Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра кібербезпеки та програмного забезпечення

**Звіт**

з виконаної лабораторної роботи № 3

Дисципліна: Паралельні та розподілені обчислення

на тему

«Керування процесом паралельних обчислень.»

Виконав :

студент академічної групи КІ-15

Аннаєв А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перевірив :

Викладач

Минайленко Р. М.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кропивницкий- 2017

**Мета:** Одержати навички керування процесом паралельних обчислень.

**Завдання:**

- Створити проект.

- Вибрати колір, довжину і ширину прямокутника відповідно до варіанта.

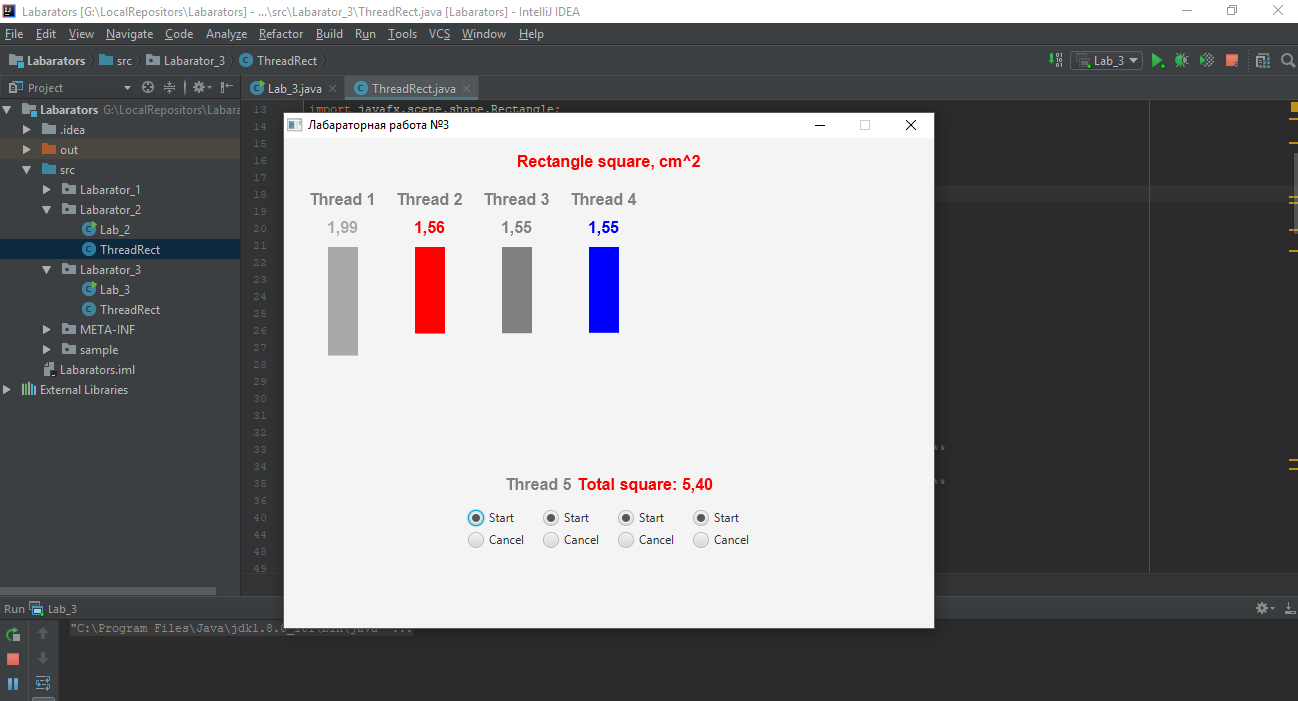
- Створити задачу і паралельні потоки обчислень.

- У кожному потоці обчислити площу прямокутника в сантиметрах квадратних.

- В окремому потоці обчислити сумарну площу прямокутників у сантиметрах   
 квадратних.

- Виконати програму.

Хід роботи



package Labarator\_3;

import javafx.application.Application;

import javafx.application.Platform;

import javafx.concurrent.Task;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.geometry.Rectangle2D;

import javafx.scene.Scene;

import javafx.scene.control.RadioButton;

import javafx.scene.control.ToggleGroup;

import javafx.scene.effect.DropShadow;

import javafx.scene.layout.HBox;

import javafx.scene.layout.Pane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.scene.text.Text;

import javafx.stage.Screen;

import javafx.stage.Stage;

import java.awt.event.ActionEvent;

import java.util.Vector;

public class Lab\_3 extends Application {

VBox rootPane;

Text totalSquereLabel;

Vector<Task<Void>> tasks;

Vector<ThreadRect> rects;

final double RECT\_WIDTH = 30;

final double RECT\_HEIGHT = 190;

final int TASK\_DURATION = 100;

final int TOTAL\_TASK\_DURATION = 200;

private double dpi;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Определение нового потока с параметрами

