



GRAPHICAL USER INTERFACE WIDGETを使う

オブジェクト指向プログラミング特論

只木進一:工学系研究科

JAVAとGUI

- 多くのプログラミング言語では
 - GUIは言語とは別のライブラリ
 - 例:c/c++とX11、GTK
 - プラットフォーム依存
- Javaでは
 - GUIライブラリが言語と同封されて配布される
 - プラットフォーム独立
 - 各プラットフォームのウィンドウマネージャ利用可

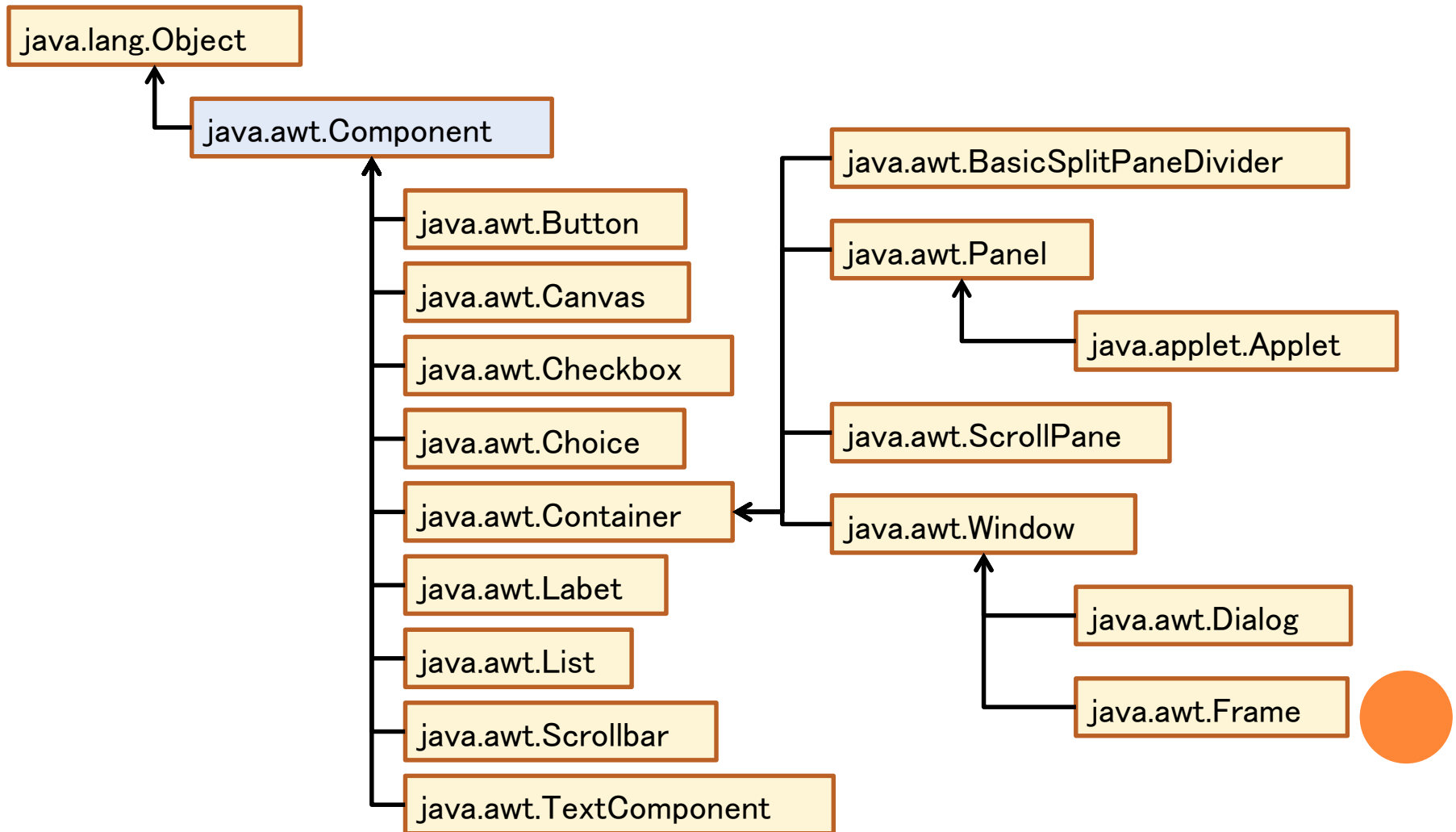


JAVAとGUI

- java.awt
 - Abstract Window Toolkit
- 基本グラフィックス
 - 色(Color)、線の属性(BasicStroke)、フォント
- 基本widget
 - パネル、ボタンなど部品群
- 基本イベント(java.awt.event)
 - マウス、キーボード、widgetの属性変化



