



ソフトウェア工学実習 Software Engineering Practice (第02回)

SEP02-002 オブジェクト指向とは
実践編

慶應義塾大学・理工学部・管理工学科
飯島 正

iiijima@ae.keio.ac.jp

こんにちは。
この授業は、
ソフトウェア
工学実習
です



- ・ 前提科目

- ・ 2年生秋学期のソフトウェア工学
 - ・ （未履修者は頑張ってついてきてください。
必要なら補習をします）

- ・ この授業の目標

- ・ この授業の目標は、
オブジェクト指向の概念を
実感をもって理解し活用できるようになる
ことです。

この
授業の
目標は…



1. はじめに

• 1. はじめに

- **目的**：GUI (Graphical User Interface) を通して
MVC (Model-View-Controller) の概念に基づいて
オブジェクト指向を理解する
- まずは、ごく簡単に、**オブジェクトの概念**をお話しします。
- 次は、とにかく、**GUI (Graphical User Interface)** を
作ってみましょう!!
- 次回は、**NetBeans**という
IDE (Integrated Development Environment: 統合開発環境)
も使ってみます

はじめに...



【実践編】 GUIオブジェクト

フレーム（ウィンドウ）
ラベル
ボタン

GUIの
プログラミング
を通して、
オブジェクトを
理解しましょう



クラスは、本当は、**オブジェクトの定義**です

- でも、
今日は、**新たにオブジェクトを定義する**
のではなく、
既定義のオブジェクトを使ってみる
ということをしましょう。
- 具体的には、
ウィンドウを意味するフレーム (JFrame)
GUI部品のラベル (JLabel) や
ボタン (JButton)
です。

GUIのための
既に定義されている
クラスを
利用しましょう

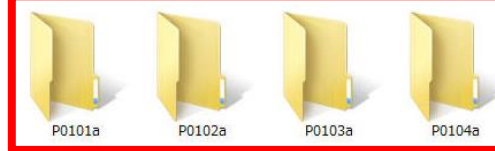


GUI(Graphical User Interface)

最初の一步

SEP01

6



例題101: 単純なフレーム : P101Frame.java



例題102: ラベル付きフレーム : P102LabelFrame.java



例題103: ボタンの表示: P103ButtonFrame.java



例題104: ボタンのアクション: P104ButtonFrame.java

この手順で
実習を
行います

今回は、zipファイルで例題プログラムを配布します

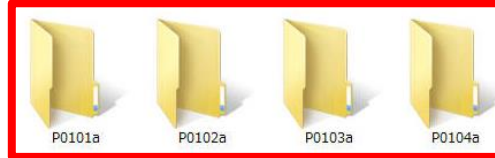


GUI(Graphical User Interface)

最初の一步

SEP01

7



例題101: 単純なフレーム : P101Frame.java



例題102: ラベル付きフレーム : P102LabelFrame.java



例題103: ボタンの表示: P103ButtonFrame.java



例題104: ボタンのアクション: P104ButtonFrame.java

最初は、
空っぽの
ウィンドウを
表示します

今回は、zipファイルで例題プログラムを配布します



例題101: 単純なフレーム : P101Frame.java

- ・ ウィンドウ（フレーム）を画面に表示します

```
import javax.swing.*;
```

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

```
public class P101Frame extends JFrame {  
    P101Frame () {  
        super ( "JFrame: P101Frame" );  
        setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );  
        pack ();  
        setBounds ( 10, 10, 400,300);  
    }  
    public static void main ( String [] args ) {  
        P101Frame aFrame = new P101Frame ();  
        aFrame.setVisible ( true );  
    }  
}
```

空っぽの
ウィンドウを
画面に
表示します



例題101: 単純なフレーム : P101Frame.java

- ・ ウィンドウ（フレーム）を画面に表示します

```
import javax.swing.*;  
import java.awt.*;
```

フレームオブジェクトの定義

```
public class P101Frame extends JFrame {  
    P101Frame () {  
        super ( "JFrame: P101Frame" )  
  
        setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );  
        pack ();  
        setBounds ( 10, 10, 400,300 );  
    }  
  
    public static void main ( String [] args ) {  
        P101Frame aFrame = new P101Frame ();  
        aFrame.setVisible ( true );  
    }  
}
```

クラス定義

スーパークラス
(上位クラス/基底クラス)

既存のJFrame
クラスを拡張して
新たなクラスを
定義します。

ベースとなるスーパークラスJFrameを拡張(extend)して、
新しいクラスP101Frameを定義しています。
このメカニズムを継承(inheritance)といいます



例題101: 単純なフレーム : P101Frame.java

- ・ ウィンドウ（フレーム）を画面に表示します

```
import javax.swing.*;
import java.awt.*;
```

パッケージのimport(輸入)

```
public class P101Frame extends JFrame {
    P101Frame() {
        super("JFrame: P101Frame");
```

JFrameクラスの定義は、
Javax.swingパッケージの中にある。

```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pack();
        setBounds(10, 10, 400, 300);
    }
```

スーパークラス
(上位クラス/基底クラス)

```
    public static void main(String[] args) {
        P101Frame aFrame = new P101Frame();
        aFrame.setVisible(true);
    }
}
```

