情報処理技法
(Javaプログラミング)2

第11回
操作に対して処理が行われるGUI(3)

人間科学科コミュニケーション専攻
白銀 純子

第11回の内容

- キーボードのキーを押したときの処理
- 項目を選択したときの処理
- マウス操作の処理
- ポップアップメニュー
- 簡易メッセージ・入力欄表示

前回の復習問題の解答

- GUIにおいて、ボタンが押されたときに何らかの処理がされるとき、人間がボタンを押すときから処理が開始されるまでの仕組みについて、簡単に説明しなさい。

解答例:
「リスナ」という機能が、人間がボタンを押すのを待っている。人間がボタンを押すと、リスナがそれを感知し、マウスでのボタンクリックなど、どのようなユーザイベントが発生したかを特定する。そして、そのユーザイベントに対応した処理を開始する。

複数のリスナの宣言

複数のリスナを使うときは?

- 1つのウィンドウ内で、複数種類のユーザイベントが起こるとき…
  - JComboBoxで項目を選択すると、処理をする
    - 1つ目のJComboBoxで都道府県を選択すると、2つ目のJComboBoxに市区町村が設定される, etc.
  - JTextField上でマウスを右クリックすると、処理をする
    - ポップアップメニューを表示し、コピーや貼り付けをする, etc.
  - JButtonを右クリックすると、処理をする
    - OKボタンを右クリックすると、ヘルプメニューを表示する, etc.
  - etc.

複数のリスナの宣言が必要

複数のリスナの宣言(1)

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

public class クラス名 extends JFrame implements リスナ名1, リスナ名2, ... {
   GUI部品の変数宣言
   public クラス名() { /* コンストラクタ */
      . . . . . .
      イベントが発生する部品の変数名.addリスナの名前(this);
      . . . . . .
   }
   public void リスナ1のメソッド名(イベント名 e) { /* リスナ1で決められたメソッド */
   }
   public void リスナ2のメソッド名(イベント名 e) { /* リスナ2で決められたメソッド */
   }
   public static void main(String[] args) {
      new クラス名();
   }
}
```

複数のリスナの宣言(2)

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

public class クラス名 extends JFrame implements リスナ名1, リスナ名2, ... {
    GUI部品の変数宣言
    public クラス名() { /* コンストラクタ */
        ...
    }
    public void リスナ1のメソッド名(イベント名 e) { /* リスナ1で決められたメソッド */
    }
    public void リスナ2のメソッド名(イベント名 e) { /* リスナ2で決められたメソッド */
    }
    public static void main(String[] args) {
        new クラス名();
    }
}
```

- 利用するリスナの名前を「,」でつなげて宣言
- リスナの順序は何でもOK

複数のリスナの宣言(3)

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

public class クラス名 extends JFrame implements リスナ名1, リスナ名2, ... {
    GUI部品の変数宣言
    public クラス名() { /* コンストラクタ */
        ...の名前(this);
    }
    public void リスナ1のメソッド名(イベント名 e) { /* リスナ1で決められたメソッド */
    }
    public void リスナ2のメソッド名(イベント名 e) { /* リスナ2で決められたメソッド */
    }
    public static void main(String[] args) {
        new クラス名();
    }
}
```

- 各リスナで定義されているメソッドをオーバーライドして処理を記述
- 宣言したリスナ全てのメソッドが必要

よく使うリスナ

KeyListener

KeyListener(1)

- キーボードのキーを押したときのリスナ
  - 分類されているパッケージ: java.awt.event
- オーバーライドするメソッド名: keyTyped, keyPressed, keyReleased
  - keyTyped: キーがポンと押されたときの処理を書くメソッド
  - keyPressed: キーをぐっと押しっぱなしにしたときの処理を書くメソッド
  - keyReleased: キーを押してはなしたときの処理を書くメソッド
    - 3つ全てオーバーライドすることが必要
    - 使わないメソッドの内容は空でOK
- オーバーライドするメソッドの引数: KeyEvent
  - この引数の「getKeyChar()」メソッドを使って、どのキーが押されたかを取得
  - 押されたキーが何であるかによって、if文で処理を書き分け

KeyListener(2)

- getKeyChar: 押されたキーがどれかをchar型で返すメソッド
- char型
  - 1文字だけを表現するデータ型
  - 変数でない文字は「'」で囲む
    - String型と違い、「"」でないので注意! (「"」はString型、「'」はchar型)

KeyListener(3)

- KeyListenerの使い方例

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

public class KeySample extends JFrame implements ActionListener, KeyListener {
  JTextField addressText;

  public KeySample() {
    ......
    addressText = new JTextField();
    addressText.addKeyListener(this);
    ......
  }
```

