МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Вычислительные системы и технологии»

**ОТЧЕТ**

по лабораторной работе №1

по дисциплине

Шаблоны проектирования программного обеспечения

ВЫПОЛНИЛИ:

Студенты группы \_\_\_22-ИВТ-2\_\_\_\_\_\_\_\_

(шифр группы)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Киселев А. Р.

(подпись) (фамилия, и.,о.)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Назаров А. И.

(подпись) (фамилия, и.,о.)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРОВЕРИЛ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жевнерчук Д. В.

(подпись) (фамилия, и.,о.)

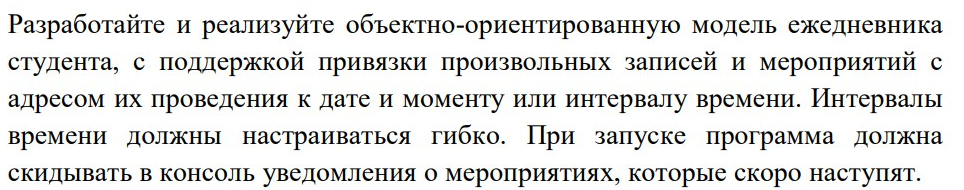
Работа защищена «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Нижний Новгород 2024

**Задание**

**Вариант 17**



**Описание проекта**

В проекте создается два потока, Генератор и Читатель событий. У Генератора частота равна 5 раз в секунду, а у Читателя 200 раз в секунду. Однако, несмотря на это, читатель будет всегда ждать пока генератор в реализацию одноместного контейнера добавит объект.

Время выполнения основного потока ограничено до 1 секунды, следовательно Читатель должен будет вывести сообщения 4 или 5 раз.

**Приложение 1**

**Программный код**

src/main/java/org/lab03/Event.java:

package org.lab03;

import org.springframework.stereotype.Component;

import java.time.LocalDateTime;

@Component

public class Event {

private String name = "unknown";

private String place = "unknown";

private LocalDateTime dateTime = LocalDateTime.MIN;

public Event() {

}

public Event(String name, String place, LocalDateTime dateTime) {

this.name = name;

this.place = place;

this.dateTime = dateTime;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPlace() {

return place;

}

public void setPlace(String place) {

this.place = place;

}

public LocalDateTime getDateTime() {

return dateTime;

}

public void setDateTime(LocalDateTime dateTime) {

this.dateTime = dateTime;

}

public String toString() {

return "Name: " + getName() + "; Place: " + getPlace() + "; DateTime: " + getDateTime();

}

}

src/main/java/org/lab03/EventHolder.java:

package org.lab03;

public class EventHolder {

private Event event;

public void set(Event event) {

synchronized (this) {

this.event = event;

notifyAll();

}

}

public Event get() throws InterruptedException {

synchronized (this) {

if (event == null) {

wait();

}

}

Event res = event;

event = null;

return res;

}

}

src/main/java/org/lab03/Logger.java:

package org.lab03;

import org.lab03.iface.ILogger;

import org.lab03.iface.IPrinter;

import java.time.LocalDateTime;

import java.util.HashSet;

public class Logger implements ILogger {

private String format = "${time}:${level}:${message}";

private HashSet<IPrinter> subscribers = new HashSet<>();

public String getFormat() {

return format;

}

public void setFormat(String format) {

this.format = format;

}

public void subscribe(IPrinter printer) {

subscribers.add(printer);

}

public void unsubscribe(IPrinter printer) {

subscribers.remove(printer);

}

@Override

public void debug(String message) {

log("DEBUG", message);

}

@Override

public void info(String message) {

log("INFO", message);

}

@Override

public void warning(String message) {

log("WARNING", message);

}

private void log(String level, String message) {

String formatted = getFormat();

formatted = formatted.replace("${time}", LocalDateTime.now().toString());

formatted = formatted.replace("${level}", level);

formatted = formatted.replace("${message}", message);

for (IPrinter printer : subscribers) {

printer.handle(formatted);

}

}

}

src/main/java/org/lab03/Main.java:

package org.lab03;

import org.lab03.thread.Generator;

import org.lab03.thread.Reader;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

public static void main(String[] args) throws InterruptedException {

ClassPathXmlApplicationContext ctx = new ClassPathXmlApplicationContext("app.xml");

Logger logger = ctx.getBean("logger", Logger.class);

logger.subscribe(System.out::println);

Thread th1 = new Thread(new Generator(ctx, 5));

Thread th2 = new Thread(new Reader(ctx, 200));

th1.start();

th2.start();

Thread.sleep(1000);

th1.interrupt();

th2.interrupt();

ctx.close();

}

}

src/main/java/org/lab03/Profiler.java:

package org.lab03;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

import org.lab03.iface.ILogger;

import org.springframework.stereotype.Component;

import java.time.LocalTime;

