Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №1

По дисциплине: «Современные платформы программирования»

Выполнила:
Студентка 3 курса
Группы ПО-6
Юсковец М.А.
Проверил:
Монтик Н.С.

Брест, 2023

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

Ход работы:

Задание:

(Вариант 1)

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.

$$\sum_{k=0}^{n} \frac{1}{k!} = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!}$$

Текст программы:

Main.java

```
package com.example.spp_lab1;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
        primaryStage.setTitle("Lab 1");
        primaryStage.setScene(new Scene(root, 300, 275));
        primaryStage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

Controller.java

```
package com.example.spp_lab1;
```

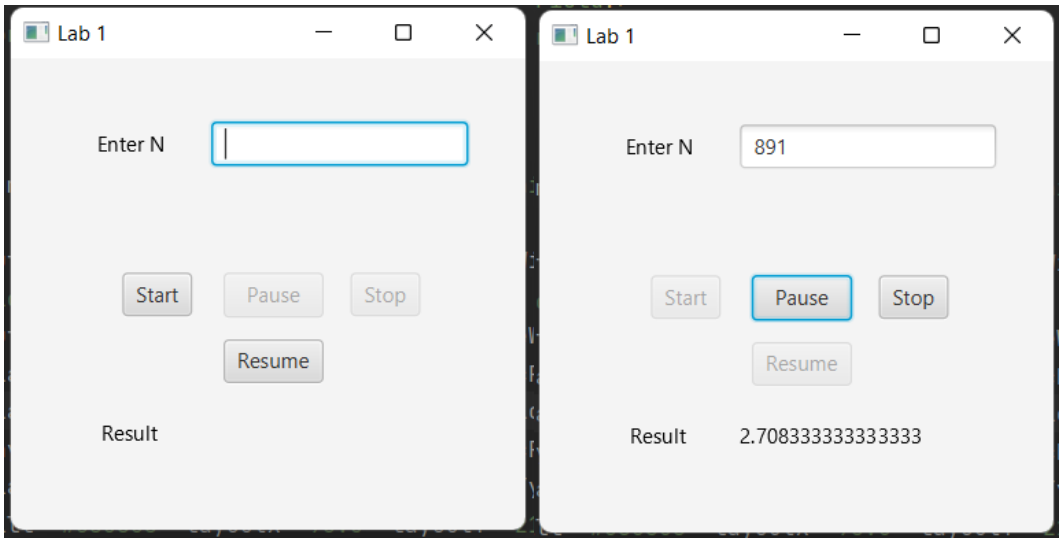
```
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
public class Controller {
    private Calculator calculator;
    private boolean isRunning = false;
    @FXML
    private TextField textFieldN;
    @FXML
    private Button startBut, pauseBut, stopBut, resumeBut;
    @FXML
    private Text resultText, errorText;
    @FXML
    void initialize() {
        pauseBut.setDisable(true);
        stopBut.setDisable(true);
    }
    @FXML
    void startClick() {
        try {
            int N = Integer.parseInt(textFieldN.getText());
            calculator = new Calculator(N, this);
            startBut.setDisable(true);
            pauseBut.setDisable(false);
            stopBut.setDisable(false);
            resumeBut.setDisable(true);
            resultText.setText("");
            errorText.setText("");
            isRunning = true;
            calculator.start();
        } catch (NumberFormatException e) {
            errorText.setText("Format error! Please, enter an integer
number.");
        }
    }
    @FXML
    synchronized void pauseClick() throws InterruptedException {
        if (isRunning) {
            resumeBut.setDisable(false);
            calculator.suspend();
            isRunning = false;
        }
    }
    @FXML
    synchronized void resumeClick() {
        if (!isRunning) {
            isRunning = true;
            calculator.resume();
            resumeBut.setDisable(true);
        }
    }
    @FXML
    void stopClick() {
        startBut.setDisable(false);
        pauseBut.setDisable(true);
        stopBut.setDisable(true);
        resumeBut.setDisable(true);
        calculator.interrupt();
    }
    void updateResult(double sum) {
        resultText.setText(Double.toString(sum));
    }
}
```

Calculator.java

```java
package com.example.spp_lab1;

public class Calculator extends Thread {
    private int N;
    private final Controller controller;
    public Calculator( int N, Controller controller) {
        this.N = N;
        this.controller = controller;
    }
    @Override
    public void run() {
        double a = 1.;
        double sum = a;
        int i = 1;
        try {
            while (i < N && !isInterrupted()) {
                controller.updateResult(sum);
                a = a*( 1.0 / i );
                sum += a;
                i++;
                synchronized (controller) {
                    controller.updateResult(sum);
                }
                Thread.sleep(500);
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            e.printStackTrace();
        }
    }
}
```

Результат программы:



Вывод: приобрели навыки написания простого оконного многопоточного приложения с использованием Java API.