Swing のイベント処理を知る

前回までにボタンやテキストフィールドを使った GUI アプリケーションの構築手順を紹介してきました。今回は「ボタンをクリック（アクション）したときに、GUI アプリケーションがどう振る舞うのか」をプログラミングする方法について紹介していきます。

イベント処理とは

アプリケーションの利用者は、GUI コンポーネントを操作することで、アプリケーションが何らかの処理を実行することを期待しています。そのため、アプリケーションの開発者は、

1. 利用者の操作を何らかの手段で検知する
2. アプリケーションがどう振る舞うべきかを状況に応じて判断する
3. 利用者の期待する処理を実行する

という一連の作業をプログラミングする必要があります。しかし、これらの処理を一からプログラミングするのは大変です。そのため、Swing にはこの一連の作業を簡単にプログラミングできるような仕組みであるイベント処理のモデル（アクション、イベント、アクション・リスナー）があらかじめ用意されています。

アクション

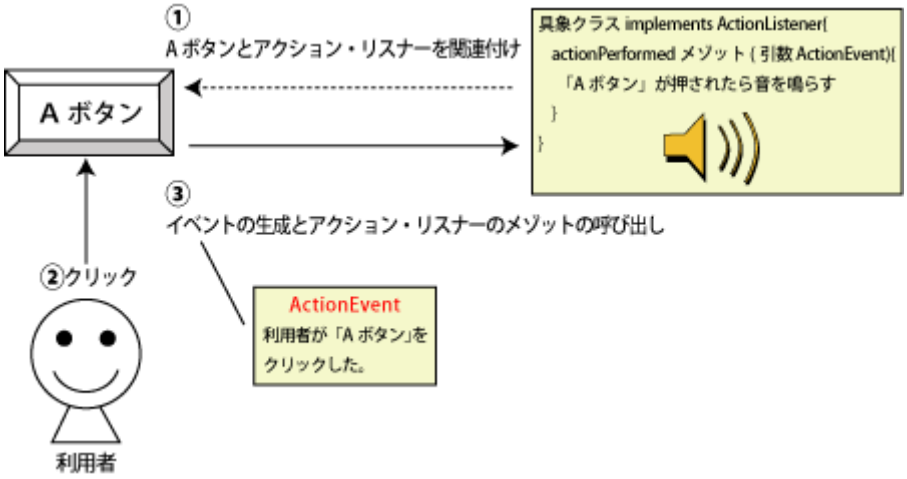
GUI コンポーネント（ボタン、テキストフィールド、メニュー等）に対して利用者が行う操作を「アクション」といいます。例えばボタンをクリックする、メニューを選択するといった、利用者の操作をアクションといいます。

イベント

利用者が操作した事実および利用者の操作に関する情報を保持するオブジェクトを［イベント］といいます。このイベントはコンポーネントに対してアクションが行われるたびに生成されます。例えば A ボタンがクリックされた、B ボタンがクリックされた、メニュー「ファイル」が選択された、といったアクションを行うと、コンポーネントはその事実や情報を保持するためのクラス「java.awt.event.ActionEvent」のインスタンスを生成します。

アクション・リスナー

利用者の操作に応じて一定の処理を行うインターフェイスを「アクション・リスナー」といいます。アクション・リスナーはインターフェイス「java.awt.event.ActionListener」であるため、具体的な処理を記述できません。そのため、例えば音を鳴らす、ダイアログボックスを表示する、といった具体的なアプリケーションの処理はそのインターフェイスを実装した具象クラスのメソッド（actionPerformed）にプログラミングすることになります。



Swing のイベント処理を知る

アプリケーションの振る舞いを実装

アクション、イベント、アクション・リスナーをより確かに理解するために、プログラムを書きながら解説します。

プロジェクトの選択

今回も以前と同じ「SwingChapter01」という名前のプロジェクトを利用します（作っていない方は最初の回の“プロジェクトの作成”を参考に作成してください）。

