ソフトウェア工学

7. オブジェクト指向とは GUI

2018年01月17日

慶應義塾大学 理工学部 管理工学科
飯島 正 (iijima@ae.keio.ac.jp)

今回は. . .

- 今回は, テストもやりますが. . .
- テスト前に, 少し, だけ講義・実習をします.
- 具体的には,
  GUI(グラフィカル ユーザ インタフェース)
  を作ってみましょう.

GUI(Graphical User Interface)
最初の一歩

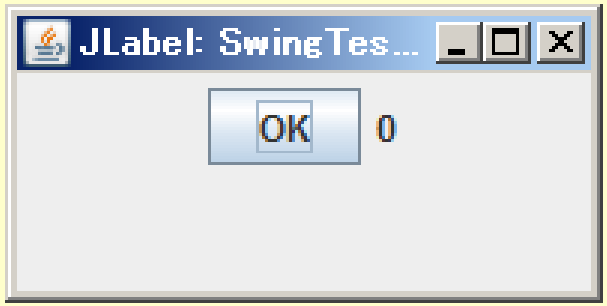
ボタンのアクション:P104ButtonFrame.java

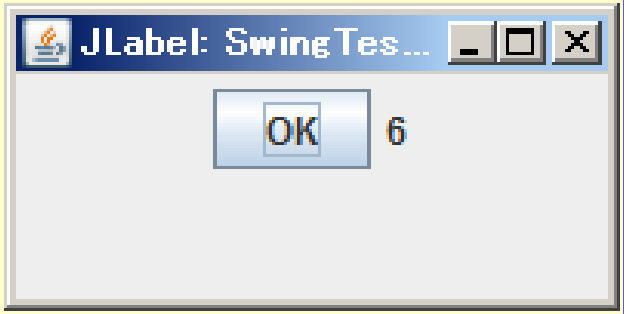
```
import javax.swing.*;
import java.awt.*;

public class P104ButtonFrame extends JFrame {
    private JLabel    aLabel;
    private OkButton  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

ボタンのアクション:P104ButtonFrame.java