JTextFieldにKeyListenerを登録
- JTextField以外の部品(JButtonなど)でも登録可能

KeyListener(4)

- KeyListenerの使い方例(続き)

```
public void keyTyped(KeyEvent e) {
  if (e.getKeyChar() == 'a') {
    /*「a」が押されたときの処理 */

  } else if (e.getKeyChar() == 'A') {
    /*「A」が押されたときの処理 */

  } else if (e.getKeyChar() == '¥n') {
    /*「Enter (Return)」が押されたときの処理 */

  } else if (...) {

  }
}
```

keyTypedメソッドの定義
- どのキーが押されたか、if文で条件分岐
- if文の条件は、「'」で文字を囲んで、getKeyCharの戻り値と比較

使わないメソッドは内容が空でOK
- ただし、書いておくことは文法上必要

```
public void keyPressed(KeyEvent e) {
}
public void keyReleased(KeyEvent e) {
}
public static void main(String[] args) {
  new KeySample();
}
}
```


KeyListener(5)

- KeyListenerのよくある使い方: Enter (Return)キーを押したときに、OKボタンを押したときと同じ処理をする
  - 処理の内容をメソッドとして定義しておく
  - 定義したメソッドを、Enter (Return)キーを押したときとOKボタンを押したときの処理として記述する

```
public void keyTyped(KeyEvent e) {
  if (e.getKeyChar() == '¥n') {  /* 「Enter (Return)」が押されたときの処理 */
    process();
  }
}
public void actionPerformed(ActionEvent e) {
  if (e.getSource() == okBut) {  /* OKボタンが押されたときの処理 */
    process();
  }
}
public void process(...) {
  ......
}
```


ItemListener

ItemListener(1)

- JComboBoxで項目が選択されたときのリスナ
  - 分類されているパッケージ: java.awt.event
- オーバーライドするメソッド: itemStateChanged
- オーバーライドするメソッドの引数: ItemEvent

ItemListener(2)

- ItemListenerのよくある使い方: JComboBoxで大きいカテゴリを選択すると、小さいカテゴリの選択項目を設定する
  - 1つ目のJComboBoxで都道府県を選択すると、2つ目のJComboBoxに市区町村の選択項目が設定される, etc.

- 処理内容
  - JComboBoxの「removeAllItems()」メソッドで、現在の登録をすべて消去する
    - addItem(...)メソッドは追加で項目を登録するだけのため
  - addItem(...)メソッドで新しい項目を登録する

ItemListener(3)

- ItemListenerの使い方例

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

public class ComboSample extends JFrame implements ActionListener, ItemListener {
  JComboBox prefCombo, cityCombo;

  public ComboSample() {
    ......
    prefCombo = new JComboBox();
    prefCombo.addItem("都道府県を選択してください。");
    prefCombo.addItem("東京都");
    prefCombo.addItem("神奈川県");
    ......
    prefCombo.addItemListener(this);

    cityCombo = new JComboBox<String>();
    ......
  }
```

JComboBoxにItemListenerを登録
- ItemListenerはJComboBoxにのみ登録可能

ItemListener(4)

- ItemListenerの使い方例(続き)