ベースとなるスーパークラスJFrameを拡張(extend)して、
新しいクラスP101Frameを定義しています。
このメカニズムを継承(inheritance)といいます

Jframeクラス
の定義は、
Javax.swing
パッケージ
の中に
あります。



例題101: 単純なフレーム : P101Frame.java

- ・ ウィンドウ（フレーム）を画面に表示します

```
import javax.swing.*;  
import java.awt.*;
```

```
public class P101Frame extends JFrame {
```

```
    P101Frame () {  
        super ( "JFrame: P101Frame" );  
        setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );  
        pack ();  
        setBounds ( 10, 10, 400,300 );  
    }
```

コンストラクタ

フレームの属性(位置と大きさ)

```
    public static void main ( String [] args ) {  
        P101Frame aFrame = new P101Frame ();  
        aFrame.setVisible ( true );  
    }
```

主メソッド

コンストラクタは、
クラス名と
同じ名前の
特殊なメソッドです。

オブジェクトの
初期化をします。
setBoundsで、
フレームの位置と
大きさを設定します



例題101: 単純なフレーム : P101Frame.java

- ・ ウィンドウ（フレーム）を画面に表示します

```
import javax.swing.*;  
import java.awt.*;
```

```
public class P101Frame extends JFrame {
```

```
    P101Frame () {
```

```
        super ( "JFrame: P101Frame" );
```

```
        setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );
```

```
        pack ();
```

```
        setBounds ( 10, 10, 400, 300 );
```

```
    public static void main ( String [] args ) {
```

```
        P101Frame aFrame = new P101Frame ();
```

```
        aFrame.setVisible ( true );
```

```
    }
```

```
}
```

コンストラクタ

フレームの属性(タイトル)

superの意味は、また、
いずれ説明しますが、
タイトルバーの文字列
が設定されます。

**コンストラクタは、
オブジェクトを作る際の
初期値設定をする
特殊なメソッド(クラスと同じ名前)**



例題101: 単純なフレーム : P101Frame.java

- ・ ウィンドウ（フレーム）を画面に表示します

```
import javax.swing.*;  
import java.awt.*;
```

```
public class P101Frame extends JFrame {
```

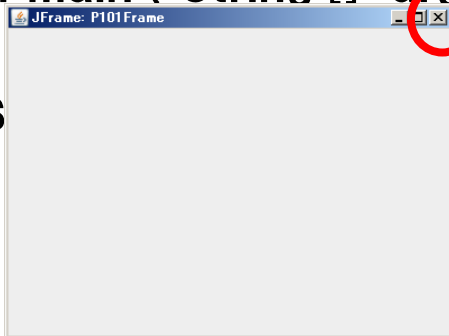
```
    P101Frame () {  
        super ( "JFrame: P101Frame" );  
        setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );  
        pack ();  
        setBounds ( 10, 10, 400,300);  
    }
```

コンストラクタ

×ボタンに対応するクローズ操作

setDefaultCloseOperation
は、ウィンドウのXボタンに
対応するクローズ操作を
指定します

```
    public static void main ( String [] args ) {  
        P101Frame a  
        aFrame.setVis  
    }  
}
```



コンストラクタは、
オブジェクトを作る際の
初期値設定をする
特殊なメソッド(クラスと同じ名前)



例題101: 単純なフレーム : P101Frame.java

- ・ ウィンドウ（フレーム）を画面に表示します

```
import javax.swing.*;  
import java.awt.*;
```

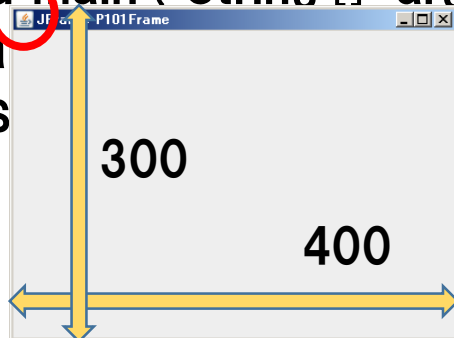
```
public class P101Frame extends JFrame {
```

```
    P101Frame () {  
        super ( "JFrame: P101Frame" );  
        setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );  
        pack ();  
        setBounds ( 10, 10, 400, 300 );  
    }
```

コンストラクタ

フレームの属性(位置と大きさ)

```
    public static void main ( String [] args ) {  
        P101Frame aFrame = new P101Frame ();  
        aFrame.setVisible ( true );  
    }
```



**コンストラクタは、
オブジェクトを作る際の
初期値設定をする
特殊なメソッド(クラスと同じ名前)**

**setBoundsは、
ウィンドウの
位置と大きさを
指定します**



例題101: 単純なフレーム : P101Frame.java

- ・ ウィンドウ（フレーム）を画面に表示します

```
import javax.swing.*;  
import java.awt.*;
```

```
public class P101Frame extends JFrame {  
    P101Frame () {  
        super ( "JFrame: P101Frame" );  
        setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );  
        pack ();  
        setBounds ( 10, 10, 400,300);  
    }  
    public static void main ( String [] args ) {  
        P101Frame aFrame = new P101Frame ();  
        aFrame.setVisible ( true );  
    }  
}
```