```
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

アクション付きのボタン:OkButton.java

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

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 ) );
    }
}
```

OkButtonクラスの定義

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
＝カウントアップして，結果を文字列に変換して表示

コンパイルと実行(バッチファイル)

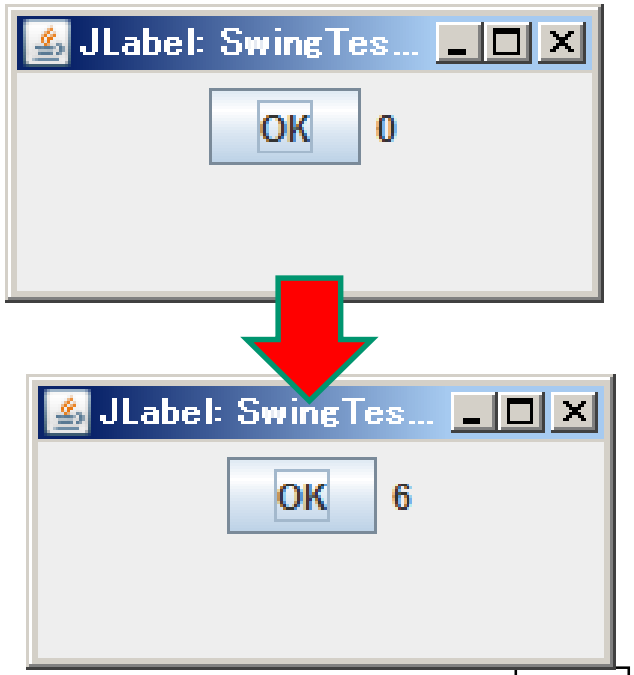
```
% javac P104ButtonFrame.java
```

→make104

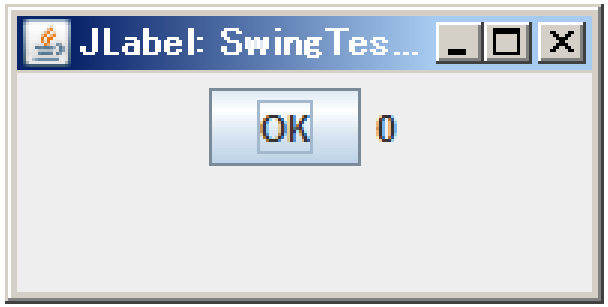
```
% java P104ButtonFrame
```

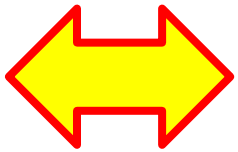
→run104

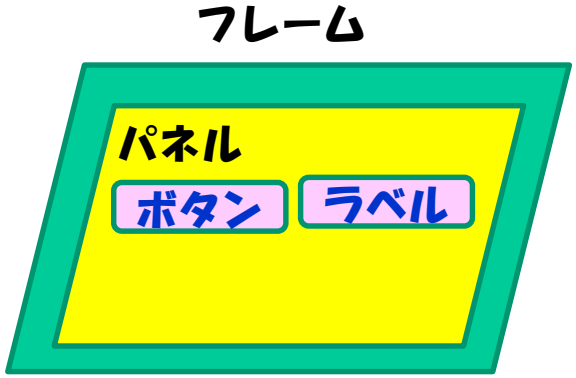
ボタンを押すと,
カウンタが進む



ボタンの表示:P104ButtonFrame.java







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)

aPanel = オブジェクト生成
new Jpanel()

パネル

add(aPanel)

フレーム

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

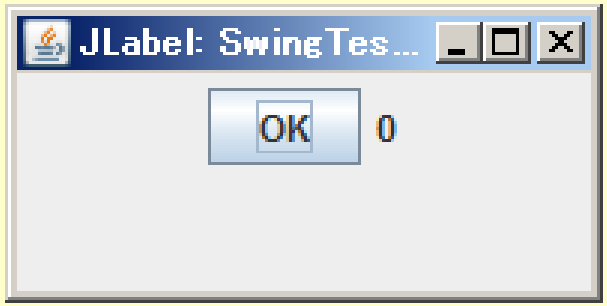
補足説明
ボタンのアクション:P104ButtonFrame.java

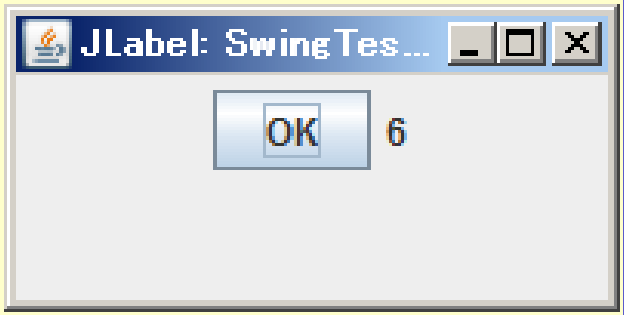
```
package p104buttonframe;
import javax.swing.*;
import java.awt.*;

public class P104ButtonFrame extends JFrame {
    private JLabel    aLabel;
    private OkButton  aButton;

    P104ButtonFrame() { … } //次ページ

    public static void main( String[] args ) {
        P104ButtonFrame aFrame = new P104ButtonFrame();
        aFrame.setVisible(true);
    }
}
```



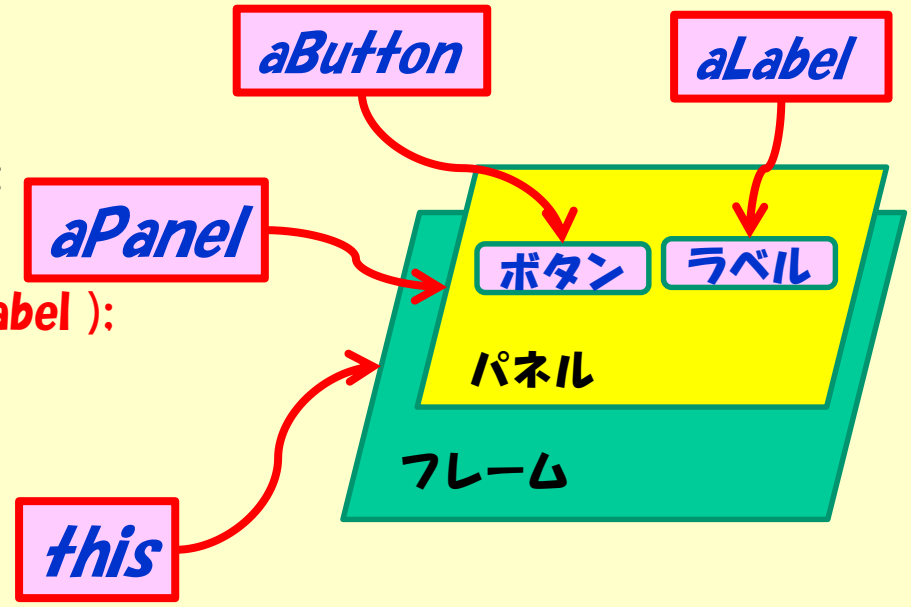


補足説明
ボタンのアクション:P104ButtonFrame.java

