**Ministerul Educaţiei, Culturii și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Departamentul Informatică și Ingineria Sistemelor**

**Disciplina: Sisteme de Operare**

**RAPORT**

**Lucrarea de laborator nr. 1**

Au efectuat:

st. gr. CR-211 L. Mihail  
M.Victor

P.Tudor

A verificat:

Rotaru Lilia

Chişinău – 2024

* 1. **Probleme propuse spre realizare:**

Creaţi o aplicaţie cu unu sau mai multe taimere utilizînd clasele Timer și TimerTask în diferite moduri:

**1)Să reacționeze la un anumit interval de timp.(M.Victor)**

**2)Să reacționeze la un anumit timp.**

**3)Să reacționeze cu o perioadă indicată.**

Utilizați clasă **anonimă**.

**3)Să reacționeze cu o perioadă indicată**

import java.awt.\*;  
import java.util.Timer;  
import java.util.TimerTask;  
import javax.swing.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 // Creating a Swing frame  
 JFrame frame = new JFrame("Timer App");  
 frame.setSize(300, 150); // Reduced height to make it look better  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 // Creating a panel to display messages and the button  
 JPanel panel = new JPanel(new GridBagLayout());  
 GridBagConstraints constraints = new GridBagConstraints();  
 constraints.gridx = 0;  
 constraints.gridy = 0;  
 constraints.insets = new Insets(10, 10, 10, 10); // Adding padding  
  
 JLabel label = new JLabel("Acțiune executată la fiecare 2 secunde");  
 panel.add(label, constraints);  
  
 constraints.gridy = 1;  
 JButton button = new JButton("Adaugă secunde");  
 panel.add(button, constraints);  
  
 // Centering the panel  
 frame.add(panel, BorderLayout.*CENTER*);  
  
 // Displaying the frame in the center of the screen  
 frame.setLocationRelativeTo(null);  
 frame.setVisible(true);  
  
 // Creating a Timer object to update the button  
 Timer buttonTimer = new Timer();  
  
 // Scheduling the task to update the button every 2 seconds  
 buttonTimer.schedule(new TimerTask() {  
 int seconds = 2; // Number of seconds  
  
 @Override  
 public void run() {  
 button.setText("" + seconds + "");  
 seconds += 2;  
 }  
 }, 0, 2000); // Scheduling updates every 2 seconds  
  
 // Adding an event listener for the button  
 button.addActionListener(e -> {  
 String text = label.getText();  
 String[] parts = text.split(" ");  
 int currentSeconds = Integer.*parseInt*(parts[parts.length - 2]);  
 currentSeconds += 2;  
 label.setText("Acțiune executată la fiecare 2 secunde");  
 });  
  
 // Creating and starting a separate execution thread to keep the application active  
 Thread thread = new Thread(() -> {  
 while (true) {  
 try {  
 Thread.*sleep*(100); // Waiting for 100 milliseconds  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
 thread.start();  
 }  
}

Concluzie:

În cadrul acestei lucrări de laborator, am avut oportunitatea de a experimenta cu utilizarea claselor Timer și TimerTask din Java pentru a crea aplicații care să reacționeze la evenimente periodice de timp. Am implementat mai multe scenarii diferite, fiecare abordând o modalitate distinctă de utilizare a acestor clase.

Am învățat că clasa Timer poate fi folosită pentru a programa sarcini care să se execute la intervale regulate de timp sau la un moment specific, iar TimerTask reprezintă sarcina care va fi executată la fiecare interval specificat. Acest lucru ne permite să automatizăm anumite acțiuni în aplicații Java, cum ar fi actualizarea unei etichete la fiecare câteva secunde sau efectuarea unei anumite acțiuni la o anumită oră.

De asemenea, am avut ocazia să lucrăm cu interfața grafică Swing pentru a crea o aplicație simplă care să utilizeze aceste funcționalități de programare a timpului.

În final, această lucrare de laborator ne-a oferit o înțelegere mai profundă a modului în care putem gestiona timpul în aplicațiile noastre Java, folosind clasele Timer și TimerTask pentru a programa și executa sarcini la intervale specifice.