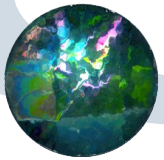
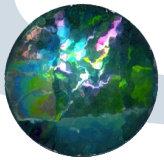
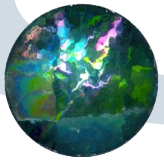


Graphical User Interface
widgetを使う



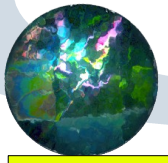
JavaとGUI

- 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

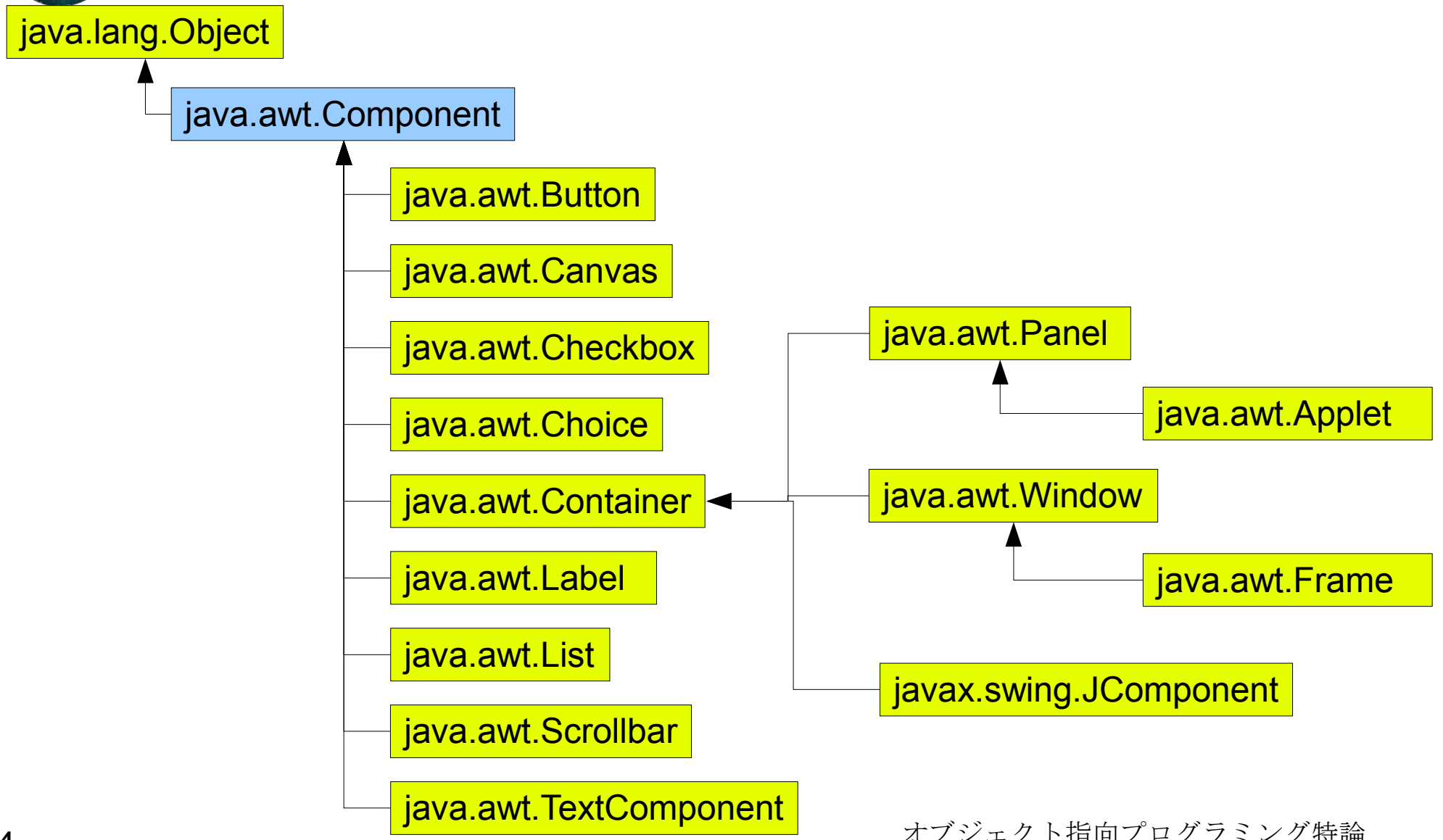


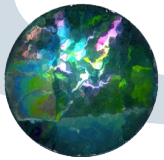
JavaとGUI

- 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



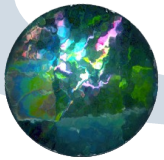
java.awtのwidgets