```
P104ButtonFrame() {
    super( "JLabel: SwingTestFrame" );
    this.aLabel = new JLabel( "0" );
    this.aButton = new OkButton( aLabel );

    JPanel aPanel = new JPanel();
    aPanel.add( this.aButton );
    aPanel.add( this.aLabel );
    this.add( aPanel ); // これはadd(aPanel)と同じ, このフレームへの貼り付け

    this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    this.pack();
    this.setBounds( 10, 10, 200,100 );
}
```



そのオブジェクト(フレーム)自身を示すthis.は省略可

アクション付きのボタン:OkButton.java

```
package p104buttonframe;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

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 ) );
    }
}
```

OkButtonクラスの定義

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
＝カウントアップして，結果を文字列に変換して表示

Javaでのクラス定義の書き方(入門編)

```
public class クラス名 {
    変数宣言;
    メソッド宣言;
}
```

属性
- インスタンス変数
- クラス変数(static変数)

操作
- インスタンス・メソッド
- クラス・メソッド (staticメソッド)

いろいろな用語:

構成要素	一般表現	Smalltalk-80	Java	C++
内部状態	属性	変数	フィールド変数	データメンバ
振る舞い	操作	メソッド	メソッド	メンバ関数

Javaでのインスタンス生成

- クラス定義中のコンストラクタ定義
  - クラス名と同じ名前の特別なメソッド
  - 返却値の型などは指定しない
  - 引数を渡せる,
  - 引数の数やその型が異なれば複数定義できる(オーバーローディング)

コンストラクタの呼び出し
new クラス名()
引数が無くても()を付ける

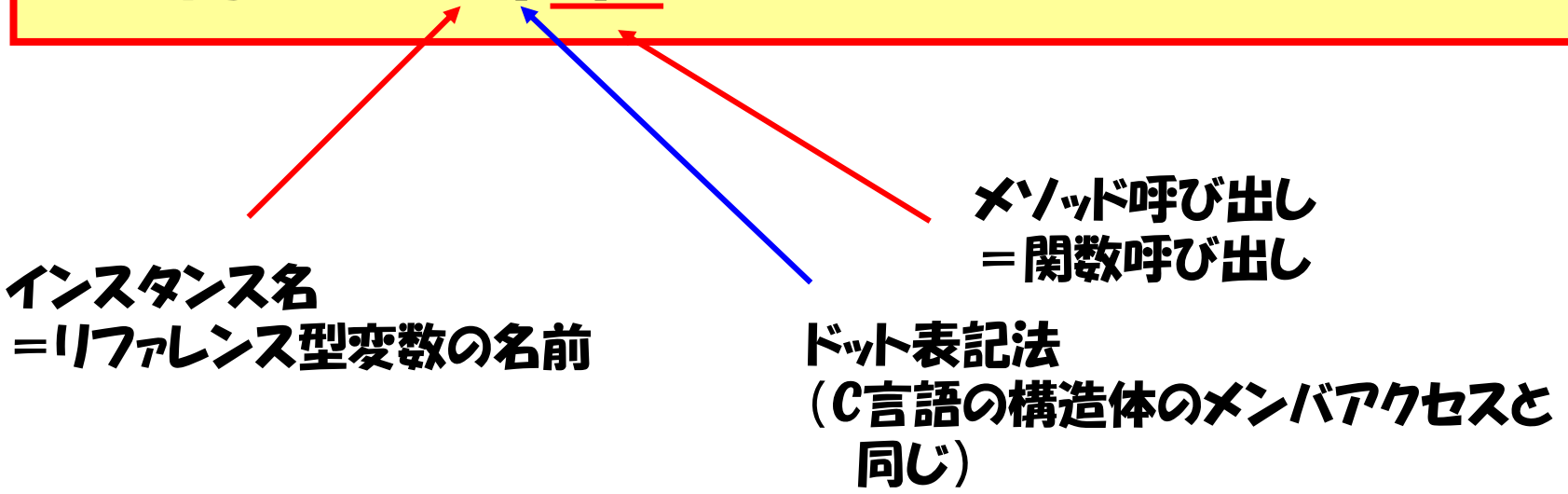
例:
new Stack();
new Stack(100);

配列
- 基本データ型配列
  int a[] = new int[20];
- クラス参照型配列
  クラス b[] = new クラス[100];

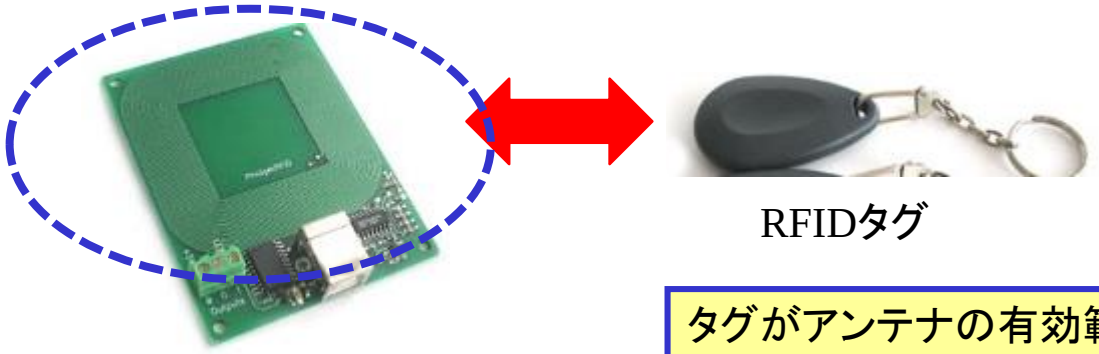
正確に言えば
クラス名と同じ名前を
持っている特殊なメソッド
がコンストラクタである.

Javaでの(インスタンス)メソッド呼出し

- インスタンス名. メソッド名(実引数,...);
  - 例えば, s1.pop();



イベント駆動プログラム



RFIDタグ

タグがアンテナの有効範囲に
入ったり，出たりするイベント
によって計算が起動される

カウンター

GUI(Graphical User I/F)の場合．
ボタンの上で
マウスがクリックされると
アクションイベントが発生し，
カウントアップする．

※簡単なデモ

イベント駆動プログラム

- 手順

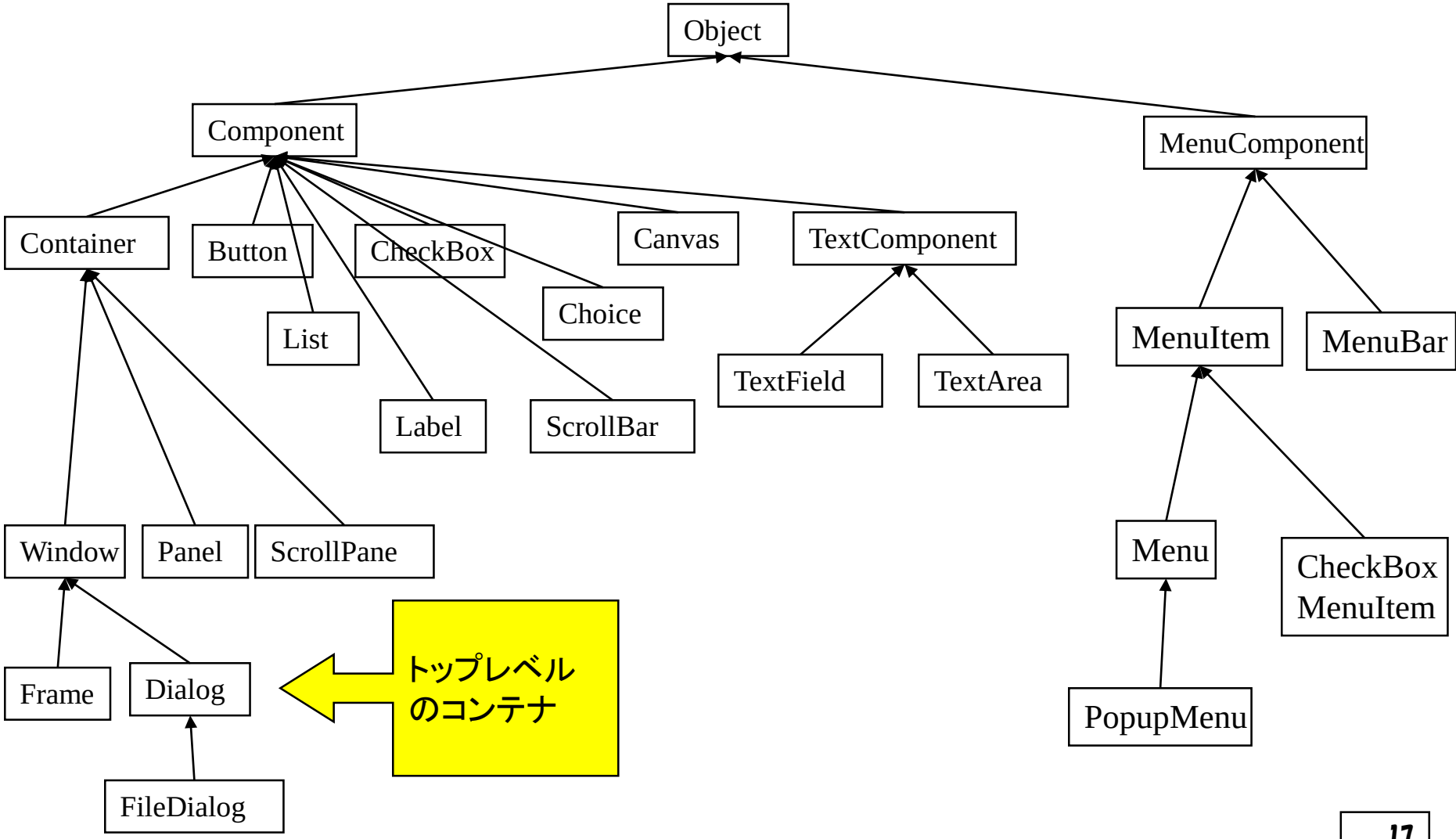
  ①予めコンポーネントにイベントリスナを登録しておく

  ②そのコンポーネント(イベントソース)で，
