```java
import javafx.application.*;
import javafx.scene.*;
import javafx.scene.layout.*;
import javafx.scene.control.*;
import javafx.scene.paint.*;
import javafx.scene.image.*;
import javafx.scene.effect.*;
import javafx.scene.text.*;
import javafx.scene.input.*;
import javafx.scene.canvas.*;
import javafx.scene.shape.*;
import javafx.stage.*;
import javafx.event.*;
import javafx.geometry.*;
import javafx.collections.*;

// 追加文のインポート（タイマー用）
import java.util.*;
// 追加文のインポート（画像ファイル保存用）
import javafx.embed.swing.*;
import javax.imageio.*;
import java.io.*;

//////////////////////////////////////////////////////////
// 課題４　マウスを用いたタイムトライアルゲーム「いらいら…
//////////////////////////////////////////////////////////

public class Assignment12_4 extends Application
{
  // コース
  private final Image[] paths;
  // 背景
  private final Image[] backs;
  // メッセージ一覧
  private final String[] msgs;
  // ラジオボタン名一覧
  private final String[] rb_name;
  // ファイル名候補一覧
  private final String[] tf_file;

  // ゲーム画面の状態変数
  private Scene scene;                  // 全体画面
  private Canvas cv;                    // ゲームパネル
  private Label elapsedTime;            // 時間ラベル
  private Label topTime;                // 最短時間ラベル
  private Label showStatus;             // 情報ラベル
  private int mode;                     // ゲーム進捗 0:Free 1:Ready 2:Game 3:Finish
  private int indexPath;                // コース番号
  private int indexBack;                // 背景番号
  private long startStamp;              // スタート時刻
  private long currentStamp;            // 現在時刻
  private ArrayList<TimedPos> curPos;   // 現在カーソル移動履歴
  private ArrayList<ArrayList<TimedPos>> topPos;// 最速カーソル移動履歴
  private RadioButton[] setLvs;         // コース切替ラジオボタン
  private CheckBox autoSave;            // 新記録自動画像保存
  private TextField autoSaveFile;       // 新記録自動画像保存ファイル

  /////////////////
  // コンストラクタ
  public Assignment12_4(){
    // コース＆背景の読込
    paths = new Image[]{
        new Image("level1.png"),
        new Image("level2.png"),
        new Image("level3.png")};
    backs = new Image[]{
        new Image("level1_bg.jpg"),
        new Image("level2_bg.jpg"),
        new Image("level3_bg.jpg")};
    // メッセージ一覧の設定
    msgs = new String[]{
        "マウスカーソルを青い円または矩形まで移動しましょう．",
        "準備完了です．赤い円を目指して正確にコースをはみ出さずにマウスカーソルを進めましょう．",
        "スタートです．",
        "ゴールしました！"};
    // ラジオボタン名と識別子
    rb_name = new String[]{"初級","中級","上級"};
    // ファイル名候補一覧
    tf_file = new String[]{"太郎.png","次郎.png","三郎.png","花子.png","ポチ.png","たま.png","Sample.png","ScreenShot.png","Image.png"};
  }

  //////////////////////
  // アプリ準備メソッド
  public void start(Stage stage) throws Exception{
    // ゲーム画面の状態変数の初期設定
    // - ゲームパネルの生成
    cv = new Canvas(1024, 768);
    // - ゲームパネルへイベントハンドラの登録
    MouseEventHandler mh = new MouseEventHandler();
```

```
    cv.addEventHandler(MouseEvent.ANY, mh);
    // - 時間＆最短＆情報ラベルの生成
    elapsedTime = new Label("-");
    topTime = new Label("-");
    showStatus = new Label("-");
    // - 時間＆最短＆情報ラベルの設定
    elapsedTime.setPrefWidth(1024);
    elapsedTime.setBackground(new Background(new BackgroundFill(Color.GREEN,null,null)));
    elapsedTime.setTextFill(Color.LIGHTGREEN);
    elapsedTime.setFont(new Font("HGGothicM",18));
    elapsedTime.setAlignment(Pos.CENTER);
    topTime.setPrefWidth(1024-70*rb_name.length-170-250-10);
    topTime.setBackground(new Background(new BackgroundFill(Color.LIGHTGREEN,null,null)));
    topTime.setTextFill(Color.GREEN);
    topTime.setFont(new Font("HGGothicM",18));
    topTime.setAlignment(Pos.CENTER_RIGHT);