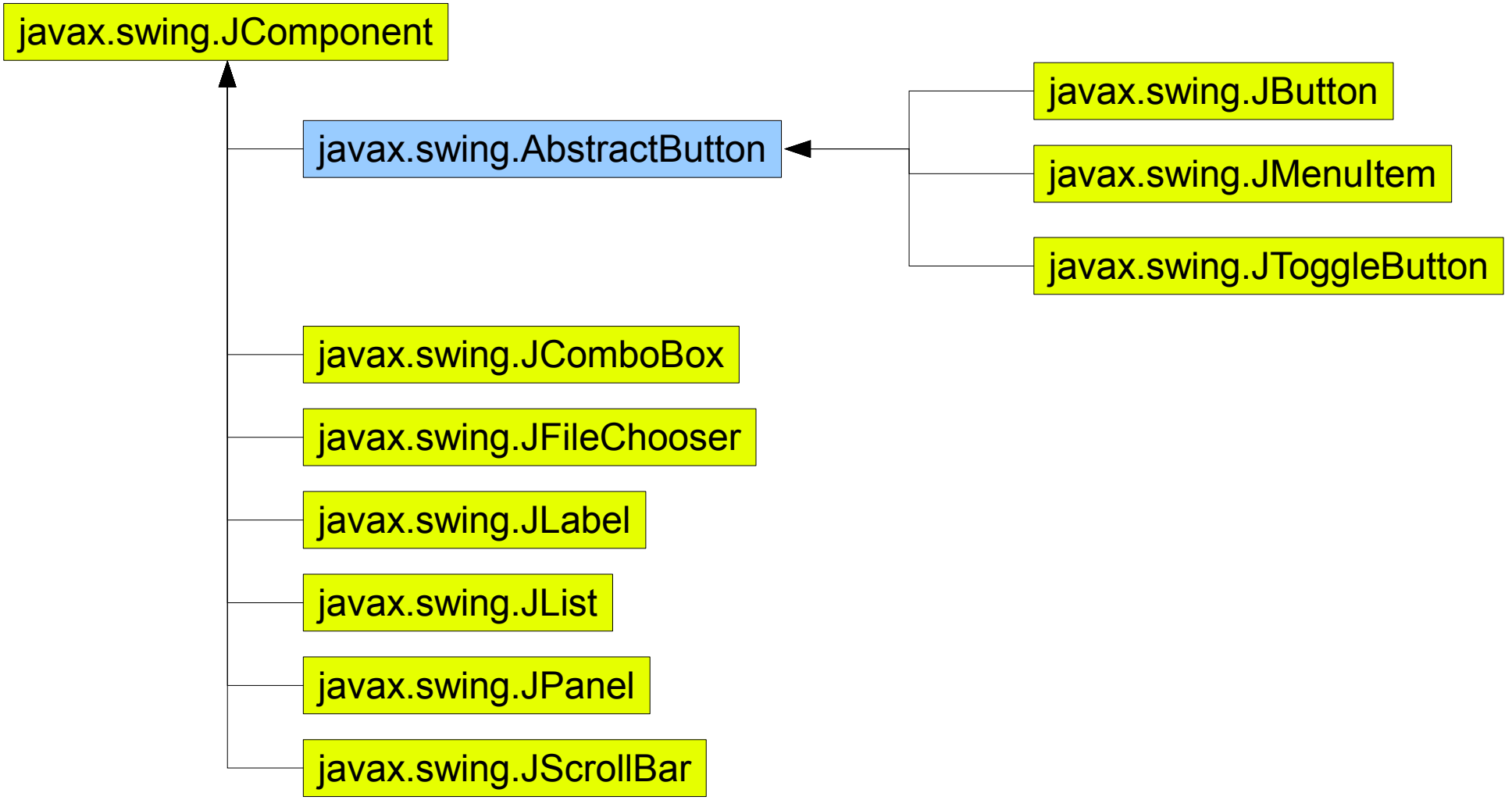


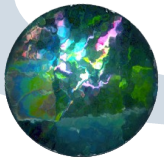
java.awtからjavax.swingへ

- 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
  - イベント通信



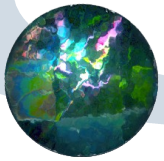
javax.swing


javax.swing.JComponent
javax.swing.AbstractButton
javax.swing.JButton
javax.swing.JMenuItem
javax.swing.JToggleButton
javax.swing.JComboBox
javax.swing.JFileChooser
javax.swing.JLabel
javax.swing.JList
javax.swing.JPanel
javax.swing.JScrollBar