JAVA.AWTのWIDGETS階層

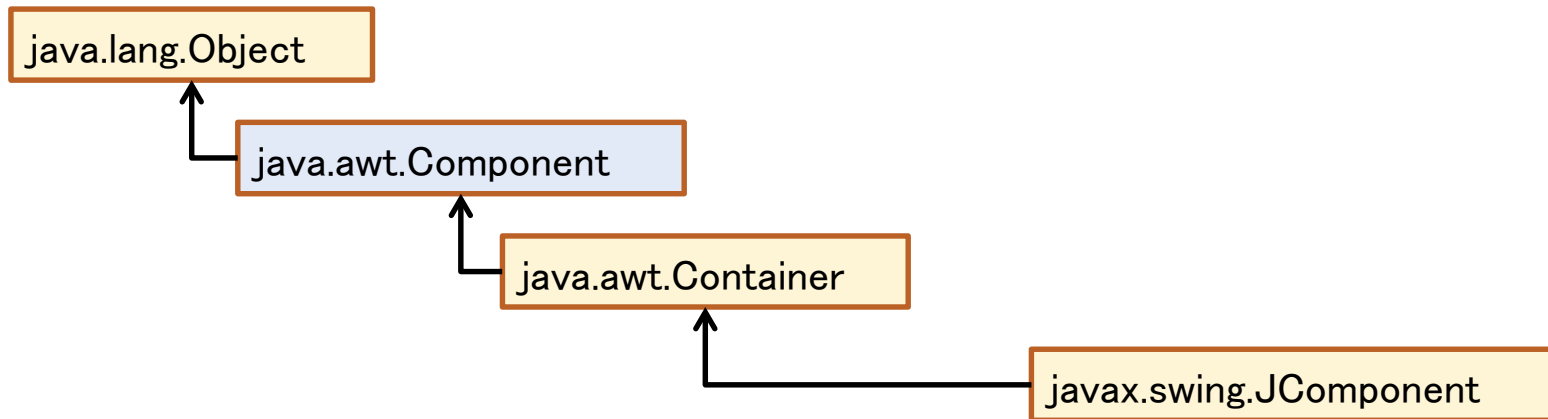


JAVA.AWTからJAVAX.SWINGへ

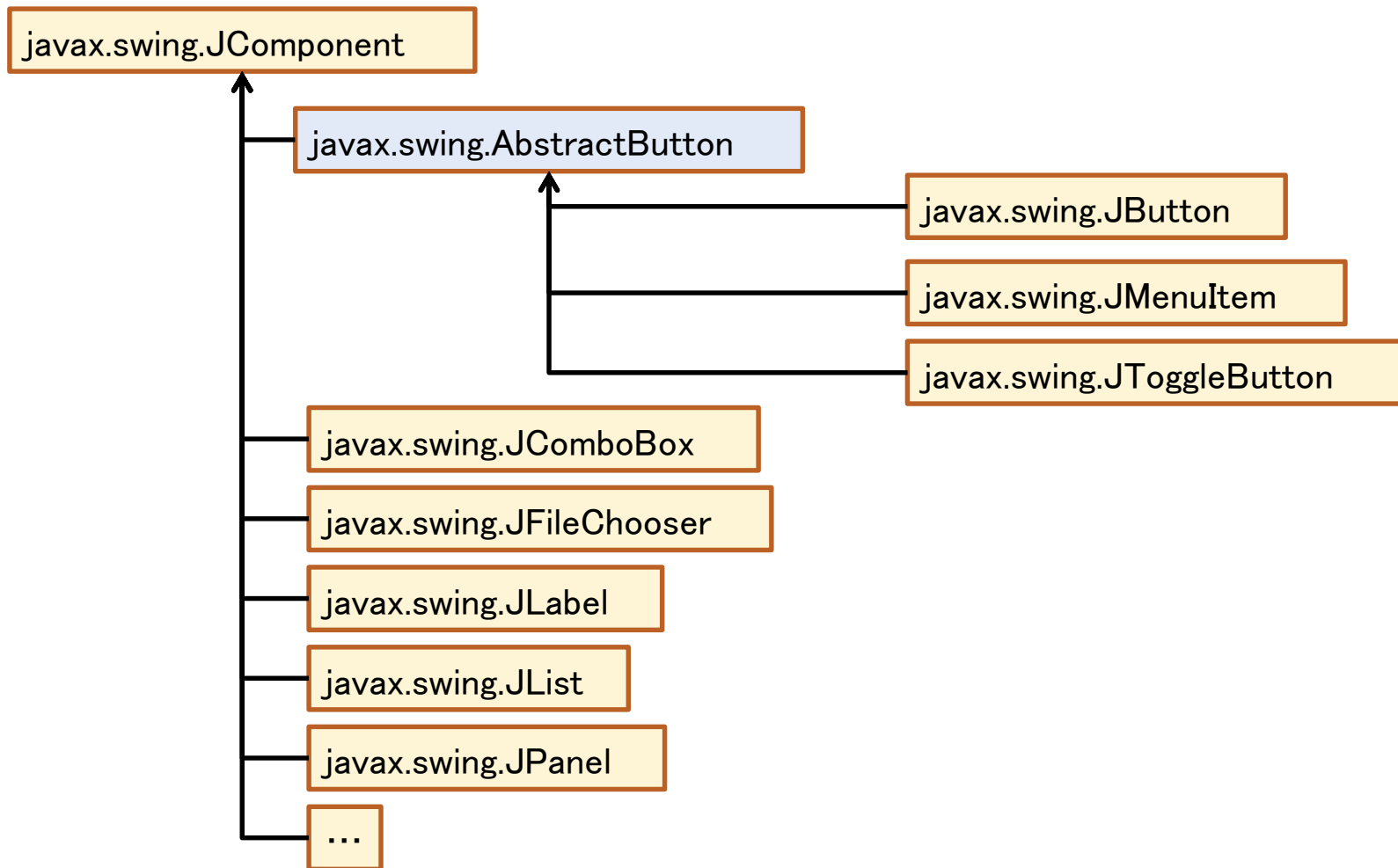
- 部品の充実
- プラットフォームからの完全独立
 - ウィンドウマネージャーとの連携
 - Look-and-Feelの分離
- 軽量化
- スレッド対応
 - イベント間通信



JAVAX.SWINGのクラス階層



JAVAX.SWINGのクラス階層



SWINGコンポーネントの役割

- javax.swing.JFrame
 - アプリケーションのウィンドウ
- javax.swing.JPanel
 - 様々なwidgetの台
 - 図形描画
- javax.swing.JButton
 - ボタン
- javax.swing.JLabel
 - 文字ラベル



例:動作の無いGUI:編集画面

widget一覧

widgetの階層

選択した widgetの属性

SampleGUI - NetBeans IDE 6.1

ファイル(E) 編集(E) 表示(V) ナビゲート(N) ソース(S) リファクタリング(A) 構築(B) 実行(R) プロファイル(P) バージョン管理(I) ツール(T) ウィンドウ(W) ヘルプ(H)

ファイル クラス

pubDevelopment

SampleGUI

ソースパッケージ

guiWithoutAction

MainFrame.java

テストパッケージ

ライブラリ

テストライブラリ

SimpleClient

SIR

StaffNameService

SurveyPersonalDataDevelopment

<init> - ナビゲータ インспекタ

フォーム MainFrame

他のコンポーネント

[JFrame]

BorderLayout

buttons [JPanel]

FlowLayout

jButton1 [JButton]

jComboBox1 [JComboBox]

ソース(S) デザイン(D)

「ツール」 > 「パレットマネージャー」 > 「Swing/AWT コンポーネント」メニュー項目

パレット

レイヤー-図面

Swing コントロール

label ラベル

ボタン

トグルボタン

チェックボックス

ラジオボタン

ボタングループ

コンボボックス

MainFrame.java - ...