  イベントが発生する
  (例えばボタン上でマウスボタンが押される)

  ③イベントソースに登録されている
  全イベントリスナに，そのイベントが通知される

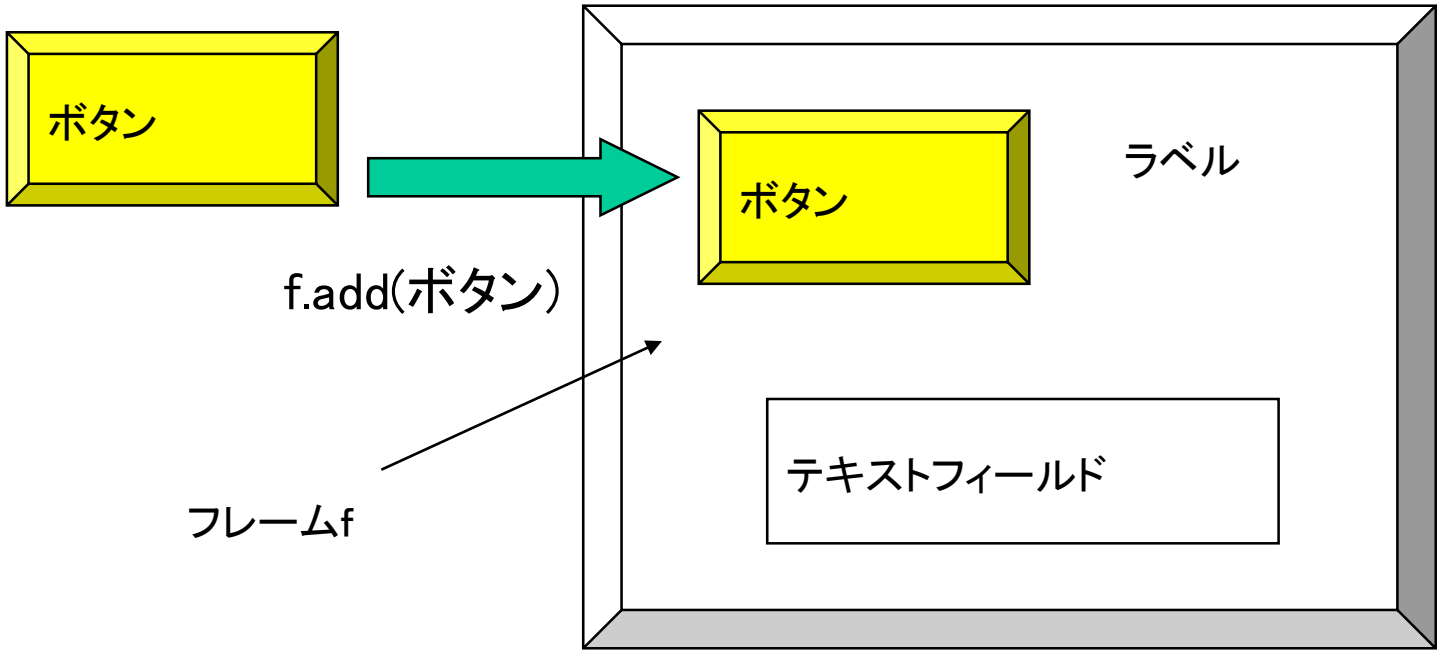
  ④リスナで，対応するメソッド(イベントハンドラ)
  が起動される

GUIコンポーネントの階層

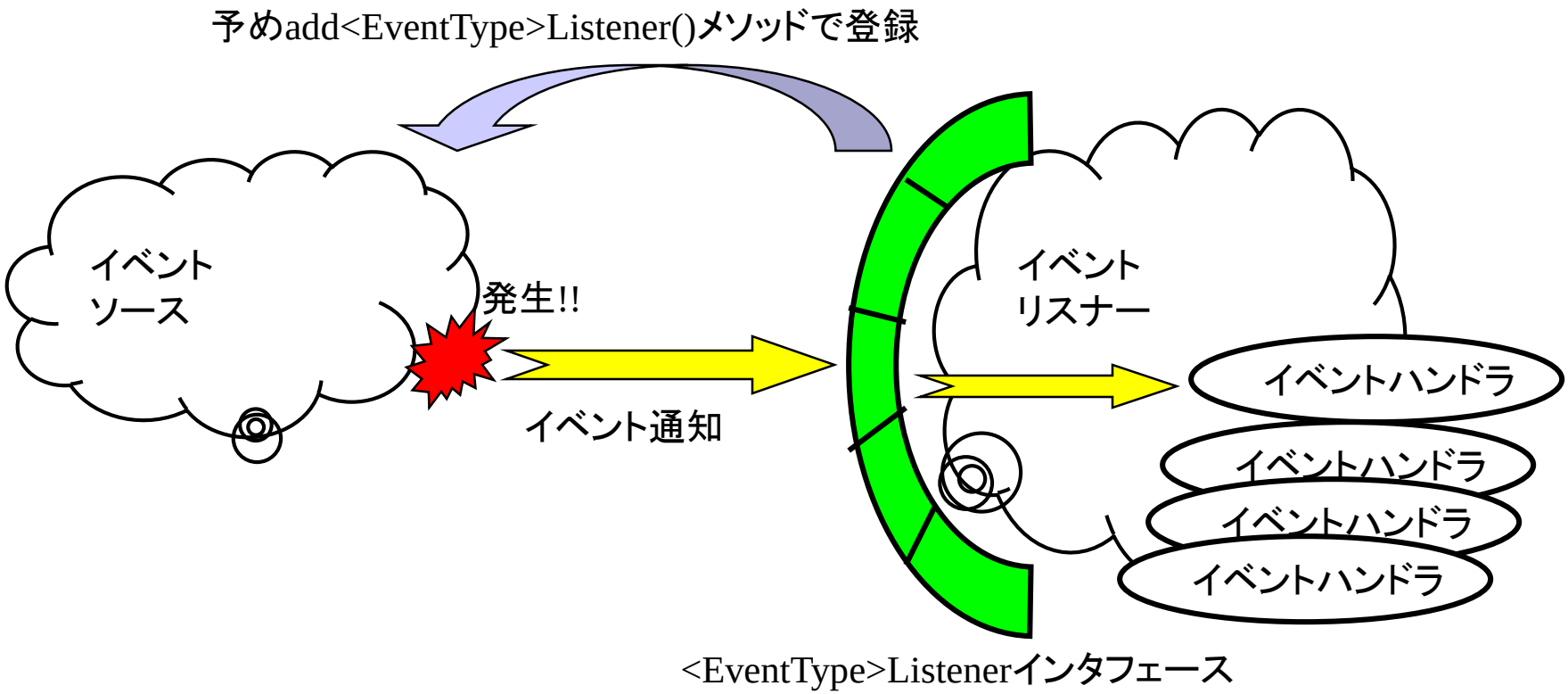


GUIコンテナ

- FrameやPanelといったコンテナに，GUIコンポーネントを貼り付けていく（ add()メソッド）



【参考】Delegation-based Listenerモデル（委譲に基づくリスナ・モデル）



イベントをメッセージで表現する

【参考】GUIコンポーネント毎のイベントとハンドラ(1/3)

Component	ComponentEvent	componentMoved()
		componentResizes()
		componentShown()
		componentHidden()
	FocusEvent	focusGained()
		focusLost()
	KeyEvent	keyPressed()
		keyReleased()
		keyTyped()
	MouseEvent	mouseClicked()
		mouseEntered()
		mouseExited()
		mousePressed()
		mouseReleased()
		mouseDragged()
		mouseMoved()

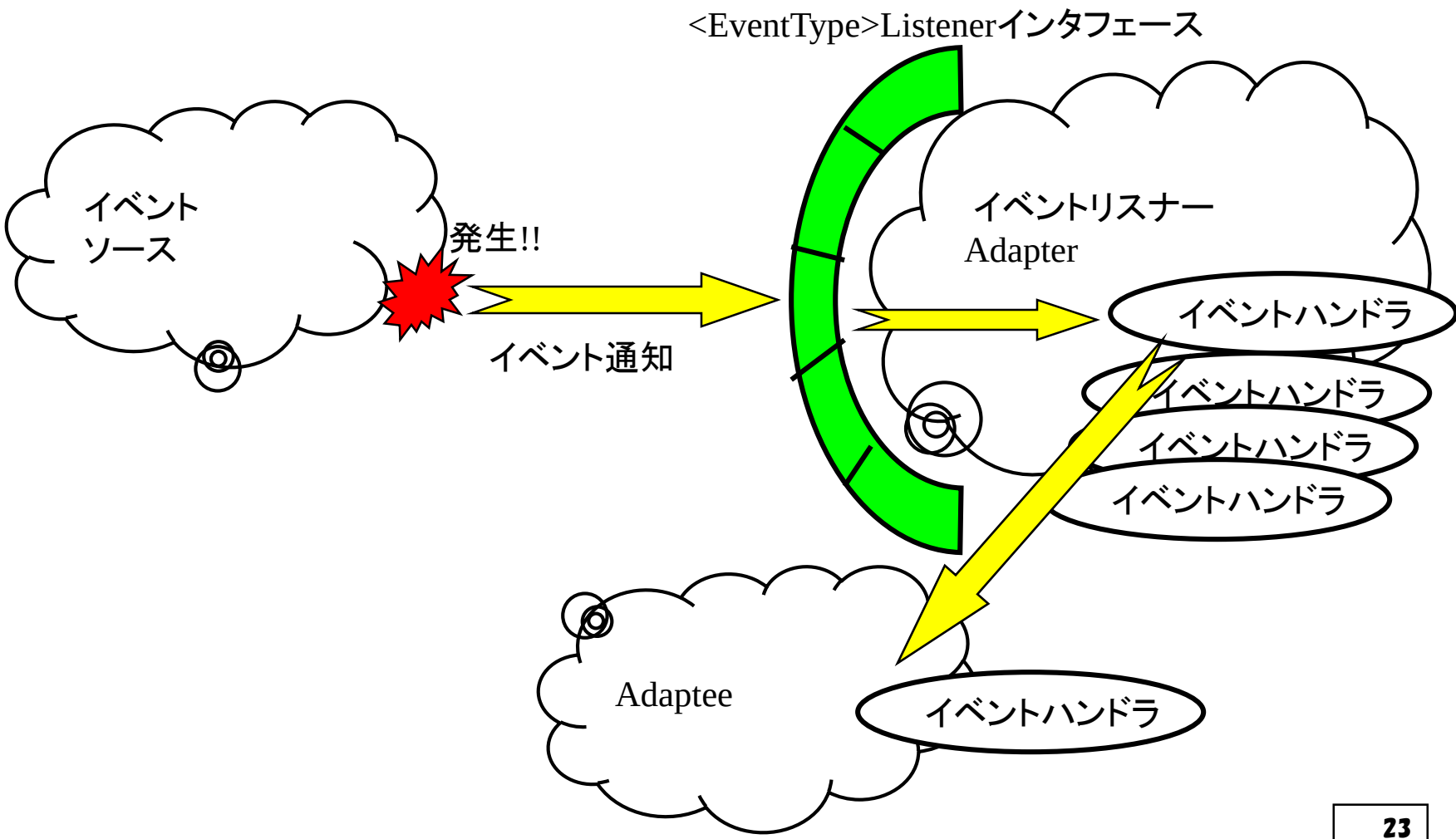
【参考】GUIコンポーネント毎のイベントとハンドラ(2/3)

Button	ActionEvent	actionPerformed()
MenuItem		actionPerformed()
List		actionPerformed()
