Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра кібербезпеки та програмного забезпечення

**Звіт**

з виконаної лабораторної роботи № 2

Дисципліна: Паралельні та розподілені обчислення

на тему

«Створення додатка для паралельних обчислень.»

Виконав :

студент академічної групи КІ-15

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

Перевірив :

Викладач

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

Кропивницкий- 2017

**Мета:** Одержати навички створення додатків для паралельних обчислень.

**Завдання:**

- Створити проект.

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

- Створити задачі і паралельні потоки обчислень.

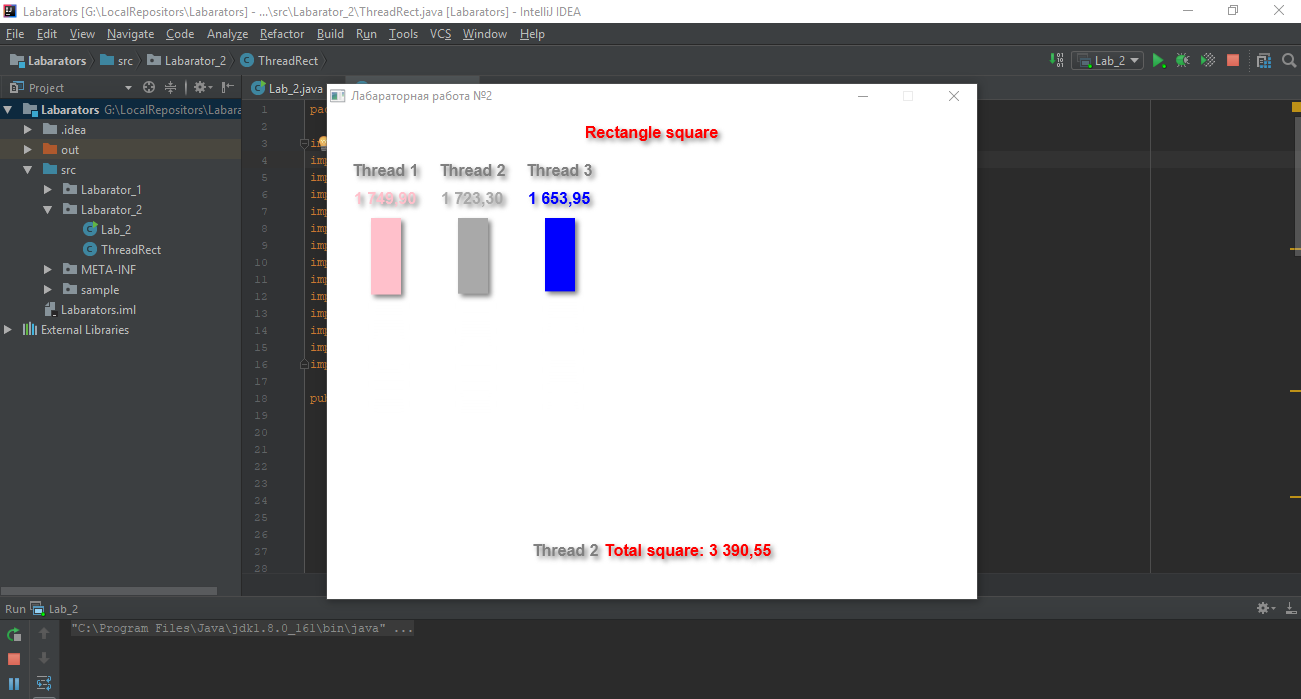
- У кожному потоці обчислити площу прямокутника в пікселах.

- В окремому потоці обчислити сумарну площу прямокутників у пікселах.

- Виконати програму

Хід роботи

Вариант №8



package Labarator\_2;

import javafx.application.Application;

import javafx.application.Platform;

import javafx.concurrent.Task;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

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.Stage;

import java.util.Vector;

public class Lab\_2 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;

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

// Определение нового потока с параметрами //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private void createScene() {

rootPane = new VBox();

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

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

rootPane.setSpacing(10);

HBox headerBox = new HBox();

HBox centerHBox = new HBox();

HBox footerHBox = new HBox();

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

headerBox.setAlignment(Pos.CENTER);

headerBox.getChildren().addAll(headerLabel);