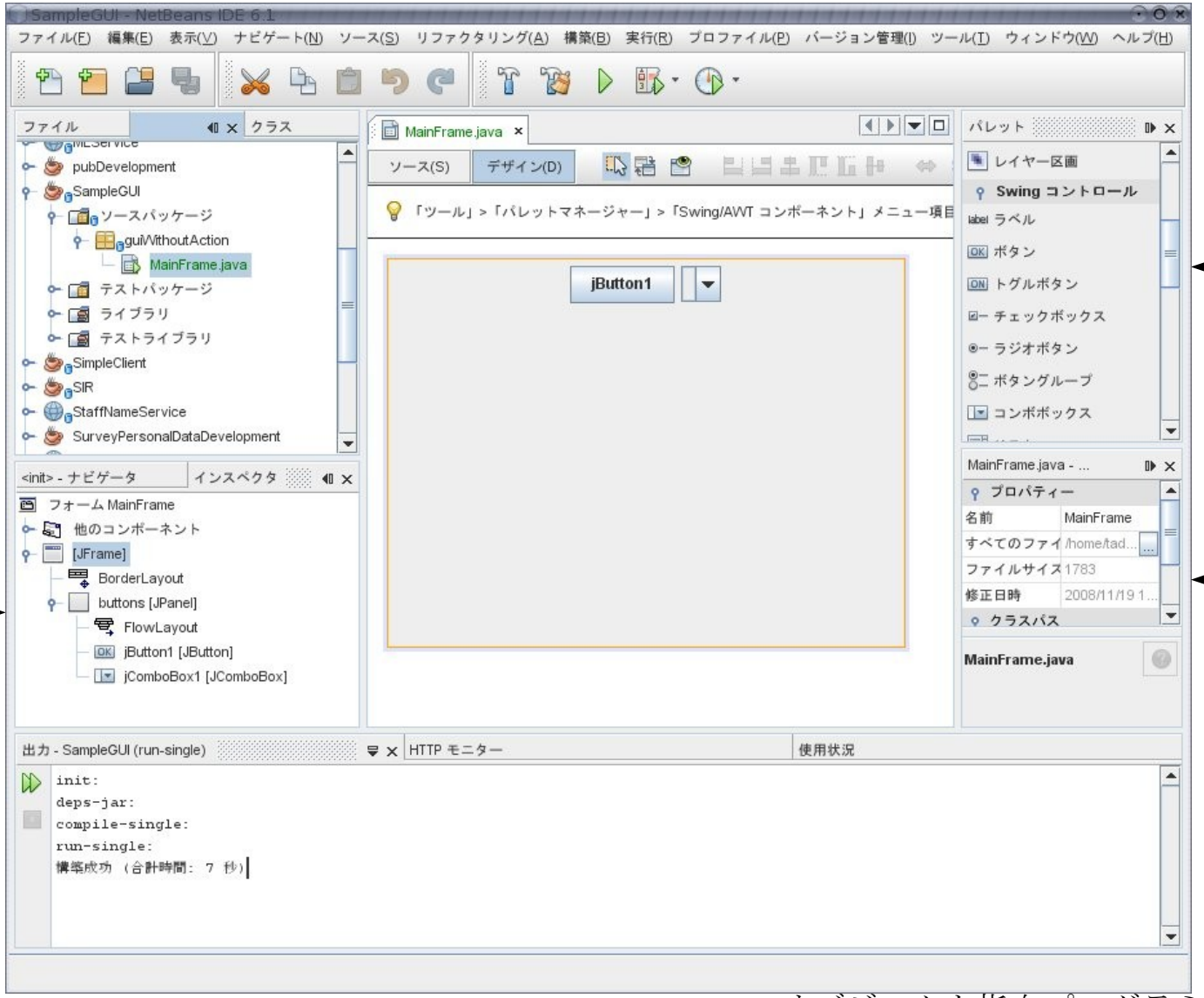


swingコンポーネントの役割

- 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



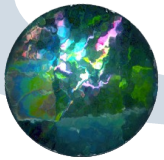
例：動作の無いGUI



widgets
パーツ

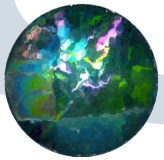
Widgetの階層

選択した
widgetの
プロパティ



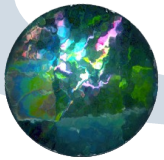
NetBeansでGUIを作る

- 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」



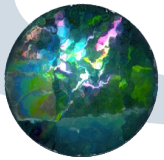
widgetの配置

- 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



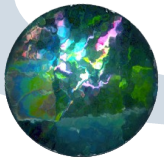
NetBeansでGUIを作る時に注意

- 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



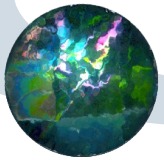
サンプルプログラム

- 動作の無いGUI
  - guiWithoutAction
- 動作の有るGUI
  - guiWithAction
- ファイルの選択
  - fileChooser

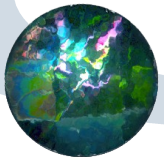



```
package guiWithoutAction;

public class MainFrame extends javax.swing.JFrame {
    public enum MENU{
        MENU1,MENU2,MENU3;
    }
    /** Creates new form MainFrame */
    public MainFrame() {
        initComponents();
        for(MENU m:MENU.values()){
            jComboBox1.addItem(m);
        }
        pack();
    }
```

initComponents()は
自動生成される



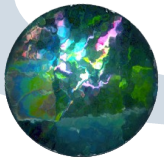
```
    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MainFrame().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JPanel buttons;
    private javax.swing.JButton jButton1;
    private javax.swing.JComboBox jComboBox1;
    // End of variables declaration

}
```



自動的に生成されているコード

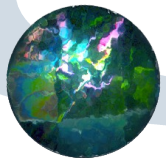
- エディタで編集できない

```
private void initComponents() {
                                                    widgetの生成
    buttons = new javax.swing.JPanel();
    jButton1 = new javax.swing.JButton();
    jComboBox1 = new javax.swing.JComboBox();                ウィンドウを閉じる動作

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jButton1.setText("jButton1");
    buttons.add(jButton1);                                   widgetの追加

    buttons.add(jComboBox1);

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

    pack();
}
```