プロパティ

名前 MainFrame

すべてのファイル /home/tad...

ファイルサイズ 1783

修正日時 2008/11/19 1

クラスパス

MainFrame.java

出力 - SampleGUI (run-single)

init:

deps-jar:

compile-single:

run-single:

構築成功 (合計時間: 7 秒)

NETBEANSでGUIを作る

- 通常と同様にプロジェクトを作成する
- JFrame作成
 - 「新規」→「JFrameフォーム」
 - レイアウト設定:「ボーダーレイアウト」



WIDGETの配置

- マウスによる配置
 - 「ナビゲーション」ウィンドウ内で
 - パレットからドラッグ
- JPanel作成
 - レイアウト設定
- widget配置
- widget動作設定



NETBEANSでGUIを作る時に注意

- GUIの配置情報はクラス名.formファイルに
 - クラス名.javaファイルからは編集できない部分がある
- 部品を拡張した自分のクラスも操作できる
- 実際の作成デモンストレーション



サンプルプログラム

- 動作の無いGUI
 - `guiWithoutAction`
- 動作の有るGUI
 - `guiWithAction`
- ファイルの選択
 - `fileChooser`
- タイマー
 - `simpleTimer`



GUIWITHOUTACTION: ソースコード前半

```
package guiWithoutAction;

public class MainFrame extends javax.swing.JFrame {
    //メニュー
    public enum MENU {
        MENU1, MENU2, MENU3;
    }

    /** Creates new form MainFrame */
    public MainFrame() {
        initComponents();
        /** メニューの設定 */
        for (MENU m : MENU.values()) {
            jComboBox1.addItem(m);
        }
        pack();
    }
}
```



initComponents()は
要素を配置するメソッド。
自動生成されている。



GUIWITHOUTACTION: ソースコード後半

```
public static void main(String args[]) {  
    java.awt.EventQueue.invokeLater(new Runnable() {  
  
        public void run() {  
            new MainFrame().setVisible(true);  
        }  
    });  
}  
// Variables declaration - do not modify  
private javax.swing.JPanel buttons;  
private javax.swing.JButton jButton1;  
private javax.swing.JComboBox<MENU> jComboBox1;  
// End of variables declaration  
}
```



自動的に生成されているコード

○ エディタで編集できない

```
// <editor-fold defaultstate="collapsed" desc="Generated Code">
```

```
private void initComponents() {
```

```
    buttons = new javax.swing.JPanel();
```

```
    jButton1 = new javax.swing.JButton();
```

```
    jComboBox1 = new javax.swing.JComboBox<MENU>();
```

widgetの生成

```
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```
    jButton1.setText("jButton1");
```

```
    buttons.add(jButton1);
```

buttonsへの配置

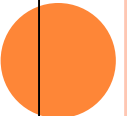
```
    buttons.add(jComboBox1);
```

```
    getContentPane().add(buttons, java.awt.BorderLayout.CENTER);
```

```
    pack();
```

```
}// </editor-fold>
```

