МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

 «БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ИИТ

ЛАБОРАТОРНАЯ РАБОТА №8

По дисциплине: «Современные платформы программирования»

Выполнил:

Студент ФЭИС

3-го курса, группы ПО-5

Белко В. А.

Проверил:

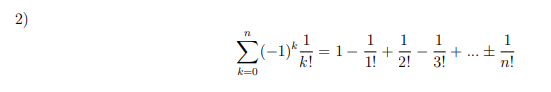
Крощенко А. А.

Брест 2022

**Вариант: 2**

**Цель работы:** приобрести навыки написания простого оконного многопоточного приложения с использованием Java API

**Задание**: Разработать оконное приложение с использованием Java API, использующее один вспомогательный поток, вычисляющий заданную сумму и выполняющий вывод результата вычисления (как конечный, так и промежуточные) в любой визуальный компонент. Все исходные данные вводятся в соответствующие визуальные компоненты. В программе должны быть предусмотрены функции

приостановки, возобновления и полной остановки выполнения потока с выводом соответствующего сообщения. В случае быстрого выполнения потока и, как следствие, невозможности демонстрации функций приостановки, продумать искусственное «торможение» потока для достижения заданных целей. Обработать исключения.

Код программы:

**CalculRow**

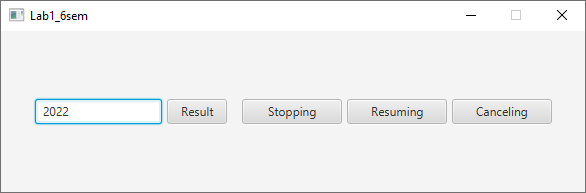
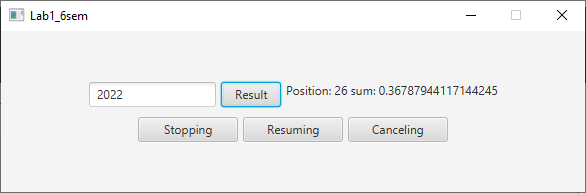
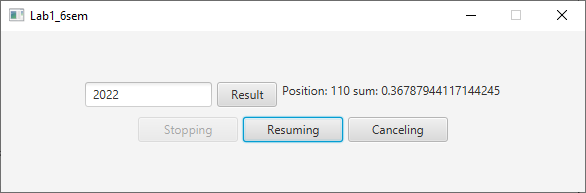
package com.company;  
import javafx.concurrent.Task;  
public class CalculRow extends Task {  
 private int randNum;  
 public volatile boolean stopping;  
 CalculRow(int randNum) {  
 this.randNum = randNum;  
 }  
 @Override  
 protected Object call() {  
 try {  
// Для 2 варианта - коф. в рекуррентной формуле после упрощений имеет вид -1/n  
// Первый (при n = 0) член ряда равен 1, значит идем от n=1 и далее  
 double current = 0, previous = 1, sum = previous;  
 for (int n = 1; n <= randNum; n++) {  
 while (stopping) { }  
 current = previous \* (- 1 / (double)n);  
 previous = current;  
 sum += current;  
 updateMessage("Position: " + n + " sum: " + sum);  
 Thread.*sleep*(100);  
 }  
 }  
 catch (InterruptedException ex) {  
 System.*out*.println(ex.getMessage());  
 }  
  
 return null;  
 }  
}

**MultiThread**

package com.company;  
public class MultiThread {  
 private CalculRow calcul;  
 private Thread thread;  
 public void start() {  
 if(calcul != null) {  
 thread.start();  
 }  
 }  
 public void stop() {  
 if(calcul != null) {  
 calcul.stopping = true;  
 }  
 }  
 public void resume() {  
 if(calcul != null) {  
 calcul.stopping = false;  
 }  
 }  
 public void interrupt() {  
 if(calcul != null) {  
 thread.interrupt();  
 }  
 }  
 public void setCalculator(CalculRow calcul) {  
 if (this.calcul != null) {  
 interrupt();  
 }  
 this.calcul = calcul;  
 thread = new Thread(calcul);  
 }  
}

**Main**

package com.company;  
import javafx.application.Application;  
import javafx.geometry.Orientation;  
import javafx.scene.layout.HBox;  
import javafx.stage.Stage;  
import javafx.scene.Scene;  
import javafx.scene.layout.FlowPane;  
import javafx.scene.control.Label;  
import javafx.scene.control.Button;  
import javafx.scene.control.TextField;  
import javafx.geometry.Pos;  
public class Main extends Application {  
 private Button stopThread;  
 private Button resumeThread;  
 private Button interruptThread;  
 private Button calculate;  
 private HBox threadControls;  
 private HBox calculationControls;  
 private TextField serialNumberInput;  
  
 private Scene scene;  
 private FlowPane root;  
 private Label calculationOutput;  
 private MultiThread thread;  
 public static void main(String[] args) {  
 Application.*launch*(args);  
 }  
 @Override  
 public void start(Stage stage) {  
 thread = new MultiThread();  
 root = createRoot();  
 scene = new Scene(root);  
 setStage(stage);  
 stage.show();  
 }  
 private FlowPane createRoot() {  
 calculationControls = createCalculationControl();  
 threadControls = createControls();  
 FlowPane result = new FlowPane(calculationControls, threadControls);  
 result.setAlignment(Pos.*CENTER*);  
 result.setOrientation(Orientation.*HORIZONTAL*);  
 result.setVgap(10);  
 result.setHgap(10);  
 return result;  
 }  
 private void setStage(Stage stage) {  
 stage.setTitle("Lab1\_6sem");  
 stage.setResizable(false);  
 stage.setScene(scene);  
 stage.setMinHeight(200);  
 stage.setMinWidth(600);  
 stage.setMaxHeight(200);  
 stage.setMaxWidth(600);  
 }  
 private HBox createControls() {  
 stopThread = new Button("Stopping");  
 stopThread.setPrefWidth(100);  
 stopThread.setOnAction(event -> {  
 thread.stop();  
 stopThread.setDisable(true);  
 resumeThread.setDisable(false);  
 });  
 resumeThread = new Button("Resuming");  
 resumeThread.setPrefWidth(100);  
 resumeThread.setOnAction(event -> {  
 thread.resume();  
 stopThread.setDisable(false);  
 resumeThread.setDisable(true);  
 });  
 interruptThread = new Button("Canceling");  
 interruptThread.setPrefWidth(100);  
 interruptThread.setOnAction(event -> {  
 thread.interrupt();  
 stopThread.setDisable(true);  
 resumeThread.setDisable(true);  
 });  
 return new HBox(5, stopThread, resumeThread, interruptThread);  
 }  
 private HBox createCalculationControl() {  
 serialNumberInput = new TextField();  
  
 serialNumberInput.setPrefColumnCount(10);  
 calculationOutput = new Label();  
 calculate = new Button("Result");  
 calculate.setPrefWidth(60);  
 calculate.setOnAction(event -> {  
 try {  
 int number = Integer.*parseInt*(serialNumberInput.getText());  
 CalculRow seriesCalculator = new CalculRow(number);  
 thread.setCalculator(seriesCalculator);  
 calculationOutput.textProperty().bind(seriesCalculator.messageProperty());  
 thread.start();  
 stopThread.setDisable(false);  
 resumeThread.setDisable(false);  
 interruptThread.setDisable(false);  
 }  
 catch (NumberFormatException ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 });  
 return new HBox(5, serialNumberInput, calculate, calculationOutput);  
 }  
}



**Вывод:** в ходе выполнения лабораторной работы были получены навыки написания простого оконного многопоточного приложения с использованием Java API.