Choice	ItemEvent	itemStateChanged()
CheckBox		itemStateChanged()
CheckBox MenuItem		itemStateChanged()
Text Component	TextEvent	textValueChanged()
TextField	ActionEvent	actionPerformed()

【参考】GUIコンポーネント毎のイベントとハンドラ(3/3)

Window	WindowEvent	windowClosed()
		windowClosing()
		windowOpened()
		windowIconified()
		windowDeiconified()
ScrollBar	AdjustmentEvent	adjustmentValueChanged()

【参考】Adapterの利用



GUI：コンポーネントのサイズや文字の指定

コンポーネントの大きさ指定

各コンポーネントの大きさ
　→レイアウトマネージャーをnullに設定しない限り、
　　最終的にレイアウトマネージャーが管理する。

例）
・BorderLayout　→配置する位置によって元々のコンポーネントの
　　大きさを尊重する部分と尊重せずに変更してしまう部分とができる

・GridLayouでは元々のコンポーネントのサイズは無視する

・FlowLayoutのように元々のコンポーネントのサイズを
　そのまま尊重するレイアウトマネージャーもある

コンポーネントの大きさ指定

推奨サイズ	void	setPreferredSize(Dimension preferredSize)	コンポーネントの適切なサイズを設定する.
			preferredSize が null　の場合、UI で適切なサイズを要求する.
			preferredSize - 新しい推奨サイズ、または null
