Swing によるデスクトップアプリケーション開発

Swing の基本（JFrame を使う）

　このテキストの内容は Java の Swing ライブラリを使って GUI アプリケーションを作りたいと考えている方を対象にしています。GUI アプリケーションとはウィンドウ、ボタン、メニューやアイコンといった部品を使ったユーザーインターフェイス（操作画面）を提供するデスクトップアプリケーションのことです。

　Web のサーブレットや JSP を学習する機会や使う機会が多い中で、あらためてオブジェクト指向の基本原則をしっかり学びたい方にもお勧めできます。

Swing とは

　Java には GUI アプリケーションを作るためのクラスライブラリ（クラス、インターフェイス群）として「AWT（Abstract Windows Toolkit）」と「Swing（スウィング）」があります。Swing は Java SE の開発当初から含まれている GUI ライブラリで**現在も標準ライブラリとして使用可能**です。

はるか昔のデスクトップアプリ開発では、AWT を利用することでしか GUI アプリケーションは作れませんでした。しかし、AWT には「動作する OS によって GUI 部品（ボタン、ラベル等）の見栄えが異なる」「利用できる GUI 部品の種類が少ない」といった問題があり、それらを改良（クラス拡張）する形で Swing が登場しました。

Swing の特徴としては AWT より描画が柔軟（軽量コンポーネント）、ボタンやテーブル、ツリービューなど、多数の GUI 部品が提供されている、そして**イベント駆動型のアーキテクチャ**である、といったことがあげられます。

JavaFX は分離された

一方で Swing の後継として、**JavaFX（FX）**が GUI ライブラリとして登場しましたが、Java 8（2014 年）で本格統合されるも、その後は**分離**（Java 11 以降は別インストール）されてしまいました。

これは、UI 開発の主流はすでに **Web 技術（HTML/CSS/JS,JSP）** や モバイルアプリ（iOS/Android） に移っていたため、JavaFX は活用される場が少なかったためです。また、JavaFX は GUI 関連の巨大なライブラリであり、**使用しない人にとっては邪魔**でした。こうした理由で、IT 企業における開発者の採用率があまり伸びなかったのが原因のようです。

用語の意味

用語	意味
コンポーネント	UI 部品の 1 つ 1 つ「ボタン、ラベルなど」
コンテナ	コンポーネントを「中に入れて配置できる容器」
レイアウト	コンポーネントの「並べ方」
イベント	ユーザーの操作（クリック、入力など）に対する反応「アクション」