buttonsを配置



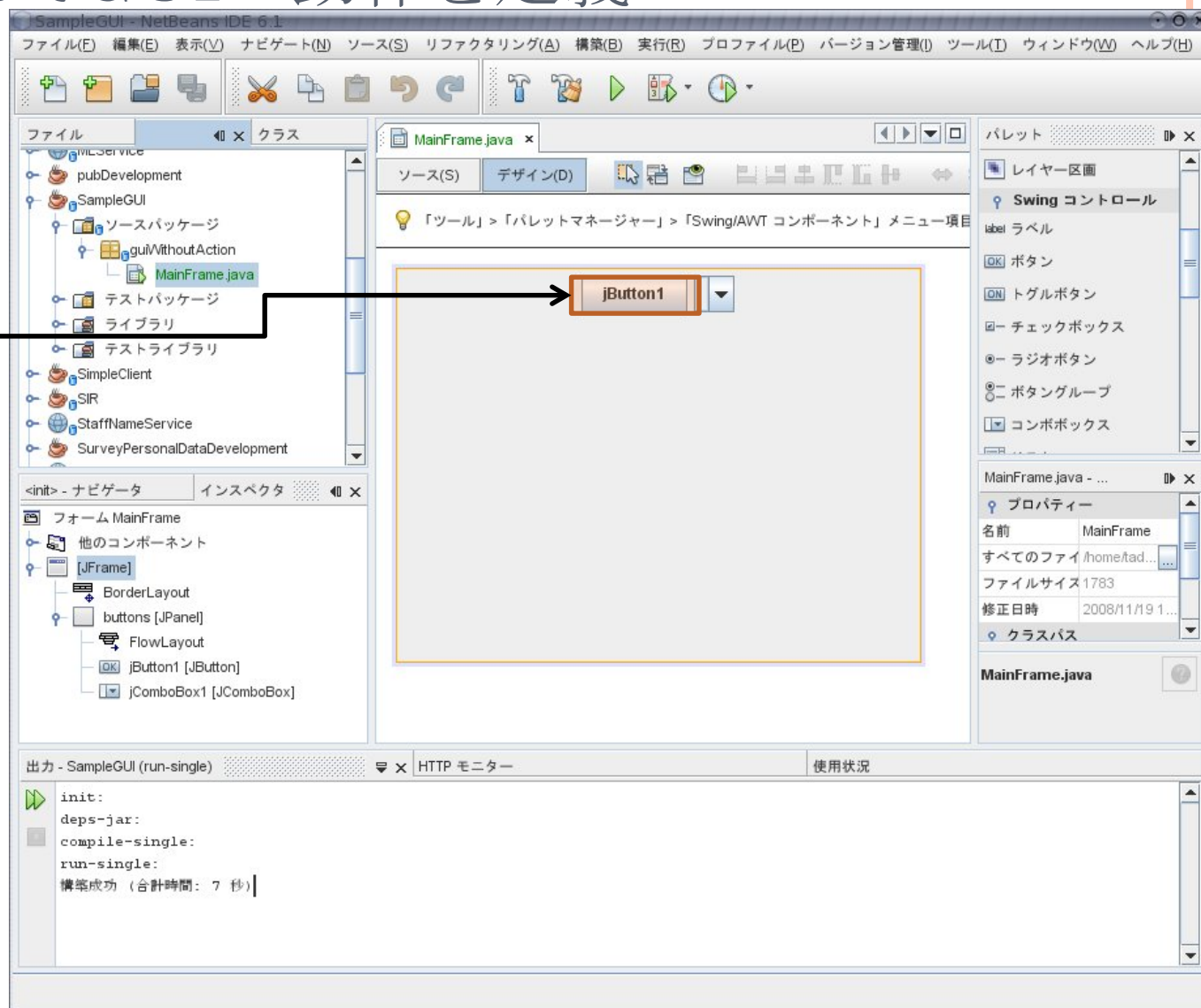
レイアウトマネージャ

- JFrameやJPanel内のwidgetの配置を管理
 - java.awt.BorderLayout
 - north (上端)、south (下端)、east (右端)、west (左端)、および center (中央)の領域にwidgetを配置
- java.awt.FlowLayout
 - widgetを一方向に配置
- java.awt.GridBagLayout
 - 矩形グリッドにwidgetを配置



NETBEANSでGUIの動作を定義

widgetをダブル
クリックして動作を
定義



- ボタンなどにactionListenerを定義する。
- actionを定義する。
- イベントを定義する



動作定義の例: ボタンの動作

```
private void initComponents() {  
    //省略  
  
    jButton1.addActionListener(new java.awt.event.ActionListener() {  
        public void actionPerformed(java.awt.event.ActionEvent evt) {  
            jButton1ActionPerformed(evt);  
        }  
    });  
    //省略  
}
```

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    System.out.println("jButton1 が押されました");  
}
```

動作記述

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    System.out.println("jButton1 が押されました");  
}
```

```
private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {  
    MENU m = (MENU) jComboBox1.getSelectedItem();  
    System.out.println(m.toString() + "が選ばれました");  
}
```

