プログラミング言語IIIB(Java) テーマ17

イベント処理

「(画面上の)ボタンが押された」「テキストが入力された」「マウスが動いた」「マウスのボタンが押された」「キーが押された」等, ユーザーが何らかの操作を行ったタイミングで, それに反応する(=コードを動作させる)ための仕組みがイベント処理機構である.

Java Swingでは, 発生した事象をイベント(Event), そのイベントを発生させたGUI部品をイベントソース(Event Source), そのイベントを処理するインスタンスをイベントリスナー(Event Listener)と呼ぶ. 予めイベントソースに対してイベントリスナーを登録しておくことによって, イベントが発生したタイミングで, イベントリスナーを呼び出してもらう事が可能となる.

図: イベントが発生した時に, そのイベントが, イベントリスナーに伝えられる(=呼び出される).

例えば, テーマ13で初めてGUIで作成したボタンのプログラムでは, ボタンがイベントソースでthisがイベントリスナーとなっている. ボタンに対して予め「このボタンが押されたらthisが処理するので呼び出してください」と登録しておくことによって, 画面上のボタンが押された時に, thisのactionPerformedが呼び出される.

※thisではない別のインスタンスをイベントリスナーにすることも可能.

```
public class EventExample extends JFrame implements ActionListener
                                          ~~~~~~~~~~~~~~~~~~~~~~~~~~ここが大事
{
    private JButton b1;

    public EventExample()
    {
        this.b1 = new JButton("緊急停止ボタン");
        this.b1.addActionListener(this);
        // ↑イベントソースthis.b1のイベントリスナーにthisを登録する.
        super.getContentPane().add(this.b1);
    }

    public void actionPerformed(ActionEvent e) // 引数のeにはイベントに関する情報が入っている.
    {
        if (e.getSource() == this.b1) // もしイベントソースがthis.b1だったら,
        {
            System.exit(0); // this.b1に応じた処理を行う. (この例では実行終了)
        }
    }
```

上記プログラムで, イベント反応3点セットがそろっていることに注目. どれ一つ欠かす事ができない.

```
implements ActionListener
this.b1.addActionListener(this)
public void actionPerformed(ActionEvent e)
```

イベントにはActionEventの他にも様々なものがあり, それぞれ対応するイベントリスナーのインターフェースの名前と, 呼び出されるメソッドの名前が決まっている. 以下に主なイベントを挙げる. スペルや大文字・小文字の違いには良く注意すること. ※特にメソッド名は先頭の小文字と途中の大文字に注意.

操作例	イベント型	リスナー名	メソッド名
画面上のボタンが押された, テキストが入力された	ActionEvent	ActionListener	actionPerformed
マウスが動いた, マウスボタンが押された	MouseEvent	MouseListener	mouseClicked等
キーが押された, キーが離された	KeyEvent	KeyListener	keyPressed等
スライダーが動いた	ChangeEvent	ChangeListener	stateChanged
ウィンドウが消された, ウィンドウが最小化された	WindowEvent	WindowListener	windowClosed等

なお, 一つのインスタンスが複数の種類のイベントリスナーになることも可能である. 以下にActionEventとChangeEventの両方を処理するクラスの例を示す.

```
public class EventExample extends JFrame
                        implements ActionListener, ChangeListener
                        // 複数のリスナー名をカンマで区切って並べる.
{
    // イベント処理メソッドを複数書く.
    public void actionPerformed(ActionEvent e)
    {
```

```
        // ボタン操作によるActionEventを処理するコード
    }
    public void stateChanged(ChangeEvent e)
    {
        // スライダー操作によるChangeEventを処理するコード
    }
}
```

JTextFieldクラス

1行の文字列入力欄であり,「入力欄の大きさ(文字単位)」を引数に指定して, newで生成する. ユーザーに短いテキストデータを入力させる場合に用いる. 文字列を入れてEnterキーを押すと, イベントリスナーに「1行分のテキストデータが入力された」という意味のActionEventが送られる. また, 入力用途のみではなく, JLabelの様に処理結果などの文字列表示にも利用できる.

```
// フィールドで
private JTextField tf1;
// コンストラクタで
this.tf1 = new JTextField(26); // 引数は欄の大きさ(文字数)
// actionPerformedで
if (e.getSource() == this.tf1) {
    String s = this.tf1.getText(); // 欄からテキストデータを取り出す. セットも可能.
    sを何に使うかはプログラムの目的次第
```