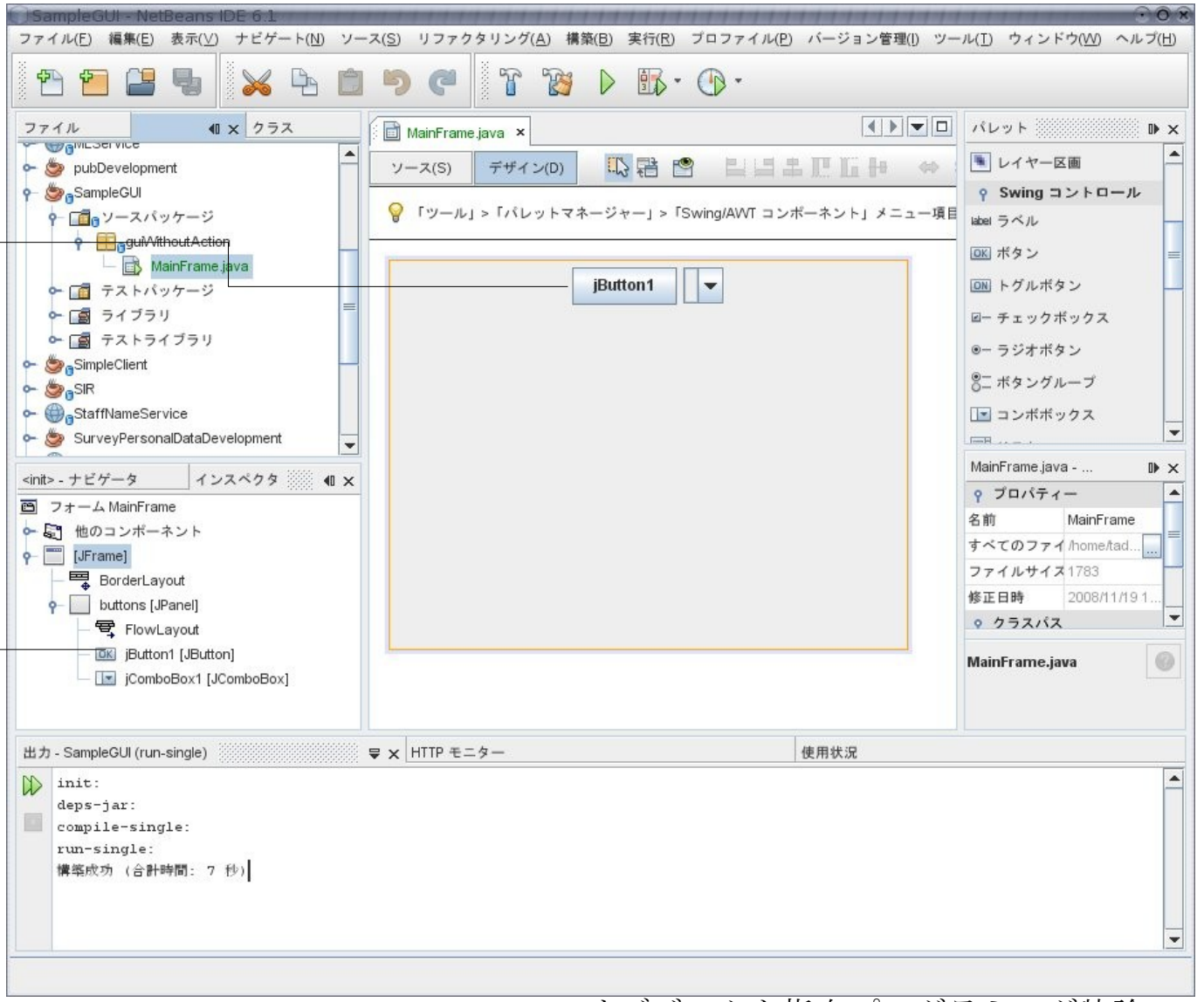
レイアウトマネージャ

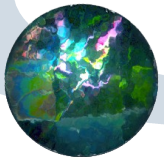
- JFrameやJPanel内のwidgetの配置を管理
- java.awt.BorderLayout
  - north (上端)、south (下端)、east (右端)、west (左端)、および center (中央)の領域にwidgetを配置
- java.awt.FlowLayout
  - widgetを一方向に配置
- java.awt.GridBagLayout
  - 矩形グリッドにwidgetを配置



NetBeansでGUIの動作を定義

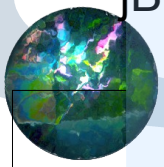
コンポーネントを
ダブルクリックして
動作を定義する



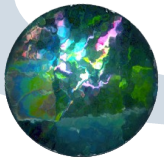


- ボタンなどにactionListenerを定義する。
- actionを定義する。
- イベントを定義する

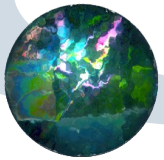
```
private void initComponents() {
    //省略
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });
    jComboBox1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jComboBox1ActionPerformed(evt);
        }
    });
}
```



