МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №8

Специальность ПО5

Выполнил:
А.А. Игнатюк,
студент группы ПО-5

Проверил:
А.А. Крощенко,
ст. преп. кафедры ИИТ,
«__» _____ 2022 г.

Брест 2022

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

Вариант 5.

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.

$$5) \quad \sum_{k=1}^{n} \frac{1}{(2k-1)(2k+1)} = \frac{1}{1 \cdot 3} + \frac{1}{3 \cdot 5} + \ldots + \frac{1}{(2n-1)(2n+1)}$$

Рисунок 1 - Исходная формула.

Спецификация ввода:

Спецификация вывода:

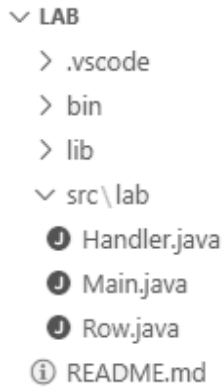

Рисунок 2 - Структура проекта.

Код программы и результаты тестирования:

Row.java 1 ×

src > lab > Row.java > ...

```java
package lab;

import javafx.concurrent.Task;

public final class Row extends Task {
    private volatile boolean running;
    private final int n;

    public Row(final int n) {
        this.running = true;
        this.n = n;
    }

    public final void start() {
        this.running = true;
    }

    public final void stop() {
        this.running = false;
    }

    @Override
    protected final Object call() {
        try {
            int previousOdd = 1, nextOdd = previousOdd + 2;
            double result = 0.0;

            for (int i = 0; i < n; ++i) {
                while (!running) {
                }

                result += 1.0 / (previousOdd * nextOdd);
                previousOdd = nextOdd;
                nextOdd += 2;

                updateMessage("Result for n = " + String.valueOf(i + 1) + " : " + String.valueOf(result));
                Thread.sleep(millis: 100);
            }
        } catch (final InterruptedException exception) {
            System.out.println(exception.getMessage());
        }

        return null;
    }
}

```

Рисунок 3 - Исходный код файла Row.java.

Handler.java ×

src > lab > Handler.java > ...

```java
package lab;

public final class Handler {
    private Row row;
    private Thread thread;

    private final boolean isSet() {
        return this.row != null && this.thread != null;
    }

    public Handler() {
        this.row = null;
        this.thread = null;
    }

    public final void start() {
        if (this.isSet()) {
            this.thread.start();
        }
    }

    public final void stop() {
        if (this.isSet()) {
            this.row.stop();
        }
    }

    public final void resume() {
        if (this.isSet()) {
            this.row.start();
        }
    }

    public final void interrupt() {
        if (this.isSet()) {
            this.thread.interrupt();
        }
    }

    public final void setRow(final Row row) {
        if (this.isSet()) {
            this.interrupt();
        }

        this.row = row;
        this.thread = new Thread(this.row);
    }
}

```

Рисунок 4 - Исходный код файла Handler.java.

Main.java

src > lab > Main.java > ...

```java
package lab;

import javafx.scene.layout.FlowPane;
import javafx.scene.Scene;
import javafx.scene.control.TextField;
import javafx.scene.control.Label;
import javafx.scene.layout.HBox;
import javafx.scene.control.Button;
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.geometry.Orientation;
import javafx.stage.Stage;

public final class Main extends Application {
    private FlowPane pane;
    private Scene scene;

    private TextField input;
    private Label output;

    private HBox threadControls;
    private HBox processControls;
    private HBox outputBox;

    private Button stopThread;
    private Button resumeThread;
    private Button interruptThread;
    private Button process;

    private Row row;
    private Handler handler;

    Run | Debug
    public final static void main(final String[] args) {
        Application.launch(args);
    }

    @Override
    public final void start(final Stage stage) {
        initThreadControls();
        initProcessControls();

        this.pane = new FlowPane(this.processControls, this.threadControls, this.outputBox);
        this.pane.setAlignment(Pos.CENTER);
        this.pane.setOrientation(Orientation.HORIZONTAL);
        this.pane.setVgap(10);
        this.pane.setHgap(10);

        this.scene = new Scene(this.pane);
        this.handler = new Handler();

        stage.setTitle("lab");
        stage.setResizable(false);
        stage.setMinHeight(320);
        stage.setMinWidth(480);
        stage.setScene(scene);
        stage.show();
    }

```

Рисунок 5 - Исходный код файла Main.java.

Продолжение рисунка 5.

```
59      private final void initThreadControls() {
60          this.stopThread = new Button("Stop");
61          this.stopThread.setPrefWidth(100);
62          this.stopThread.setDisable(true);
63
64          this.stopThread.setOnAction(event -> {
65              this.handler.stop();
66              this.stopThread.setDisable(true);
67              this.resumeThread.setDisable(false);
68          });
69
70          this.resumeThread = new Button("Resume");
71          this.resumeThread.setPrefWidth(100);
72          this.resumeThread.setDisable(true);
73
74          this.resumeThread.setOnAction(event -> {
75              this.handler.resume();
76              this.stopThread.setDisable(false);
77              this.resumeThread.setDisable(true);
78          });
79
80          this.interruptThread = new Button("Interrupt");
81          this.interruptThread.setPrefWidth(100);
82          this.interruptThread.setDisable(true);
83
84          this.interruptThread.setOnAction(event -> {
85              this.handler.interrupt();
86              this.stopThread.setDisable(true);
87              this.resumeThread.setDisable(true);
88              this.interruptThread.setDisable(true);
89              this.process.setDisable(false);
90          });
91
92          this.threadControls = new HBox(5, this.stopThread, this.resumeThread, this.interruptThread);
93      }
94
95      private final void initProcessControls() {
96          this.input = new TextField();
97          this.output = new Label();
98
99          this.process = new Button("Process");
100         this.process.setOnAction(event -> {
101             try {
102                 int n = Integer.parseInt(this.input.getText());
103
104                 this.row = new Row(n);
105                 this.handler.setRow(this.row);
106                 this.output.textProperty().bind(this.row.messageProperty());
107                 this.handler.start();
108
109                 this.stopThread.setDisable(false);
110                 this.resumeThread.setDisable(true);
111                 this.interruptThread.setDisable(false);
112                 this.process.setDisable(true);
113             } catch (final NumberFormatException exception) {
114                 System.out.println(exception.getMessage());
115             }
116         });
117
118         this.processControls = new HBox(5, this.input, this.process);
119         this.outputBox = new HBox(5, this.output);
120     }
121 }
122
```

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