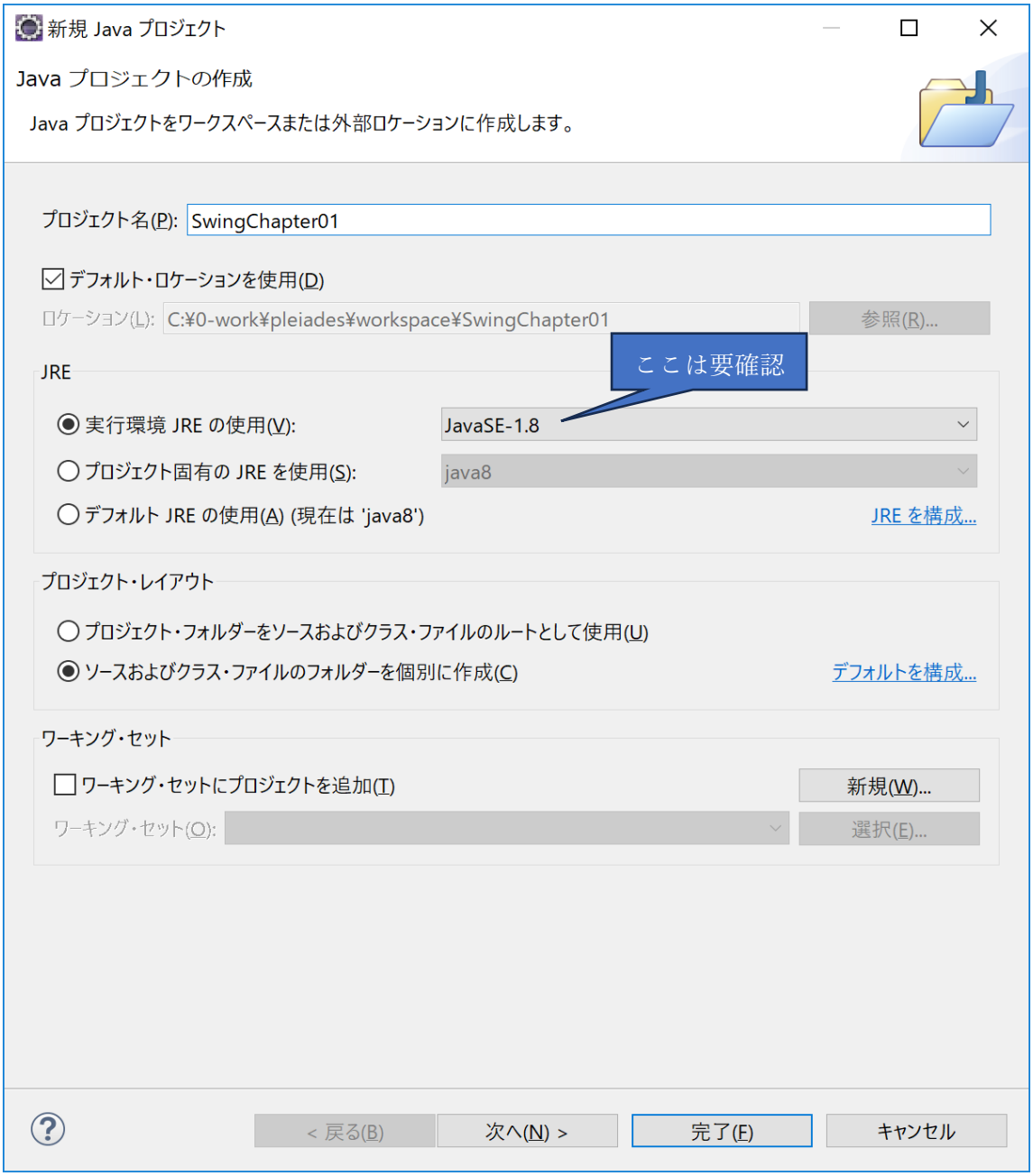
簡単な GUI の作成（BuilderTool を使わずに！）

Swing の基本的な技術や用語を説明する前に、まずは簡単な GUI アプリケーションを WindowBulder ツールを使わずに作ってみましょう。

プロジェクトの作成

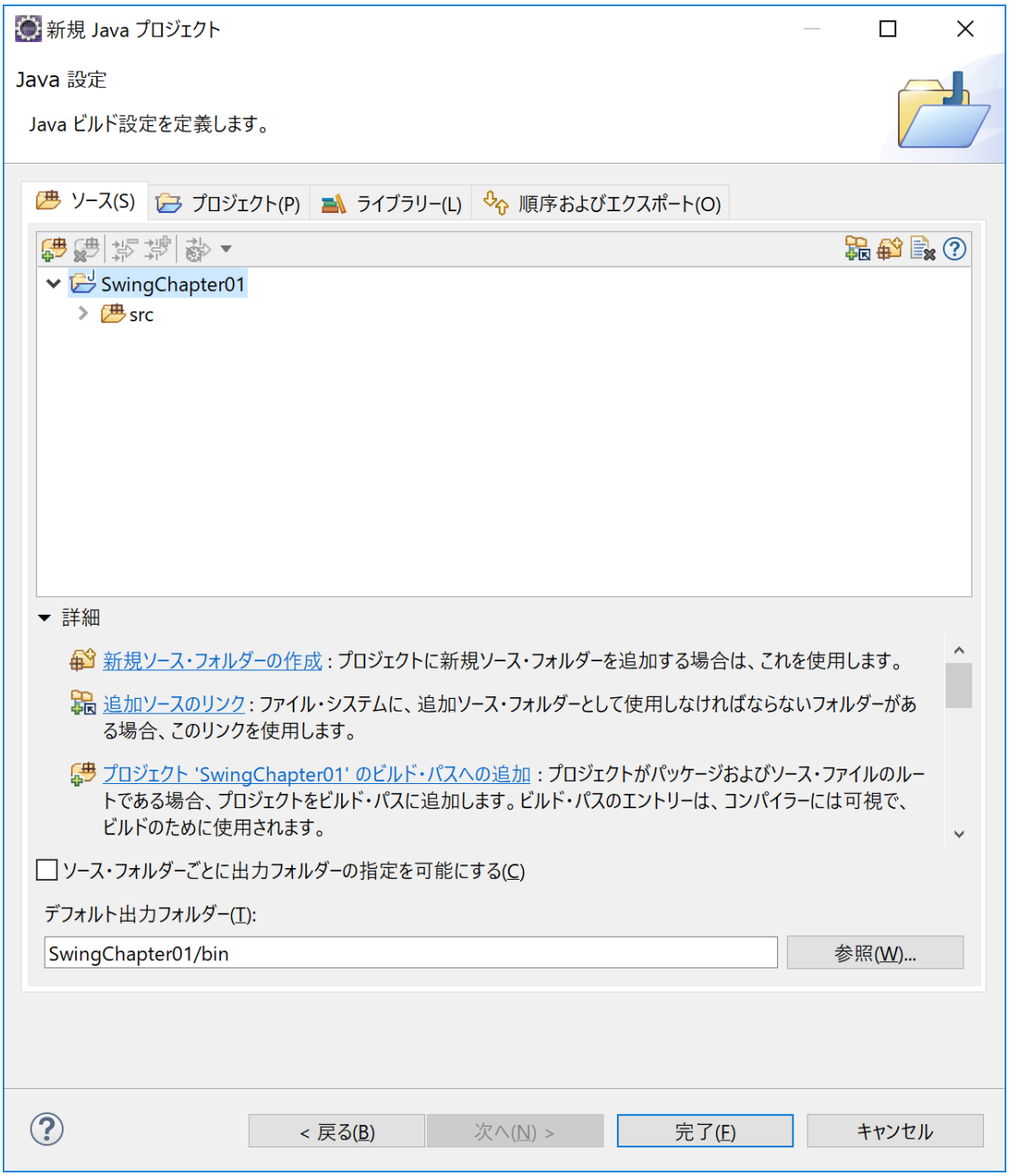
「SwingChapter01」という名前のプロジェクトを作成します。

1. ［ファイル］→［新規］→［Java プロジェクト］を選択
2. ［Java プロジェクトの作成］ダイアログでプロジェクト名に「SwingChapter01」を入力



3.［次へ］をクリック

・Java 設定を確認して［完了］をクリック



クラスの作成

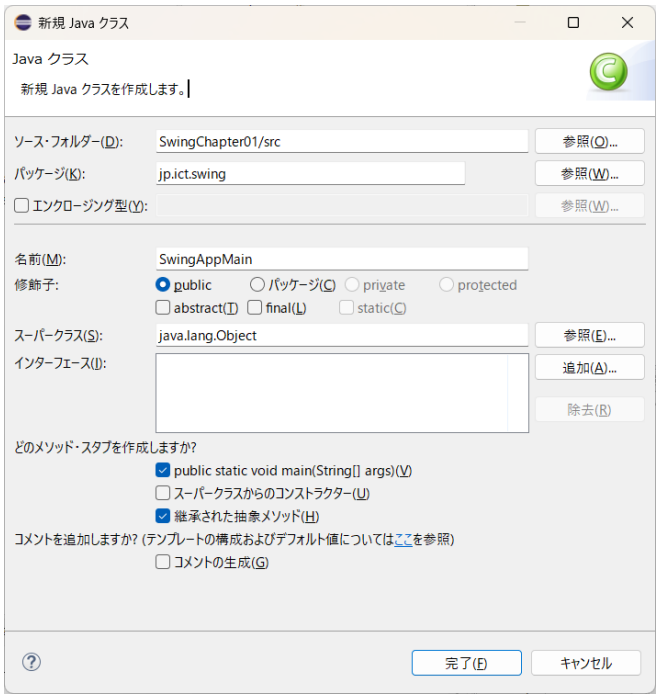
GUI アプリケーションの起動およびウィンドウの作成を担当するクラス「SwingAppMain」を作成します。

1.新規 Java クラスを作成する

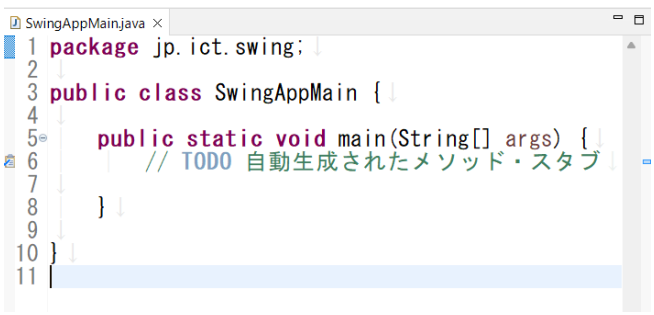
「SwingChapter01」プロジェクトを右クリック →［新規］→［クラス］を選択
[新規 Java クラス]ダイアログで

- パッケージに「jp.ict.swing」を入力
- 名前に「SwingAppMain」を入力
- public static void main(String[] args)をチェックし、

  ［完了］をクリック



[完了]をクリックすると、以下のようなクラスが作成されます。



2.以下のコードを入力する

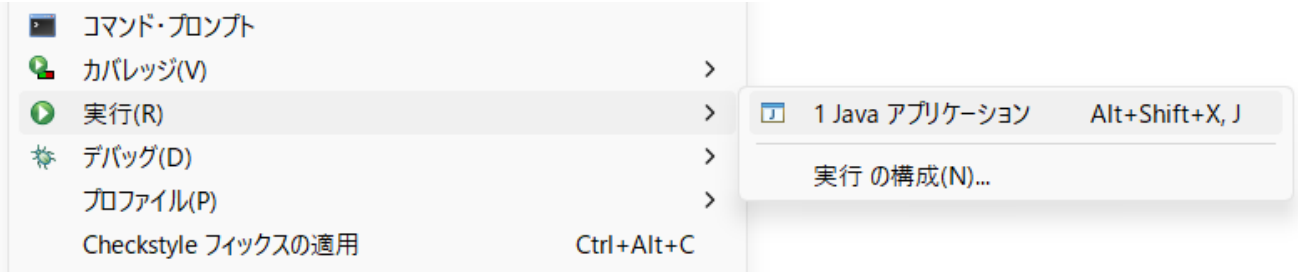
- ウィンドウの初期サイズ
- ウィンドウの表示場所（画面の中央に表示）
- ウィンドウの表示