    showStatus.setPrefWidth(1024);
    showStatus.setBackground(new Background(new BackgroundFill(Color.WHITE,null,null)));
    showStatus.setTextFill(Color.DIMGREY);
    showStatus.setFont(new Font("HGGothicM",18));
    // - ゲーム進捗の設定
    mode = 0;
    showStatus.setText(msgs[mode]);
    // - コース＆背景番号の設定
    indexPath = 0;
    indexBack = 0;
    // - スタート＆現在時刻の設定
    startStamp = 0;
    currentStamp = 0;
    // - カーソル移動履歴の生成
    curPos = new ArrayList<TimedPos>();
    topPos = new ArrayList<ArrayList<TimedPos>>();
    for(int i=0; i<rb_name.length; i++)
      topPos.add(new ArrayList<TimedPos>());
    // - コース切替ラジオボタンの生成と設定
    setLvs = new RadioButton[rb_name.length];
    ToggleGroup tg = new ToggleGroup();
    RadioEventHandler reh = new RadioEventHandler();
    for(int i=0; i<setLvs.length; i++){
      setLvs[i] = new RadioButton(rb_name[i]);
      setLvs[i].setId(""+i);
      setLvs[i].setToggleGroup(tg);
      setLvs[i].setPrefWidth(70);
      setLvs[i].setTextFill(Color.GREEN);
      setLvs[i].setFont(new Font("HGGothicM",12));
      if(i==0) setLvs[i].setSelected(true);
      else setLvs[i].setSelected(false);
      setLvs[i].addEventHandler(ActionEvent.ANY, reh);
    }

    setRadioButtonDisable(false);
    // - 新記録自動画像保存用チェックボックスの生成と設定
    autoSave = new CheckBox("新記録時自動画像保存");
    autoSave.setPrefWidth(170);
    autoSave.setTextFill(Color.GREEN);
    autoSave.setFont(new Font("HGGothicM",12));
    autoSave.setSelected(false);
    // - 新記録自動画像保存ファイル用フィールドの生成と設定
    autoSaveFile = new TextField();
    autoSaveFile.setPrefWidth(250);
    autoSaveFile.setPromptText("保存画像ファイル名を入力（screenshot.png）");

    // コンテキストメニューの作成と登録
    ContextMenu cm = new ContextMenu();
    ObservableList<MenuItem> lstmi = cm.getItems();
    for(int i=0; i<tf_file.length; i++){
      MenuItem mi = new MenuItem(tf_file[i]);
      mi.setId(""+i);
      mi.setGraphic(new ImageView("fileicon.png"));
      lstmi.add(mi);
    }
    // - イベントハンドラをメニューへ登録
    TextFieldEventHandler tfen = new TextFieldEventHandler();
    cm.addEventHandler(ActionEvent.ANY, tfen);
    // - コンテキストメニューをテキストフィールドへ貼付
    autoSaveFile.setContextMenu(cm);

    // ゲーム画面の描画
    drawCanvas();

    // レイアウトの準備
    // - レベル変更パネル用
    GridPane gp = new GridPane();
    for(int i=0; i<setLvs.length; i++)
      gp.add(setLvs[i], i, 0);
    gp.add(autoSave, 4, 0);
    gp.add(autoSaveFile, 5, 0);
    gp.add(topTime, 6, 0);
    gp.setPadding(new Insets(5));
    gp.setBackground(null);

    // - ゲームパネル用
    VBox vb = new VBox();
    ObservableList<Node> lst = vb.getChildren();
```

```
    lst.add(gp);
    lst.add(elapsedTime);
    lst.add(cv);
    lst.add(showStatus);
    vb.setBackground(null);
    // - メニュー追加用
    BorderPane bp = new BorderPane();
    bp.setCenter(vb);
    bp.setBackground(null);

    // シーンの生成
    scene = new Scene(bp);
    scene.setFill(Color.LIGHTGREEN);

    // ステージの設定
    stage.setScene(scene);
    stage.setTitle("いらいらマウス");
    stage.setResizable(false);
    // ウィンドウサイズを固定する場合に
    // 以下をしないと余白が勝手に出現します
    // バグです！そのうち修正されると思います
    stage.sizeToScene();
    // ステージの表示
    stage.show();
  }

  //////////////////////////
  // キャンバス描画メソッド
  private void drawCanvas(){
    GraphicsContext gc = cv.getGraphicsContext2D();

    // ゲーム背景とコースを描画
    gc.drawImage(backs[indexBack], 0, 0);
    gc.drawImage(paths[indexPath], 0, 0);

    // 最速カーソル移動履歴の表示（赤色）
    // - モード０：全部表示
    // - モード１：非表示
    // - モード２：経過時間まで表示
    // - モード３：全部表示
    ArrayList<TimedPos> crtArlst = topPos.get(indexPath);