```
public void itemStateChanged(ItemEvent e) {
  String value = (String) prefCombo.getSelectedItem();

  if (value.equals("東京都")){
    cityCombo.removeAllItems();
    cityCombo.addItem("杉並区");
    cityCombo.addItem("武蔵野市");
    ....
  } else if (value.equals("神奈川県")){
    cityCombo.removeAllItems();
    cityCombo.addItem("横浜市");
    cityCombo.addItem("川崎市");
    ......
  }
}
public static void main(String[] args) {
  new ComboSample();
}
}
```

大きいカテゴリのJComboBoxで
選択された項目が何であるかで条件分岐

現在のJComboBoxの登録内容を消去

小さいカテゴリの選択項目を新たに登録

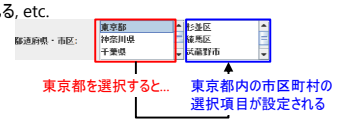
ListSelectionListener

ListSelectionListener(1)

- JListで項目が選択されたときのリスナ
- オーバーライドするメソッド: valueChanged
- オーバーライドするメソッドの引数: ListSelectionEvent

ListSelectionListener(2)

- ListSelectionListenerのよくある使い方: JListで大きいカテゴリを選択すると、小さいカテゴリの選択項目を設定する
  - 1つ目のJListで都道府県を選択すると、2つ目のJListに市区町村の選択項目が設定される, etc.



- 処理内容
  - setListData(...)メソッドで新しい項目を登録する
    - setListData(...)メソッド: JListに項目を設定するためのメソッド(引数はString型の配列)
    - JComboBoxのような登録内容の削除は不要

ListSelectionListener(3)

- ListSelectionListenerの使い方例

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

public class ListSample extends JFrame implements ListSelectionListener {
  JList prefList, cityList;

  public ListSample() {
    ......
    String[] prefs = {"東京都", "神奈川県", "千葉県", "埼玉県", "群馬県"};
    prefList = new JList<String>(prefs);
    prefList.addListSelectionListener(this);
    ......
  }
```

JListにListSelectionListenerを登録
- ListSelectionListenerはJListにのみ登録可能

ListSelectionListener(4)

- ListSelectionListenerの使い方例(続き)

```
public void valueChanged(ListSelectionEvent e) {
    String value = (String) prefList.getSelectedValue();
    if (value.equals("東京都")) {
      String[] cities = {"杉並区", "練馬区", "武蔵野市", "三鷹市"};
      cityList.setListData(cities);
    } else if (value.equals("神奈川県")) {
      String[] cities = {"横浜市", "川崎市", "横須賀市", "鎌倉市"};
      cityList.setListData(cities);
    }
  }
    public static void main(String[] args) {
      new ListSample();
    }
}
```

大きいカテゴリのJListで選択された項目が何であるかで条件分岐

小さいカテゴリの選択項目を新たに登録

MouseListener

MouseListener(1)

- マウスを操作したときのリスナ
  - 分類されているパッケージ: java.awt.event
- オーバーライドするメソッド名: mouseClicked, mouseEntered, mouseExited, mousePressed, mouseReleased
  - mouseClicked: マウスのボタンがポンと押されたときの処理を書くメソッド
  - mouseEntered: マウスカーソルが部品の上に来たときの処理を書くメソッド
  - mouseExited: マウスカーソルが部品の上から出て行ったときの処理を書くメソッド
  - mousePressed: マウスのボタンをぐっと押しっぱなしにしたときの処理を書くメソッド
  - mouseReleased: マウスのボタンを押してはなしたときの処理を書くメソッド
    - 5つ全てオーバーライドすることが必要
    - mouseClicked, mousePressed, mouseReleasedは3つのボタンのどれを押しても反応
    - 使わないメソッドの内容は空でOK

MouseListener(2)

- オーバーライドするメソッドの引数: MouseEvent
  - この引数の「getButton()」メソッドを使って、どのボタンが押されたかを取得
  - この引数の「getClickCount()」メソッドを使って、ボタンが何回押されたかを取得
    - ダブルクリックの判定などに利用
  - 押されたボタンがどれであるか、何回押されたかによって、if文で処理を書き分け

マウスのボタンとgetButtonメソッドの戻り値との対応

マウスのボタン	getButtonメソッドの戻り値
左ボタン	MouseEvent.BUTTON1
ホイールボタン	MouseEvent.BUTTON2
右ボタン	MouseEvent.BUTTON3

MouseListener(3)

- MouseListenerの使い方例

```
JTextField addressText;

  public MouseSample() {
    ......
    addressText = new JTextField();
    addressText.addMouseListener(this);
    ......
  }
```

JTextFieldにMouseListenerを登録
- JTextField以外の部品(JButtonなど)でも登録可能

MouseListener(4)

