Swing のキホン JFrame を使う

この実習は Java によるプログラミングを始めて間もない初心者、かつ 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）を提供するアプリケーションのことです。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 Java の基本を学習したい方にもお勧めできます。

本実習を通じて、簡単に GUI アプリケーションが作れることを皆さんに体感していただければと思います。

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。

少し昔の話になりますが、Java2（JDK1.2）が登場する以前は、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（Swing は標準でサポートされていませんでした）。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。

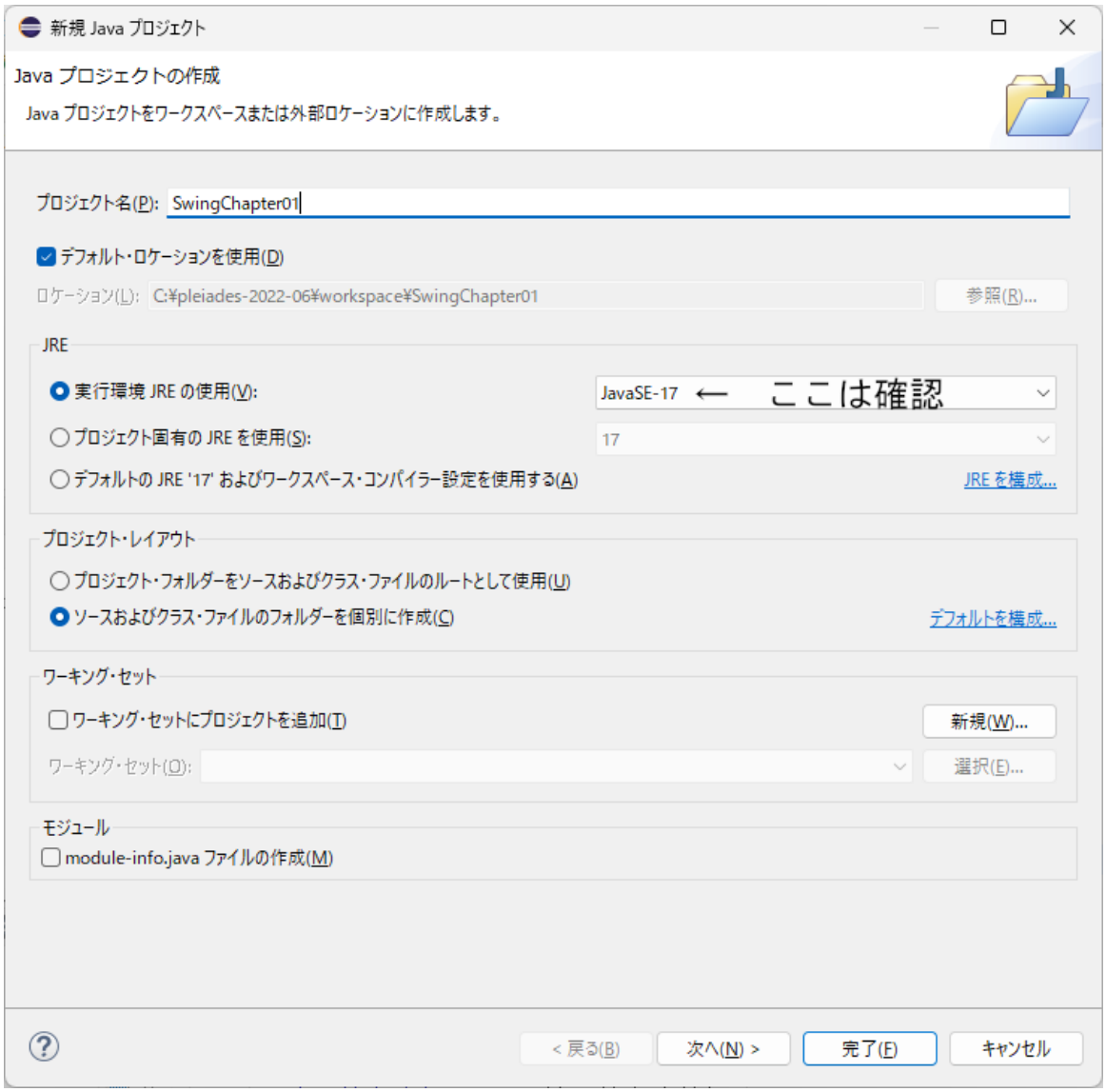
AWT、Swing のどちらを使っても同じ GUI アプリケーションを作成することは可能ですが、最近は Swing を利用して作られることの方が多いようです。