SwingAppMain.java

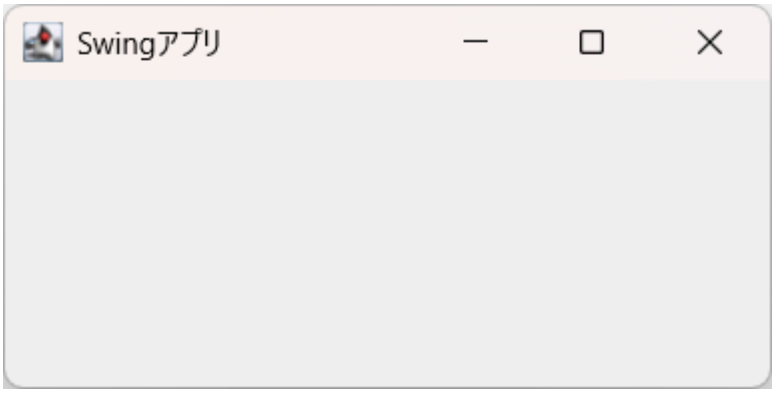
```java
package jp.ict.swing;

import javax.swing.JFrame;
public class SwingAppMain {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        JFrame mainFrame = new JFrame("Swingアプリ");
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setSize(320,160);
        mainFrame.setLocationRelativeTo(null);//ウィンドウを画面の中央に表示する
        mainFrame.setVisible(true);
    }

}
```

3.実行する

「編集画面」を右クリック →［実行］→［Java アプリケーション］をクリックして実行します。



タイトルバーに「Swing アプリ」と表示されたウィンドウが開きます。



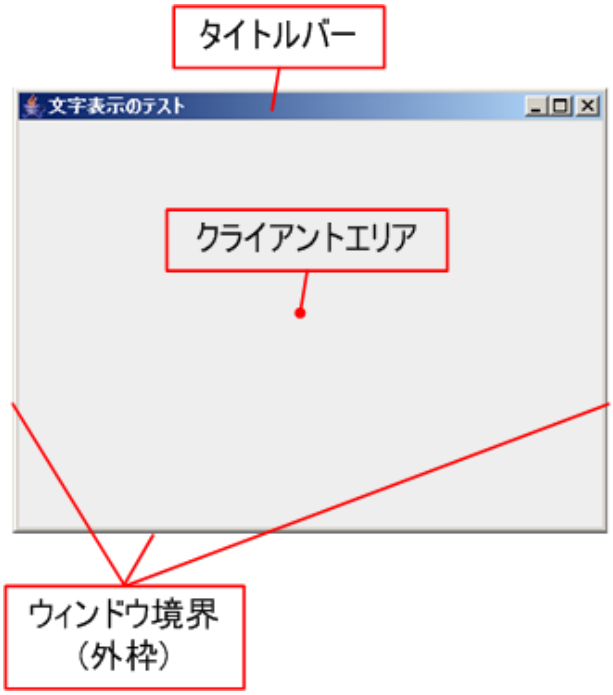
Swing で GUI アプリケーションを作成するための基礎知識は次回から解説しますが、今回登場した「JFrame」クラスについて、次回の予習として簡単に説明しておきます。

JFrame はウィンドウを作成するためのクラスです。そのため、**Swing を利用した GUI アプリケーションでは、初めに JFrame クラスのインスタンスを生成し、その後、生成したインスタンスのメソッドを介してウィンドウの初期設定をする**ことになります。

ソースコードの説明

コメントでそれぞれの行の役割を説明しています。

```
    public static void main(String[] args) {
    // TODO 自動生成されたメソッド・スタブ
    // JFrame クラスのインスタンスを生成
        JFrame mainFrame = new JFrame("サンプル");

    // 閉じるボタン押下時のアプリケーションの振る舞いを決定
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // ウィンドウの初期サイズ（幅、高さ）をピクセル単位で設定
        mainFrame.setSize(320, 160);

    // ウィンドウの表示場所を規定
        mainFrame.setLocationRelativeTo(null);

     // ウィンドウを表示
        mainFrame.setVisible(true);
    }
}
```



JFrame にコンポーネントを配置する

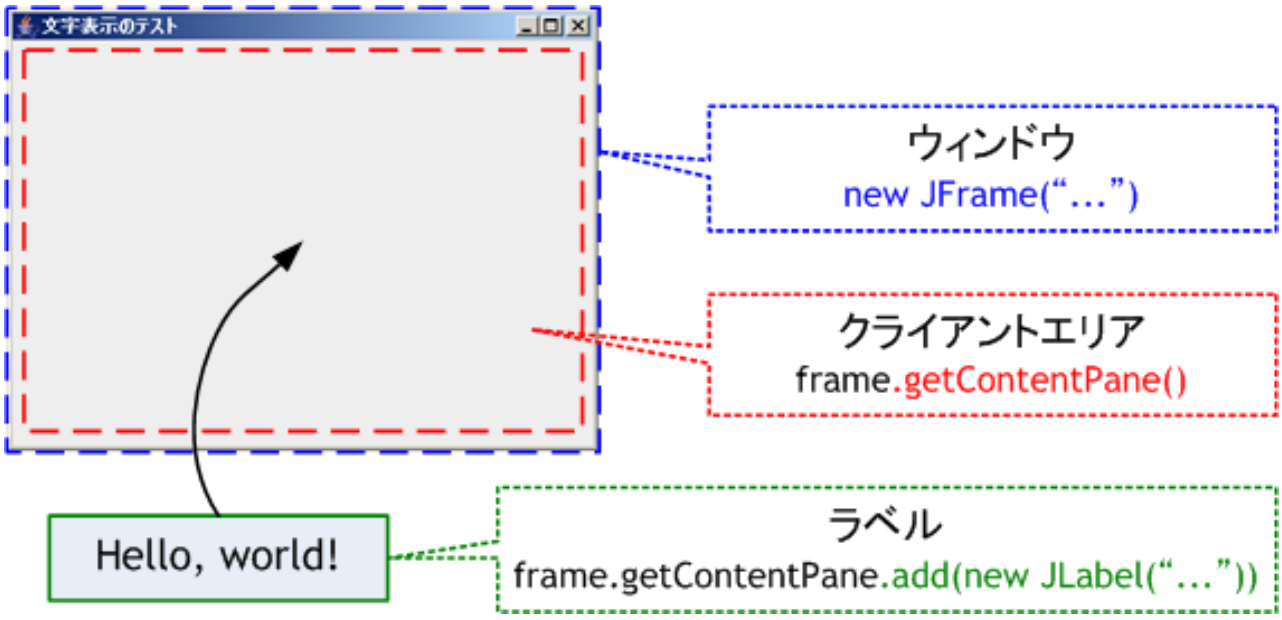
前回は、JFrame を使ってウィンドウを表示するまでの手順を説明しました。今回は、ボタンやラベルといった GUI コンポーネント（部品）を説明するとともに、それらを JFrame に配置する方法を説明します。

Swing コンポーネント(部品)の種類

実際にコンポーネントを配置する前に、Swing コンポーネントの種類について説明しておきます。Swing コンポーネントは以下の 3 つの階層に分類することができます。

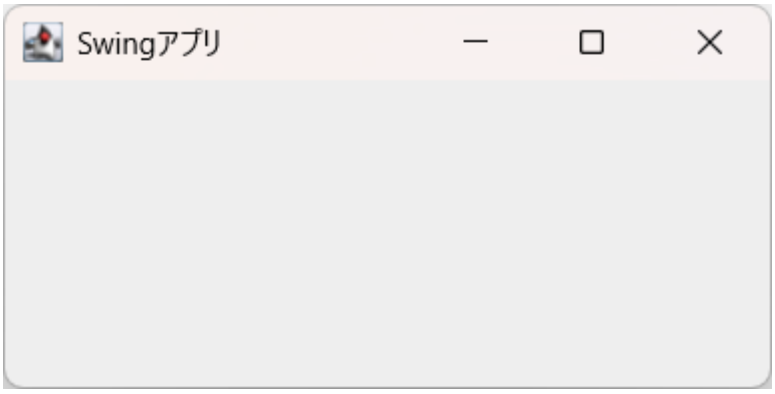
- **トップレベルコンテナ（JFrame）**
  メインウィンドウとなるコンポーネントです。ほかのコンポーネントの**土台**となります。タイトルバーや Window 枠。
- **中間コンテナ（contentPane）**
  コンポーネントを配置するための**容器**の役割を果たします。。
- **コントロール（JLable、JButton 等）**
  ボタンやラベルなどの個々の **GUI 部品**です。