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}

private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}
```

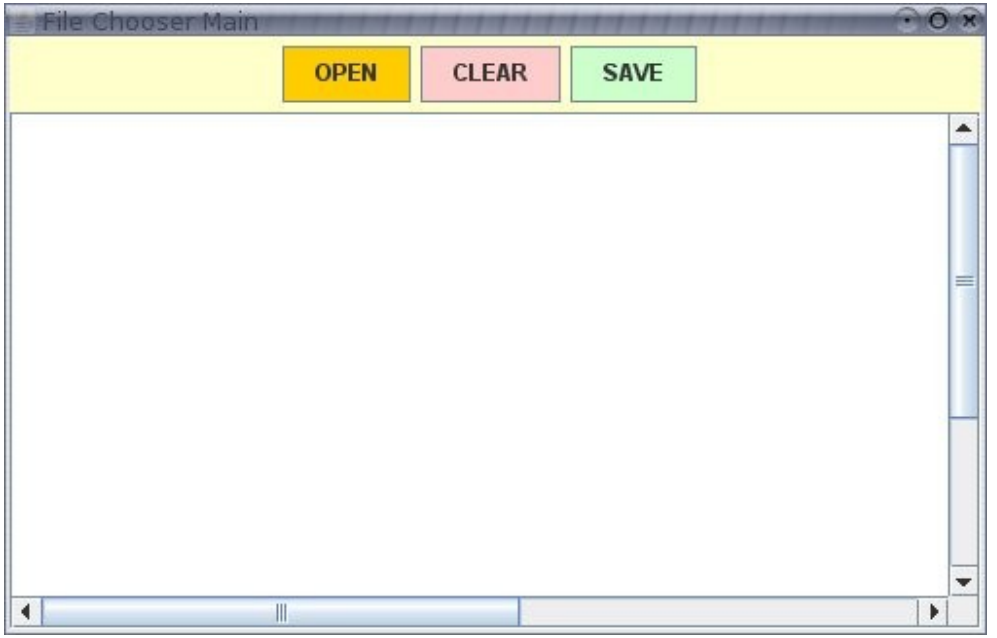


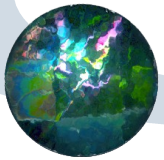
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    System.out.println("jButton1 が押されました");
}

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

```
ivate void openActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser chooser = new JFileChooser();
    FileNameExtensionFilter filter =
        new FileNameExtensionFilter("Text File", "txt");
    chooser.setFileFilter(filter);
    int returnVal = chooser.showOpenDialog(this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
        File file = chooser.getSelectedFile();
        FileUtil.openFile(file, textArea);
        fileNameLabel.setText(file.getName());
    }
}
```

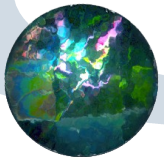
ファイル選択ダイアログ生成

ファイルフィルタ設定

ファイル選択ダイアログで
OKが押された場合

ファイルの内容をtextAreaに表示

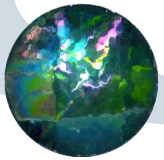
ファイル名をfileNameLabelに表示



saveボタンの動作

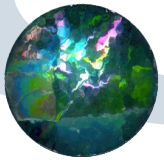
```
private void saveActionPerformed(java.awt.event.ActionEvent evt) {
    JFileChooser chooser = new JFileChooser();
    FileNameExtensionFilter filter =
            new FileNameExtensionFilter("Text File", "txt");
    chooser.setFileFilter(filter);
    int returnVal = chooser.showSaveDialog(this);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

保存用のダイアログであることに注意



標準のファイル選択GUI

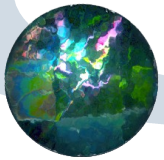
- javax.swing.JFileChooserクラス
- 標準的ファイル選択画面を生成
  - 選択状態
  - 選択したファイルの情報
- FileNameExtensionFilterを使う
  - 拡張子での制限が可能になる



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
- 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

ユーティリティとして使えるメソッドをstaticとして定義



ダイアログの生成

```
static public boolean checkOverwrite(String filename) {
        boolean b = true;
        String message = filename + "は存在します。上書きしますか？";
        int answer = JOptionPane.showConfirmDialog(
                new JFrame(), message, "上書き確認",
                JOptionPane.OK_CANCEL_OPTION);
        if (answer != JOptionPane.OK_OPTION) {
            b = false;
        }
        return b;
    }

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

MainFrame.java

```
/*
 * MainFrame.java
 * Created on 2008/11/19, 14:35
 * @author  tadaki
 */
package guiWithoutAction;

public class MainFrame extends javax.swing.JFrame {

    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();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        buttons = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jComboBox1 = new javax.swing.JComboBox();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

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

```
        buttons.add(jComboBox1);

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

        pack();
    }// </editor-fold>//GEN-END:initComponents

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new MainFrame().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JPanel buttons;
    private javax.swing.JButton jButton1;
    private javax.swing.JComboBox jComboBox1;
    // End of variables declaration//GEN-END:variables
}
```