まずは体験からスタート

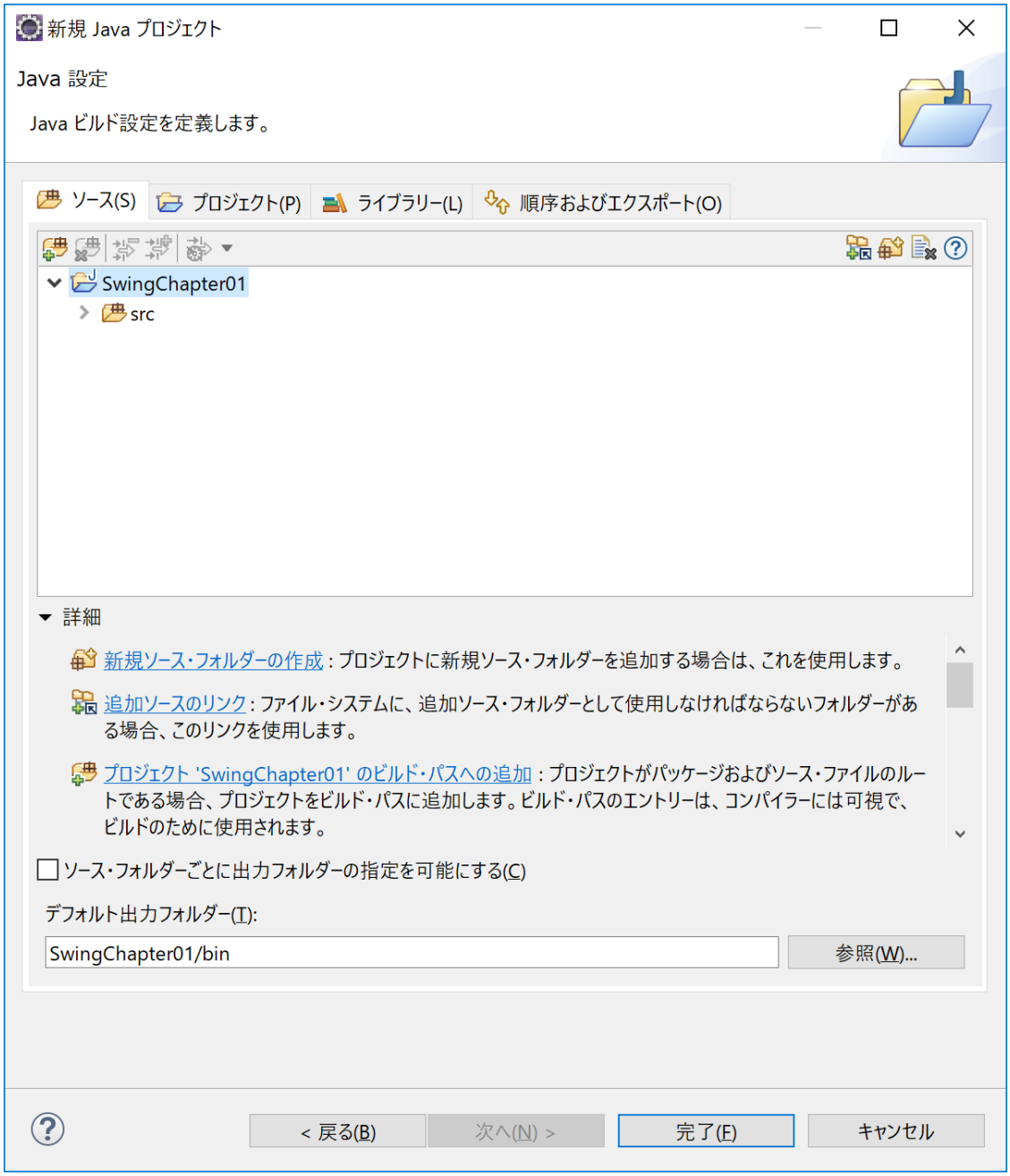
Swing の基本的な技術や用語を説明する前に、まずは簡単な GUI アプリケーション（ウィンドウを作成する）を作ってみましょう。

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］でプロジェクト名に「SwingChapter01」を入力