最大サイズ	void	setMaximumSize(Dimension maximumSize)	コンポーネントの最大サイズを定数値に設定する.
			getMaximumSize の以降の呼び出しで、常にこの値を返す. その計算のためにコンポーネントの UI が要求されることはない.
			最大サイズを null に設定すると、デフォルトの動作に戻る
			maximumSize - 要求される最大許可サイズを保持する Dimension
最小サイズ	void	setMinimumSize(Dimension minimumSize)	コンポーネントの最小サイズを定数値に設定する.
			getMinimumSize の以降の呼び出しで、常にこの値を返す。 その計算のためにコンポーネントの UI が要求されることはない
			最小サイズを null に設定すると、デフォルトの動作に戻る.
			minimumSize - このコンポーネントの新しい最小サイズ

※ディメンジョンの指定にはインスタンスを生成する⇒　new Dimension(200,100)

文字フォント，スタイル，サイズ指定

フォント(font)	文字のデザインに関するデータ

フォント名	"Dialog"	
	"DialogInput"	
	"Monospaced"	
	"Serif"	
	"SansSerif"	
	"Symbol"	

字体指定	標準	Font.PLAIN	
	太字	Font.BOLD	
	斜体	Font.ITALIC	
	太字かつ斜体	Font.BOLD | Font.ITALIC	

文字の大きさ	ポイント数で指定

Fontインスタンス の生成の例	new Font("SansSerif", Font.BOLD, 24)
	SansSerif のボールド体の24ポイント
GUI部品へのフォント指定	GUI部品のインスタンス.setFont(フォントのインスタンス)

ラベルの文字指定(1/2)

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

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

ラベルの文字指定(2/2)

```
class SwingTestFrame extends JFrame {
    private JLabel    aLabel;
    private Container aContentPane;

    SwingTestFrame() {
        super( "JLabel: SwingTestFrame" );
        aLabel = new JLabel( "Hello World!" );
        Font font = new Font("SansSerif", Font.BOLD, 24);
        aLabel.setFont( font );
        aContentPane = getContentPane();
        aContentPane.add( aLabel );
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds( 10, 10, 400,300);
        pack();
    }
}
```

おわりに

- オブジェクト指向については，未だほんの入り口
- 3年生春学期・月/金曜(各5時限目)の
  ソフトウェア工学実習
  で続きをやります
  - Javaは，管理工学実験演習でも使いますから，
    できるだけ3年生で履修しておいてください．
- そこでは，便利なツールをつかいます．
  IDE(統合開発環境)であるNetBeans
  - とっても便利です．
  - 文法チェックなどは自動的にやってくれます．
  - コンパイルも実行もボタン一発です．

来年のソフトウェア工学実習では...