@Aspect

@Component

public class Profiler {

private ILogger logger;

public Profiler(ILogger logger) {

this.logger = logger;

}

@Around("execution(\* org.lab03.Event.\*(..))")

private Object profileEvent(ProceedingJoinPoint jp) throws Throwable {

LocalTime time = LocalTime.now();

Object r = jp.proceed();

logger.debug("profileEvent( "

+ jp.getSignature() + " -> "

+ (LocalTime.now().toNanoOfDay() - time.toNanoOfDay() + " ns")

+ ")");

return r;

}

}

src/main/java/org/lab03/iface/ILogger.java:

package org.lab03.iface;

public interface ILogger {

void debug(String message);

void info(String message);

void warning(String message);

}

src/main/java/org/lab03/iface/IPrinter.java:

package org.lab03.iface;

public interface IPrinter {

void handle(String message);

}

src/main/java/org/lab03/thread/AbstractThread.java:

package org.lab03.thread;

import org.springframework.context.support.ClassPathXmlApplicationContext;

abstract public class AbstractThread implements Runnable {

protected final ClassPathXmlApplicationContext ctx;

protected final long rate;

public AbstractThread(ClassPathXmlApplicationContext ctx, long rate) {

this.ctx = ctx;

this.rate = rate;

}

@Override

public void run() {

while (true) {

try {

tick();

Thread.sleep(1000 / rate);

} catch (InterruptedException e) {

break;

}

}

}

abstract protected void tick() throws InterruptedException;

}

src/main/java/org/lab03/thread/Generator.java:

package org.lab03.thread;

import org.lab03.Event;

import org.lab03.EventHolder;

import org.lab03.Logger;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.time.LocalDateTime;

import java.util.Random;

public class Generator extends AbstractThread {

public Generator(ClassPathXmlApplicationContext ctx, long rate) {

super(ctx, rate);

}

@Override

protected void tick() {

Random random = new Random();

Logger logger = ctx.getBean("logger", Logger.class);

EventHolder holder = ctx.getBean("holder", EventHolder.class);

Event event = ctx.getBean("event", Event.class);

event.setName("Зачет");

event.setPlace("Ауд. " + random.nextInt(6999));

event.setDateTime(LocalDateTime.now().plusDays(random.nextInt(20)));

logger.info("Generator: " + event);

holder.set(event);

}

}

src/main/java/org/lab03/thread/Reader.java:

package org.lab03.thread;

import org.lab03.Event;

import org.lab03.EventHolder;

import org.lab03.Logger;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Reader extends AbstractThread {

public Reader(ClassPathXmlApplicationContext ctx, long rate) {

super(ctx, rate);

}

@Override

protected void tick() throws InterruptedException {

Logger logger = ctx.getBean("logger", Logger.class);

EventHolder holder = ctx.getBean("holder", EventHolder.class);

Event event = holder.get();

if (event == null) {

logger.warning("Reader: event is null");

return;

}

logger.info("Reader: " + event);

}

}

**Приложение 2**

**Результаты тестирования**

2024-05-25T01:51:53.543376900 - INFO - Generator: Name: Зачет; Place: Ауд. 5639; DateTime: 2024-05-26T01:51:53.540377200

**2024-05-25T01:51:53.544377 - INFO - Reader: Name: Зачет; Place: Ауд. 5639; DateTime: 2024-05-26T01:51:53.540377200**

2024-05-25T01:51:53.746891800 - INFO - Generator: Name: Зачет; Place: Ауд. 5020; DateTime: 2024-05-29T01:51:53.746891800

**2024-05-25T01:51:53.746891800 - INFO - Reader: Name: Зачет; Place: Ауд. 5020; DateTime: 2024-05-29T01:51:53.746891800**

2024-05-25T01:51:53.948891600 - INFO - Generator: Name: Зачет; Place: Ауд. 6097; DateTime: 2024-05-26T01:51:53.948891600

**2024-05-25T01:51:53.948891600 - INFO - Reader: Name: Зачет; Place: Ауд. 6097; DateTime: 2024-05-26T01:51:53.948891600**

2024-05-25T01:51:54.149927100 - INFO - Generator: Name: Зачет; Place: Ауд. 4868; DateTime: 2024-05-28T01:51:54.149927100

**2024-05-25T01:51:54.149927100 - INFO - Reader: Name: Зачет; Place: Ауд. 4868; DateTime: 2024-05-28T01:51:54.149927100**

2024-05-25T01:51:54.351145900 - INFO - Generator: Name: Зачет; Place: Ауд. 2490; DateTime: 2024-06-13T01:51:54.351145900

**2024-05-25T01:51:54.351145900 - INFO - Reader: Name: Зачет; Place: Ауд. 2490; DateTime: 2024-06-13T01:51:54.351145900**

Process finished with exit code 0