



# Graphical User Interface widgetを使う2

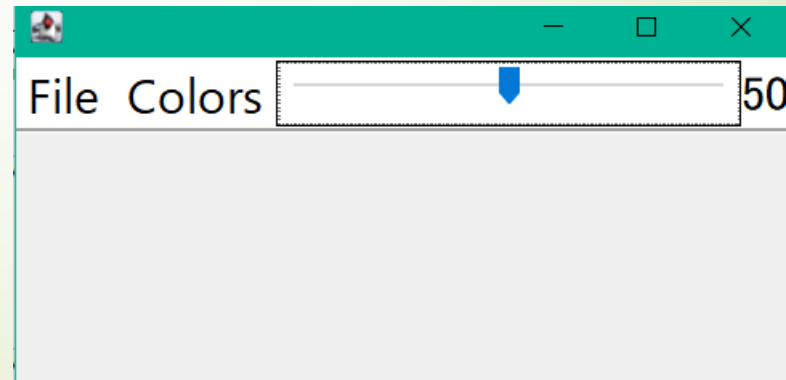
オブジェクト指向プログラミング特論

2020年度

只木進一：工学系研究科

# ボタンの動作を定義する

- **widget**には、イベントを扱う機能がある
  - メニューなどに**actionListener**を設定する。
  - 動作内容を定義する。



# NetBeansでGUIの動作を定義

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

401.1/888.0MB

Projects - O... x Files Services

Main.java x

Source Design History

To select multiple components in an area hold Shift and drag mouse over the component

File Colors

exit

exit [JMenuItem] - Navigator x

Form Main

Other Components

[JFrame]

menuBar [JMenuBar]

fileMenu [JMenu]

exit [JMenuItem]

selectColors [JMenu]

BorderLayout

panel [JPanel]

Swing Containers

Tabbed Pane

Split Pane

Panel

Scroll Pane

Tool Bar

Desktop Pane

Internal Frame

Layered Pane

Swing Controls

Label

Button

Toggle Button

Check Box

Radio Button

Button Group

Combo Box

List

Text Field

Text Area

Scroll Bar

Slider

Progress Bar

Formatted Field

Password Field

Spinner

Separator

Text Pane

exit [JMenuItem] - Properties x

Properties Events Code

Properties

action	<none>
accelerator	null
background	[240,240,240]
font	Yu Gothic UI 24 Plain
foreground	[0,0,0]
icon	<none>
mnemonic	
text	exit

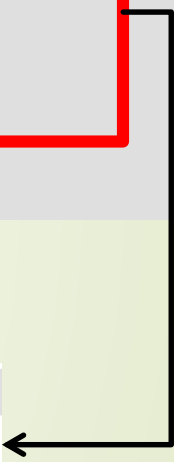
exit [JMenuItem]

widgetをダブルクリックして動作を定義

# 動作定義の例：メニューの動作

```
113 exit.setFont(new java.awt.Font("Yu Gothic UI", 0, 24)); // NOI18N
114 exit.setText("exit");
115 exit.addActionListener(new java.awt.event.ActionListener() {
116     public void actionPerformed(java.awt.event.ActionEvent evt) {
117         exitActionPerformed(evt);
118     }
119 });
120 fileMenu.add(exit);
```

```
133 private void exitActionPerformed(java.awt.event.ActionEvent evt) {
134     dispose();
135 }
```

A black arrow originates from the `exitActionPerformed(evt);` line in the first code block and points to the `exitActionPerformed` method definition in the second code block.

## 動作定義の例：色の選択

```
39 for (Colors m : Colors.values()) {  
40     JMenuItem item = new JMenuItem(m.toString());  
41     item.setFont(font);  
42     item.addActionListener(e -> colorItemPerformed(m));  
43     selectColors.add(item);  
44 }
```

```
66 /**  
67  * 色を選択した際の動作  
68  *  
69  * @param c  
70  */  
71 private void colorItemPerformed(Colors c) {  
72     System.out.println(c.toString());  
73     panel.setBackground(c.getColor());  
74 }  
75
```