Swing では、この 3 つの階層のコンポーネントを組み合わせてアプリケーションを作成していきます。JLabel などのコントロールを持ったアプリケーションを作成するには、まずトップレベルコンテナを作成し、そこに中間コンテナを配置し、さらに中間コンテナに JPanel 等のコントロールを配置するといった手順を踏む必要があります（簡単な UI ならば直接中間コンテナにコントロールを配置してもよいです）。



コンポーネントの配置

前回は、以下のようなウィンドウを作成しました。このウィンドウにラベルとボタンを配置してみましょう。



プロジェクトの選択

前回作成した「SwingChapter01」プロジェクト内の「src フォルダ」下にある「jp.ict.swing パッケージ」を展開します。

クラスの修正

SwingAppMain.java をダブルクリックしてソースコードを開きます。

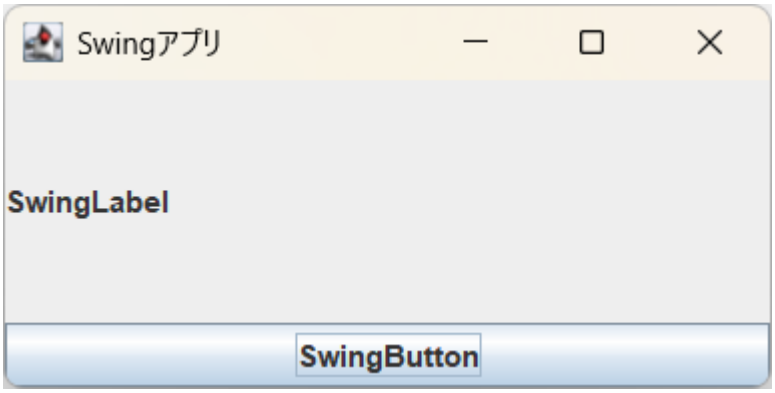
1.ソースに GUI 部品（コントロール）を追加します。
青枠の部分を追加します。

```
package jp.ict.swing;
import javax.swing.JFrame;
import java.awt.Container;
import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JLabel;
public class SwingAppMain {
	public static void main(String[] args) {
	// TODO 自動生成されたメソッド・スタブ
		JFrame mainFrame = new JFrame("サンプル");
		mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		mainFrame.setSize(320, 160);
		mainFrame.setLocationRelativeTo(null);
```

```
        // JFrame より ContentPane を取得
        Container contentPane = mainFrame.getContentPane();
        // ラベルのインスタンスを生成
        JLabel label = new JLabel("SwingLabel");
        // ボタンのインスタンスを生成
        JButton button = new JButton("SwingButton");
        // ラベルを ContentPane に配置
        contentPane.add(label, BorderLayout.CENTER);
        // ボタンを ContentPane に配置
        contentPane.add(button, BorderLayout.SOUTH);
        mainFrame.setVisible(true);
    }
}
```

実行

作成したコードを実行してみましょう。
「編集画面」を右クリック →［実行］→［Java アプリケーション］をクリックして実行します。



ラベルとボタンが配置されたウィンドウが表示されました。このように、JFrame（Window 枠）から contentPane（GUI 部品配置用中間コンテナ）を取得し、「ラベルやボタンのインスタンスを生成して中間コンテナに追加する」というコードを記述することでデスクトップアプリを実現できます。

※ ボタンに対するイベントアクションを実装していないのでボタンをクリックしても何も起きません。

各コンポーネントの役割と機能

配置した各コンポーネントについて解説をしていきましょう。

ContentPane

ContentPane とは、JFrame の表示領域です。
JFrame に表示したいコンポーネントは、ContentPane に追加する必要があります。

・ContentPane の取得

```
Container contentPane = mainFrame.getContentPane();
```

上記コードの getContentPane メソッドで、ContentPane を取得しています。ContentPane は、Container クラスの型で取得できますが、JFrame の中で JPanel として生成されています。

・JPanel クラス

JPanel は、複数のコンポーネントを配置できるコンテナです（コンポーネントを張り付ける台紙をイメージしてください）。Swing では、JPanel のような汎用コンテナにコンポーネントを張り付ける感覚で画面を作成していきます。JPanel を置くことで、レイアウトの分割や構造を整理できます。

・コンポーネントの配置

```
contentPane.add(label, BorderLayout.CENTER);
contentPane.add(button, BorderLayout.SOUTH);
```

上記コードで、ContentPane にコンポーネントを追加することができます。add メソッドの第 1 パラメータでは、ContentPane に配置したいコンポーネントを指定しています。第 2 パラメータでは、コンポーネントを配置する位置を指定しています。
BorderLayout.CENTER は、BorderLayout クラスの定数です。
BorderLayout.CENTER は、コンポーネントを ContentPane の中心に配置することを意味しています。BorderLayout.SOUTH は ContentPane の下側に配置します。

レイアウトマネージャ

コンテナ内のコンポーネントのレイアウトに関してはレイアウトマネージャが管理しています。Swing での GUI の実装には基本的にレイアウトマネージャを使います。レイアウトマネージャには BorderLayout や、GridLayout などさまざまな種類のものがあります。これらのレイアウトマネージャを使用することでコンテナ内の統一的なレイアウトを簡単に実現することができます。ここでは、BorderLayout クラスについて解説します。

・BorderLayout クラス

BorderLayout は、以下のように 5 つの領域に分けてコンポーネントを配置します。1 つの表示領域には、1 つのコンポーネントを配置することができます。設定されたコンポーネントは、その領域を満たすサイズで表現されます。

Jlabel クラス

Jlabel クラスは、文字列やイメージの表示に使用するクラスです。

・インスタンスの生成

```
JLabel label = new JLabel("SwingLabel");
```

上記コードで、JLabel クラスのインスタンスを生成しています。コンストラクタには、JLabel で表示したい文字列を設定しています。また、表示したい文字列はインスタンスを生成した後で設定することも可能です。その場合は、JLabel クラスの setText メソッドを使用します。

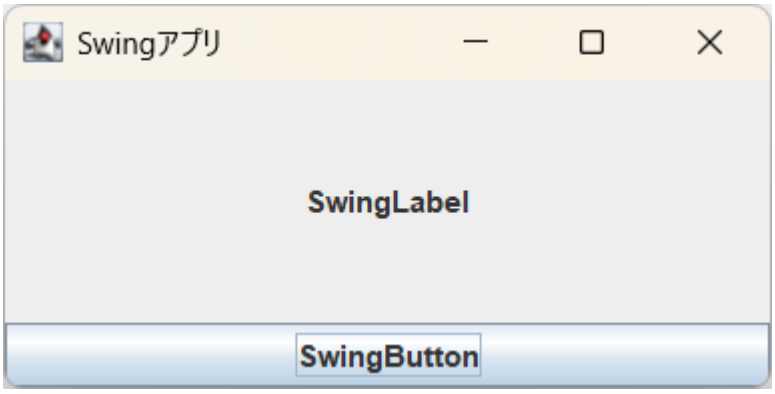
【実習】

表示文字列の配置位置の制御

JLabel は、表示文字列の配置位置を制御することができます。［SwingLabel］の文字列を中央に配置してみましょう。以下のコードを追記して実行してみてください。

- javax.swing.SwingConstants をインポート
- label インスタンスを生成した後の部分に以下のコードを追加

```
label.setHorizontalAlignment(SwingConstants.CENTER);
```