3.［次へ］を押下 ・Java 設定を確認して［完了］を押下



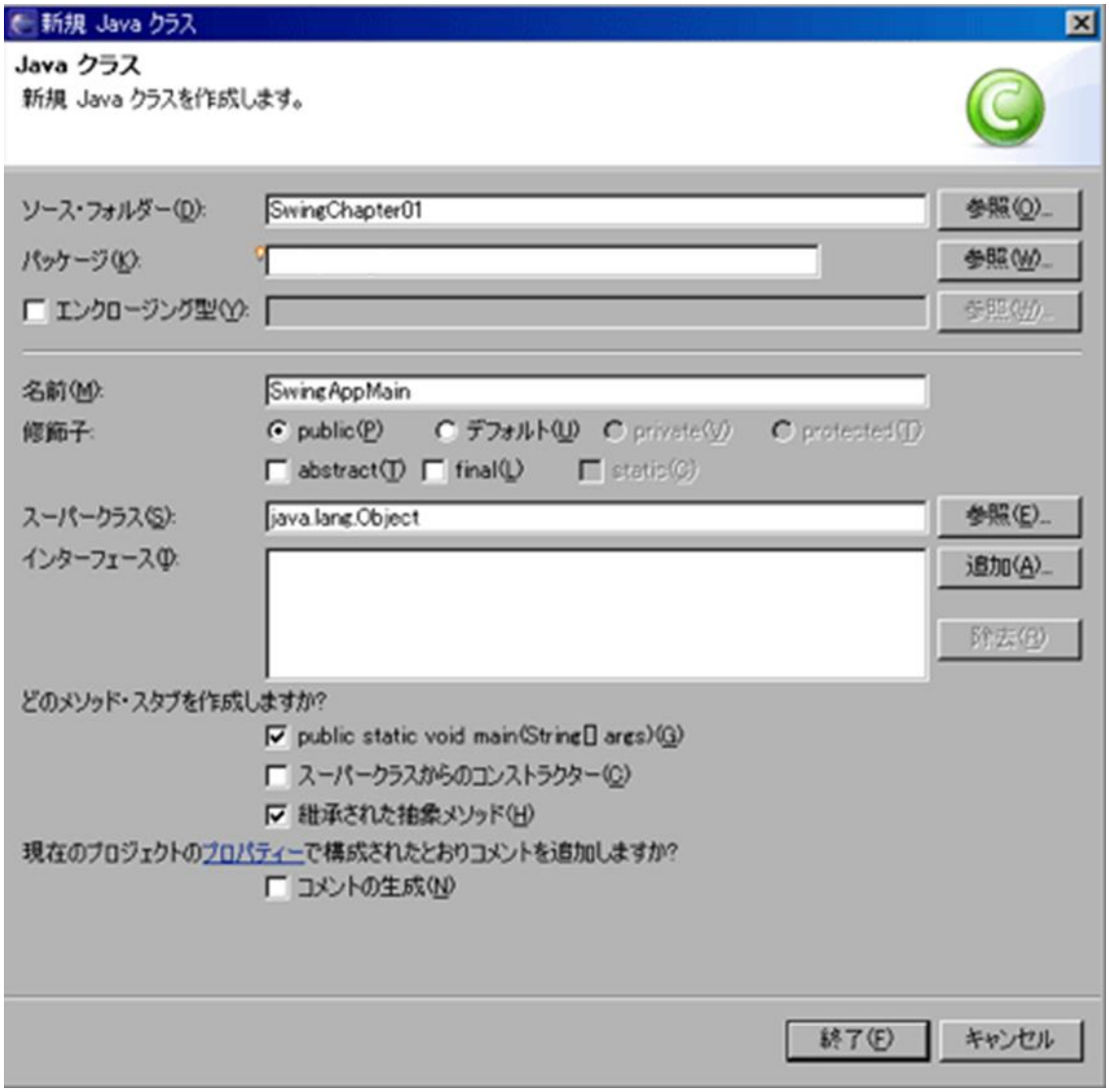
クラスの作成

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。

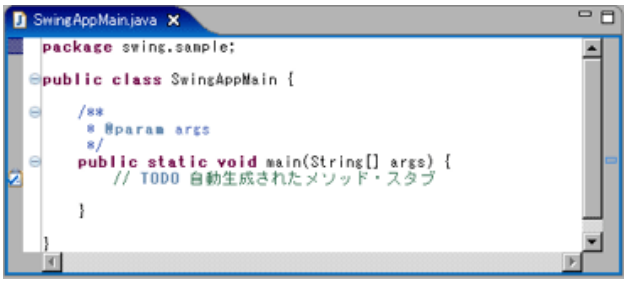
1.新規 Java クラスを作成。
［ファイル］→［新規］→［クラス］を選択、[新規 Java クラス]ダイアログで　名前：SwingAppMain を設定し［完了］を押下

