

```

1 import javafx.application.*;
2 import javafx.scene.*;
3 import javafx.scene.layout.*;
4 import javafx.scene.control.*;
5 import javafx.scene.paint.*;
6 import javafx.scene.image.*;
7 import javafx.scene.effect.*;
8 import javafx.scene.text.*;
9 import javafx.scene.input.*;
10 import javafx.stage.*;
11 import javafx.event.*;
12 import javafx.geometry.*;
13 import javafx.collections.*;
14
15 ///////////////////////////////////////////////////
16 // 問1 マウスイベントが発生したら、以下のように種類に応じ...
17 ///////////////////////////////////////////////////
18
19 public class Assignment7_1 extends Application
20 {
21     public void start(Stage stage) throws Exception
22     {
23         // レイアウトVBoxを生成／設定します
24         VBox vb = new VBox();
25
26         // シーンを生成／設定します
27         Scene scene = new Scene(vb);
28
29         // イベントハンドラを設定します
30         MyEventHandler mousehandler = new MyEventHandler();
31         scene.addEventHandler(MouseEvent.ANY, mousehandler);
32
33         // ステージを設定します
34         stage.setScene(scene);
35         stage.setTitle("マウスイベントの可視化");
36
37         // ステージを表示します
38         stage.show();
39     }
40
41     // イベントハンドラ（イベント処理）クラスの宣言
42     private class MyEventHandler implements EventHandler<MouseEvent>
43     {
44         public void handle(MouseEvent e)
45         {
46             EventType<? extends MouseEvent> type = e.getEventType();
47             if(type == MouseEvent.MOUSE_MOVED){
48                 System.out.print(">");
49             }
50             if(type == MouseEvent.MOUSE_PRESSED){
51                 System.out.print("P");
52             }
53             if(type == MouseEvent.MOUSE_RELEASED){
54                 System.out.print("R");
55             }
56             if(type == MouseEvent.MOUSE_CLICKED){
57                 System.out.println("C");
58             }
59             if(type == MouseEvent.MOUSE_DRAGGED){
60                 System.out.print("-");
61             }
62             if(type == MouseEvent.DRAG_DETECTED){
63                 System.out.print("!");
64             }
65         }
66     }
67
68     public static void main(String[] args)
69     {
70         launch(args);
71     }
72 }
73

```

```

1 import javafx.application.*;
2 import javafx.scene.*;
3 import javafx.scene.layout.*;
4 import javafx.scene.control.*;
5 import javafx.scene.paint.*;
6 import javafx.scene.image.*;
7 import javafx.scene.effect.*;
8 import javafx.scene.text.*;
9 import javafx.scene.input.*;
10 import javafx.stage.*;
11 import javafx.event.*;
12 import javafx.geometry.*;
13 import javafx.collections.*;
14
15 ///////////////////////////////////////////////////
16 // 問2 写真閲覧アプリを作成しましょう。5枚の写真を読み込...
17 ///////////////////////////////////////////////////
18
19 public class Assignment7_2 extends Application
20 {
21     private Image[] img;    // 表示する画像の配列
22     private int img_no;    // 現在表示している画像の添え字
23     private ImageView iv;   // 画像表示クラス型の変数
24
25     public void start(Stage stage) throws Exception
26     {
27         // 画像を準備します
28         img = new Image[5];
29         for(int i=0; i<img.length; i++)
30             img[i] = new Image("photo"+i+".jpg");
31
32         // 画像表示の準備をします
33         iv = new ImageView();
34         img_no = 0;
35         iv.setImage(img[img_no]);
36
37         // レイアウトVBoxを生成／設定します
38         VBox vb = new VBox();
39         ObservableList<Node> lst = vb.getChildren();
40         lst.add(iv);
41
42         // シーンを生成／設定します
43         Scene scene = new Scene(vb);
44
45         // イベントハンドラを設定します
46         MyEventHandler mousehandler = new MyEventHandler();
47         scene.addEventHandler(MouseEvent.ANY, mousehandler);
48
49         // ステージを設定します
50         stage.setScene(scene);
51         stage.setTitle("画像閲覧アプリ");
52
53         // ステージを表示します
54         stage.show();
55     }
56
57     // イベントハンドラ (イベント処理) クラスの宣言
58     private class MyEventHandler implements EventHandler<MouseEvent>
59     {
60         public void handle(MouseEvent e)
61         {
62             EventType<? extends MouseEvent> type = e.getEventType();
63             // マウスが押される度に次の画像を表示します
64             if(type == MouseEvent.MOUSE_PRESSED){
65                 img_no++;
66                 if(img_no >= img.length) img_no = 0;
67                 iv.setImage(img[img_no]);
68             }
69         }
70     }
71
72     public static void main(String[] args)
73     {
74         launch(args);
75     }
76 }
77