［SwingLabel］が中央に表示されましたでしょうか。

setHorizontalAlignment は、表示文字列の横軸に沿った配置位置を制御するメソッドです。パラメータの SwingConstants.CENTER で表示文字列を配置する位置を指定します。

SwingConstants は、JLabel が継承しているインターフェイスで、コンポーネントの配置に関する定数が定義されています。JLabel、JButton 等の SwingConstants を継承したクラスは、SwingConstants の定数を使用することで、配置の指定に関する表現が統一されています。

JButton クラス

JButton クラスは、プッシュボタンを表現するクラスです。JLabel 同様、イメージを表示することも可能です。

・インスタンスの生成

```
JButton button = new JButton("SwingButton");
```

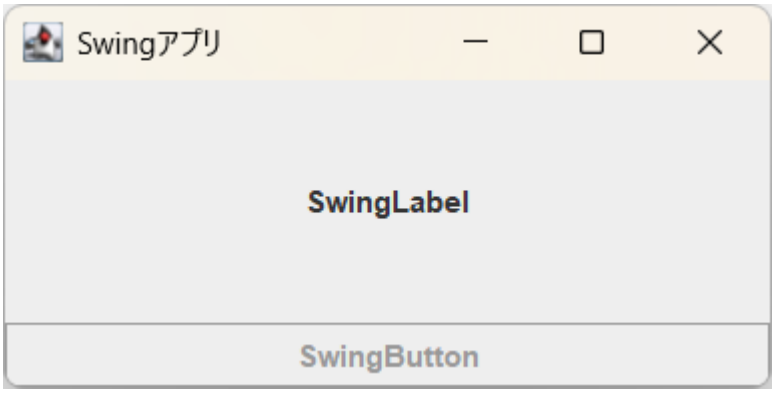
上記コードで、JButton クラスのインスタンスを生成しています。コンストラクタには、JButton で表示したい文字列を設定しています。JLabel と同じく、setText メソッドを使用することで後から表示文字列を設定することができます。

【実習】

有効無効の制御

ボタンの有効無効を制御するには、setEnabled メソッドを使用します。以下のコードを追記して実行してみてください。

- button インスタンスを生成した後の部分に以下のコードを追加
  button.setEnabled(false);



［SwingButton］が無効になりました。setEnabled メソッドは、パラメータの boolean 値で有効無効を指定することができます。true の場合が有効、false の場合が無効を意味します。

この setEnabled メソッドは、トップレベルコンテナ（JFrame 等）以外のすべての Swing コンポーネントが継承している JComponent クラスのメソッドです。よって、トップレベルコンテナ以外の Swing コンポーネントは setEnabled メソッドにより、有効無効を制御することができます。よって、JLabel も JButton と同様に setEnabled メソッドを使用して有効無効を制御することができます。

まとめ

今回は、ラベルやボタンといったコンポーネントを持つ GUI アプリケーションを作成する方法を解説しました。以下にソースコードの全体を載せておきます。

今回のソースコード

```
package jp.ict.swing;

import java.awt.BorderLayout;
import java.awt.Container;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.SwingConstants;    //koko
```

```
public class SwingAppMain {

	public static void main(String[] args) {
		// TODO 自動生成されたメソッド・スタブ
		JFrame mainFrame = new JFrame("Swing アプリ");
		mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		mainFrame.setSize(320,160);
		mainFrame.setLocationRelativeTo(null);//ウィンドウを画面の中央に表示する

		 // JFrame より ContentPane を取得
		 Container contentPane = mainFrame.getContentPane();
		// ラベルのインスタンスを生成
		JLabel label = new JLabel("SwingLabel");
		label.setHorizontalAlignment(SwingConstants.CENTER);    //koko
		// ボタンのインスタンスを生成
		JButton button = new JButton("SwingButton");
		button.setEnabled(false);                               //koko
		// ラベルを ContentPane に配置
		 contentPane.add(label, BorderLayout.CENTER);
		 // ボタンを ContentPane に配置
		contentPane.add(button, BorderLayout.SOUTH);

		mainFrame.setVisible(true);
	}
}
```

※GUI をデザインして手作業で実装するのはとても大変です。そこで、WindowBuilder ツールなどを使って効率よく開発することが推奨されます。

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説明してきました。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説明していきます。

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。

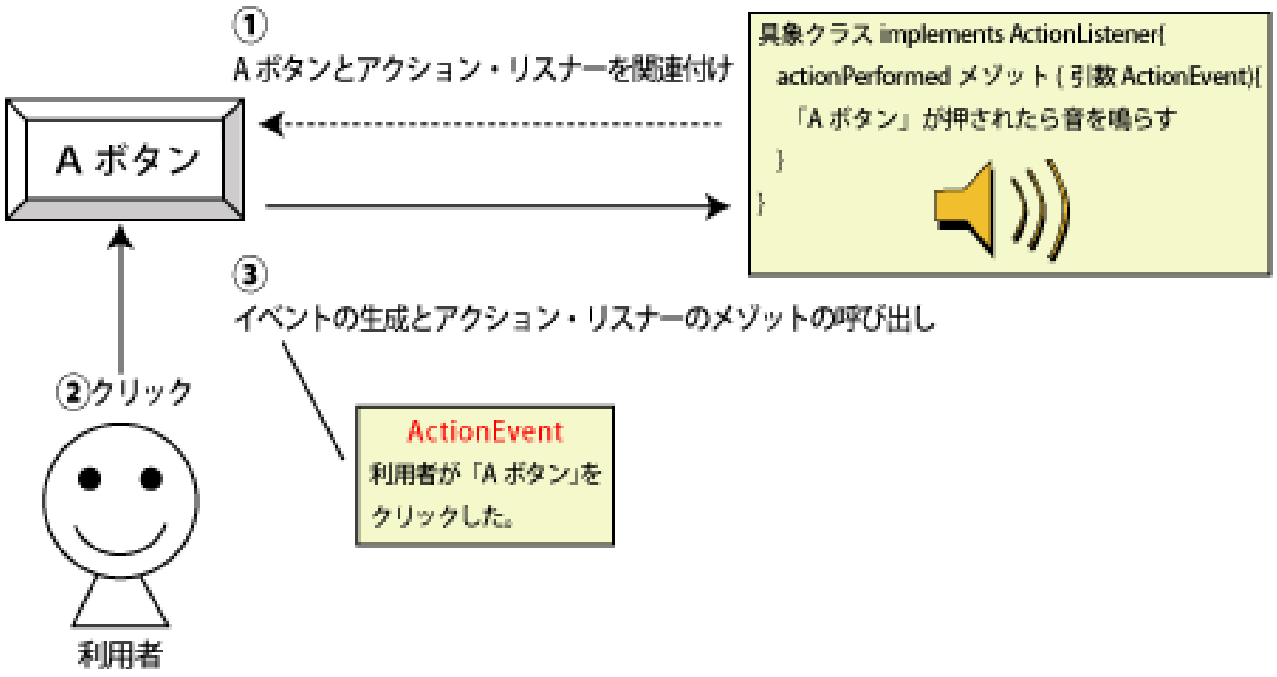
イベント

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。



WindowBuilder による実装

Java Swing における「アクション（Action）」の定義は、特に WindowBuilder などの GUI 設計ツールでボタンやメニューの共通動作を定義するためによく使われます。

Action ソースコードの構成

- private class SwingAction extends AbstractAction
  - AbstractAction は Swing の アクション（動作）のテンプレートを定義する抽象クラスです。
  - このクラスを継承して、「何が起きたときに何をするか」を定義します。
  - private にしているのは、この SwingAction をクラス内部だけで使うからです。