インスタンス(オブジェクト)生成
つまり、コンストラクタ呼び出し

コンストラクタは、
new演算子をつ
けて呼び出す

メイン・メソッド

コンストラクタは、
メイン・メソッド
の中で
New演算子をつ
けて
呼び出され、
インスタンスを
生成して
初期化します。



例題101: 単純なフレーム : P101Frame.java

- ・ ウィンドウ（フレーム）を画面に表示します

```
import javax.swing.*;  
import java.awt.*;
```

```
public class P101Frame extends JFrame {  
    P101Frame () {  
        super ( "JFrame: P101Frame" );  
        setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );  
        pack ();  
        setBounds ( 10, 10, 400,300);  
    }  
    public static void main ( String [] args ) {  
        P101Frame aFrame = new P101Frame ();  
        aFrame.setVisible ( true );  
    }  
}
```

P101Frameクラスのインスタンス
であるオブジェクトへの
「メッセージsetVisibleの送信」
=「メソッドsetVisibleの呼出し」
=「visible(可視性)属性の
値(true)の設定」

trueを
パラメータに与えた,
setVisibleメソッド
で表示します

主メソッド



コンパイルと実行: 空っぽのウィンドウを表示する

SEP01

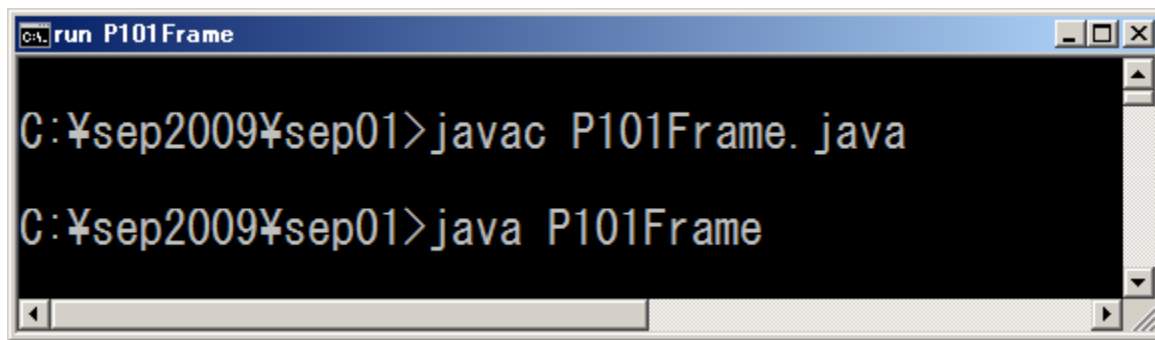
17

```
% javac P101Frame.java
```

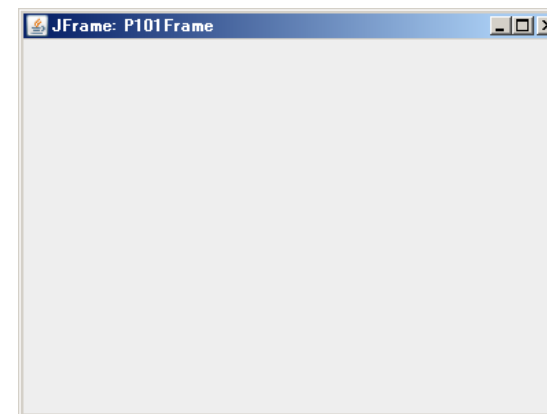
コンパイル

```
% java P101Frame
```

実行



```
run P101Frame
C:\sep2009\sep01>javac P101Frame.java
C:\sep2009\sep01>java P101Frame
```



javacコマンドで
コンパイルして,
Javaコマンドで
実行します



コンパイルと実行 (バッチファイル)

SEP01

18

```
@echo off
rem make P101Frame 2017.04.07 iijima@ae.keio.ac.jp
title make P101Frame
javac P101Frame.java
pause
```

make101.bat

```
@echo off
rem run P101Frame 2017.04.07 iijima@ae.keio.ac.jp
title run P101Frame
java P101Frame
pause
```

run101.bat

バッチファイルを作っておくと、コマンドプロンプトからコマンド
を打ち込まなくても、ダブルクリックで起動できる

バッチファイルを
作るとアイコンの
ダブルクリック
でも実行できます



コンパイルと実行 (バッチファイル)