- MouseListenerの使い方例(続き)

mouseClickedとmousePressedメソッドの定義
- どのボタンが押されたか、if文で条件分岐

```
public void mouseClicked(MouseEvent e) {
    if ((e.getButton() == MouseEvent.BUTTON1) && (e.getClickCount() == 2)) {
      /* マウスの左ボタンが2回押された(ダブルクリック)ときの処理 */
    }
}
public void mousePressed(MouseEvent e) {
    if (e.getButton() == MouseEvent.BUTTON3) {
      /* マウスの右ボタンが押されたときの処理 */
    }
}
public void mouseReleased(MouseEvent e) {
}
public void mouseEntered(MouseEvent e) {
}
public void mouseExited(MouseEvent e) {
}
```

使わないメソッドは内容が空でOK
- ただし、書いておくことは文法上必要

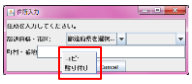
MouseListener(5)

- MouseListenerのよくある使い方
  - マウスの右ボタンを押したときの処理を書く
    - JTextFieldやJTextAreaでポップアップメニューを出してコピー&ペーストの処理, etc.
  - マウスの左ボタンをダブルクリックしたときの処理を書く
    - JListで項目をダブルクリックしたときに、項目の詳細情報のウィンドウを表示する処理, etc.

ポップアップメニュー

ポップアップメニューの作成(1)

- ポップアップメニュー: その場その場で表示するメニュー
  - 多くの場合、マウスの右クリックで表示
  - Javaでの部品名: JPopupMenu



- ポップアップメニューの作成と表示
  - JPopupMenuのオブジェクトを作成
  - JPopupMenuのオブジェクトにJMenuItemのオブジェクトを登録
    - JMenuItemの扱い方は、メニューバーを作るときと全く同じ
  - JPopupMenuのオブジェクトを「show(...)」メソッドで画面上に表示
    - どの部品上のどの座標に表示するかを、showメソッドの引数(表示する部品, x座標, y座標の順で)として指定

ポップアップメニューの作成(2)

- マウス操作に応じて表示する場合
  - MouseListenerのメソッドのMouseEvent引数のメソッドで、必要な情報を取得
    - getComponent()メソッド: どの部品上でマウス操作が行われたかを取得
    - getX()メソッド: マウス操作が行われた場所のx座標
    - getY()メソッド: マウス操作が行われた場所のy座標

この3つの情報を、JPopupMenuのオブジェクトを表示するときに使用

ポップアップメニューの作成(3)

- マウスの右クリックでポップアップメニューを表示する例

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

public class MouseSample extends JFrame implements ActionListener, MouseListener {
  JPopupMenu popup;
  JMenuItem copyItem, pasteItem;

  public MouseSample(){
    ……
  }
```

部品のオブジェクトの作成や表示はリスナのメソッド内で行うが、変数宣言はクラスのフィールドとして行う必要

ポップアップメニューの作成(4)

- マウスの右クリックでポップアップメニューを表示する例(続き)

```
public void mousePressed(MouseEvent e) {
  if (e.getButton() == MouseEvent.BUTTON3) {
    popup = new JPopupMenu();

    copyItem = new JMenuItem("コピー");
    copyItem.addActionListener(this);
    popup.add(copyItem);

    pasteItem = new JMenuItem("貼り付け");
    pasteItem.addActionListener(this);
    popup.add(pasteItem);

    popup.show(e.getComponent(), e.getX(), e.getY());
  }
}
```

マウスの右ボタンを押されたときの処理として定義

JPopupMenuの作成
- JMenuItemはオブジェクトを作成して、JPopupMenuのオブジェクトに登録

JPopupMenuオブジェクトの表示
- 「e.getComponent()」でマウス操作が行われた部品を取得
  - ✓ ポップアップメニューの表示先
- 「e.getX()」と「e.getX()」でマウス操作が行われたx座標・y座標を取得
  - ✓ ポップアップメニューを表示するx座標・y座標

ポップアップメニューの作成(5)

- マウスの右クリックでポップアップメニューを表示する例(続き)

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == copyItem) {
        /* ポップアップメニューのcopyItemが押されたときの処理 */
    } else if (e.getSource() == pasteItem) {
        /* ポップアップメニューのpasteItemが押されたときの処理 */
    } else if (e.getSource() == okBut) {
        /* OKボタンが押されたときの処理 */
    }
  }
}
```