- putValue(NAME, "SwingAction");
  - NAME → アクションの名前（例：ボタンのラベルになる）
  - "SwingAction" → ラベルや説明文など、**UI に表示される名前**に使われます

- putValue(SHORT_DESCRIPTION, "Some short description");
  - SHORT_DESCRIPTION → ツールチップ（マウスを当てたときの説明）として使われます

- public void actionPerformed(ActionEvent e)
  - アクションが実行されたときの処理をここに書きます。

例えばボタンをクリックしたときに HelloWolrd を出力する処理は以下のように実装します。

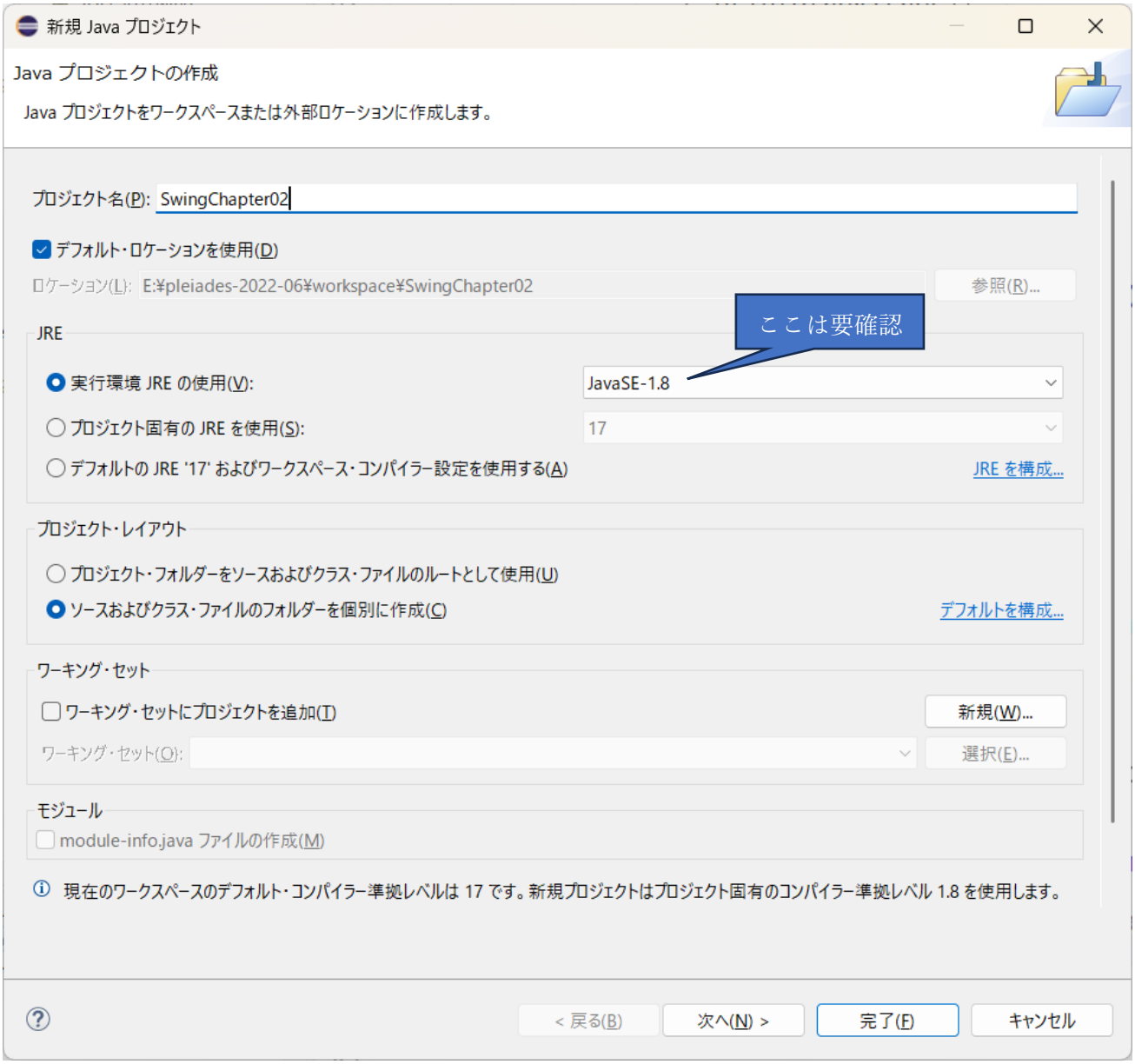
```
private class SwingAction extends AbstractAction {
    public SwingAction() {
        putValue(NAME, "SwingAction");
        putValue(SHORT_DESCRIPTION, "Some short description");
    }
    public void actionPerformed(ActionEvent e) {
        lblNewLabel.setText("HelloWorld");
    }
}
```

【実習】

WindowBuilder（Swing デザイナー）を使ったデスクトップアプリケーションの開発の流れを一通り実習します。

プロジェクトの作成

「SwingChapter02」という名前のプロジェクトを作成します。
1. ［ファイル］→［新規］→［Java プロジェクト］を選択します。
2. ［Java プロジェクトの作成］ダイアログでプロジェクト名に「SwingChapter02」を入力します。



クラスの作成

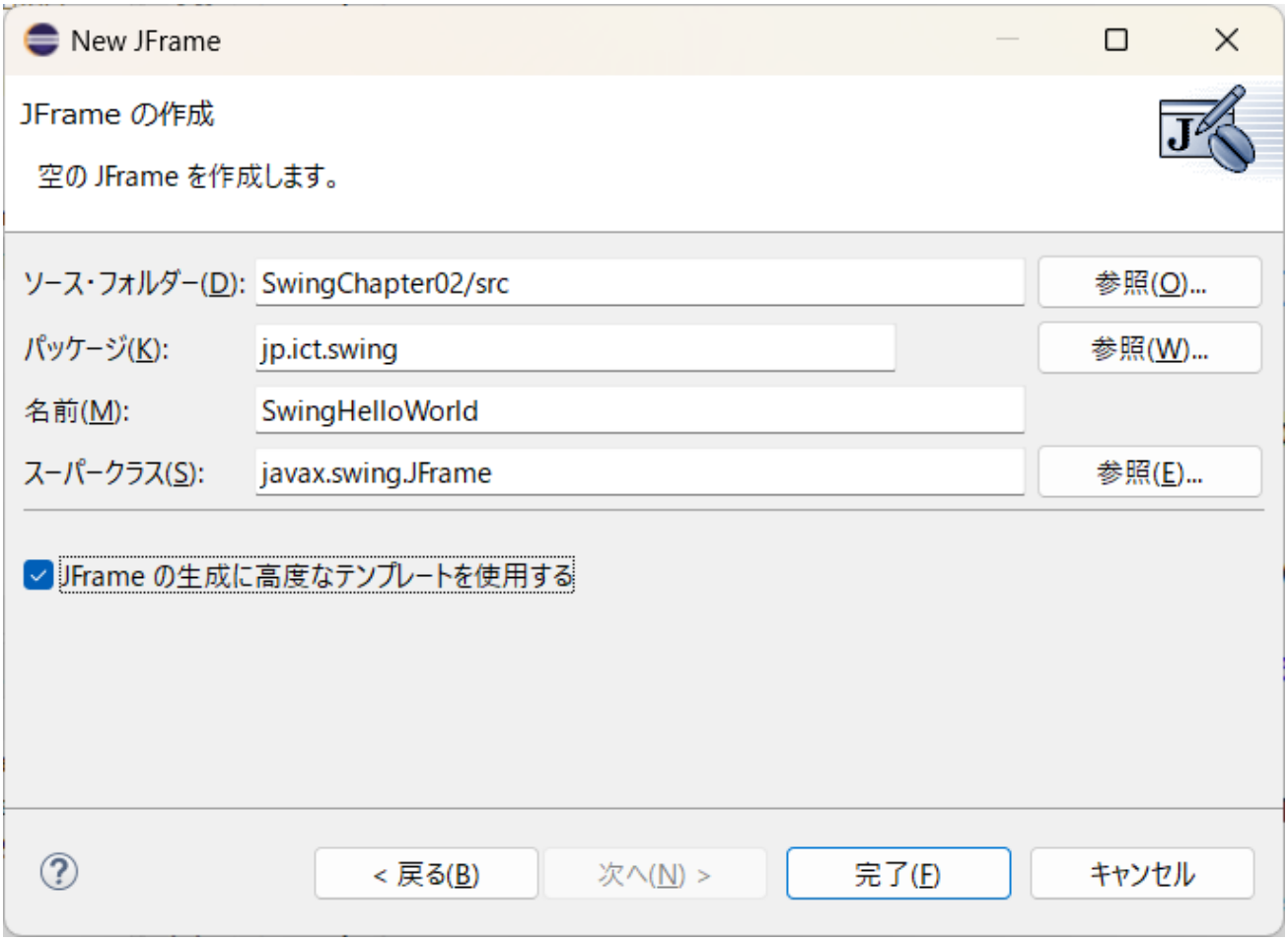
WindowBuilder による GUI アプリケーションのクラス「SwingHelloWorld」を作成します。

