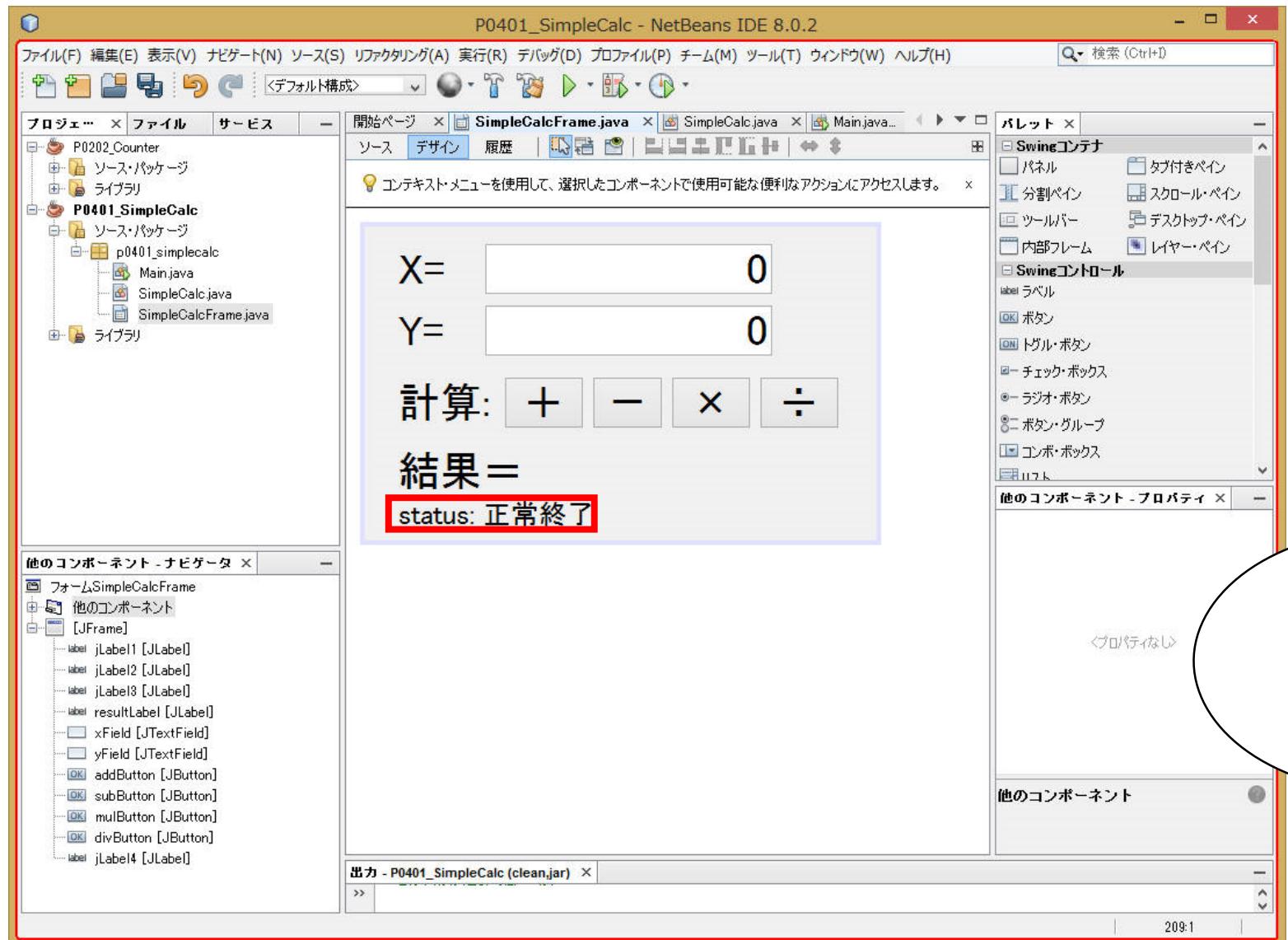
# Swingコントロールパレットからラベルをドラッグ&ドロップ



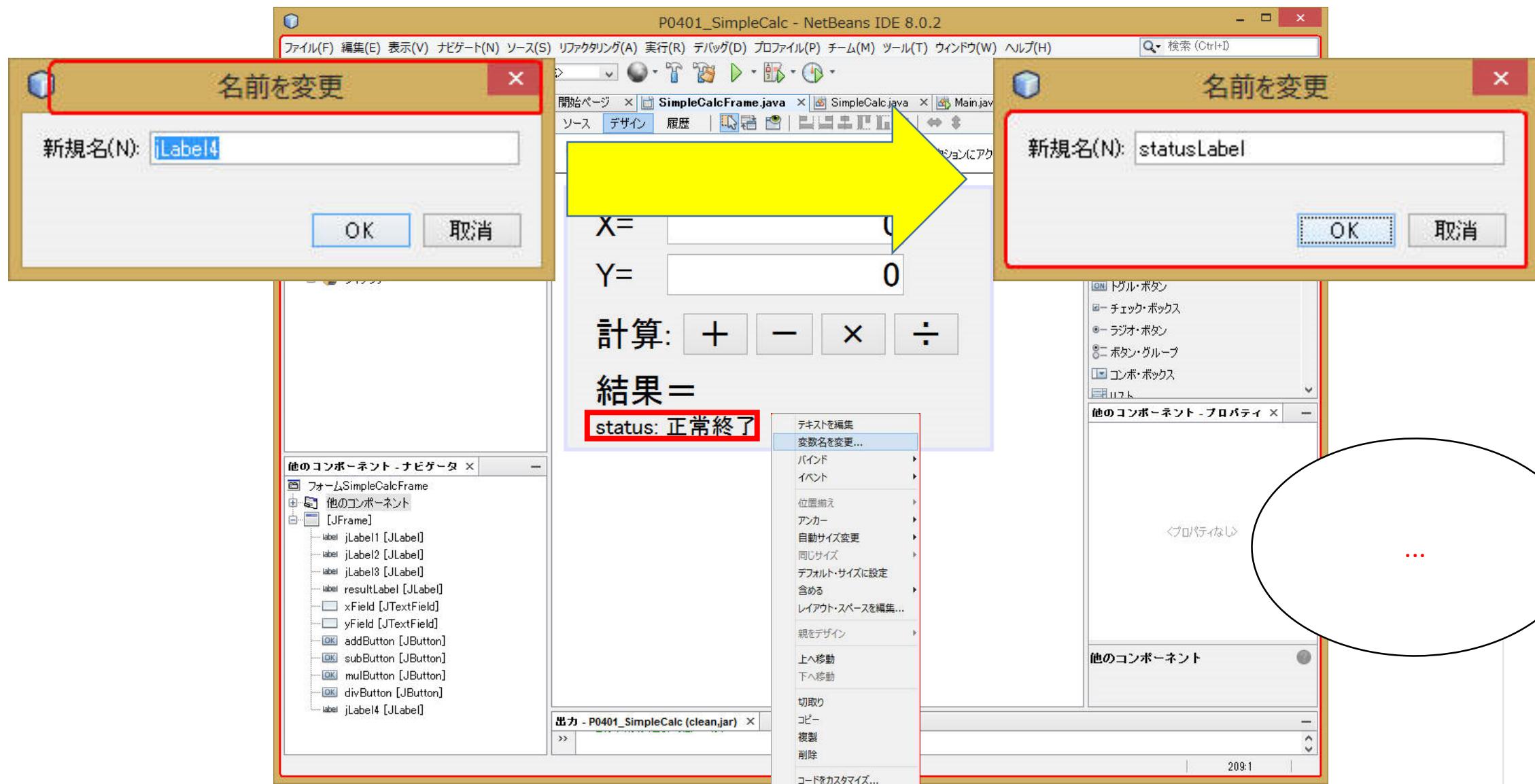
# 文字サイズを大きくして、文字列の変更



# 「Status:正常終了」としましょう



# 変数名を、 statusLabelに変更します



# 演算時のイベントハンドラのメソッド

The diagram illustrates the call hierarchy between two Java files: Main.java and SimpleCalc.java.

**Main.java** contains the following code:

```
160 /**
161 * 「簡単な電卓」にパラメータを設定する
162 */
163 private void setParameters() {
164     aCalc.setX( Integer.parseInt( xField.getText() ) );
165     aCalc.setY( Integer.parseInt( yField.getText() ) );
166 }
```

**SimpleCalc.java** contains the following code:

```
162 /**
163 */
164 private void setParameters() {
165     aCalc.setX( Integer.parseInt( xField.getText() ) );
166     aCalc.setY( Integer.parseInt( yField.getText() ) );
167 }
168 /**
169 * 「簡単な電卓」から計算結果を取得し、ビューに反映させる
170 */
171 private void updateView() {
172     resultLabel.setText( "結果= " + Integer.toString( aCalc.getResult() ) );
173 }
174 /**
175 * 加算ボタンのイベントハンドラ
176 * @param evt アクションイベント
177 */
178 private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
179     setParameters();
180     aCalc.add();
181     updateView();
182 }
183 /**
184 * 減算ボタンのイベントハンドラ
185 */
186 private void minusButtonActionPerformed(java.awt.event.ActionEvent evt) {
187     setParameters();
188     aCalc.subtract();
189     updateView();
190 }
```