- 名前に「**SwingAppMain**」を入力
- public static void main(String[] args)をチェックし、

［終了］を押下

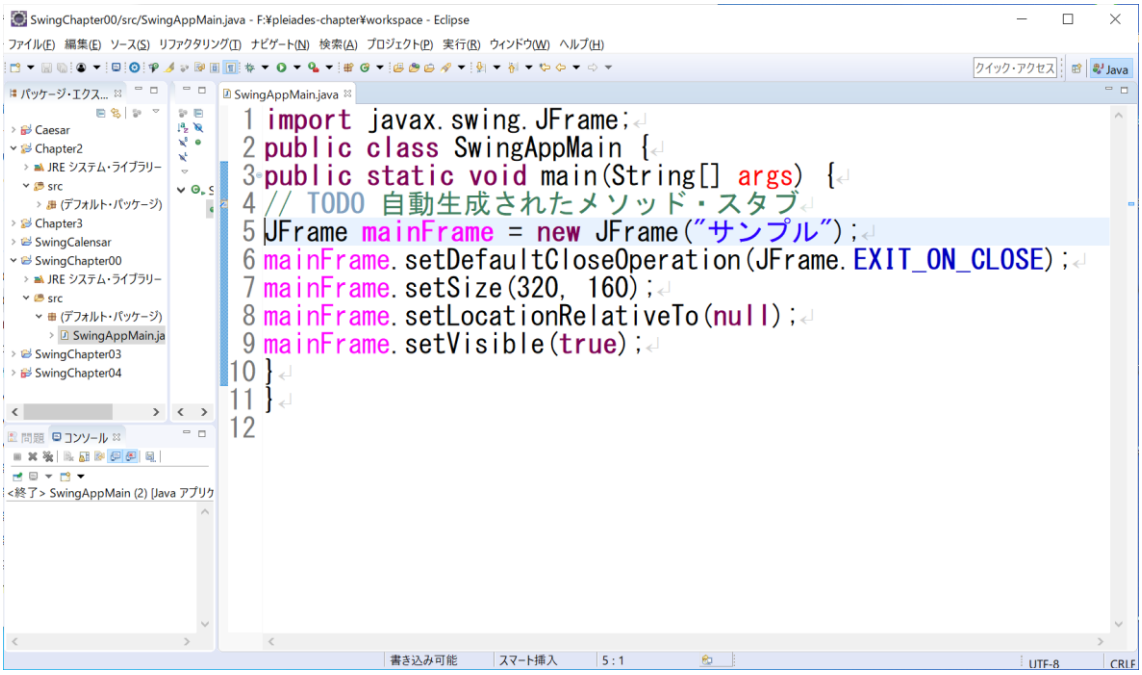


[完了]を押下すると、以下のようなクラスが作成されます。

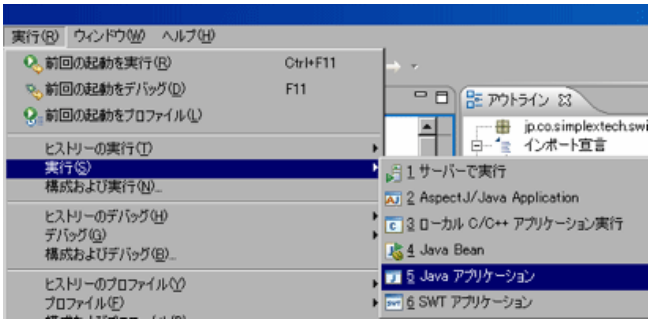


2.以下のコードを入力

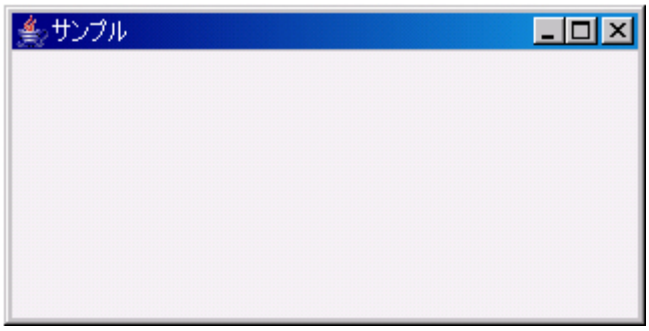
- ウィンドウの初期サイズ
- ウィンドウの表示場所（画面の中央に表示）
- ウィンドウの表示