1.「SwingChapter02」プロジェクトを右クリック →［新規］→［その他］→［WindowBuilder］→［Swing デザイナー］→［JFrame］を選択します

2. [JFrame の作成]ダイアログで

- パッケージに「jp.ict.swing」を入力
- 名前に「SwingHelloWorld」を入力

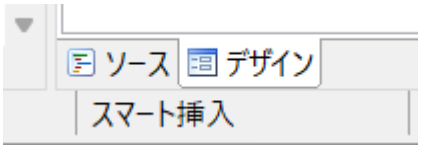
［完了］をクリックします



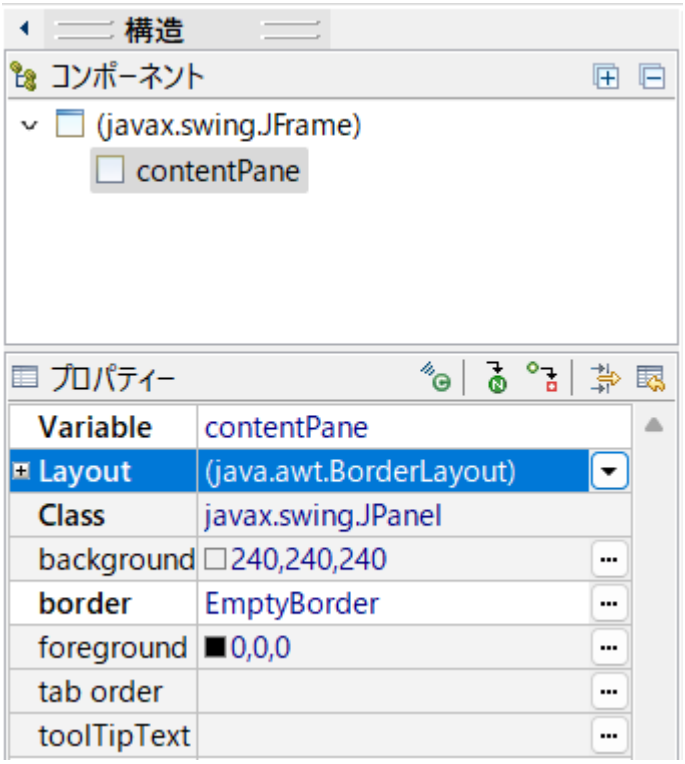
デザイナーへ切り替え

Swing デザイナーのデザイン画面に切り替えます。

1.「Eclipse エディター画面の下」にある切り替えタブの「デザイン」をクリックします。

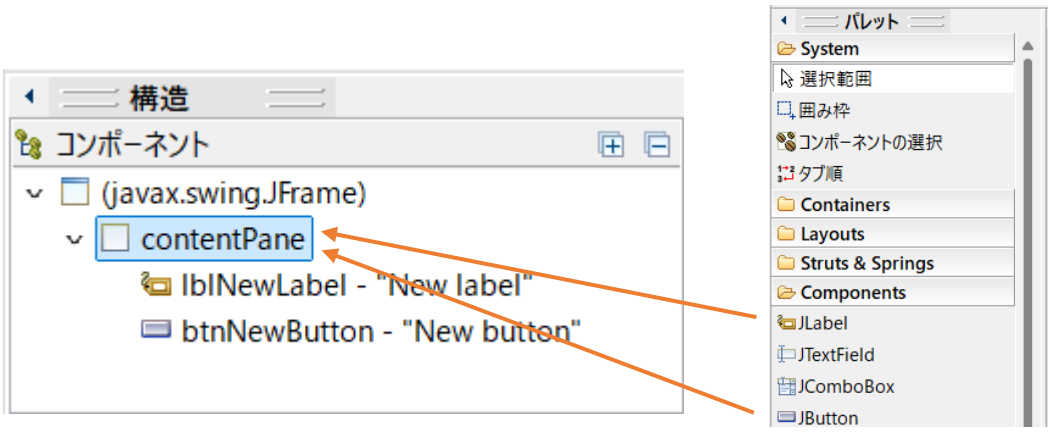


2.コンポーネント画面の「contentPain」のプロパティの Layout を BorderLayout に変更します。Swing での GUI のデザインは基本的にレイアウトマネージャを使います。



3.パレット内の Components フォルダにある JLabel と JButton をコンポーネント画面の「contentPain」に配置します。

※パレット内の JLabel をクリックしコンポーネント画面の contentPain をクリックします。同様に JButoon もクリックして配置します。



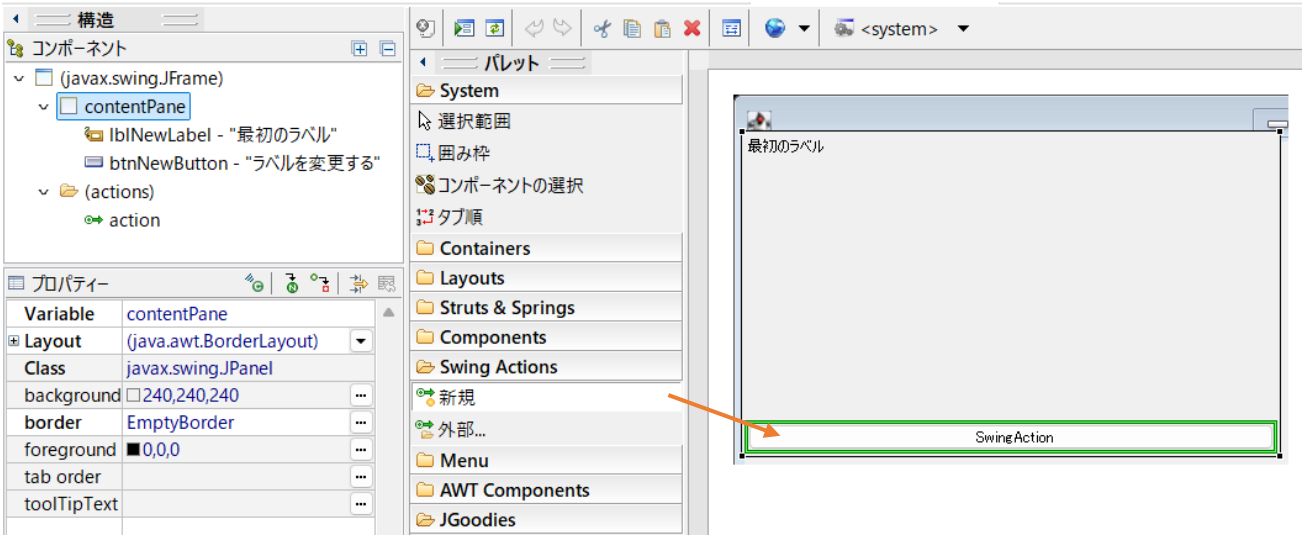
ラベルとボタンは自動的に BorderLayout の「North」と「South」の位置に配置されます。位置が不適切であればプロパティから変更してください