    if(crtArlst.size()>=2){
      gc.setLineWidth(8);
      gc.setStroke(Color.DARKRED);
      if(mode == 2){
        // 経過時間
        currentStamp = System.currentTimeMillis();
        long tmp = currentStamp-startStamp;
        // 経過時間までを描画
        for(int i=1; i<crtArlst.size(); i++){
          TimedPos s = crtArlst.get(i-1);
          TimedPos g = crtArlst.get(i);
          gc.strokeLine(s.getX(), s.getY(), g.getX(), g.getY());
          if(g.getTime() > tmp)break;
        }
      }else if(mode == 0 || mode == 3){
        for(int i=1; i<crtArlst.size(); i++){
          TimedPos s = crtArlst.get(i-1);
          TimedPos g = crtArlst.get(i);
          gc.strokeLine(s.getX(), s.getY(), g.getX(), g.getY());
        }
      }
    }
    // 現在カーソル移動履歴の表示（黄色）
    // - 全部を表示
    if(curPos.size()>=2){
      gc.setLineWidth(2);
      gc.setStroke(Color.LIGHTGREEN);
      for(int i=1; i<curPos.size(); i++){
        TimedPos s = curPos.get(i-1);
        TimedPos g = curPos.get(i);
        gc.strokeLine(s.getX(), s.getY(), g.getX(), g.getY());
      }
    }
  }

  ////////////////////////////////
  // コース切替ボタンの設定メソッド
  private void setRadioButtonDisable(boolean b){
    // 有効化／無効化の設定変更
    for(int i=0; i<setLvs.length; i++){
      setLvs[i].setDisable(b);
    }
  }

  ////////////////////////////////
  // スクリーンショット保存メソッド
  private void saveScreenToImage(){
    // - 画像として画面の保存
    WritableImage bi = scene.snapshot(null);
    File file;
    String str = autoSaveFile.getText();
    if(!str.equals(""))
```

```
      file = new File(str);
    else
      file = new File("screenshot.jpg");
    try{
      ImageIO.write(SwingFXUtils.fromFXImage(bi, null), "png", file);
    }catch(Exception ee){}
  }

  ////////////////////////////////////////////////
  // ゲーム進捗切替用マウスハンドラクラス（インナー）
  private class MouseEventHandler implements EventHandler<MouseEvent>
  {
    // 経過時間表示用
    private Timer tm;

    public MouseEventHandler(){
      // 経過時間表示用変数の初期化
      tm = null;
    }

    public void handle(MouseEvent e){
      EventType<? extends MouseEvent> type = e.getEventType();
      if(type ==  MouseEvent.MOUSE_MOVED || type ==  MouseEvent.MOUSE_DRAGGED){
        // マウス座標の取得
        double mx = e.getX();
        double my = e.getY();

        // マウス位置の色の取得
        PixelReader pl = paths[indexPath].getPixelReader();
        Color c = pl.getColor((int)mx, (int)my);
        double r = c.getRed();
        double g = c.getGreen();
        double b = c.getBlue();

        // ゲーム進捗の更新
        switch(mode){
          ///
          /// モード０〔Free〕////////////////////
          ///
          case 0:
            if(r==0.0 && g==0.0 && b==1.0){
              // モード０ → モード１
              mode = 1;
              elapsedTime.setBackground(new Background(new BackgroundFill(Color.GREEN,null,null)));
              elapsedTime.setTextFill(Color.LIGHTGREEN);
              elapsedTime.setText("-");
              showStatus.setText(msgs[mode]);
              // 現在カーソル移動履歴の削除
              curPos.clear();
              // コース切り替えボタンの無効化
              setRadioButtonDisable(true);
            }
            break;
          ///
          /// モード１〔Ready〕////////////////////
          ///
          case 1:
            if(r==1.0 && g==1.0 && b==1.0){
              // モード１ → モード０
              mode = 0;
              elapsedTime.setText("-");
              showStatus.setText(msgs[mode]);
              // コース切り替えボタンの有効化
              setRadioButtonDisable(false);
            }else if(r==0.0 && g==0.0 && b==0.0){
