Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

Кафедра ИИТ

**Лабораторная работа №8**

По дисциплине: "СПП"

Выполнил:

Студент 3 курса

Группы ПО-3

Лущ М. Г.

Проверил:

Крощенко А. А.

Брест 2021

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

Вариант 2

**Общее задание**

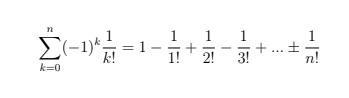
Разработать оконное приложение с использованием Java API, использующее один вспомогательный поток, вычисляющий заданную сумму и выполняющий вывод результата вычисления (как конечный, так и промежуточные) в любой визуальный компонент. Все исходные данные вводятся в соответствующие визуальные компоненты.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

Main

package sample;  
  
import javafx.application.Application;  
import javafx.geometry.Orientation;  
import javafx.scene.layout.VBox;  
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 VBox threadControlsSection;  
 private VBox calculationControlsSection;  
 private TextField serialNumberInput;  
 private Scene scene;  
 private Label threadButtonsInfo;  
 private FlowPane root;  
 private Label calculationOutput;  
  
 private ThreadManager thread;  
  
 public static void main(String[] args) {  
 Application.*launch*(args);  
 }  
  
 @Override  
 public void start(Stage stage) {  
 thread = new ThreadManager();  
  
 root = createUiRoot();  
 scene = new Scene(root);  
  
 configureStage(stage);  
 stage.show();  
 }  
  
 private FlowPane createUiRoot() {  
 calculationControlsSection = createCalculationControlSection();  
 threadControlsSection = createThreadControlSection();  
  
 FlowPane result = new FlowPane(calculationControlsSection, threadControlsSection);  
 result.setAlignment(Pos.*CENTER*);  
 result.setOrientation(Orientation.*HORIZONTAL*);  
 result.setVgap(10);  
 result.setHgap(10);  
  
 return result;  
 }  
  
 private void configureStage(Stage stage) {  
 stage.setScene(scene);  
 stage.setMinHeight(400);  
 stage.setMinWidth(500);  
 }  
  
 private VBox createThreadControlSection() {  
 threadButtonsInfo = new Label("Buttons for thread manipulating");  
  
 stopThread = new Button("Stop");  
 stopThread.setPrefWidth(80);  
 stopThread.setOnAction(event -> {  
 thread.stop();  
  
 stopThread.setDisable(true);  
 resumeThread.setDisable(false);  
 });  
  
 resumeThread = new Button("Resume");  
 resumeThread.setPrefWidth(80);  
 resumeThread.setOnAction(event -> {  
 thread.resume();  
  
 stopThread.setDisable(false);  
 resumeThread.setDisable(true);  
 });  
  
 interruptThread = new Button("Discard");  
 interruptThread.setPrefWidth(80);  
 interruptThread.setOnAction(event -> {  
 thread.interrupt();  
  
 stopThread.setDisable(true);  
 resumeThread.setDisable(true);  
 });  
  
 return new VBox(5, threadButtonsInfo, stopThread, resumeThread, interruptThread);  
 }  
  
 private VBox createCalculationControlSection() {  
 serialNumberInput = new TextField();  
 serialNumberInput.setPrefColumnCount(10);  
  
 calculationOutput = new Label();  
  
 calculate = new Button("Calculate");  
 calculate.setPrefWidth(80);  
 calculate.setOnAction(event -> {  
 SeriesCalculator seriesCalculator = new SeriesCalculator(Integer.*parseInt*(serialNumberInput.getText()));  
 thread.setCalculator(seriesCalculator);  
 calculationOutput.textProperty().bind(seriesCalculator.messageProperty());  
  
 thread.start();  
  
 stopThread.setDisable(false);  
 resumeThread.setDisable(false);  
 interruptThread.setDisable(false);  
  
 });  
  
  
 return new VBox(5, serialNumberInput, calculate, calculationOutput);  
 }  
}

ThreadManager

package sample;  
  
public class ThreadManager {  
 private SeriesCalculator calculator;  
 private Thread thread;  
  
 public void start() {  
 thread.start();  
 }  
  
 public void stop() {  
 calculator.isStop = true;  
 }  
  
 public void resume() {  
 calculator.isStop = false;  
 }  
  
 public void interrupt() {  
 thread.interrupt();  
 }  
  
 public void setCalculator(SeriesCalculator calculator) {  
 if (this.calculator != null) {  
 interrupt();  
 }  
  
 this.calculator = calculator;  
 thread = new Thread(calculator);  
 }  
}

SeriesCalculator

package sample;  
import javafx.concurrent.Task;  
  
public class SeriesCalculator extends Task {  
  
 private int seriesNumber;  
  
 public volatile boolean isStop;  
  
 SeriesCalculator(int seriesNumber)  
 {  
 this.seriesNumber = seriesNumber;  
 }  
  
 @Override  
 protected Object call() {  
 try {  
 double current = 0, prev = 1, sum = 1;  
 for (int i = 1; i <= seriesNumber; i++) {  
 while (isStop) { }  
  
 current = prev\*(-1/(double)i);  
 prev = current;  
 sum += current;  
  
 updateMessage("iteration: " + i + " value: " + sum);  
  
 Thread.*sleep*(100);  
 }  
 }  
 catch (InterruptedException e) {  
 System.*out*.println(e.getMessage());  
 }  
 return null;  
 }  
}

Результат работы:

