- アクションイベントの処理（承前）
- マウスイベントの処理
- イベントの情報の取得

★9　イベント処理（承前）

★9.3　アクションイベントの処理

★9.3.1　CheckBox のアクションイベント

別のイベント処理を考えてみよう．この例では，チェックボックスのクリック時に発生するイベントの処理を行なう．p.204 の表を見ると，次のことがわかる．

- イベントオブジェクトのクラスは
- 対応するイベントハンドラプロパティは

これらの情報に基づいてプログラムを書くと，以下のようなものになる．このプログラムは，**CSS（カスケーディング・スタイルシート）**を使ってスタイルを指定する例にもなっている．後述の拡張子 .css のファイルがそのスタイル指定のためのファイルである（詳しくは p.186 参照）．

TrueUni3.java

```
   (import 文は省略)
 9  public class TrueUni3 extends Application {
10      @Override
11      public void start(Stage pstage) {
12          ImageView ivb = new ImageView(new Image("blackuni.jpg"));
13          ImageView ivw = new ImageView(new Image("whiteuni.jpg"));

14          Label label = new Label("", ivw);
15          CheckBox cb = new CheckBox("真のお姿");

16          HBox root = new HBox(label, cb);
17          root.getStyleClass().add("hbox");

18          Scene scene = new Scene(root);
19          scene.getStylesheets().add("trueUni3.css");

20          pstage.setScene(scene);
21          pstage.setTitle("TrueUni3");
22          pstage.sizeToScene();
23          pstage.show();

24          cb.setOnAction((event)-> {
25              if(cb.isSelected()) label.setGraphic(ivb);
26              else                label.setGraphic(ivw);
27          });
28      }
        (main メソッドは省略)
32  }
```

trueUni3.css（TrueUni3 と同じディレクトリに置いておく）

```
.hbox{
    -fx-spacing: 20;           /* 子ノード間隔 */
    -fx-padding: 15 10 15 15;  /* 余白 */
    -fx-alignment: center;     /* 子ノードの位置合わせ */
}
```

★ 9.3.2 TextField 上のアクションイベント (p.212)

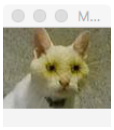
TextField というのは，1 行の文字列を表示したり入力したりできるコンポーネントである (p.198)．p.204 表 14.1 を見ると分かるように，TextField 上で Enter キーを押すと ActionEvent が発生する．以下は，そのイベント処理の例である．イベント発生時に TextField に入力されているものは text プロパティの値として取得できる．ただし，その値は文字列 (String) なので，整数値などを扱いたければ，このプログラムのように変換する必要がある．

TextFieldEvent.java

```
 : (import 文は省略)
 8 public class TextFieldEvent extends Application {
 9
10    @Override
11    public void start(Stage pstage) {
12       TextField tf = new TextField("なんか入力してね");
13       BorderPane root = new BorderPane();
14       root.setCenter(tf);
15       Scene scene = new Scene(root);
16       pstage.setScene(scene);
17       pstage.setTitle("TextFieldEvent");
18       pstage.sizeToScene();
19       pstage.show();
20
21       // tf でのイベント処理
22       tf.setOnAction((event->{
23          // TextField に入力された文字列は  Text プロパティの値として入手できる
24          System.out.println("テキストフィールドの文字列は  "+tf.getText());
25          // 整数値を受け取りたければ，次のように変換すればよい
26          int x = 0;
27          try {
28             // int のラッパークラス Integer のクラスメソッド parseInt で int 型に変換
29             x = Integer.parseInt(tf.getText());
30          } catch(NumberFormatException error) {
31             // 上記の変換がうまくできないときはこちらが実行される
32             x = 5963;
33          }
34          System.out.println("整数に変換した値は  "+x);
35       }));
36    }
 :    (main メソッドは省略)
41 }
```

★9.4 マウスイベントの処理 (p.215)

マウスカーソルの位置や操作に応じたイベント処理の仕方を学ぼう．アクションイベントの場合と同様に，p.204 の表 14.1 を見てみよう．マウスを動かしたりマウスボタンを押したりした時のイベントクラスは MouseEvent である．MouseEvent では,「マウスボタンをクリックした」,「マウスカーソルが領域に入った」といった動作ごとに**イベントタイプ**という分類がクラス変数として定義されている．表の 3 列目を見ると，上記の例の前者は MOUSE_CLICKED，後者は MOUSE_ENTERED に対応していることが分かる．イベントハンドラ用のプロパティはイベントタイプ毎に用意されており，前者なら onMouseClicked，後者なら onMouseEntered を使う．



マウスイベントを扱う最初の例として，上記のようなイベント処理を考えよう．これを実現するプログラムは以下のようになる．

MouseEventSample.java（教科書のとは別物）

```
   (import 文は省略)
 9  public class MouseEventSample extends Application {
10      @Override
11      public void start(Stage pstage) {
12          Image white = new Image("whiteuni.jpg");
13          Image black = new Image("blackuni.jpg");
14          ImageView iv = new ImageView(white);

15          Label lab1 = new Label("", iv);
16          Label lab2 = new Label();

17          BorderPane root = new BorderPane();
 :          (これまでと同様なので省略)

25          lab1.setOnMouseEntered((event) -> {
26                  iv.setImage(black);
27                  lab2.setText("やあ");
28              });

29          lab1.setOnMouseExited((event) -> {
30                  iv.setImage(white);
31                  lab2.setText("ばいばい");
32              });
33      }
 :      (main メソッドは省略)
37  }
```

★ 9.5 イベントの情報の取得

マウスイベントを扱う場合，マウスクリックされた位置やどのマウスボタンが押されたかに応じた処理を書きたいことがある．他のイベントについても同様に，イベントが発生した場所やイベントの状態（どのノードで発生したか，ウィンドウ内の位置，どのキーが押されたか，etc.）に応じた処理を書きたいことがある．イベントオブジェクトにはこれらの情報が格納されており，適切なメソッドを呼び出してそれらを取得することができる（p.208 表 14.2 参照 (☆ 1)）．

以下のプログラムは，前節のものをベースに，マウスクリックの位置に応じたイベント処理を追加したものである．

☆ 1) この表に含まれていないものもたくさんある．例えば，MouseEvent にはどのマウスボタンが押されたか調べるためのメソッドも用意されている．詳しく知りたい時は，使いたいイベントクラスの API を参照しよう．

MouseEventSample.java（改造版）

```
 7  import javafx.scene.input.*;
   （他の import 文は省略）
 9  public class MouseEventSample extends Application {
10      @Override
11      public void start(Stage pstage) {

 :          （start メソッドのここまでの内容は元と同じ）

34          lab1.setOnMouseClicked((event) -> {
35                  double w = lab1.getWidth();
36                  double h = lab1.getHeight();
37                  double x = event.getX();
38                  double y = event.getY();

39                  double d = (x-w/2)*(x-w/2) + (y-h/2)*(y-h/2);

40                  if(d < 20*20) lab2.setText("いたい");
41                  else          lab2.setText("");
42              });
43      }
 :      （main メソッドは省略）
47  }
```

教科書第 14 章の 14.4 節以降の内容はスキップします．