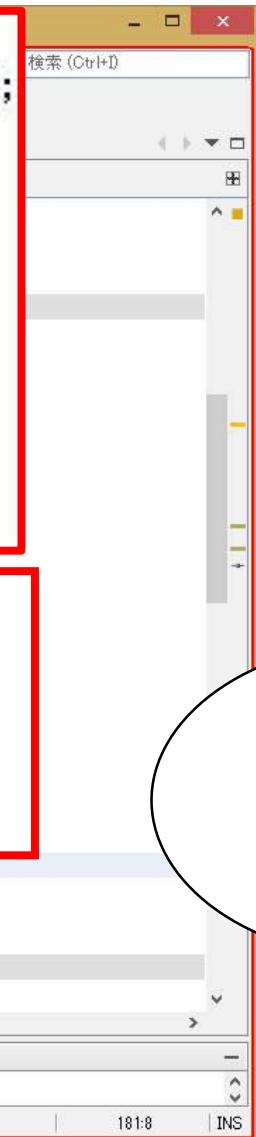
**Annotations:**

- A red box highlights the `setParameters()` method in **Main.java**.
- A red box highlights the `updateView()` method in **SimpleCalc.java**.
- A red box highlights the `addButtonActionPerformed()` method in **Main.java**.
- A red box highlights the `minusButtonActionPerformed()` method in **SimpleCalc.java**.
- Yellow arrows indicate the flow of calls:
  - An arrow points from the `setParameters()` call in **Main.java** to the `setParameters()` method in **SimpleCalc.java**.
  - An arrow points from the `updateView()` call in **Main.java** to the `updateView()` method in **SimpleCalc.java**.
  - An arrow points from the `setParameters()` call in the `addButtonActionPerformed()` method in **Main.java** to the `setParameters()` method in **SimpleCalc.java**.
  - An arrow points from the `updateView()` call in the `addButtonActionPerformed()` method in **Main.java** to the `updateView()` method in **SimpleCalc.java**.
  - An arrow points from the `setParameters()` call in the `minusButtonActionPerformed()` method in **SimpleCalc.java** back to the `setParameters()` method in **SimpleCalc.java**.

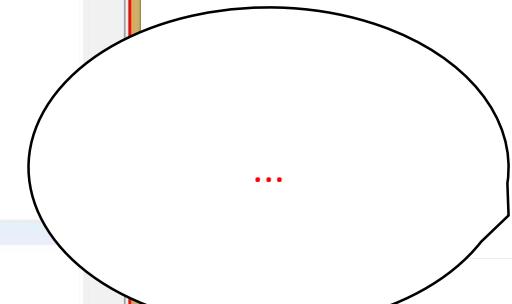


# updateView() メソッドに エラーメッセージ表示機能を追加

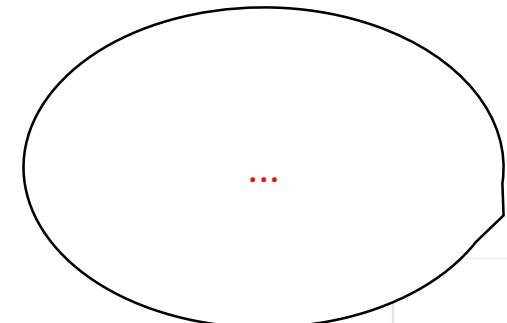
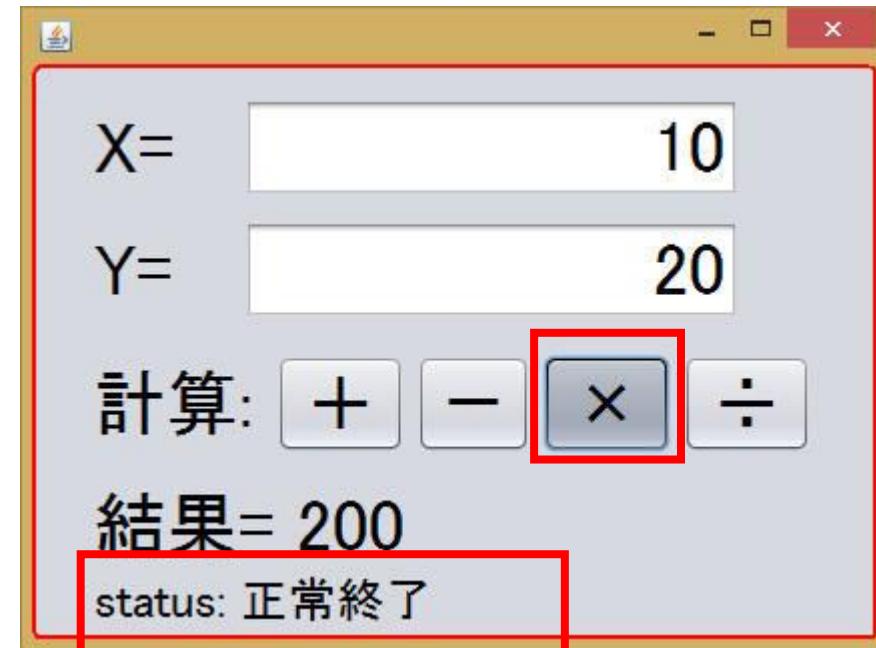
```
170     private void updateView() {  
171         resultLabel.setText("結果= " + Integer.toString( aCalc.getResult() ) );  
172         String errorMessage = aCalc.getErrorMessage();  
173         if ( errorMessage != null ) {  
174             statusLabel.setText("status: " + errorMessage );  
175             statusLabel.setForeground(new java.awt.Color(255, 0, 0));  
176         } else {  
177             statusLabel.setText("status: 正常終了");  
178             statusLabel.setForeground(new java.awt.Color(0, 0, 0));  
179         }  
180     }
```