```

```

1 import javafx.application.*;
2 import javafx.scene.*;
3 import javafx.scene.layout.*;
4 import javafx.scene.control.*;
5 import javafx.scene.paint.*;
6 import javafx.scene.image.*;
7 import javafx.scene.effect.*;
8 import javafx.scene.text.*;
9 import javafx.scene.input.*;
10 import javafx.stage.*;
11 import javafx.event.*;
12 import javafx.geometry.*;
13 import javafx.collections.*;
14
15 ///////////////////////////////////////////////////
16 // 問3 「パノラマ画像閲覧アプリ」を作成しましょう。横4000...
17 ///////////////////////////////////////////////////
18
19 public class Assignment7_3 extends Application
20 {
21     private boolean slideLeft; // スライドの向き
22     private ImageView iv;      // 画像ファイル
23                                // - ハンドラから参照するため
24     private boolean flag_stopThread;
25                                // 画像のスライドスレッドの稼働フラグ
26
27     public void start(Stage stage) throws Exception
28     {
29         // 画像を準備します
30         Image im = new Image("pano1.jpg");
31
32         // ビューポートを準備します
33         // - 基となる画像の縦の長さを一辺とする正方形
34         // - 初期値は画像の中央に設定
35         Rectangle2D rt = new Rectangle2D(im.getWidth()/2+im.getHeight()/2, 0, im.getHeight(), im.getHeight());
36
37         // 画像枠の準備をします
38         iv = new ImageView(im);
39         iv.setViewport(rt);
40
41         // レイアウトVBoxを生成/設定します
42         VBox vb = new VBox();
43         ObservableList<Node> lst = vb.getChildren();
44         lst.add(iv);
45
46         // シーンを生成/設定します
47         Scene scene = new Scene(vb);
48
49         // イベントハンドラを設定します
50         MyEventHandler mousehandler = new MyEventHandler();
51         scene.addEventHandler(MouseEvent.ANY, mousehandler);
52
53         // 画像をスライドするスレッドを開始します
54         flag_stopThread = true;
55         MyThread mt = new MyThread();
56         mt.start();
57
58         // イベントハンドラを設定します
59         MyEventHandler2 mousehandler2 = new MyEventHandler2();
60         stage.addEventHandler(WindowEvent.ANY, mousehandler2);
61
62         // ステージを設定します
63         stage.setScene(scene);
64         stage.setTitle("パノラマ画像閲覧アプリ");
65         stage.setResizable(false);
66         // ウィンドウサイズを固定する場合に
67         // 以下をしないと余白が勝手に出現します
68         // バグです！ そのうち修正されと思います
69         stage.sizeToScene();
70
71         // ステージを表示します
72         stage.show();
73     }
74
75     // 画像のスライドを行うスレッドクラス
76     private class MyThread extends Thread implements Runnable
77     {
78         private Rectangle2D rt;
79
80         public void run(){
81             // 現在のビューポートを取得
82             rt = iv.getViewport();
83
84             // ビューポートを左または右へ移動
85             while(flag_stopThread){
86                 if(slideLeft){
87                     // 左へスライド
88                     if(rt.getMinX()-1 > 0)
89                         rt = new Rectangle2D(rt.getMinX()-1.5, 0, rt.getWidth(), rt.getHeight());
90                 }else{
91                     // 右へスライド
92                     if(rt.getMinX()+1 < iv.getImage().getWidth()-rt.getWidth())

```

```

93         rt = new Rectangle2D(rt.getMinX()+1.5, 0, rt.getWidth(), rt.getHeight());
94     }
95     iv.setViewport(rt);
96
97     try{
98         sleep(5);
99     }catch(Exception e){}
100 }
101 }
102 }
103
104 // イベントハンドラ（イベント処理）クラスの宣言
105 private class MyEventHandler implements EventHandler<MouseEvent>
106 {
107     public void handle(MouseEvent e)
108     {
109         EventType<? extends MouseEvent> type = e.getEventType();
110         // マウスが動く度に次の画像を表示します
111         if(type == MouseEvent.MOUSE_MOVED){
112             double mx = e.getX();
113             Rectangle2D rt = iv.getViewPort();
114
115             // マウスが画像の半分より左にあるか右にあるか判断します
116             // - 左にあると、画像を左へスライド
117             // - 右にあると、画像を右へスライド
118             if(mx > rt.getWidth()/2)
119                 slideLeft = false;
120             else
121                 slideLeft = true;
122         }
123     }
124 }
125
126 // イベントハンドラ（イベント処理）クラスの宣言
127 private class MyEventHandler2 implements EventHandler<WindowEvent>
128 {
129     public void handle(WindowEvent e)
130     {
131         EventType<WindowEvent> type = e.getEventType();
132         // ウィンドウが閉じられようとするとスレッドを停止
133         if(type == WindowEvent.WINDOW_CLOSE_REQUEST){
134             flag_stopThread = false;
135         }
136     }
137 }
138
139 public static void main(String[] args)
140 {
141     launch(args);
142 }
143 }
144

```