MainFrame.java

```
/*
 * MainFrame.java
 * Created on 2008/11/19, 14:35
 * @author  tadaki
 */
package guiWithAction;

public class MainFrame extends javax.swing.JFrame {

    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();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        buttons = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jComboBox1 = new javax.swing.JComboBox();
        jSlider1 = new javax.swing.JSlider();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jButton1.setText("jButton1");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
        buttons.add(jButton1);

        jComboBox1.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jComboBox1ActionPerformed(evt);
            }
        });
        buttons.add(jComboBox1);

        jSlider1.setMinorTickSpacing(10);
        jSlider1.setPaintLabels(true);
        jSlider1.setPaintTicks(true);
        jSlider1.addChangeListener(new
javax.swing.event.ChangeListener() {
            public void stateChanged(javax.swing.event.ChangeEvent evt) {
                jSlider1StateChanged(evt);
            }
        });
        buttons.add(jSlider1);

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

        pack();
    }// </editor-fold>//GEN-END:initComponents

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton1ActionPerformed
    System.out.println("jButton1 が押されました");
}//GEN-LAST:event_jButton1ActionPerformed

private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jComboBox1ActionPerformed
    MENU m = (MENU) jComboBox1.getSelectedItem();
    System.out.println(m.toString() + "が選ばれました");
}//GEN-LAST:event_jComboBox1ActionPerformed

private void jSlider1StateChanged(javax.swing.event.ChangeEvent evt)
{//GEN-FIRST:event_jSlider1StateChanged
    int v = jSlider1.getValue();
```

```
        System.out.println("jSlider1の値が" + String.valueOf(v) + "になりました。");
    }//GEN-LAST:event_jSlider1StateChanged

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new MainFrame().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JPanel buttons;
    private javax.swing.JButton jButton1;
    private javax.swing.JComboBox jComboBox1;
    private javax.swing.JSlider jSlider1;
    // End of variables declaration//GEN-END:variables
}
```

FileChooserMain.java

```
/*
 * FileChooserMain.java
 *
 * Created on 2008/11/21, 16:08
 */
package fileChooser;

import java.io.File;
import javax.swing.JFileChooser;
import javax.swing.filechooser.FileNameExtensionFilter;

/**
 *
 * @author  tadaki
 */
public class FileChooserMain extends javax.swing.JFrame {

    /** Creates new form FileChooserMain */
    public FileChooserMain() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        buttons = new javax.swing.JPanel();
        open = new javax.swing.JButton();
        clear = new javax.swing.JButton();
        save = new javax.swing.JButton();
        fileNameLabel = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        textArea = new javax.swing.JTextArea();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("File Chooser Main");
```

```
        buttons.setBackground(new java.awt.Color(255, 255, 204));

        open.setBackground(new java.awt.Color(255, 204, 0));
        open.setText("OPEN");
        open.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                openActionPerformed(evt);
            }
        });
        buttons.add(open);

        clear.setBackground(new java.awt.Color(255, 204, 204));
        clear.setText("CLEAR");
        clear.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                clearActionPerformed(evt);
            }
        });
        buttons.add(clear);

        save.setBackground(new java.awt.Color(204, 255, 204));
        save.setText("SAVE");
        save.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                saveActionPerformed(evt);
            }
        });
        buttons.add(save);
        buttons.add(fileNameLabel);

        getContentPane().add(buttons, java.awt.BorderLayout.NORTH);

        textArea.setColumns(80);
        textArea.setRows(20);
        jScrollPane1.setViewportView(textArea);

        getContentPane().add(jScrollPane1, java.awt.BorderLayout.CENTER);

        pack();
    }// </editor-fold>//GEN-END:initComponents

 private void openActionPerformed(java.awt.event.ActionEvent evt)
 {//GEN-FIRST:event_openActionPerformed
     JFileChooser chooser = new JFileChooser();
```

```
        FileNameExtensionFilter filter =
                new FileNameExtensionFilter("Text File", "txt");
        chooser.setFileFilter(filter);
        int returnVal = chooser.showOpenDialog(this);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = chooser.getSelectedFile();
            textArea.setText(FileUtil.openFile(file));
            textArea.setVisible(true);
            fileNameLabel.setText(file.getName());
        }
    }//GEN-LAST:event_openActionPerformed

    private void clearActionPerformed(java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_clearActionPerformed
        textArea.setText(null);
    }//GEN-LAST:event_clearActionPerformed

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

        /**
         * @param args the command line arguments
         */
        public static void main(String args[]) {
            java.awt.EventQueue.invokeLater(new Runnable() {

                public void run() {
                    new FileChooserMain().setVisible(true);
                }
            });
        }
        // Variables declaration - do not modify//GEN-BEGIN:variables
        private javax.swing.JPanel buttons;
```