※それぞれのプロパティからラベル名を変更してください。

4.パレット内の SwingActions フォルダにある「新規」のショートカットをボタンに配置します。

※このとき「新規」をクリックした後 Eclipse 右側のデザイン画面をクリックして配置します。これによりソースコードに AbstractAction が生成されます。

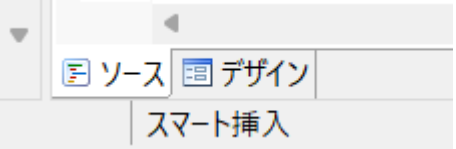
※このようにデザイン画面を利用しないと配置できない部品があります！！



エディターへ切り替え

Eclipse のソースコード編集画面に切り替えます。

1.「Eclipse エディター画面の下」にある切り替えタブの「ソース」をクリックします。



2. ①アクションが実行されたときの処理を実装します。

```
private class SwingAction extends AbstractAction {
    public SwingAction() {
        putValue(NAME, "ラベルを変更する");
        putValue(SHORT_DESCRIPTION, "HelloWorldを出力します");
    }
    public void actionPerformed(ActionEvent e) {
        lblNewLabel.setText("HelloWorld");          //koko
    }
}
```

※ putValue（NAME を設定しないとボタンの表示名が変更されません！

3. ②インスタンス変数の宣言を追加します。

```
public class SwingHelloWorld extends JFrame {

    private JPanel contentPane;
    private final Action action = new SwingAction();

    private JLabel lblNewLabel; //koko

    /**
     * Launch the application.
     */
```

4. ③ローカル変数の宣言を削除します。

```
public SwingHelloWorld() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

    setContentPane(contentPane);
    contentPane.setLayout(new BorderLayout(0, 0));

    lblNewLabel = new JLabel("最初のラベル");      //koko
    contentPane.add(lblNewLabel, BorderLayout.NORTH);

    JButton btnNewButton = new JButton("ラベルを変更する");
    btnNewButton.setAction(action);
    contentPane.add(btnNewButton, BorderLayout.SOUTH);
}
```

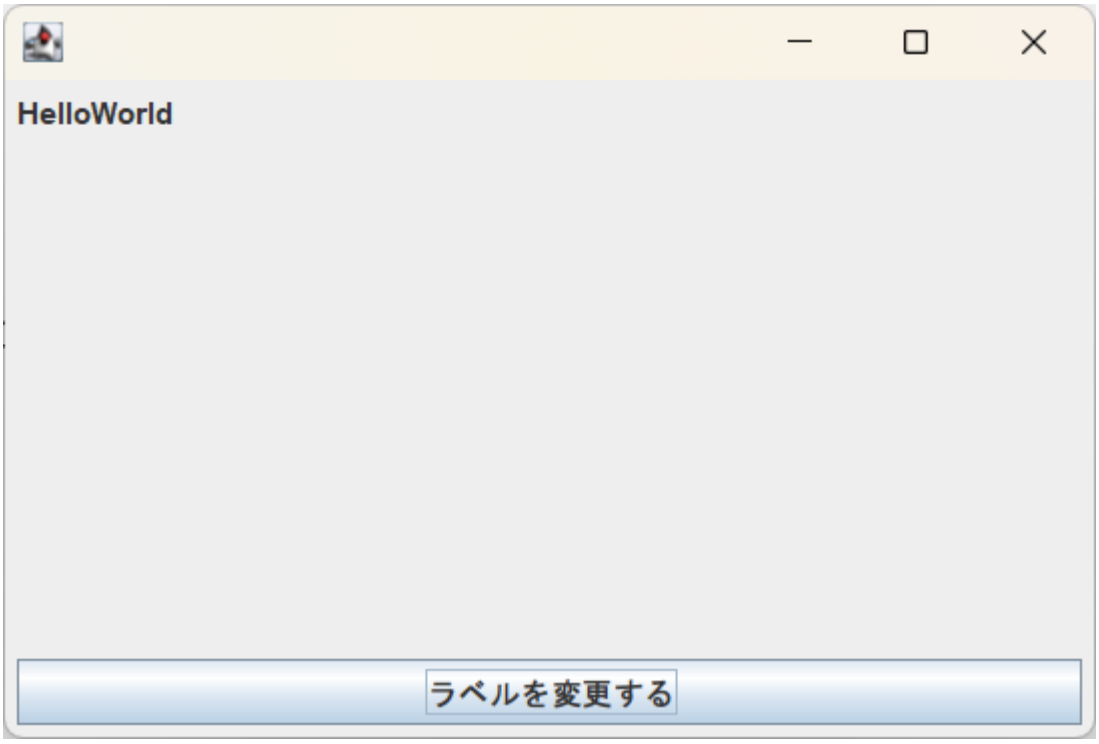
5.実行します。

作成したコードを実行してみましょう。

※ 保存するのを忘れないように

「ソースコード編集画面」を右クリック →［実行］→［Java アプリケーション］をクリックして実行します。

ボタンをクリックすると「最初のラベル」の文字が「HelloWorld」に変更されます。



WindowBuilder を使った場合 GUI をデザインした後のボタンの action を実装するには

３つのソースコードの変更が必要です。← **[重要]**

① アクションが実行されたときの処理を実装
② インスタンス変数の宣言を追加
③ ローカル変数の宣言を削除

【演習】

以下の GUI になるようデザインとソースを変更して下さい。
クリアボタンをクリックすると「最初のラベル」に表示が戻ります。

【最後に】

Swing の設計には、**オブジェクト指向の基本原則**がしっかりと取り入れられています。

オブジェクト指向の特徴	Swing での具体例
クラスとオブジェクト	JButton, JLabel, JFrame など、すべてクラスで構成され、インスタンス化して使います。例：new JButton(″OK″)
カプセル化	各コンポーネントは内部の状態（例えばテキストや位置など）を持ち、setText() や setBounds() などのメソッドで操作します。直接フィールドにアクセスできないように設計されています。
継承	JComponent → JLabel, JButton, JPanel など、Swing の多くのクラスは共通のスーパークラスから継承されています。
ポリモーフィズム	JComponent 型の変数で、JButton, JLabel, JPanel など異なる種類のコンポーネントを扱うことができます。
イベント駆動型設計	ActionListener や MouseListener など、インターフェースを使ってイベント処理をオブジェクトとして分離・実装できます。

Swing でデスクトップアプリケーションを作成することでオブジェクト指向の概念を再認識してください。

【補足】

Swing での GUI デザインの実装には基本的にはレイアウトマネージャを使いますが、もちろんこれを無効にして自由にコンポーネントを配置することも可能です。以下にレイアウトマネージャを使わない GUI 設計の手順を示します。

※レイアウトマネージャを無効にして自由にコンポーネントを配置する手順

contentPane → プロパティ → Layout → BorderLayout に変更

contentPane → JPanel を配置 → 「center」位置に変更

「center」位置の panel → プロパティ → Layout → 絶対レイアウトに変更（absolute）

これで panel に自由にコンポーネントの配置が可能となります。

（配置例）