クラスの作成

GUI アプリケーションの起動、コンポーネントの配置、コンポーネントのアクションに応じた処理を担当するクラス「SwingApp3」を作成します。

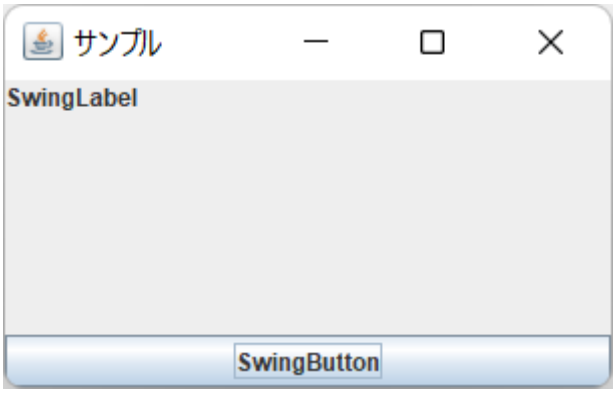
```
1
2  import java.awt.*;
3  import java.awt.event.*;
4  import javax.swing.*;
5
6  public class SwingApp3 extends JFrame implements ActionListener{
7     public static void main(String[] args) {
8        new SwingApp3();
9     }
10    JButton button;                                          //koko
11    JLabel label1;                                           //koko
12    JLabel label2;                                           //koko
13    Container contentPane;                                   //koko
14    JFrame mainFrame;                                        //koko
15    public SwingApp3() {
16       mainFrame = new JFrame("サンプル");
17       mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18       mainFrame.setSize(320, 200);
19       mainFrame.setLocationRelativeTo(null);
20 //ここから
21       contentPane = mainFrame.getContentPane();            //koko
22       // ラベルのインスタンスを生成
23       label1 = new JLabel("SwingLabel");                   //koko
24       // ラベルのインスタンスを生成
25       label2 = new JLabel("HelloSwingWorld");              //koko
26       // ボタンのインスタンスを生成
27       button = new JButton("SwingButton");
28       button.addActionListener(this);                      //koko
29       // ラベルをContentPane（コンポーネント）に配置
30       contentPane.add(label1, BorderLayout.NORTH);         //koko
31       // ボタンをContentPane（コンポーネント）に配置
32       contentPane.add(button, BorderLayout.SOUTH);
33 //ここまで
34       mainFrame.setVisible(true);
35    }
36    public void actionPerformed(ActionEvent event) {
37       if(event.getSource()==button) {
38          contentPane.add(label2, BorderLayout.CENTER);     //koko
39          mainFrame.setVisible(true);
40       }
41    }
42 }
```

自動的に展開されます

赤枠部分がアクション・リスナー処理の実装です。

アプリケーションの実行

「SwingApp3」クラスを実行すると、以下のウィンドウが表示されます。



「SwingButton」をクリックすることで HelloSwingWorld の文字列が追加されます。

プログラムの解説

1.アクション・リスナーの実装

ボタンに対して利用者の操作（アクション）に反応するためには「ActionListener」を implements したクラスを定義する必要があります。今回の例では「SwingApp3」自身がその役割を担っています。

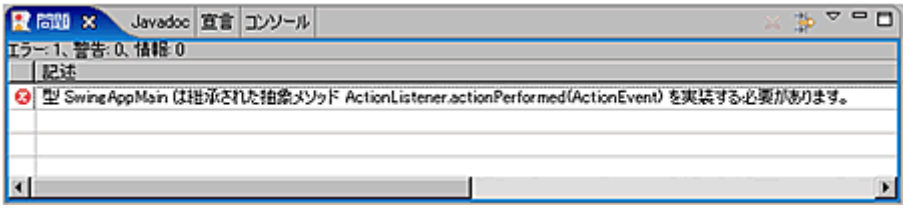
```
public class SwingApp3 implements ActionListener {
```

「ActionListener」インターフェイスは implements するだけでは機能しません。このインターフェイスは抽象メソッド「actionPerformed」を宣言しているためです（**注**）。

そのため「SwingApp3」固有の「actionPerformed」メソッドを定義します。

```
public void actionPerformed(ActionEvent event){}
```

注：「actionPerformed」メソッドを定義しない場合、コンパイルエラーが発生します。



2.アクション・リスナーの登録

```
// ボタンとアクション・リスナーの関連付け
button.addActionListener(this);//」ボタンとアクション・リスナーの関連付け
```

ボタンはアクションがあった場合に、どのクラス（メソッド）を実行すればよいかをあらかじめ知っておく（関連付けておく）必要があります。そのため「addActionListener」を実行し、ボタンとアクション・リスナーを実装したクラスとを関連付けます。

「addActionListener」の引数には「ActionListener」を implements したクラスのインスタンスを設定します。今回の例では「SwingApp3」自身となるため「this」を設定しています。

上記のコードを記述することでボタンがクリックされたとき「SwingApp3」クラス固有の「actionPerformed」が実行されるようになります。

3. 利用者の操作に応じた処理の実装

ボタンに対してアクションを行うと「ActionEvent」のインスタンスが生成されます。

今回の例では、ボタンに対して行われたアクション（クリック）の結果として生成された「actionPerformed」メソッドにおいてテキストを追加する処理を行っています。

```
// 利用者の操作に応じた処理を実装
public void actionPerformed(ActionEvent event)
```

```
  public void actionPerformed(ActionEvent event) {
   if(event.getSource()==button) {
        contentPane.add(label2, BorderLayout.CENTER);     //koko
        mainFrame.setVisible(true);
   }
  }
```