SEP01

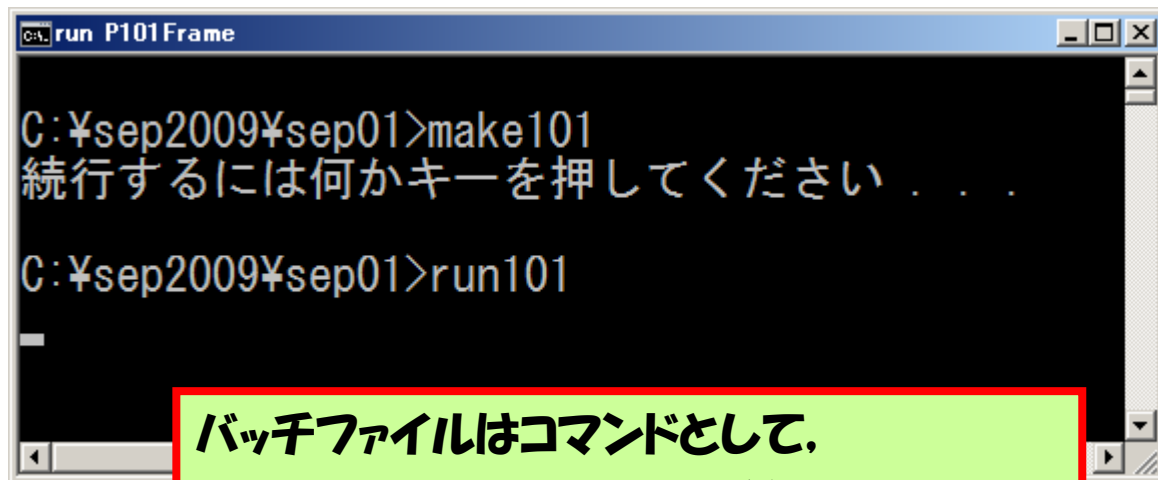
19

```
% javac P101Frame.java
```

→make101

```
% java P101Frame
```

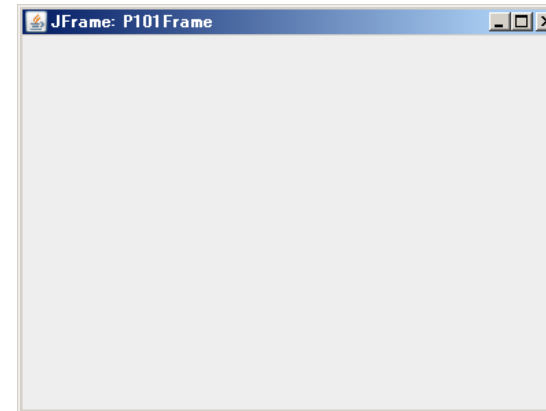
→run101



```
C:\¥sep2009¥sep01>make101
続行するには何かキーを押してください . . .

C:\¥sep2009¥sep01>run101
```

バッチファイルはコマンドとして、
コマンドプロンプトから打ち込んでもよい



バッチファイルは、
コマンドとして
コマンドプロンプト
から入力しても
実行できます

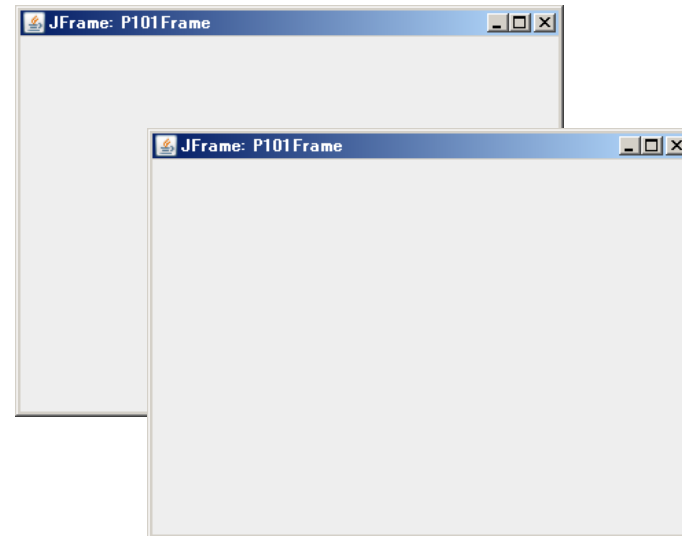


課題101a:フレームを2枚作る（同じ属性）

SEP01

20

- ・ウィンドウの枚数を2枚にしてください。



同じクラスから、
複数のオブジェクト
を生成することが
できます



GUI(Graphical User Interface)

最初の一步

SEP01

21

例題101: 単純なフレーム : P101Frame.java



例題102: ラベル付きフレーム : P102LabelFrame.java



例題103: ボタンの表示: P103ButtonFrame.java



例題104: ボタンのアクション: P104ButtonFrame.java

ウィンドウの
上に
ラベルを
置きましょう

今回は, zipファイルで例題プログラムを配布します



例題102: ラベル付きフレーム : P102LabelFrame.java

SEP01

22

- ・ フレームの上にラベルを乗せる

```
import javax.swing.*;  
import java.awt.*;
```

```
class P102LabelFrame extends JFrame {
```

```
    private JLabel aLabel;
```

```
    P102LabelFrame () {
```

```
        super ( "JLabel: LabelFrame" );
```

```
        aLabel = new JLabel ( "Hello World!" );
```

```
        add ( aLabel );
```

```
        setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );
```

```
        pack ();
```

```
        setBounds ( 10, 10, 400,300);
```

```
    }
```

```
    static void main ( String [] args ) {
```

```
        P102LabelFrame aFrame = new P102LabelFrame ();
```

```
        aFrame.setVisible (true);
```

```
    }
```

```
}
```



JLabelクラスで
ラベルを生成し、
フレームに
追加します



例題102: ラベル付きフレーム : P102LabelFrame.java

SEP01

23

- ・ フレームの上にラベルを乗せる

```
import javax.swing.*;  
import java.awt.*;
```

```
class P102LabelFrame extends JFrame {
```

```
    private JLabel aLabel;
```