## 動作定義の例 : slider

```
46 //スライダーの生成
47 slider = new JSlider();
48 slider.setFont(font);
49 slider.setPaintTicks(true);
50 slider.setBorder(BorderFactory.createLineBorder(Color.BLACK));
51 slider.setBackground(Color.white);
52 //動作の定義
53 slider.addChangeListener(e -> sliderStateChanged(e));
54 menuBar.add(slider); //メニューバーへの追加
```

```
76 /**
77  * スライダーを操作した際の動作
78  *
79  * @param evt
80  */
81 private void sliderStateChanged(javax.swing.event.ChangeEvent evt) {
82     int v = slider.getValue();
83     label.setText(String.valueOf(v));
84 }
```

# 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

```
105 private void openMenuActionPerformed(java.awt.event.ActionEvent evt) {  
106     //file chooserを生成し、テキストファイルに限定  
107     JFileChooser chooser = new JFileChooser();  
108     chooser.setCurrentDirectory(dir);  
109     chooser.setFileFilter(  
110         new FileNameExtensionFilter("Text File", "txt"));  
111  
112     int returnVal = chooser.showOpenDialog(this); //ダイアログの表示  
113     if (returnVal == JFileChooser.APPROVE_OPTION) {  
114         File file = chooser.getSelectedFile(); //選択したファイル  
115         textArea.setText(FileUtilGUI.openFile(file)); //textAreaへ表示  
116         textArea.setVisible(true);  
117         filename = file.getName();  
118         setTitle(applicationName+" "+filename);  
119         dir = file.getParentFile();  
120     }  
121 }
```

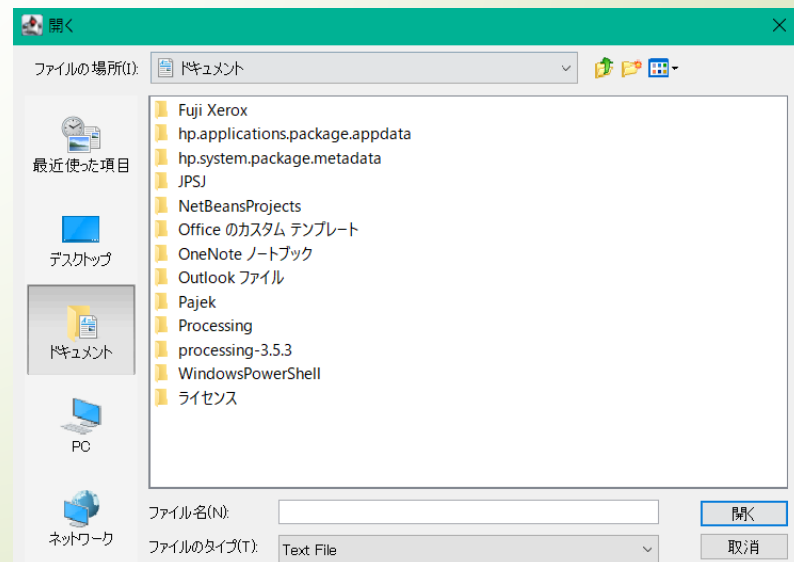


# saveボタンの動作

```
131 private void saveTextActionPerformed(java.awt.event.ActionEvent evt) {  
132     //file chooserを生成し、テキストファイルに限定  
133     JFileChooser chooser = new JFileChooser();  
134     chooser.setCurrentDirectory(dir);  
135     chooser.setFileFilter(new FileNameExtensionFilter("Text File", "txt"));  
136  
137     int returnVal = chooser.showSaveDialog(this);  
138     if (returnVal == JFileChooser.APPROVE_OPTION) {  
139         File file = chooser.getSelectedFile();  
140         FileUtilGUI.saveFile(file, textArea.getText());  
141         filename = file.getName();  
142         this.setTitle(applicationName+" "+filename);  
143         dir = file.getParentFile();  
144     }  
145 }
```

# 標準のファイル選択GUI 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で拡張子制限



# ファイル操作のクラス

- **FileUtilGUI**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

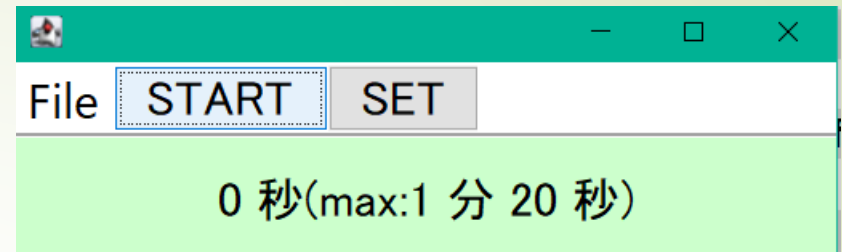
```
124  /**
125   * エラーを示すダイアログを表示
126   *
127   * @param message エラーメッセージ
128   */
129  static public void showError(String message) {
130      JOptionPane.showMessageDialog(
131          new JFrame(), message, "エラー発生",
132          JOptionPane.ERROR_MESSAGE);
133  }
134
135  /**
136   * メッセージを示すダイアログを表示
137   *
138   * @param message エラーメッセージ
139   */
140  static public void showMessage(String message) {
141      JOptionPane.showMessageDialog(
142          new JFrame(), message, "メッセージ",
143          JOptionPane.INFORMATION_MESSAGE);
144  }
```

