プログラミング言語IIIB(Java) テーマ15

JCheckBoxクラス

選択されて「いる」か「いない」かの二つの状態を持つボタンで, isSelectedメソッドでその状態を調べることができる. 以下に使用例を示す.

```
// ---- フィールドで ----
private JCheckBox cb1, cb2; // チェックボックス2つ
private JButton judge;      // 判定ボタン

// ---- コンストラクタで ----
this.cb1 = new JCheckBox("数学が得意");
this.cb2 = new JCheckBox("英語が得意", true); // trueを付けるとデフォルトで選択状態にできる.
this.judge = new JButton("判定");

// 部品の配置方法は→テーマ14
// 判定ボタンに反応させるには→テーマ13

// ---- actionPerformedメソッドで ----
if (e.getSource() == this.judge) { // いま押されたボタンが判定ボタンだったら
    if (this.cb1.isSelected() && !(this.cb2.isSelected())) { // 数学チェックボックスが選択されている状態で
                                                             // かつ, 英語が選択されていない状態だったら
        System.out.println("いいですね! でも英語も重要ですよ!");
    }
}
```

チェックボックスは通常, 他の判定等のボタンが押された時に, チェックボックスの選択状態を調べて用いるため, いま押されたボタンがチェックボックスと等しいかどうかは判定しない.

しかし, 自分のやりたいアイデア実現のために, 仕組みを理解した上で敢えて行っても良い. タッチ操作のデバイス等, 最近ではチェックボックス操作に直ちに反応するUIも増えてきている.

```
// ---- チェック操作に直ちに反応させる方法 ----
if (e.getSource() == this.cb1) { // いま押されたボタンが数学チェックボックスだったら
    ...
}
```

JRadioButtonクラス

JCheckBoxクラスと同じく, 選択されて「いる」か「いない」かの二つの状態を持つボタンである.

ただし, ラジオボタンは複数のボタンをグループ(ButtonGroup)にまとめ, そのグループの中からは同時にいずれか一つだけ選択できる様に設定して用いる. 「ラジオボタン」の名称は, 大昔のラジオやテレビのチャンネル選択ボタンに由来している. 以下に使用例を示す.

```
// ---- フィールドで ----
private JRadioButton rb1, rb2, rb3;
private JButton kettei;

// ---- コンストラクタで ----
this.rb1 = new JRadioButton("就職");
this.rb2 = new JRadioButton("専攻科進学");
this.rb3 = new JRadioButton("大学編入学", true);

ButtonGroup group = new ButtonGroup(); // ボタングループを生成して(ローカル変数で良い)
group.add(this.rb1);                   // そのグループにボタンを登録しておく.
group.add(this.rb2);                   // これにより, グループのうち1つだけが選択可能になる.
group.add(this.rb3);

// なお, group.add(...)とpanel.add(...)は ****それぞれ別に**** 行う必要があるので注意.

//  【大切なのでもう一度】
// パネルへの追加のためのpanel.add(...)も必要!!

this.kettei = new JButton("進路決定");
// 部品の配置方法は→テーマ14
// 進路決定ボタンに反応させるには→テーマ13

// ---- actionPerformedメソッドで ----
if (e.getSource() == this.kettei) { // いま押されたボタンが進路決定ボタンだったら
    if (this.rb1.isSelected()) {    // 就職ラジオボタンが選択されている状態だったら
        System.out.println("実習系科目には特にしっかり取り組みましょう!");
    }
}
```

ラジオボタンも, 通常は他の決定等のボタンが押された時に, ラジオボタンの選択状態を調べて用いる.

注意: Java(Swing)では, ButtonGroupはあくまで意味的なグループであり, 画面上の配置のためのパネルへの追加とは無関係である. ラジオボタンを画面に配置するには, ButtonGroupへのaddとは別に, 通常通りパネルへのaddが必要となる. ※世の中にはWindows Forms(Windows上でのC#によるGUI開発)の様に, 画面上の配置と意味的なグループが一致する開発環境も存在する.

おまけ: アイコン画像付きボタンの作り方

```
// 画像ファイルを用意しておいて
ImageIcon icon1 = new ImageIcon("icon1.png"); // 画像ファイルを読み込み
this.button1 = new JButton("お気に入りに登録", icon1);
                                              ~~~~~ ボタンをnewするときに与える
// これだけなので, ぜひ使ってみましょう.
```

課題

以下の課題のレポートは, レポートファイルreport15.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. [練習] 上記のJCheckBoxの例を実際に動作させてみよ. (完成したソース, 仕組みについての考察をレポート)
   - 考察のヒント: チェックボックスの状態はどこに保存されていると考えられるか? さらにヒント: new演算子は何をしているか?
   - 部品を画面に配置する方法で悩む人は, テーマ14課題1のプログラムを流用すると良い.
   - 判定ボタンが押されたときに反応させる方法は, テーマ13のAppTest3を参考にすると良い. (イベント反応3点セットが必要)

```
implements ActionListener
this.judge.addActionListener(this);
public void actionPerformed(ActionEvent e) { }
```

2. [練習] 上記のJRadioButtonの例を実際に動作させてみよ. (完成したソース, 仕組みについての考察をレポート)
   - 考察のヒント: ButtonGroupはどの様な仕組みでラジオボタンが一つだけ選択される様に制御していると考えられるか? さらにヒント: ラジオボタンの状態, ゲッター, セッター

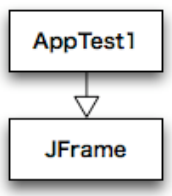
以下の課題3と課題4は, どちらに先に取り組んでも良い.

3. JCheckBoxとJRadioButtonを両方用いて, 自由にGUIプログラムを作成せよ. (ソース, 使用方法をレポート)
   - ヒント: JCheckBoxやJRadioButtonは, アンケート調査やゲームのコンフィグ画面の様に多数の選択肢をユーザーに提示し, ユーザーの選択状態に応じてその後の処理内容を変化させる画面に適している. ※しかし, 自分のやりたいアイデアがある場合は必ずしもこの通りにする必要はない.
   - 上級者向け: 選択肢に加えて, キーボード入力による自由記述欄も利用したい場合は, JTextFieldやJTextAreaを調べて試してみると良い. また, これらの部品は入力だけではなく処理結果の表示にも利用できる.
4. これまでのGUI課題で登場した部品(JFrame, JButton, JLabel, JPanel, JCheckBox, JRadioButton等)とその祖先について, 継承関係を調査し, UMLクラス図(概略版)を描いて提出せよ. 木構造の図が完成するはずである. ツールで作図して印刷しても良いが, 手描きで十分である. A4用紙, 表紙1枚付き, 左上綴じで作成すること.
   - ヒント: Java API仕様でJButton等のクラスを調べると, そのスーパークラス(親)が分かる. 先頭のjava.lang.やjava.awt.等は大量のクラスを分類整理するためのパッケージ名なので無視して良い.

```
---- JButtonのAPI仕様の例 ----
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.AbstractButton
          javax.swing.JButton

JButton のスーパークラスは AbstractButton
さらに AbstractButton のスーパークラスは JComponent ... という継承関係を表している.
```

   - 全てのクラスは祖先を辿(たど)ると最終的にObjectクラスへ行き着く. JButton以外の, JLabel等のクラスから先祖を辿っても, 先祖のどこかで合流するため, 全体としては木構造となる.
   - UMLクラス図では, 継承関係をサブクラス(子)からスーパークラス(親)へ向けて矢印を描いて表現する. 継承関係の矢印は白抜き三角形である. 三角形の中に線を貫通させたり, 三角形を黒く塗ってはいけない.



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.