ラベルオブジェクトの作成

```
    P102LabelFrame () {  
        super ( "JLabel: LabelFrame" );
```

```
        aLabel = new JLabel ( "Hello World!" );
```

```
        add ( aLabel );
```

```
        setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );
```

```
        pack ();
```

```
        setBounds ( 10, 10, 400,300);
```

```
    }
```

```
    static void main ( String [] args ) {
```

```
        P102LabelFrame aFrame = new P102LabelFrame ();
```

```
        aFrame.setVisible (true);
```

```
    }
```

```
}
```



ラベル
オブジェクトを
生成します



例題102: ラベル付きフレーム : P102LabelFrame.java

SEP01

24

- ・ フレームの上にラベルを乗せる

```
import javax.swing.*;  
import java.awt.*;
```

```
class P102LabelFrame extends JFrame {
```

```
    private JLabel aLabel;
```

```
    P102LabelFrame () {  
        super ( "JLabel: LabelFrame" );
```

```
        aLabel = new JLabel ( "Hello World!" );
```

```
        add ( aLabel );
```

```
        setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );
```

```
        pack ();
```

```
        setBounds ( 10, 10, 400,300 );
```

ラベルをフレームに貼り付け

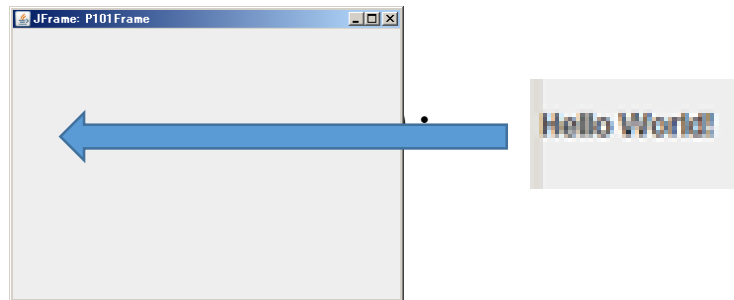
```
    }
```

```
    static void main ( String [] args ) {  
        P102LabelFrame aFrame = new  
        aFrame.setVisible (true);  
    }
```

```
}
```



ラベルオブジェクトを
addメソッドで
フレームに
追加します



コンパイルと実行

SEP01

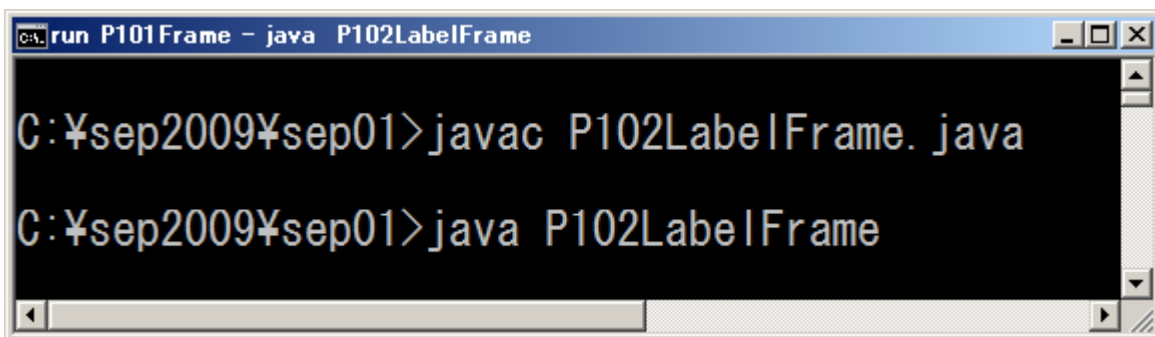
25

```
% javac P102LabelFrame.java
```

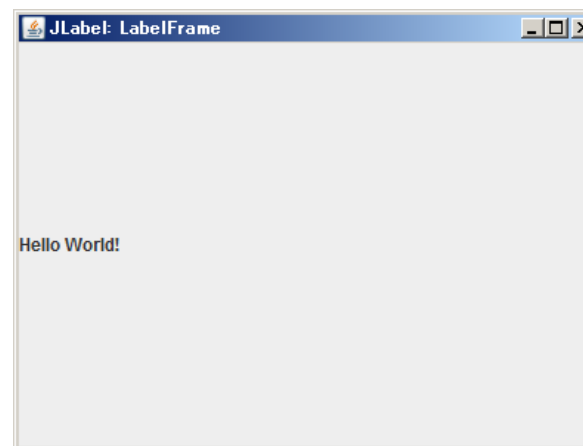
コンパイル

```
% java P102LabelFrame
```

実行



```
run P101Frame - java P102LabelFrame
C:\¥sep2009¥sep01>javac P102LabelFrame.java
C:\¥sep2009¥sep01>java P102LabelFrame
```



javacコマンドで
コンパイルして、
Javaコマンドで
実行します



コンパイルと実行 (バッチファイル)