4.［実行］→［Java アプリケーション］を選択して、実行する



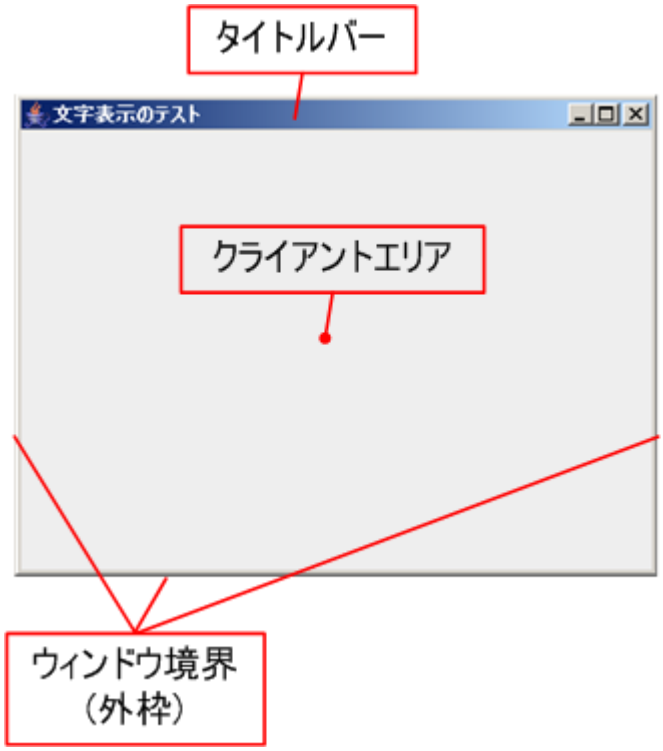
実行した結果、タイトルバーに「サンプル」と表示されたウィンドウが開きます。



非常にシンプルな GUI アプリケーションですが、数行のソースコードを記述するだけで Java でも簡単に GUI アプリケーションが作れることを体験していただけたと思います。

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ご紹介しておきましょう。

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、必ず初めに 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。



ウィンドウを表示するだけのシンプルな GUI アプリケーションを作る方法を解説しました。この例では起動メソッド内にすべてのコードを記述しているため、オブジェクト指向の考え方に基づき GUI クラスを作成してインスタンス化するようコードを変更します。

プロジェクトは既存を使用

前回作成したに「SwingChapter01」という名前のプロジェクトに新たにクラスを追加します。

クラスの作成

パッケージ・エクスプローラから

「SwingChapter01」を右クリック → 「新規」 → 「クラス」

で作成します。

クラス名は「**SwingApp1**」とします。

```
1
2 import javax.swing.JFrame;
3
4 public class SwingApp1 extends JFrame{
5    public static void main(String[] args) {
6       new SwingApp1();
7    }
8    public SwingApp1() {
9       JFrame mainFrame = new JFrame("サンプル");
10      mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11      mainFrame.setSize(320, 200);
12      mainFrame.setLocationRelativeTo(null);
13
14      mainFrame.setVisible(true);
15 }
16 }
```