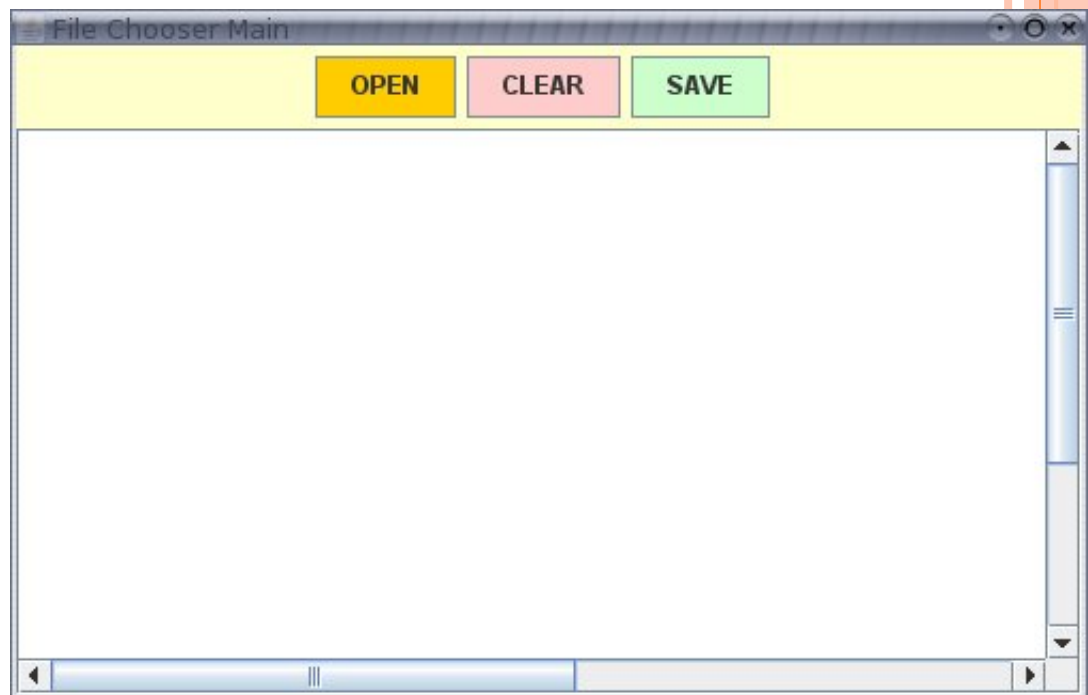
```
private void jSlider1StateChanged(javax.swing.event.ChangeEvent evt) {  
    int v = jSlider1.getValue();  
    System.out.println("jSlider1の値が" + String.valueOf(v) + "になりました。");  
}
```



例:ファイル選択

○ 機能

- ファイルを選択する
- テキストとして表示する
- ファイルを保存する
- エラーダイアログを表示する



OPENボタンの動作

```
private void openActionPerformed(java.awt.event.ActionEvent evt) {  
    //file chooserを生成し、テキストファイルに限定  
    JFileChooser chooser = new JFileChooser();  
    chooser.setFileFilter(new FileNameExtensionFilter("Text File", "txt"));  
  
    int returnVal = chooser.showOpenDialog(this); //ダイアログの表示  
    if (returnVal == JFileChooser.APPROVE_OPTION) {  
        File file = chooser.getSelectedFile();  
        textArea.setText(FileUtil.openFile(file));  
        textArea.setVisible(true);  
        fileNameLabel.setText(file.getName());  
    }  
}
```

SAVEボタンの動作

```
private void saveActionPerformed(java.awt.event.ActionEvent evt) {  
    JFileChooser chooser = new JFileChooser();  
    chooser.setFileFilter(new FileNameExtensionFilter("Text File", "txt"));  
    int returnVal = chooser.showSaveDialog(this);  
    if (returnVal == JFileChooser.APPROVE_OPTION) {  
        File file = chooser.getSelectedFile();  
        FileUtil.saveFile(file, textArea.getText());  
        fileNameLabel.setText(file.getName());  
    }  
}
```



標準のファイル選択GUI

- javax.swing.JFileChooserクラス
- 標準的ファイル選択画面を生成
 - 選択状態
 - 選択したファイルの情報
- FileNameExtensionFilterを使う
 - 拡張子での制限が可能になる