// Setup thread rectangles

// First rectangle

addThreadRectangle(centerHBox,

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

TASK\_DURATION);

addThreadRectangle(centerHBox,

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

TASK\_DURATION);

addThreadRectangle(centerHBox,

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

TASK\_DURATION);

Text totalThreadLabel = createLabel("Thread 2 ", Color.GRAY);

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

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

footerHBox.setAlignment(Pos.BOTTOM\_CENTER);

// Setup shadow

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(2\*RECT\_HEIGHT);

rootPane.getChildren().addAll(headerBox, centerHBox, footerHBox);

}

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

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

// Определение нового потока с параметрами

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

private void addThreadRectangle(Pane destPane,

ThreadRect rect, int taskDurationMsecs) {

// установка параметров прямоугольника с константными значениями

rect.setWidth(RECT\_WIDTH);

rect.setHeight(RECT\_HEIGHT);

// определение расположения прямоугольника

VBox v = rect.createLayout();

// запуск

rect.startAnimation();

// добавление в родительскую ветвь

destPane.getChildren().addAll(v);

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

tasks.add(t);

rects.add(rect);

// Start thread

Thread th = new Thread(t);

th.setName("Rect Thread");

th.start();

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

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

// 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.getSquere();

}

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{

// инициализация списка задач

tasks = new Vector<>();

// инициализация списка прямоугольником переопределенные под работу с потоками

rects = new Vector<>();

// Описание программы

primaryStage.setTitle("Лабараторная работа №2");

// Не изменяемый размер

primaryStage.setResizable(false);

// создание сцены

createScene();

// главный поток(суммирующий)

createMainThread();

// обработчик закрытия окна

primaryStage.setOnCloseRequest(event -> {

for (Task<Void> t: tasks) {

t.cancel();

}

});

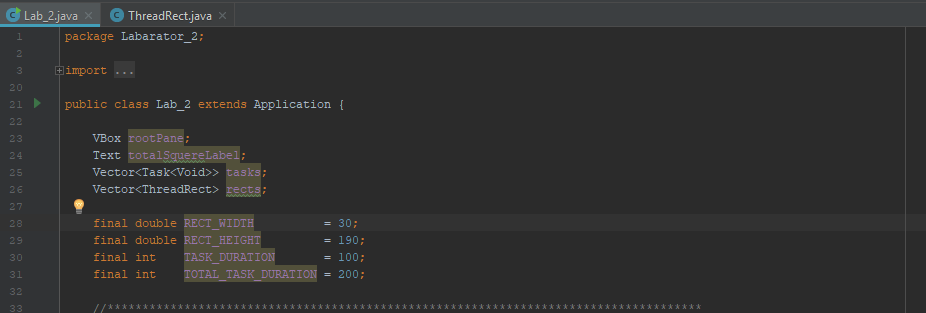
Scene scene = new Scene(rootPane, 640, 480, Color.TRANSPARENT);

primaryStage.setScene(scene);

primaryStage.show();

}

}



package Labarator\_2;

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.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 {

// текущая площать прямоугольника

private double squere = 0.0f;

// экземпляр обьекта прямоугольник

private Rectangle rect;

// размещение прямоугольника главная форма обьекта прямоугольник

private VBox mainLayout;

// имя потока

private String threadName;

// поле вывода значения площади

private Text squereLabel;

// максимальное допустимое значения прямоугольника

private double rectMaxHeight;

// цвет прямоугольника

private Color rectColor;

// интервал анимации

private int animMsec;

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

// Геттеры и Сеттеры

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

public double getSquere()

{

return squere;

}

public void setWidth(double width)

{

rect.setWidth(width);

}

public void setHeight(double height)

{

rect.setHeight(height);

}

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

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

// Конструктор

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

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

animMsec = animationMsecs;

threadName = threadTitle;

rectMaxHeight = maxHeight;

rect = new Rectangle();

rectColor = color;

rect.setFill(rectColor);

createLayout();

}

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

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

// Запуск анимации

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

public void startAnimation(){

final Timeline 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(animMsec), kv1);

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

timeline.getKeyFrames().add(kf1);

timeline.getKeyFrames().add(kf2);

timeline.play();

}

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

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

// 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(() -> {

squere = rect.getHeight() \* rect.getWidth();

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

});

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;

}

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

}