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private void createScene() {

totalSquereLabel = createLabel("0.0", Color.RED);

rootPane = new VBox();

rootPane.setPadding(new Insets(15, 15, 15, 15));

rootPane.setSpacing(10);

HBox headerBox = new HBox();

HBox centerHBox = new HBox();

HBox toggleHBox = new HBox();

HBox totalHBox = new HBox();

Text headerLabel = createLabel("Rectangle square, cm^2", Color.RED);

headerBox.setAlignment(Pos.CENTER);

headerBox.getChildren().addAll(headerLabel);

// Setup thread rectangles

// First rectangle

addThreadRectangle(centerHBox, toggleHBox,

new ThreadRect(dpi, 1000, "Thread 1", RECT\_HEIGHT, Color.DARKGREY),

TASK\_DURATION);

// Second rectangle

addThreadRectangle(centerHBox, toggleHBox,

new ThreadRect(dpi, 1000, "Thread 2", RECT\_HEIGHT, Color.RED),

TASK\_DURATION);

// Third rectangle

addThreadRectangle(centerHBox, toggleHBox,

new ThreadRect(dpi, 1000, "Thread 3", RECT\_HEIGHT, Color.GREY),

TASK\_DURATION);

// Fourth rectangle

addThreadRectangle(centerHBox, toggleHBox,

new ThreadRect(dpi, 1000, "Thread 4", RECT\_HEIGHT, Color.BLUE),

TASK\_DURATION);

toggleHBox.setSpacing(18);

toggleHBox.setPadding(new Insets(5, 5, 5, 5));

toggleHBox.setAlignment(Pos.CENTER);

Text totalThreadLabel = createLabel("Thread 5 ", Color.GRAY);

Text footerLabel = createLabel("Total square: ", Color.RED);

totalHBox.getChildren().addAll(totalThreadLabel, footerLabel, totalSquereLabel);

totalHBox.setAlignment(Pos.BOTTOM\_CENTER);

DropShadow dropShadow = new DropShadow();

dropShadow.setRadius(5.0);

dropShadow.setOffsetX(3.0); // Shadow offset (X and Y axis)

dropShadow.setOffsetY(3.0);

dropShadow.setColor(Color.GRAY); // Shadow color

rootPane.setEffect(dropShadow);

centerHBox.setMinHeight(1.5\*RECT\_HEIGHT);

rootPane.getChildren().addAll(headerBox, centerHBox, totalHBox, toggleHBox);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Определение нового потока с параметрами

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private void addThreadRectangle(Pane rectsPane, Pane togglePane, ThreadRect rect, int

taskDurationMsecs)

{

rect.setWidth(RECT\_WIDTH);

rect.setHeight(RECT\_HEIGHT);

VBox v = rect.createLayout();

rect.play();

rectsPane.getChildren().addAll(v);

Task<Void> t = rect.squareCalculateTask(taskDurationMsecs);

tasks.add(t);

rects.add(rect);

ToggleGroup tg = new ToggleGroup();

RadioButton rb\_1 = new RadioButton("Start");

RadioButton rb\_2 = new RadioButton("Cancel");

rb\_1.setToggleGroup(tg);

rb\_2.setToggleGroup(tg);

tg.selectToggle(rb\_1);

VBox vB = new VBox();

vB.getChildren().addAll(rb\_1, rb\_2);

vB.setSpacing(5);

togglePane.getChildren().add(vB);

// Start thread

Thread th = new Thread(t);

th.setName("Rect Thread");

th.start();

rb\_1.setOnAction((javafx.event.ActionEvent ActionEvent) ->{

Task<Void> newT = rect.squareCalculateTask((taskDurationMsecs));

tasks.add(newT);

Thread newTH = new Thread(newT);

newTH.setName("Rect Thread");

newTH.start();

rect.play();

});

rb\_2.setOnAction((javafx.event.ActionEvent ActionEvent)-> {

t.cancel();

tasks.remove(t);

rect.pause();

});

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Определение текста

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private Text createLabel(String text, Color color) {

Text t = new Text();

t.setText(text);

t.setFont(Font.font("Arial", FontWeight.BOLD, 16));

t.setFill(color);

return t;

} //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Определение суммарного значения площадей

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private void createMainThread(){

Task<Void> task = new Task<Void>() {

@Override

protected Void call() throws Exception {

while (true) {

Platform.runLater(() -> {

double s = 0.0f;

for (ThreadRect r: rects) {

s += r.getSquare();

}

totalSquereLabel.setText(String.format("%1$,.2f", s));

});

Thread.sleep(TOTAL\_TASK\_DURATION);

}

}

};

tasks.add(task);

Thread th = new Thread(task);

th.setName("Total thread");

th.start();

} //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Точка входа в программу

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public static void main(String[] args)