ファイル操作のクラス

○ FileUtilとして別に分けている

- 再利用可能
- インスタンスを生成しない
- 全てをstaticで定義

○ 機能

- ファイルから文字列を読み込む
- ファイルに文字列を保存する
- 書き込み可能性を確認する
- ダイアログを表示する
- ファイル名の拡張子を得る

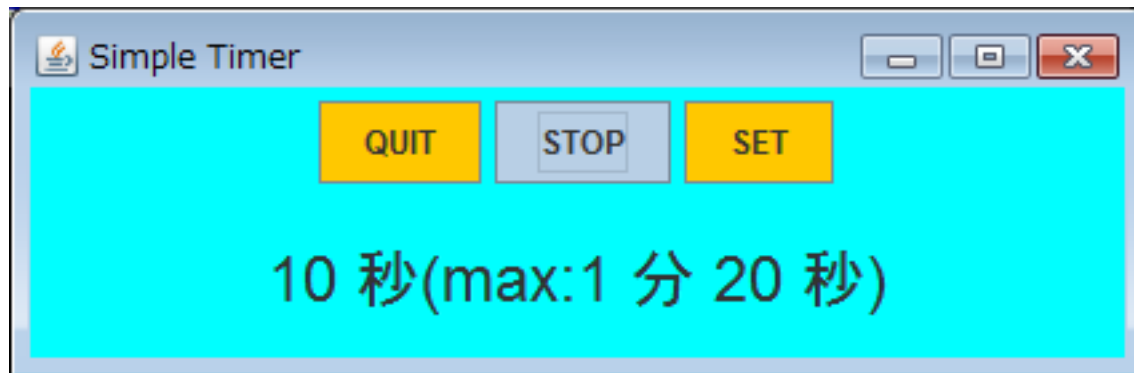


ダイアログの生成

```
static public void showError(String message) {  
    JOptionPane.showMessageDialog(  
        new JFrame(), message, "エラー発生",  
        JOptionPane.ERROR_MESSAGE);  
}  
  
static public void showMessage(String message) {  
    JOptionPane.showMessageDialog(  
        new JFrame(), message, "メッセージ",  
        JOptionPane.INFORMATION_MESSAGE);  
}
```



例:タイマー



○ ボタン

- 開始・停止のトグルボタン
- 終了ボタン
- 制限時間設定ボタン

○ タイマー本体

- JLabelの継承クラス
- 時刻を表示

○ 時間設定パネル

- 分・秒を設定
- JOptionPaneに組み込む



タイマー本体:TIMER

- JLabelの継承クラス
 - 時間を文字列にして表示
- インターフェイスRunnableを実装
 - スレッドとして、自律的に時間を進める
- 開始時の時刻nowと現在の時刻との差
 - 秒に変換
 - 文字列に変換して表示



本体: SIMPLETIMER

- START/STOPのトグルボタン
 - Timerクラスにstart/stopを送る
- QUITボタン
 - 終了
- SETボタン
 - Dialogを表示して、制限時間を設定



制限時間設定パネル: SETTIMEPANEL

- JOptionPaneのmessage objectとして使う
- 分と秒を設定するテキストフォーム
- OKボタンを押すとダイアログが閉じる
 - 上記の分と秒を読みだす



制限時間設定パネルの表示

```
private void setTimeActionPerformed(java.awt.event.ActionEvent evt) {  
    //停止する  
    toggle.setSelected(false);  
    toggle.setText("START");  
    timerLabel.stop();  
    //設定用Dialogの表示  
    int answer = JOptionPane.showOptionDialog(new JFrame(), setTimePanel,  
        "時間設定", JOptionPane.OK_CANCEL_OPTION,  
        JOptionPane.QUESTION_MESSAGE, null, null, null);  
    if(answer == JOptionPane.OK_OPTION){  
        //OKが押されたときに、制限時間を設定  
        int m = setTimePanel.getMinute();  
        int s = setTimePanel.getSecond();  
        timerLabel.setMax(60*m+s);  
    } else {  
        setTimePanel.setDefault();  
    }  
}
```

