Лабораторная работа № 4

Тема работы:

Проектирование механизма планировщика задач.

Теоретическая часть:

Класс **Timer** из базового пакета **java.util** выполняет роль планировщика задач. В экземпляре этого класса планируется задачи, которые должны быть выполнены в определенный момент времени. Задачи могут быть запланированы на одноразовое или многоразовое выполнение.

Сами задачи с запланированным выполнением реализуются через класс **TimerTask**. Этот абстрактный класс, а также он имеет метод **run()** полученный из интерфейса **Runnable**, в котором, через переопределение метода, описывается алгоритм запланированной задачи.

Методы класса Timer для планирования задач:

- **schedule(TimerTask task, long delay)** - задание планируется к выполнению через период в миллисекундах, переданный в параметре delay
- **schedule(TimerTask task, long delay, long period)** – задание планируется к выполнению через период в миллисекундах, переданный в параметре delay. Затем задание повторяется повторно периодически - каждые period миллисекунд
- **schedule(TimerTask task, Date when)** - задание планируется на время, указанное в параметре when
- **schedule(TimerTask task, Date when, long period)** – задание планируется на время, указанное в параметре when. Затем задание выполняется повторно периодически - каждые period миллисекунд
- **scheduleAtFixedRate(TimerTask task, long delay, long period)** - задание планируется к выполнению через период в миллисекундах, переданный в параметре delay. Затем задание выполняется повторно периодически - каждые period миллисекунд. Время каждого повтора задаётся относительно первого запуска.
- **scheduleAtFixedRate(TimerTask task, Date when, long period)** - задание планируется к выполнению на время, указанное в параметре when. Задание затем выполняется повторно периодически - каждые period миллисекунд. Время каждого повтора задаётся относительно первого запуска.

```
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
	}
}
```

Официальная документация на класс Timer:

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

Официальная документация на класс TimerTask:

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

Основное задание:

Основная задача состоит в создании планировщика задач, в котором будут запланированы определенные задачи на выполнение в определенное время (возможно даже, в течение определенного интервала). Реализовать алгоритм, в котором задействованы как минимум 3 экземпляра класса Timer, где каждый член команды реализует отдельную запланированную задачу и эти задачи будут отправлены в планировщик задач.

- **Средний вариант:** реализовать экземпляры класса Timer всеми тремя способами – отдельный класс, прямо в main-е и анонимный класс.
- **Сложный вариант:** реализовать экземпляры класса Timer всеми тремя способами, а также их взаимосвязь.

Варианты для реализации задания:

- Модель распределенной системы выполнения задач. В этой системе есть центральный класс контроллера и несколько рабочих классов. Каждый рабочий класс будет выполнять определенную задачу через различные промежутки времени и сообщать о ходе ее выполнения контроллеру.
- Симуляция планирования и изготовления объекта на 3D принтере. Существует основная модель принтера, которой необходимо создать таймеры для планирования изготовления объекта на принтере в определённое время, таймер, отвечающий за процесс распечатывания и процесс последующей обработки объекта.
- Симуляция триатлона. Задача заключается в создании нескольких таймеров активизирующегося на базе предыдущих, каждый таймер зависит от участка пути, который проходит участник. Нужно создать меню, в котором нужно указать скорость участников для различных результатов таймеров. Также каждый участник афиширует количество пройденного пути.
- Симуляция работы в крупной IT компании. В системе необходимо учесть взаимодействие между разными отделами и зависимость во времени обработки общего проекта. Возможна реализации множества проектов одной компанией.
- Реализовать принцип планирования публикации текста/медиа. Запланированные публикации должны быть получены от конечного пользователя и отправлены в планировщик задач.
- Симуляция работы архитектора и последующей постройки. Задача заключается в создании нескольких таймеров: отвечающий за создание архитектуры здания,

построении каркаса здания и таймера для продажи готовой конструкции. Нужно учесть, что при постройке здания, некоторое этажи могут быть построены ранее других.

По завершению работы, составьте отчет, в котором должно быть – Ваша фамилия, имя, группа, тема работы, Ваш вариант для реализации задания, краткое описание реализации задания, ссылку на исходный код на GitHub. Исходный код push-ите в вашу ветку в соответствующем репозитории - https://github.com/FCIM-SO/Practice-Work-RU. Сохранить отчет в формате PDF и отправить на ELSE - https://else.fcim.utm.md/mod/assign/view.php?id=43456.