ポップアップメニューの項目が選択されたときの処理
- ウィンドウ上の他の部品(JButtonなど)とあわせて、if文で条件分岐

ポップアップメニューの作成(6)

- ちなみに...JTextFieldとJTextAreaで切り取り(カット)・コピー・貼り付け(ペースト)を行うには...
  - cut()メソッド: 選択された文字列を切り取り(カット)するメソッド
  - copy()メソッド: 選択された文字列をコピーするメソッド
  - paste()メソッド: 選択された文字列を貼り付け(ペースト)するメソッド

ポップアップメニューの作成(7)

- コピー&ペースト処理の記述例

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

public class MouseSample extends JFrame implements ActionListener, MouseListener {
    JPopupMenu popup;
    JMenuItem copyItem, pasteItem;
    JTextField addressText;

    ......

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == copyItem) { /* ポップアップメニューのcopyItemが押されたときの処理 */
            addressText.copy();
        } else if (e.getSource() == pasteItem) { /* ポップアップメニューのpasteItemが押されたときの処理 */
            addressText.paste();
        }
    }
}
```


簡易メッセージ・入力欄の表示

簡易メッセージ・入力欄

- ちょっとしたメッセージや入力をしたい!
  - 確認メッセージ
  - 警告メッセージ
  - エラーメッセージ
  - 1つだけ入力
  - etc.

いちいちウィンドウを作るのは面倒!

JOptionPane

JOptionPane(概要)

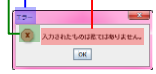
- 1行だけのメッセージや入力欄を簡易表示するためのウィンドウ
  - OK/CancelボタンやYes/Noボタンつき
  - 表示内容によって異なるメソッド利用
  - メソッドにメッセージ内容などを引数として設定
  - JOptionPaneが表示されているときは、他のウィンドウの操作不可

JOptionPane(showMessageDialog)

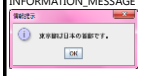
- 1行のメッセージを表示
  - OK/CancelやYes/Noなどの利用者の判断を求めないメッセージ(戻り値なし)
- 引数は4つ
  - 1つ目の引数: 常に「null」(ウィンドウの表示先を表すが、通常はnullにしておく)
  - 2つ目の引数: 表示するメッセージ(String型)
  - 3つ目の引数: ウィンドウのタイトル(String型)
  - 4つ目の引数: メッセージのタイプ(タイプによって表示されるアイコンが違う)
    - JOptionPane. ERROR_MESSAGE: エラーメッセージ
    - JOptionPane.INFORMATION_MESSAGE: 情報を提示
    - JOptionPane.WARNING_MESSAGE: ワーニングメッセージ
    - JOptionPane.QUESTION_MESSAGE: 質問メッセージ
    - JOptionPane.PLAIN_MESSAGE: アイコンなしでメッセージを表示

JOptionPane(showMessageDialog)(例)

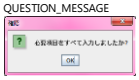
```
JOptionPane.showMessageDialog(null, "入力されたものは数ではありません。", "エラー", JOptionPane.ERROR_MESSAGE);
```



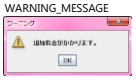
INFORMATION_MESSAGE



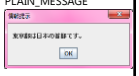
QUESTION_MESSAGE



WARNING_MESSAGE



PLAIN_MESSAGE



JOptionPane(showConfirmDialog)(1)

- 1行のメッセージを表示
  - OK/CancelやYes/Noなどの利用者の判断を求めるメッセージ
- 戻り値(int型)によって、どのボタンが押されたかを取得
  - JOptionPane.OK_OPTION: OKボタンが押されたときの戻り値
  - JOptionPane.CANCEL_OPTION: Cancelボタンが押されたときの戻り値
  - JOptionPane.YES_OPTION: Yesボタンが押されたときの戻り値
  - JOptionPane.NO_OPTION: Noボタンが押されたときの戻り値

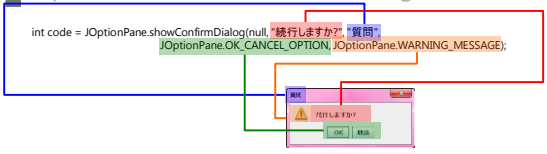
int型の変数を用意して戻り値を受け取り、if文で条件分岐して処理

JOptionPane(showConfirmDialog)(2)

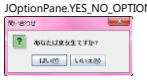
- 引数は5つ
  - 1つ目の引数: 常に「null」(ウィンドウの表示先を表すが、通常はnullにしておく)
  - 2つ目の引数: 表示するメッセージ(String型)
  - 3つ目の引数: ウィンドウのタイトル(String型)
  - 4つ目の引数: OK/CancelやYes/Noのボタンのタイプ
    - JOptionPane.OK_CANCEL_OPTION: OK/Cancelボタンを表示
    - JOptionPane.YES_NO_OPTIOn: Yes/Noボタンを表示
    - JOptionPane.YES_NO_CANCEL_OPTION: Yes/No/Cancelボタンを表示
  - 5つ目の引数: メッセージのタイプ(タイプによって表示されるアイコンが違う)
    - 種類はshowMessageDialogメソッドと同じ

JOptionPane(showConfirmDialog)(例)