JFrame にコンポーネントを配置する

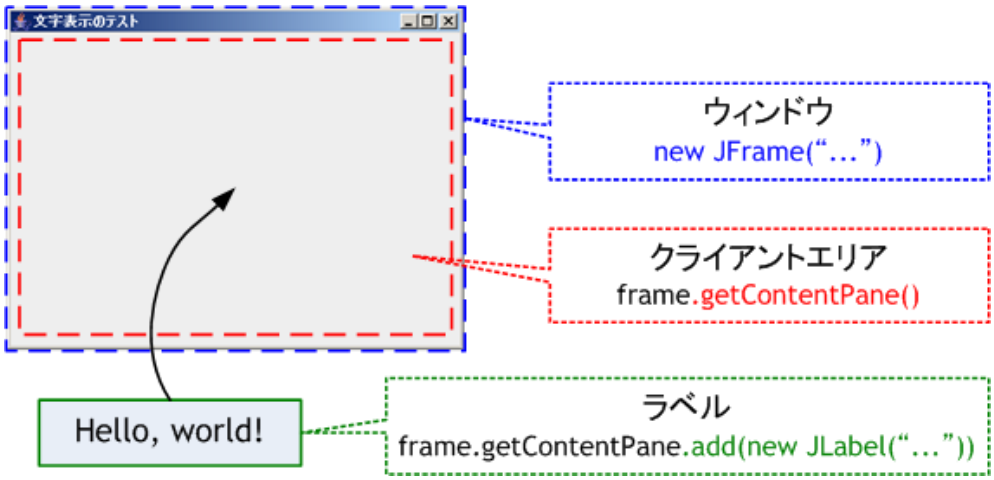
前回は、JFrame を使ってウィンドウを表示するまでの手順を紹介しました。今回は、ボタンやラベルといった GUI コンポーネントを紹介するとともに、それらを JFrame に配置する方法を紹介しましょう。

Swing コンポーネントの種類

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。

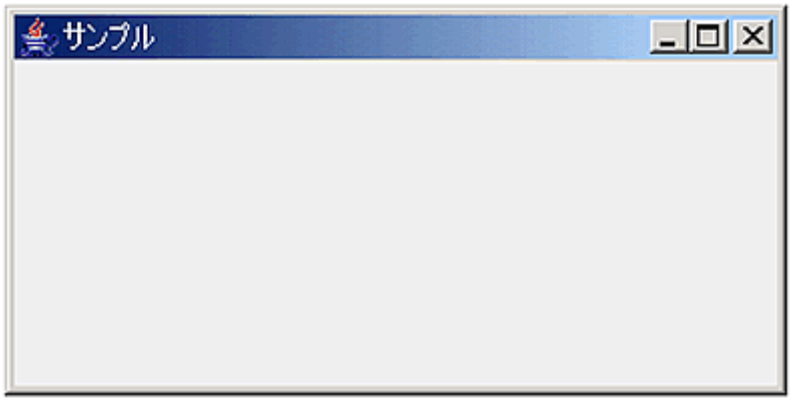
- トップレベルコンテナ（JFrame 等）
  メインウィンドウとなるコンポーネントです。ほかのコンポーネントの土台となります。
- 中間コンテナ（JPanel 等）
  コンポーネントを配置するためのコンテナの役割を果たします。。
- コントロール（JLable、JButton 等）
  ボタンやラベルなどの個々の GUI 部品です。

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。それでは、実際にラベルやボタンをフレームに配置してみましょう。



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。



プロジェクトは既存を使用

前回作成したに「SwingChapter01」という名前のプロジェクトに新たにクラスを追加します。

クラスの作成

パッケージ・エクスプローラから「SwingChapter01」を右クリックして「新規」→「クラス」でも作成できますが、以下の手順でも作業可能です。

1.メニュー[ファイル]→[新規]→[クラス]を選択し、[新規 Java クラス]ダイアログを開き、以下の作業を行う。[終了]を押下してソースファイルを作成する。

- 名前に「**SwingApp2**」を入力
- public static void main(String[] args)をチェック

2.ソースに以下のクラスのインポートを追加する。

- javax.swing.JButton
- javax.swing.JFrame
- javax.swing.JLabel
- java.awt.BorderLayout