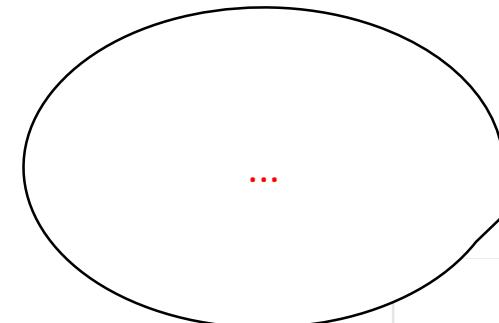
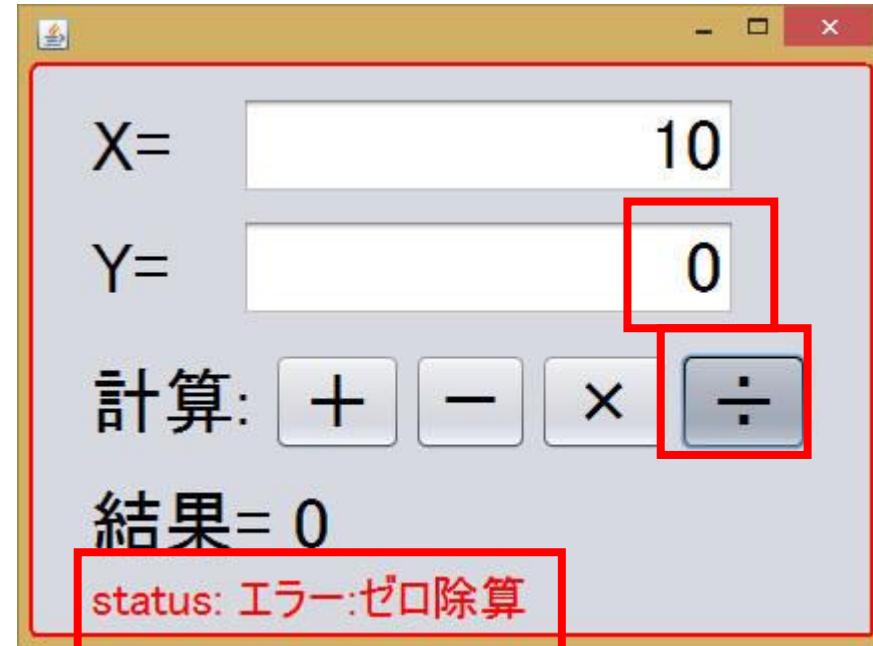
```
189     /*  
190      *  
191      * private void updateView() {  
192          resultLabel.setText("結果= " + Integer.toString( aCalc.getResult() ) );  
193          String errorMessage = aCalc.getErrorMessage();  
194          if ( errorMessage != null ) {  
195              statusLabel.setText("status: " + errorMessage );  
196              statusLabel.setForeground(new java.awt.Color(255, 0, 0));  
197          } else {  
198              statusLabel.setText("status: 正常終了");  
199              statusLabel.setForeground(new java.awt.Color(0, 0, 0));  
200          }  
201      */  
202      * 加算ボタンのイベントハンドラー  
203      * @param evt アクションイベント  
204      */  
205      private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {  
206          setParameters();  
207      }  
208  
```



# 乗算は正常終了



# ゼロ除算は、エラー表示



# 続けて、正常な計算をしますと…

