- 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
- グラフィックス

★9　イベント処理（承前）

★9.3　アクションイベントの処理（承前）

★9.3.1　TextField 上のアクションイベント (p.212)

前回の練習問題で作成した Counter クラスでは TextField を用いたが，イベント処理は Button に対するものだった．今回は，これを TextField のイベント処理を行うように改造し，TextField に数を直接入力できるようにしてみよう．

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．その時 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 (p.198)．ただし，その値は文字列 (String) なので，今の場合は int 型に変換しないといけない．

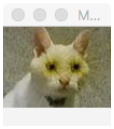
改造を施したプログラム（の一部）を以下に示す．

Counter.java（改造版）

```
   (import 文は省略)
 9  public class Counter extends Application {
10      int count = 0; // カウント用の変数
11      TextField tf;

12      @Override
13      public void start(Stage pstage) {
14          tf = new TextField(String.valueOf(count));

:           (この部分は前回のと同じなので省略)

35          tf.setOnAction((event)-> { procInput(); });
36      }

37      void procInput(){
38          int prev = count;
39          try{
40              count = Integer.parseInt(tf.getText());
41          } catch(NumberFormatException error){
42              count = prev;
43          }
44          tf.setText(String.valueOf(count));
45      }

 :      (main メソッドは省略)
49  }
```

★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．

★10 グラフィックス

画面上に自分で線を引いたり図形を描いたりする方法を学ぼう．

★10.1 描画の概要 (p.224)

JavaFX では，図形を描く方法が二通り提供されている．

Shape クラスのサブクラスを組み合わせて描画する方法 Shape クラスのサブクラスとして線分，円，矩形 (☆ 2) などの基本的な図形が用意されている（p.225 表 15.1）．これらのオブジェクトをシーングラフに追加することで描画する．個々のオブジェクトにイベント処理を設定することも可能．

☆ 2) 長方形のこと

Canvas クラスのオブジェクト上に描画する方法 Canvas クラスは，GraphicsContext という描画のためのクラスのオブジェクトをプロパティとして持っている．この GraphicsContext クラスのメソッドを使って図形を描画する．上記と違って個々の図形はオブジェクトではないく，個別にイベント処理を設定したりできない．そのかわり描画は高速である．

上記いずれの方法を利用する場合でも，図形をどの位置にどのような大きさで描くかを指定するためには座標系を定めないといけない．JavaFX では，Node クラスのオブジェクトは個別に**ローカル座標系**と呼ばれる座標系を持っている．左上角が原点，右向きに x 軸，下向きに y 軸をとった座標系で，座標の値は double 型で表される (☆ 3)．

☆ 3) p.225 15.1.2 節の説明と脚注の図参照．

★ 10.2 Shape クラス (p.226)

ここではプログラムの例を示すにとどめる．図形に対するイベント処理も行うものになっている．

ShapeSample.java

```
 1  import javafx.application.Application;
 2  import javafx.stage.*;          // Stage
 3  import javafx.scene.*;          // Scene
 4  import javafx.scene.layout.*;   // Pane, HBox
 5  import javafx.scene.shape.*;    // Rectangle, Circle
 6  import javafx.scene.paint.*;    // Color
 7  import javafx.event.*;          // MouseEvent
 8
 9  public class ShapeSample extends Application {
10
11      @Override
12      public void start(Stage pstage){
13
14          Pane left = new Pane();       // 描画用 Pane コンテナその 1
15          left.setPrefSize(200, 200);   // コンテナの大きさ
16
17          Pane right = new Pane();      // 描画用 Pane コンテナその 2
18          right.setPrefSize(200, 200);
19          right.setStyle("-fx-background-color: #87CEEB;"); // 背景色を設定
20
21          HBox root = new HBox(left, right); // ルートノード．左にその 1，右にその 2
22
23          drawRectangles(left); // 図形を描画（矩形）
24          drawCircles(right);   // 図形を描画（円）
25
26          Scene scene = new Scene(root);
27          pstage.setTitle("ShapeSample");
28          pstage.setScene(scene);
29          pstage.show();
30      }
```

(次頁へつづく)

ShapeSample.java（つづき）

```
31
32      // 変数 p が表す Pane 上に矩形を描画
33      void drawRectangles(Pane p) {
34
35          // 矩形その 1 （左上の x 座標，y 座標，幅，高さ）
36          Rectangle rect1 = new Rectangle(20, 30, 40, 60);
37          rect1.setStroke(Color.RED); // 線の色
38          rect1.setStrokeWidth(3);    // 線の太さ
39          rect1.setFill(Color.PINK);  // 図形内部の塗りつぶしの色
40
41          // 矩形その 2
42          Rectangle rect2 = new Rectangle(100, 100, 50, 50);
43          rect2.setFill(Color.RED);
44
45          // 2 つの Rectangle オブジェクトを p にのせる
46          p.getChildren().addAll(rect1, rect2);
47
48          // 矩形その 2 のイベントハンドラを設定（クリックごとに色が変わる）
49          rect2.setOnMouseClicked((event) -> {
50                  if(rect2.getFill() == Color.RED) rect2.setFill(Color.BLUE);
51                   else                            rect2.setFill(Color.RED);
52              });
53      }
54
55      // 変数 p が表す Pane 上に円を描画
56      void drawCircles(Pane p) {
57
58          // 円 （中心の x 座標，y 座標，半径）
59          Circle cir = new Circle(30, 30, 20);
60          cir.setFill(Color.BLUE);  // 図形内部の塗りつぶしの色
61
62          // Circle オブジェクトを p にのせる
63          p.getChildren().addAll(cir);
64
65          // イベントハンドラを設定（クリックごとに位置が変わる）
66          cir.setOnMouseClicked((event) -> {
67                  cir.setCenterX(Math.random() * p.getWidth());
68                  cir.setCenterY(Math.random() * p.getHeight());
69              });
70      }
71
72      public static void main(String[] args) {
73          launch(args);
74      }
75  }
```