              // 開始時間の記録
              startStamp = System.currentTimeMillis();
              // モード１ → モード２
              mode = 2;
              elapsedTime.setText(String.format("経過時間（秒）  %06.2f", 0.0));
              showStatus.setText(msgs[mode]);
              // 現在カーソル移動履歴の追加
              curPos.add(new TimedPos(0, mx, my));
              // アニメーションの開始
              tm = new Timer();
              tm.schedule(new MyTimer(), 0, 50);
            }
            break;
          ///
          /// モード２〔Game〕////////////////////
          ///
          case 2:
            if(r==1.0 && g==1.0 && b==1.0){
              // 経過時間表示タイマーの終了
              tm.cancel();
              tm = null;
              // モード２ → モード０
              mode = 0;
              elapsedTime.setText("-");
              showStatus.setText(msgs[mode]);
              // 現在カーソル移動履歴に最終位置の追加
              currentStamp = System.currentTimeMillis();
              curPos.add(new TimedPos(currentStamp-startStamp, mx, my));
```

```
            // コース切り替えボタンの有効化
            setRadioButtonDisable(false);
          }else if(r==1.0 && g==0.0 && b==0.0){
            // 経過時間表示タイマーの終了
            tm.cancel();
            tm = null;
            // 終了時間の記録
            currentStamp = System.currentTimeMillis();
            // モード２ → モード３
            mode = 3;
            elapsedTime.setText(String.format("経過時間（秒） %06.2f", (currentStamp-startStamp)/1000.0));
            showStatus.setText(msgs[mode]);
            // 現在カーソル移動履歴に最終位置の追加
            curPos.add(new TimedPos(currentStamp-startStamp, mx, my));
            // 最速カーソル移動履歴の更新
            ArrayList<TimedPos> crtArlst = topPos.get(indexPath);
            if(crtArlst.size() != 0){
              // - 既に履歴がある場合
              TimedPos cur = curPos.get(curPos.size()-1);
              TimedPos top = crtArlst.get(crtArlst.size()-1);
              if(cur.getTime() < top.getTime()){
                elapsedTime.setBackground(new Background(new BackgroundFill(Color.DARKRED,null,null)));
                elapsedTime.setTextFill(Color.PINK);
                elapsedTime.setText("【新記録】" + elapsedTime.getText()+" 【新記録】");
                crtArlst.clear();
                for(int i=0; i<curPos.size(); i++)
                  crtArlst.add(curPos.get(i));
                // - 最短ラベルの更新
                top = crtArlst.get(crtArlst.size()-1);
                topTime.setText(String.format("最短記録（秒） %06.2f", top.getTime()/1000.0));
                // - 画像として画面の保存
                if(autoSave.isSelected())
                  saveScreenToImage();
              }
            }else{
              // - 記録が１つもない場合
              elapsedTime.setBackground(new Background(new BackgroundFill(Color.DARKRED,null,null)));
              elapsedTime.setTextFill(Color.PINK);
              elapsedTime.setText("【新記録】" + elapsedTime.getText()+" 【新記録】");
              for(int i=0; i<curPos.size(); i++)
                crtArlst.add(curPos.get(i));
              // - 最短ラベルの更新
              TimedPos top = crtArlst.get(crtArlst.size()-1);
              topTime.setText(String.format("最短記録（秒） %06.2f", top.getTime()/1000.0));
              // - 画像として画面の保存
              if(autoSave.isSelected())
                saveScreenToImage();
            }
          }else{
            // 現在カーソル移動履歴の追加
            currentStamp = System.currentTimeMillis();
            curPos.add(new TimedPos(currentStamp-startStamp, mx, my));
          }
          break;
        ///
        /// モード３〔Finish〕//////////////////
        ///