SEP01

26

make102.bat

```
@echo off
rem make P102LabelFrame 2017.04.07 iijima@ae.keio.ac.jp
title make P102LabelFrame
javac P102LabelFrame.java
pause
```

run102.bat

```
@echo off
rem run P102LabelFrame 2017.04.07 iijima@ae.keio.ac.jp
title run P102LabelFrame
java P102LabelFrame
pause
```

バッチファイルを作っておくと、コマンドプロンプトからコマンド
を打ち込まなくても、ダブルクリックで起動できる

バッチファイルを
作るとアイコンの
ダブルクリック
でも実行できます



コンパイルと実行 (バッチファイル)

SEP01

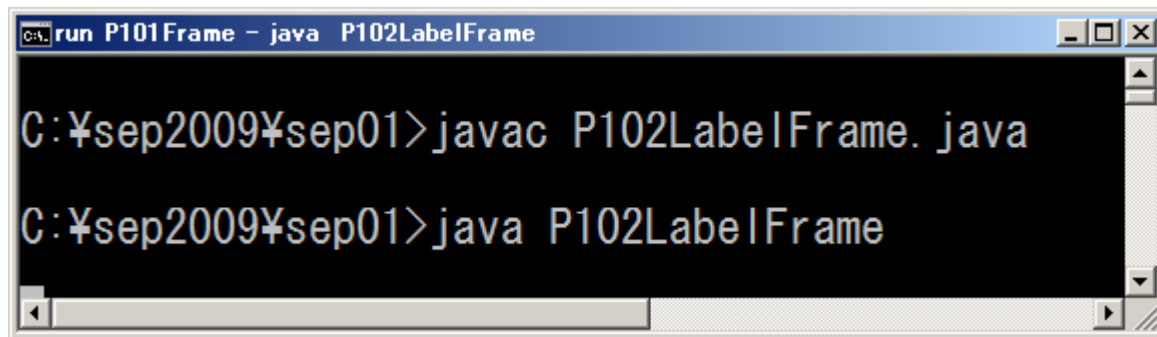
27

% javac P102LabelFrame.java

→make102

% java P102LabelFrame

→run102



```
C:\¥sep2009¥sep01>javac P102LabelFrame.java
C:\¥sep2009¥sep01>java P102LabelFrame
```



バッチファイルは、
コマンドとして
コマンドプロンプト
から入力しても
実行できます



GUI(Graphical User Interface)

最初の一步

SEP01

28

例題101: 単純なフレーム : P101Frame.java



例題102: ラベル付きフレーム : P102LabelFrame.java



例題103: ボタンの表示: P103ButtonFrame.java



例題104: ボタンのアクション: P104ButtonFrame.java

ボタンも
置きましょう。
でもボタンを
クリックしても
何もしません

今回は, zipファイルで例題プログラムを配布します



ボタンの表示:P103ButtonFrame.java

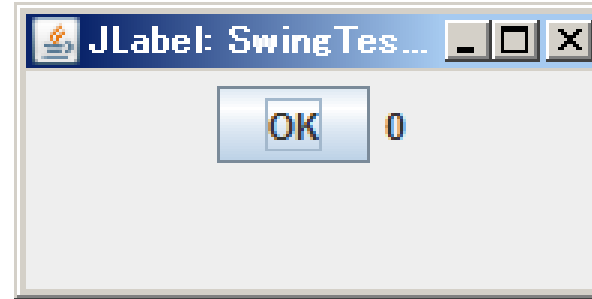
SEP01

29

- ・ フレームの上にボタンとラベルを乗せる
 - ・ (ボタンをクリックしたときのハンドリング・アクションは未定義 → つまり, 何もしない)

```
import javax.swing.*;  
import java.awt.*;
```

```
public class P103ButtonFrame extends JFrame {  
    private JLabel  aLabel;  
    private JButton aButton;
```



```
P103ButtonFrame () { ... } //次ページ
```

```
public static void main ( String [] args ) {  
    P103ButtonFrame aFrame = new P103ButtonFrame ();  
    aFrame.setVisible (true);  
}  
}
```

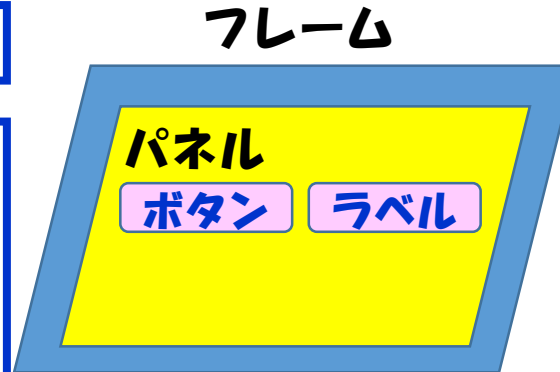
ボタンには
Jbuttonクラスを
使います。



ボタンの表示:P103ButtonFrame.java

- ・ フレームの上にボタンとラベルを乗せる
 - ・ (ボタンをクリックしたときのハンドリング・アクションは未定義 → つまり, 何もしない)

```
P103ButtonFrame () {  
    super ( "JLabel: SwingTestFrame" );  
    aLabel = new JLabel ( "0" );  
    aButton = new JButton ( "OK" );  
  
    JPanel aPanel = new JPanel ();  
    aPanel.add ( aButton );  
    aPanel.add ( aLabel );  
    add ( aPanel );  
  
    setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );  
    pack ();  
    setBounds ( 10, 10, 200, 100 );  
}
```