FileChooserMain.java

```
    private javax.swing.JButton clear;
    private javax.swing.JLabel fileNameLabel;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JButton open;
    private javax.swing.JButton save;
    private javax.swing.JTextArea textArea;
    // End of variables declaration//GEN-END:variables
}
```

FileUtil.java

```java
/**
 *
 * @author tadaki
 */
package fileChooser;

import javax.swing.JFrame;
import javax.swing.JOptionPane;
import java.io.*;

public class FileUtil {

    /**
     * fileの内容を文字列として返す
     * @param file 読み込むfile
     * @return fileから読み込まれた文字列
     */
    static public String openFile(File file) {
        if (!file.canRead()) {//ファイルが読めることを確認
            showError(file.getName() + " は読めません");
            return null;
        }
        //読み出し開始
        BufferedReader in = null;
        try {
            in = new BufferedReader(
                    new InputStreamReader(new FileInputStream(file)));
        } catch (FileNotFoundException ex) {
            showError(ex.getMessage());
            return null;
        }
        StringBuilder buf = new StringBuilder();
        try {
            String line;
            String nl = System.getProperty("line.separator");

            while ((line = in.readLine()) != null) {
                buf.append(line);
                buf.append(nl);
            }
            in.close();
        } catch (IOException ex) {
            showError(ex.getMessage());
            return null;
```

```
            }
            return buf.toString();
        }

        /**
         * textAreaの内容をfileに保存する。
         * @param file  保存先file
         * @param 保存する文字列
         */
        static public void saveFile(File file, String text) {
            if (!checkWritable(file)) {
                return;
            }
            try {
                //保存開始
                BufferedWriter out;
                try {
                    out = new BufferedWriter(
                            new OutputStreamWriter(new
 FileOutputStream(file)));
                } catch (FileNotFoundException ex) {
                    showError(ex.getMessage());
                    return;
                }
                out.write(text);
                out.close();
            } catch (IOException ex) {
                showError(ex.getMessage());
            }
        }

        /**
         * file への書き込み可能性を確認
         * @param file
         * @return 書き込み可能ならばtrue
         */
        static public boolean checkWritable(File file) {
            boolean isWritable = true;
            if (file.isFile()) {//ファイルが存在する場合
                if (!file.canWrite()) {//上書き可能性の確認
                    showError(file.getName() + " に書き込めません");
                    return false;
                } else {
                    if (!checkOverwrite(file.getName())) {
```

```
                    return false;
                }
            }
        } else {
            try {
                if (!file.createNewFile()) {//新規ファイル作成
                    showError(file.getName() + " を生成できません");
                    return false;
                }
            } catch (IOException ex) {
                showError(ex.getMessage());
                return false;
            }
        }
        return isWritable;
    }

    /**
     * 上書き保存の確認ダイアログを表示
     * @param filename  保存先ファイル名
     * @return 上書きする場合true
     */
    static public boolean checkOverwrite(String filename) {
        boolean b = true;
        String message = filename + "は存在します。上書きしますか？";
        int answer = JOptionPane.showConfirmDialog(
                new JFrame(), message, "上書き確認",
                JOptionPane.OK_CANCEL_OPTION);
        if (answer != JOptionPane.OK_OPTION) {
            b = false;
        }
        return b;
    }

    /**
     * エラーを示すダイアログを表示
     * @param message エラーメッセージ
     */
    static public void showError(String message) {
        JOptionPane.showMessageDialog(
                new JFrame(), message, "エラー発生",
                JOptionPane.ERROR_MESSAGE);
    }
```

FileUtil.java

```
    /**
     * メッセージを示すダイアログを表示
     * @param message エラーメッセージ
     */
    static public void showMessage(String message) {
        JOptionPane.showMessageDialog(
                new JFrame(), message, "メッセージ",
                JOptionPane.INFORMATION_MESSAGE);
    }

    /**
     * ファイル名中の拡張子を調べる
     * @param filename
     * @return 拡張子の文字列
     */
    static public String getExtention(String filename) {
        String ext = null;
        int index = filename.lastIndexOf(".");
        if (index > 0) {
            ext = filename.substring(index + 1);
            if (ext.length() < 1) {
                ext = null;
            }
        }
        return ext;
    }
}
```