        case 3:
          if(r==1.0 && g==1.0 && b==1.0){
            // モード３ → モード０
            mode = 0;
            // ※時間ラベルの変更はしない
            showStatus.setText(msgs[mode]);
            // コース切り替えボタンの有効化
            setRadioButtonDisable(false);
          }else if(r==0.0 && g==0.0 && b==0.0){
            // モード３ → モード０
            mode = 0;
            // ※時間ラベルの変更はしない
            showStatus.setText(msgs[mode]);
            // コース切り替えボタンの有効化
            setRadioButtonDisable(false);
          }
          break;
      }
    }
    // 再描画
    drawCanvas();
  }

  ////////////////////////////////////////////////
  // アニメーションタイマークラス（ダブルインナー）
  private class MyTimer extends TimerTask
  {
    public void run(){
      Platform.runLater(new MyRunLater());
    }
  }

  ////////////////////////////////////////////
  // アニメーション描画クラス（ダブルインナー）
  private class MyRunLater implements Runnable
```

```
462     {
463       public void run(){
464         if(mode == 2){
465           // 経過時間の表示
466           currentStamp = System.currentTimeMillis();
467           elapsedTime.setText(String.format("経過時間（秒） %06.2f", (currentStamp-startStamp)/1000.0));
468           // 再描画
469           drawCanvas();
470         }
471       }
472     }
473   }
474
475   ////////////////////////////////////////////////////////////
476   // コース番号切り替え用ラジオボタンハンドラクラス（インナー）
477   private class RadioEventHandler implements EventHandler<ActionEvent>
478   {
479     public void handle(ActionEvent e){
480       // ラジオボタンと識別番号の取得
481       RadioButton bt = (RadioButton)e.getTarget();
482       int id = Integer.parseInt(bt.getId());
483       if(mode == 0){
484         // ↑モード０の時にしかコールされないけど念のため…
485         indexPath = id;
486         indexBack = id;
487         // 現在カーソル移動履歴の削除
488         curPos.clear();
489         // - 最短ラベルの更新
490         ArrayList<TimedPos> crtArlst = topPos.get(indexPath);
491         if(crtArlst.size() != 0){
492           TimedPos top = crtArlst.get(crtArlst.size()-1);
493           topTime.setText(String.format("最短記録（秒） %06.2f", top.getTime()/1000.0));
494         }else{
495             topTime.setText("-");
496         }
497         // - 経過時間ラベルの更新
498         elapsedTime.setBackground(new Background(new BackgroundFill(Color.GREEN,null,null)));
499         elapsedTime.setTextFill(Color.LIGHTGREEN);
500         elapsedTime.setText("-");
501         // 再描画
502         drawCanvas();
503       }
504     }
505   }
506
507   ////////////////////////////////////////////////////
508   // ファイル名変更用メニューハンドラクラス（インナー）
509   private class TextFieldEventHandler implements EventHandler<ActionEvent>
510   {
511     public void handle(ActionEvent e){
512       // メニュー項目と識別番号の取得
513       MenuItem mi = (MenuItem)e.getTarget();
514       int id = Integer.parseInt(mi.getId());
515       // テキストフィールドのファイル名変更
516       System.out.println(id);
517       autoSaveFile.setText(tf_file[id]);
518     }
519   }
520
521   ///////////////////////
522   // マウス移動履歴クラス
523   private class TimedPos
524   {
525     private final long time; // 経過タイムスタンプ
526     private final double hx; // カーソル位置X
527     private final double hy; // カーソル位置Y
528
529     public TimedPos(long t, double x, double y){
530       time = t;
531       hx = x;
532       hy = y;
533     }
534     // 各種取得メソッド
535     public long getTime(){return time;}
536     public double getX(){return hx;}
537     public double getY(){return hy;}
538   }
539
540   public static void main(String[] args)
541   {
542     launch(args);
543   }
544 }
545
546
```