パネルの上に,
ボタンとラベルを
配置し,
パネルをフレームに
追加します



コンパイルと実行 (バッチファイル)

SEP01

31

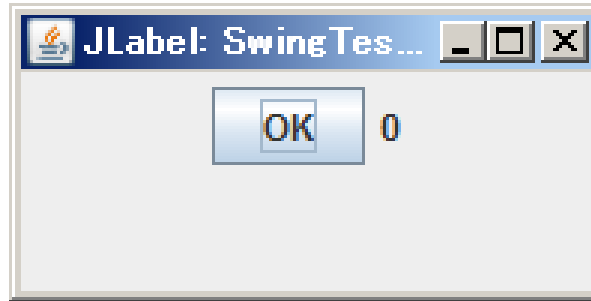
```
% javac P103ButtonFrame.java
```

→make103

```
% java P103ButtonFrame
```

→run103

ボタンにアクションを
設定していないので、
ボタンを押しても何もしない



バッチファイルは、
コマンドとして
コマンドプロンプト
から入力しても
実行できます



ボタンのアクション:P104ButtonFrame.java

SEP01

32

```
import javax.swing.*;
```

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

```
public class P104ButtonFrame extends JFrame {
```

```
    private JLabel    aLabel;
```

```
    private JButton   aButton;
```

```
    P104ButtonFrame () { ... } //次ページ
```

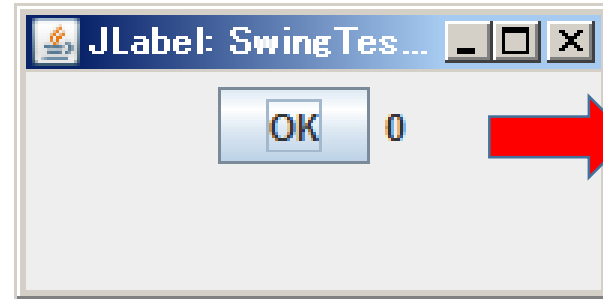
```
    public static void main ( String [] args ) {
```

```
        P104ButtonFrame aFrame = new P104ButtonFrame ();
```

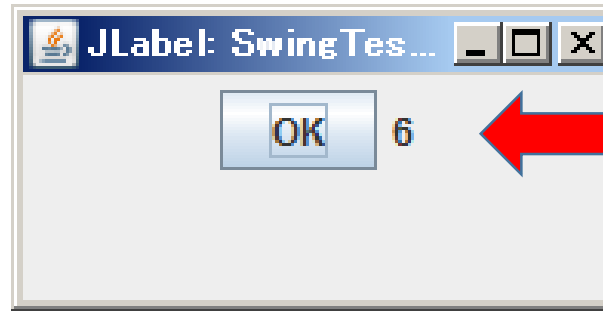
```
        aFrame.setVisible (true);
```

```
    }  
}
```

**JButtonクラスを、アクション定義付きの
Okボタンクラスに置き換える**



ボタンクリックという
イベント発生



対応するイベントハンドラ
というアクションが
起動される

ボタンを
クリックすると
カウントアップする
ようにしましょう



ボタンのアクション:P104ButtonFrame.java

SEP01

33

```
P104ButtonFrame () {  
    super ( "JLabel: SwingTestFrame" );  
    aLabel = new JLabel ( "0" );  
    aButton = new OkButton ( aLabel );  
  
    JPanel aPanel = new JPanel ();  
    aPanel.add ( aButton );  
    aPanel.add ( aLabel );  
    add ( aPanel );  
  
    setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );  
    pack ();  
    setBounds ( 10, 10, 200,100 );  
}
```

**JButtonクラスを、
アクション定義付きの
Okボタンクラス
に置き換える**

**Jbuttonを
アクション定義付
きのOkButton
クラスに置き換え
ます**



アクション付きのボタン:OkButton.java

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

OkButtonクラスの定義⇒Jbuttonクラスにカウント機能を追加する

```
class OkButton extends JButton implements ActionListener {
```

```
    private int count = 0;
```

```
    private JLabel aLabel;
```

```
    OkButton ( JLabel aLabel ) {
```

```
        super ( "OK" );
```

```
        addActionListener (this);
```

```
        this.aLabel = aLabel;
```

```
    }
```

```
    public void actionPerformed ( ActionEvent e ) {
```

```
        count++;
```

```
        aLabel.setText (Integer.toString ( count ) );
```

```
    }
```

```
}
```

**ボタンを押した回数を
記憶する変数(整数型)**

**ボタンが押された時の
アクション(イベントハンドラ)
= カウントアップして、
結果を文字列に
変換して表示**

**OkButton
クラスを定義しま
す。**



コンパイルと実行 (バッチファイル)