# ファイル上書きを確認する dialog

```
106  /**
107  * 上書き保存の確認ダイアログを表示
108  *
109  * @param filename 保存先ファイル名
110  * @return 上書きする場合true
111  */
112  static public boolean checkOverwrite(String filename) {
113      boolean b = true;
114      String message = filename + "は存在します。上書きしますか?";
115      int answer = JOptionPane.showConfirmDialog(
116          new JFrame(), message, "上書き確認",
117          JOptionPane.OK_CANCEL_OPTION);
118      if (answer != JOptionPane.OK_OPTION) {
119          b = false;
120      }
121      return b;
122  }
```

```
81 static public boolean checkWritable(File file) {
82     boolean isWritable = true;
83     if (file.isFile()) { //ファイルが存在する場合
84         if (!file.canWrite()) { //上書き可能性の確認
85             showError(file.getName() + " に書き込めません");
86             return false;
87         } else {
88             if (!checkOverwrite(file.getName())) {
89                 return false;
90             }
91         }
92     } else {
93         try {
94             if (!file.createNewFile()) { //新規ファイル作成
95                 showError(file.getName() + " を生成できません");
96                 return false;
97             }
98         } catch (IOException ex) {
99             showError(ex.getMessage());
100             return false;
101         }
102     }
103     return isWritable;
104 }
```

# 例：タイマー



- メニュー
  - 開始・停止のトグルボタン
  - 終了メニュー
  - 制限時間設定メニュー
- 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



# 時間の経過とともに、時間表示を更新

## ■ Timer クラス

- Runnable インターフェイス
- 開始時刻と現在時刻の差

```
29      @Override
30      public void run() {
31          while (running) {
32              setTime();
33              try {
34                  Thread.sleep(100);
35              } catch (InterruptedException ex) {
36              }
37          }
38      }
```

```
97  /**
98  * 時刻設定
99  *
100 * @return 終了時間を過ぎるとfalse
101 */
102 public boolean setTime() {
103     Calendar c = Calendar.getInstance();
104     //開始時刻から現在までの秒数
105     int d = (int) (c.getTimeInMillis() - startDate.getTimeInMillis()) / 1000;
106     setTimeString(d);
107     if (d >= max) { //終了を過ぎている
108         setForeground(foregroundOver);
109         return false;
110     }
111     setVisible(true);
112     return true;
113 }
```

```
43 //START/STOPのトグルボタンの生成と配置
44 toggle = new JToggleButton("START");
45 toggle.setFont(font);
46 toggle.addActionListener(evt -> toggleActionPerformed(evt));
47 menuBar.add(toggle);
48
49 //タイマーの設定のダイアログを起動するボタンの生成と配置
50 setButton = new JButton("SET");
51 setButton.setFont(font);
52 setButton.addActionListener(e -> setTimeActionPerformed(e));
53 menuBar.add(setButton);
```

# SETボタンの動作

```
81 private void setTimeActionPerformed(java.awt.event.ActionEvent evt) {  
82     //停止する  
83     toggle.setSelected(false);  
84     toggle.setText("START");  
85     timerLabel.stop();  
86     //設定用Dialogの表示  
87     int answer = JOptionPane.showOptionDialog(new JFrame(), setTimePanel,  
88         "時間設定", JOptionPane.OK_CANCEL_OPTION,  
89         JOptionPane.QUESTION_MESSAGE, null, null, null);  
90     if (answer == JOptionPane.OK_OPTION) {  
91         //OKが押されたときに、制限時間を設定  
92         int m = setTimePanel.getMinute();  
93         int s = setTimePanel.getSecond();  
94         timerLabel.setMax(60 * m + s);  
95     } else {  
96         setTimePanel.setDefault();  
97     }  
98 }
```

## 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