JTextAreaクラス

複数行に渡る長い文字列を表示/入力/編集できるGUI部品である.「縦と横の大きさ(文字単位)」を指定して, newで生成し, ウィンドウ上に配置すると, 自由にカーソル移動や文字入力のできる領域ができあがる.

```
// フィールドで
private JTextArea ta1;
// コンストラクタで
this.ta1 = new JTextArea(26, 52); // この場合は縦26行, 横52文字の入力欄ができる.
// actionPerformedで
String s = this.ta1.getText(); // 欄からテキストデータを取り出す. セットも可能.

this.ta1.append("追加する文字列"); // appendメソッドで追加も可能.
```

課題

以下の課題のレポートは, レポートファイルreport17.txtを作成してアップロードにより提出すること. レポートファイルの1行目には出席番号・名前・回を忘れずに記入すること.

1. JTextFieldに1行入力してEnterを押すと, 入力された文字列がJTextAreaに追加されていくGUIプログラムを作成せよ. (ソースをレポート)

   ヒント: まずJTextFieldとJTextAreaをウィンドウ内に配置するだけの張り子(見た目だけ)を作ると良い. その後, イベント処理を書き足していく.

   ```
   イベントソースがJTextFieldだったときに, ※つまりEnterキーが押されたときに
     JTextField から getText() で文字列を取り出し,
     その文字列を JTextArea に append() で追加する.
     改行のため System.getProperty("line.separator") も追加する.
     JTextField には空文字列 "" を setText() して入力欄をクリアする.
   ```

   ※以下の課題では, 自分のソースファイルをファイルの読み書きの動作テストに使わない方が安全である. 特に書き込みは大変危険である. 自分のソースファイルが消えて悲しいことにならない様に気を付けましょう.

   ---

2. ファイル名を入力する欄(JTextField), ファイルを開くボタン(JButton), ファイルの内容を表示するテキストエリア(JTextArea)から成るGUIプログラムを作成せよ. (ソースをレポート)

   ヒント: まずJTextField, JButton, JTextAreaをウィンドウ内に配置するだけの張り子を作ると良い. その後, イベント処理を書き足していく.

   ```
   JButton が押されたときに,
     JTextField から getText() でファイル名を取り出し,
     そのファイルを開き,
     ファイルの内容を全て読み込んで長い文字列を作成し,
     その長い文字列を JTextArea に setText() でセットする.
   ```

3. 課題2のプログラムに, さらに保存ボタンを追加し, ファイルを読み込んで編集し, 再びファイルに書き出せるプログラム(テキストエディタ)を作成せよ. (保存機能に関する部分のソースをレポート)

   注意: 課題2とは分けて記入すること. 課題2はファイルの読み込みのみができるエディタに関して, 課題3はそれに追加した保存機能に関してのみ記入である.

   ヒント: まず張り子の保存ボタンを追加し, その後, イベント処理を書き足していく.

   ```
   保存の JButton が押されたときに,
     JTextField から getText() でファイル名を取り出し,
   ```

```
さらに JTextArea から getText() で長い文字列を取り出し，
ファイルに長い文字列を書き込む．
```

4. 課題3のプログラムに, さらにステータスバー(JLabel)やスクロールバー(JScrollPane), ファイル選択ダイアログ(JFileChooser), ファイルの新規作成, 検索, 置換等, 締め切りまでに可能な範囲で自由に機能を追加してみよ. 一般的なテキストエディタにはない独創的なアイデアも良い. (ソース, 追加機能の解説, 将来的にさらに追加してみたい機能をレポート)

   注意: 課題3とは分けて記入すること. 課題3はファイルの読み書きのみができるエディタに関して, 課題4はそれ以上に追加した機能に関してのみ記入である.

アドバイス: 時間に余裕が無ければ, スクロールバーが最も簡単なので, それで妥協して良い.

レポート

- 内容: 課題中に指示されている通り. 必要な項目を全て記載しているか, 十分に確認すること.