SEP01

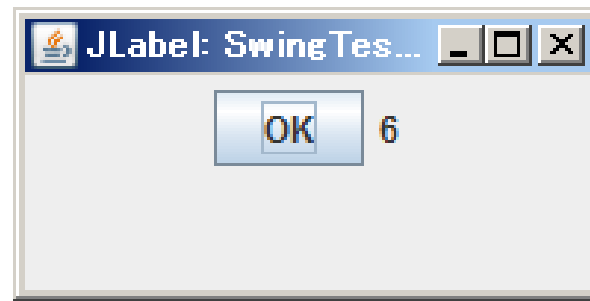
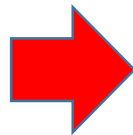
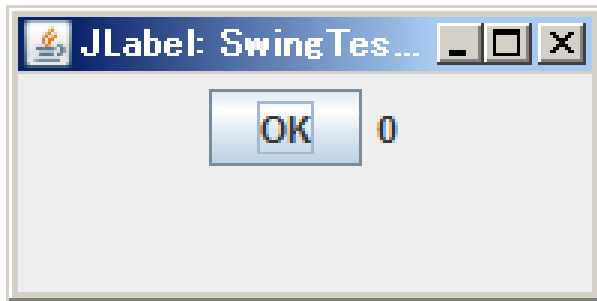
35

```
% javac P104ButtonFrame.java
```

→make104

```
% java P104ButtonFrame
```

→run104



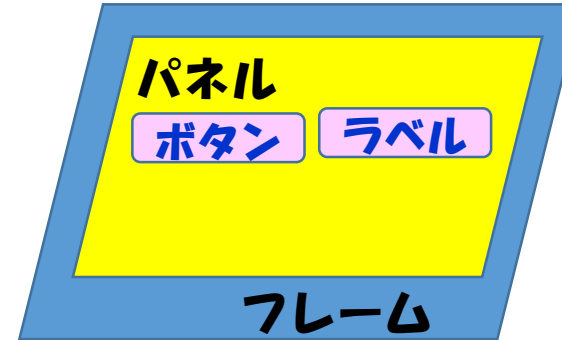
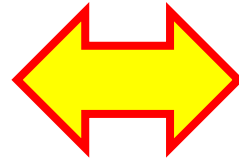
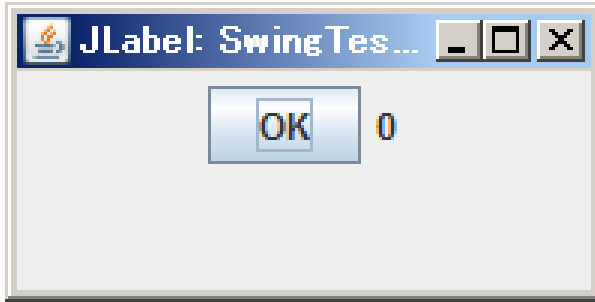
ボタンを押すと、カウンタが進む

バッチファイルは、
コマンドとして
コマンドプロンプト
から入力しても
実行できます



ボタンの表示:P104ButtonFrame.java

- ボタンやラベルを直接フレームに追加せず、パネルを介しています。
これはなぜでしょう？



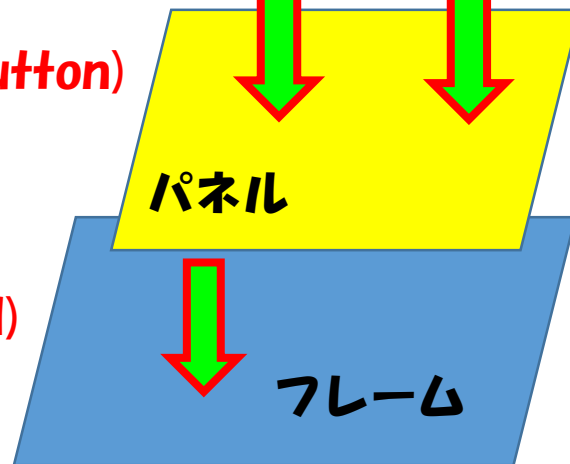
`aButton = new JButton("OK")`
オブジェクト生成

ボタン ラベル

`aLabel = new JLabel("0")`
オブジェクト生成

`aPanel.add(aButton)`

`aPanel.add(aLabel)`



`add(aPanel)`

`aPanel = new JPanel()`
オブジェクト生成

`aFrame =`
`new P104ButtonFrame()`
オブジェクト生成

ボタンやラベルは、
直接フレームに
追加せず、
パネルを挟んでいます。
なぜでしょう？



原理的なことは理解できたので、以降はIDEを使いましょう

- 統合開発環境NetBeansを使いましょう
 - とりあえずはレイアウトなどの細かい仕掛けを意識しないで、大局を理解しましょう
 - デザインについての詳細は、もっとレベルが上がってからやります。
 - イベントハンドラの名前もとりあえずは、NetBeansが名前を付け替えてくれているものを使いながら、少しずつ、Java本来の書法を理解していきましょう。
- HelloWorld
 - プロジェクト名: P0201
- Counter
 - プロジェクト名: P0202_Counter
 - 「いいね!」ボタンを作ろう!!
- 今日の課題
 - (1) P0202b_Counter ... downボタン, resetボタンを作しましょう
 - (2) P0202c_Counter ... downボタンで、count変数の値が負にならないように保護しましょう。
 - (3) P0202d_Counter ... 高度な課題。スライダーでカウントを表現しましょう。

今後は、NetBeansを使っていきましょう。

