Lucru de laborator № 4

Tema lucrării laboratoare:

Elaborarea unui mecanism de planificare a activităţii sistemului de operare.

Partea teoretică:

Clasa **Timer** din pachetul de bază java.util acționează ca un programator de sarcini. O instanță a acestei clase programează sarcinile care urmează să fie executate la un anumit moment. Sarcinile pot fi programate pentru execuție unică sau multiplă.

Sarcinile programate în sine sunt implementate prin intermediul clasei **TimerTask.** Aceasta este o clasă abstractă și are, de asemenea, o metodă **run()** derivată din interfața **Runnable**, care, prin suprapunerea metodelor, descrie algoritmul sarcinii programate.

Metode ale clasei Timer pentru programarea sarcinilor:

- **schedule(TimerTask task, long delay)** - sarcina este programată pentru execuție după perioada în milisecunde trecută în parametrul delay.
- **schedule(TimerTask task, long delay, long period)** – sarcina este programată pentru execuție după perioada în milisecunde trecută în parametrul delay. Apoi, sarcina se repetă periodic - la fiecare perioadă de milisecunde.
- **schedule(TimerTask task, Date when)** - sarcina este programată pentru momentul specificat în parametrul when.
- **schedule(TimerTask task, Date when, long period)** – sarcina este programată pentru timpul specificat în parametrul when. Apoi, sarcina se repetă periodic - la fiecare perioadă de milisecunde.
- **scheduleAtFixedRate(TimerTask task, long delay, long period)** - sarcina este programată pentru execuție după perioada în milisecunde trecută în parametrul delay. Apoi, sarcina se repetă periodic - la fiecare milisecundă de perioadă. Timpul fiecărei repetări este stabilit în raport cu prima execuție.
- **scheduleAtFixedRate(TimerTask task, Date when, long period)** - sarcina este programată pentru execuție în timpul specificat în parametrul when. Sarcina este apoi repetată periodic - la fiecare milisecunde period. Momentul fiecărei repetări este stabilit în raport cu prima execuție.

```
import java.util.Timer;
import java.util.TimerTask;

class AdditionTask extends TimerTask
{
	int a;
	int b;
	AdditionTask(int i, int j)
	{
		this.a = i;
		this.b = j;
	}

	@Override
	public void run()
	{
		System.out.println("Performing the Addition(takes 2000ms)");
		try {
			Thread.sleep(2000);
		}
		catch (InterruptedException e) {
			e.printStackTrace();
		}
		System.out.println("The sum is: " + (a+b));
	}
}
public class Demo
{
	public static void main(String[] args)
	{
		AdditionTask t1 = new AdditionTask(10, 20);
		AdditionTask t2 = new AdditionTask(40, 60);
		Timer timer = new Timer();
		timer.schedule(t1, 0);//Scheduling the first task without any delay
		timer.schedule(t2, 5000);//Scheduling the second task after a delay
of 5000ms
	}
}
```

Documentația oficială pentru clasa Timer:

https://docs.oracle.com/javase/8/docs/api/java/util/Timer.html

Documentația oficială pentru clasa TimerTask:

https://docs.oracle.com/javase/8/docs/api/java/util/TimerTask.html

Sarcina principală:

Sarcina principală este de a crea un programator de sarcini în care anumite sarcini vor fi programate pentru a fi executate la un anumit moment (poate chiar într-un anumit interval). Implementați un algoritm prin care fiecare membru al echipei realizează o sarcină programată separată, iar aceste sarcini vor fi trimise la planificatorul de sarcini.

La realizarea laboratorului trebuie de creat 3 sau mai multe Timer. Creați Timer în moduri diferite. (clasă anonimă, clasă separată, direct în main etc.), trebuie să vă gândiți la interacțiunea dintre Timer, independenta de sarcina Timer-ilor.

- **Opțiunea usoara:** realizati Timer-ile.
- **Opțiunea de mijloc:** realizati timerile utilizand diferite moduri de creare.
- **Opțiune dificilă:** realizati timerile utilizand diferite moduri de creare si interactiunea intre ele.

Opțiuni de realizare:

- Un model al unui sistem distribuit de execuție a sarcinilor. În acest sistem, există o clasă de control central și mai multe clase de lucrători. Fiecare clasă de lucrători va efectua o sarcină specifică la intervale diferite și va raporta progresul acesteia controlorului.
- Simulare a planificării și fabricării unui obiect pe o imprimantă 3D. Există un model de imprimantă de bază care trebuie să creeze cronometre pentru programarea fabricării unui obiect pe imprimantă la un anumit moment, un cronometru responsabil pentru procesul de imprimare și procesul de procesare ulterioară a obiectului.
- Simulare de triatlon. Sarcina este de a crea mai multe cronometre care se activează pe baza cronometrelor anterioare, fiecare cronometru depinzând de secțiunea de traseu pe care concurentul o parcurge. Trebuie să creați un meniu în care trebuie să specificați viteza participanților pentru diferitele rezultate ale cronometrelor. De asemenea, fiecare participant anunță cantitatea de distanță parcursă.
- Simulare a muncii într-o mare companie IT. În sistem este necesar să se ia în considerare interacțiunea dintre diferite departamente și dependența de timp a procesării unui proiect comun. Este posibilă realizarea mai multor proiecte de către o singură companie.

- Realizarea principiului de programare a publicării textului/mediului. Publicațiile programate ar trebui să fie primite de la utilizatorul final și trimise către planificatorul de sarcini.
- Simulare a lucrărilor de arhitectură și a construcției ulterioare. Sarcina este de a crea mai multe cronometre: cronometrul responsabil pentru crearea arhitecturii clădirii, construirea cadrului clădirii și cronometrul pentru vânzarea construcției finite. Este necesar să se țină cont de faptul că, atunci când se construiește o clădire, unele etaje pot fi construite înaintea altora.

După finalizarea activității, scrieți un raport, care trebuie să includă - numele, prenumele, grupul, tema de lucru, varianta de implementare a sarcinii, o scurtă descriere a implementării sarcinii, un link către codul sursă pe GitHub. Împingeți codul sursă în ramura dvs. în depozitul corespunzător - https://github.com/FCIM-SO/Practice-Work-RO. Salvați raportul în format PDF și trimiteți-l la ELSE - https://else.fcim.utm.md/mod/assign/view.php?id=43456.