3.[JFrame]クラスのインスタンス(mainFrame)を生成し初期設定を行う(詳細は第1回クラスの作成を参照してください)。

4.mainFrame の ContentPane を取得する contentPane。

5.[JLabel]クラスのインスタンス(label)を生成する。

6.[JButton]クラスのインスタンス(button)を生成する。

7.contentPane に label を追加する。

8.contentPane に button を追加する。

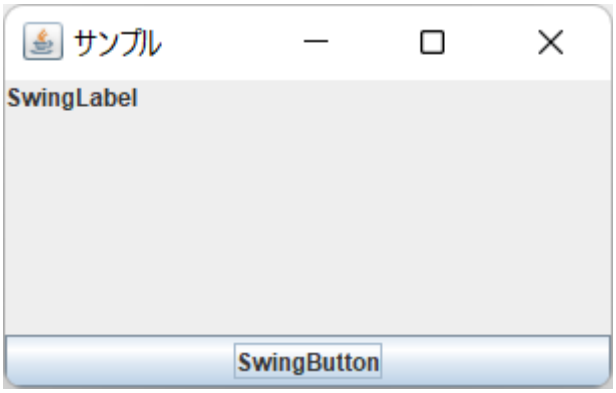
9.ウィンドウを表示する。

以下のソースコードを作成します。

```
1
2  import java.awt.BorderLayout;
3  import java.awt.Container;
4
5  import javax.swing.JButton;
6  import javax.swing.JFrame;
7  import javax.swing.JLabel;
8
9  public class SwingApp2 extends JFrame{
10    public static void main(String[] args) {
11       new SwingApp2();
12    }
13    public SwingApp2() {
14       JFrame mainFrame = new JFrame("サンプル");
15       mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16       mainFrame.setSize(320, 200);
17       mainFrame.setLocationRelativeTo(null);
18 //ここから
19       Container contentPane = mainFrame.getContentPane();
20       // ラベルのインスタンスを生成
21       JLabel label = new JLabel("SwingLabel");
22       // ボタンのインスタンスを生成
23       JButton button = new JButton("SwingButton");
24       // ラベルをContentPane (コンポーネント) に配置
25       contentPane.add(label, BorderLayout.NORTH);
26       // ボタンをContentPane (コンポーネント) に配置
27       contentPane.add(button, BorderLayout.SOUTH);
28 //ここまで
29       mainFrame.setVisible(true);
30 }
31 }
```

実行

作成したコードを実行してみましょう。メニュー[実行]→[Java アプリケーション]を選択します。



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、「ラベル、ボタンのインスタンスを生成し、コンテナに追加する」という簡潔なコードを記述することでコンポーネントの配置を実現できます。感覚的にも理解しやすいですね。

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 のような中間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。

・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 クラスは ContentPane のデフォルトのレイアウトマネージャです。

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。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。

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。図２の［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に、以下コードを追記

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

［SwingLabel］が中央に表示されました。

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");
```

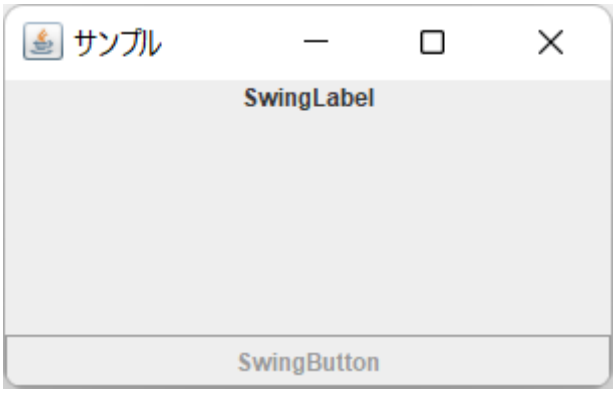
上記コードで、JButton クラスのインスタンスを生成しています。コンストラクタには、JButton で表示したい文字列を設定しています。JLabel と同じく、setText メソッドを使用することで後から表示文字列を設定することができます。

・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に、以下のコードを追記してください。

```
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。JLabel も 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。次回はテキストフィールドなど、さらに多くのコンポーネントを利用した GUI アプリケーションを作成する方法を解説したいと思います。

今回のソースコード全体

```
1
2 import java.awt.BorderLayout;
3 import java.awt.Container;
4
5 import javax.swing.JButton;
6 import javax.swing.JFrame;
7 import javax.swing.JLabel;
8 import javax.swing.SwingConstants;
9
10 public class SwingApp2 extends JFrame{
11    public static void main(String[] args) {
12       new SwingApp2();
13    }
14    public SwingApp2() {
15       JFrame mainFrame = new JFrame("サンプル");
16       mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17       mainFrame.setSize(320, 200);
18       mainFrame.setLocationRelativeTo(null);
19 //ここから
20       Container contentPane = mainFrame.getContentPane();
21       // ラベルのインスタンスを生成
22       JLabel label = new JLabel("SwingLabel");
23       label.setHorizontalAlignment(SwingConstants.CENTER);     //koko
24       // ボタンのインスタンスを生成
25       JButton button = new JButton("SwingButton");
26       button.setEnabled(false);                                //koko
27       // ラベルをContentPane（コンポーネント）に配置
28       contentPane.add(label, BorderLayout.NORTH);
29       // ボタンをContentPane（コンポーネント）に配置
30       contentPane.add(button, BorderLayout.SOUTH);
31 //ここまで
32       mainFrame.setVisible(true);
33 }
34 }
```