ちなみに、addButton と clearButton など複数のボタンに対して行われたアクション（クリック）の結果として生成された「ActionEvent」クラスのインスタンスの情報を基にアプリケーションの振る舞い（actionPerformed）を変えることも可能です。

```
【例】
// ユーザーの操作対象を判断
if(event.getSource() == addButton) {
// テキストエリアへ文字列を追加
textArea.append(textField.getText() + "¥n");
}
if(event.getSource() == clearButton) {
// テキストエリアの文字列を全消去
textArea.setText(null);
}
}
```

まとめ

今回は利用者の操作に応じてアプリケーションがどう振る舞うのかをプログラミングする方法（Swing に用意されているイベント処理のモデル）について解説しました。

演習課題

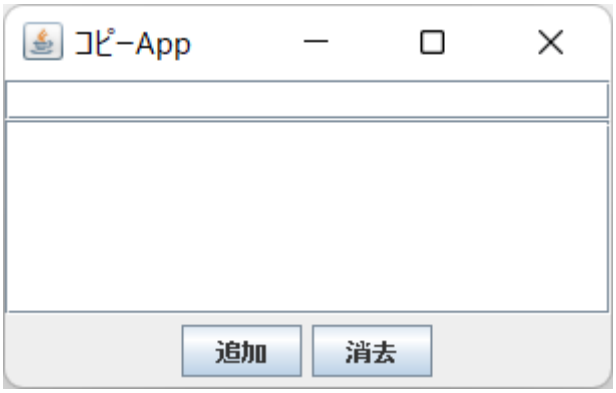
テキストフィールドに書き込んだ文字列をテキストエリアにコピーする GUI アプリケーションを作成します。

以下、SwingApp3 をコピーし SwingApp4 を作成してください。

```
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;

public class SwingApp4 extends JFrame implements ActionListener{

  private JFrame mainFrame;
  private Container contentPane;
  private JTextField textField;
  private JTextArea textArea;
  private JScrollPane scrollPane;
  private JPanel buttonPane;
  private JButton addButton;
  private JButton clearButton;
// コンストラクタ
 public SwingApp4(){
  mainFrame = new JFrame("コピーApp");
  mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  mainFrame.setSize(320, 200);
  mainFrame.setLocationRelativeTo(null);
  contentPane = mainFrame.getContentPane();
  textField = new JTextField();
  textArea = new JTextArea();
  scrollPane = new JScrollPane(textArea);
  addButton = new JButton("追加");
  clearButton = new JButton("消去");

//  「追加」ボタンとアクション・リスナーの関連付け
  addButton.addActionListener(this);
//  「消去」ボタンとアクション・リスナーの関連付け
  clearButton.addActionListener(this);

  buttonPane = new JPanel();
  buttonPane.add(addButton);
  buttonPane.add(clearButton);
  contentPane.add(textField, BorderLayout.NORTH);
  contentPane.add(scrollPane, BorderLayout.CENTER);
  contentPane.add(buttonPane, BorderLayout.SOUTH);
  mainFrame.setVisible(true);
 }
```

```
42 // 利用者の操作に応じた処理を実装
43  public void actionPerformed(ActionEvent event){
44 // ユーザの操作対象を判断
45    if(event.getSource() == addButton) {
46 // テキストエリアへ文字列を追加
47      textArea.append(textField.getText() + "¥n");
48    }
49    if(event.getSource() == clearButton) {
50 // テキストエリアの文字列を全消去
51      textArea.setText(null);
52    }
53  }
54 // アプリケーションの起動
55  public static void main(String[] args) {
56      new SwingApp4();
57  }
58 }
```

「SwingApp4」クラスを実行すると、以下のウィンドウが表示されます。



このアプリケーションでは、以下の処理を行っています。

1. 利用者の操作（[追加]ボタンのクリックというアクション）に応じてテキストフィールドに入力されている文字列をテキストエリアに追加
2. 利用者の操作（[消去]ボタンのクリックというアクション）に応じてテキストエリアの文字列をすべて消去

参考

eclipse での class ファイル使用

プロジェクトのプロパティのダイアログで、
『Java のビルド・パス』の『ライブラリ』タブの
~~『クラス・フォルダーの追加』ボタン、~~あるいは『外部クラス・フォルダーの追加』ボタンから
class ファイルのあるフォルダを指定してください。

その class ファイルがパッケージ指定されているものならパッケージ構成と同じサブフォルダを作って配置してください。