```
int code = JOptionPane.showConfirmDialog(null, "続行しますか?", "質問",
                                         JOptionPane.OK_CANCEL_OPTION, JOptionPane.WARNING_MESSAGE);
```



JOptionPane.YES_NO_OPTION



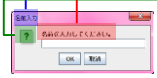
JOptionPane.YES_NO_CANCEL_OPTION

JOptionPane(showInputDialog)

- 1行の入力欄を表示
- 戻り値は入力された文字列(String型)
  - String型の変数を用意して戻り値を受け取り
- 引数は4つ
  - 1つ目の引数: 常に「null」(ウィンドウの表示先を表すが、通常はnullにしておく)
  - 2つ目の引数: 表示するメッセージ(String型)
  - 3つ目の引数: ウィンドウのタイトル(String型)
  - 4つ目の引数: メッセージのタイプ(タイプによって表示されるアイコンが違う)
    - 種類はshowMessageDialogメソッドと同じ

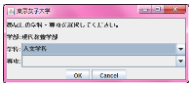
JOptionPane(showInputDialog)(例)

```
String name =
        JOptionPane.showInputDialog(null, "名前を入力してください。", "名前入力", JOptionPane.QUESTION_MESSAGE);
```



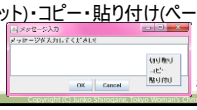
やってみよう!(1)

1. 下記のウィンドウで、学科を選択すると専攻の選択項目が設定されるプログラム



※OK・Cancelボタンの処理は何もなくて良い

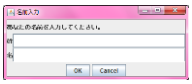
2. 1. のウィンドウのJComboBoxをJListに変更したプログラム
3. 下記のウィンドウで、JTextArea上でマウスの右クリックをすると、「切り取り」、「コピー」、「貼り付け」のポップアップメニューが表示されて切り取り(カット)・コピー・貼り付け(ペースト)の処理ができるプログラム



※OK・Cancelボタンの処理は何もなくて良い

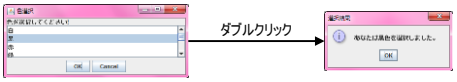
やってみよう!(2)

- 名前を入力するウィンドウで、下記のことができるプログラム
  - 「OK」ボタンを押したとき、「あなたの名前はXXYYですね!」と標準出力で出力する
    - 「XX」は「姓」の欄、「YY」は「名」の欄に入力された文字列
  - 「姓」の欄、「名」の欄でEnter (Return)キーを押したときに、「OK」ボタンを押したときと同じ処理をする



やってみよう!(3)

- 下記のように、色の一覧(JList)上で色名をダブルクリックすると、どの色を選択したかのメッセージをJOptionPaneで表示するプログラム



※OK・Cancelボタンの処理は何もなくて良い

補講と期末試験のお知らせ

- 補講: 1月20日(水) 5限 9301教室
- 期末試験: 1月26日(火) 3限 9301教室
  - 内容: 後期の講義内容すべて
    - 用語の意味の選択・説明
    - 概念に関する説明
    - 実技