{

launch(args);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Override

public void start(Stage primaryStage) throws Exception{

this.dpi = Screen.getPrimary().getDpi();

Rectangle2D r = Screen.getPrimary().getBounds();

if(this.dpi == -1)

this.dpi = 103.96; // DPI for 1366x768 and 15.6 inches

tasks = new Vector<>();

rects = new Vector<>();

primaryStage.setTitle("Лабараторная работа №3");

primaryStage.setResizable(false);

createScene();

createMainThread();

// обработчик закрытия окна

primaryStage.setOnCloseRequest(event -> {

for (Task<Void> t: tasks) {

t.cancel();

}

});

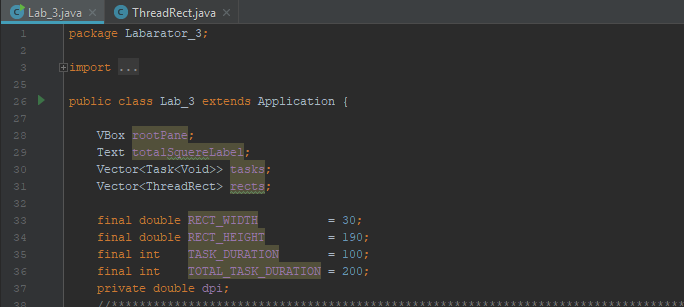
Scene scene = new Scene(rootPane, 640, 480, Color.TRANSPARENT);

primaryStage.setScene(scene);

primaryStage.show();

}

}



package Labarator\_3;

import javafx.animation.FadeTransition;

import javafx.animation.KeyFrame;

import javafx.animation.KeyValue;

import javafx.animation.Timeline;

import javafx.application.Platform;

import javafx.concurrent.Task;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.effect.DropShadow;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.shape.Rectangle;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.scene.text.Text;

import javafx.util.Duration;

public class ThreadRect {

private final Timeline timeline;

private final double CENT\_PER\_INCH = 0.393700787;

private double square = 0.0f;

private Rectangle rect;

private VBox mainLayout;

private String threadName;

private Text squereLabel;

private double rectMaxHeight;

private Color rectColor;

private int animMsecs = 0;

private double dpi;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Геттеры и Сеттеры

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public double getSquare() { return square; }

public void setWidth(double width) {rect.setWidth(width); }

public void setHeight(double height) { rect.setHeight(height); }

public VBox getLayout(){ return mainLayout;}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Конструктор

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public ThreadRect(double dpi, int animationMsecs, String threadTitle, double maxHeight, Color color)

{

animMsecs = animationMsecs;

threadName = threadTitle;

rectMaxHeight = maxHeight;

rect = new Rectangle();

rectColor = color;

rect.setFill(rectColor);

this.dpi = dpi;

timeline = new Timeline();

timeline.setCycleCount(Timeline.INDEFINITE);

timeline.setAutoReverse(true);

final KeyValue kv1 = new KeyValue(rect.heightProperty(), 0);

final KeyValue kv2 = new KeyValue(rect.heightProperty(), rectMaxHeight);

final KeyFrame kf1 = new KeyFrame(Duration.millis(animMsecs), kv1);

final KeyFrame kf2 = new KeyFrame(Duration.millis(animMsecs), kv2);

timeline.getKeyFrames().add(kf1);

timeline.getKeyFrames().add(kf2);

createLayout();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Запуск анимации

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public void play() {

timeline.play();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Остановка анимации

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public void pause() {

timeline.pause();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Настройка размещения прямоугольника на форме

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public VBox createLayout()

{

mainLayout = new VBox();

mainLayout.setPadding(new Insets(10, 10, 10, 10));

mainLayout.setSpacing(10);

mainLayout.setAlignment(Pos.TOP\_CENTER);

Text titleLabel = createLabel(threadName, Color.GRAY);

squereLabel = createLabel("0.0", rectColor);

mainLayout.getChildren().addAll(titleLabel, squereLabel, rect);

return mainLayout;

} //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Задача подсчета значения площади прямоугольника в отдельном потоке

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public Task<Void> squareCalculateTask(int sleepMsec){

Task<Void> task = new Task<Void>() {

@Override

protected Void call() throws Exception {

while (true) {

Platform.runLater(() -> {

square = ((rect.getHeight() / dpi / CENT\_PER\_INCH) \* (rect.getWidth() /

dpi / CENT\_PER\_INCH)) ;

squereLabel.setText(String.format("%1$,.2f", square));

});

Thread.sleep(sleepMsec);

}

}

};

return task;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Создание и настройка отображения текстовой информации

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private Text createLabel(String text, Color color){

Text t = new Text();

t.setText(text);

t.setFont(Font.font("Arial", FontWeight.BOLD, 16));

t.setFill(color);

return t;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}

