Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационные системы и анализ данных

Центр программной инженерии

ОТЧЁТ

к лабораторной работе №2

по дисциплине

|  |
| --- |
| «Технологии разработки программных комплексов» |
| Распараллеливание потоков |

наименование темы

Вариант №28

Выполнил cтудент ИСТБ-21-2 Н.В. Зимин

номер группы подпись И. О. Фамилия

дата

Проверил Доцент З.А. Бахвалова

Должность подпись И. О. Фамилия

дата

Иркутск – 2024

# Содержание

[1 Постановка задачи 3](#_Toc164936456)

[2 UML-диаграмма классов 4](#_Toc164936457)

[3 Результаты тестирования работы программы 6](#_Toc164936458)

[4 Исходный код 7](#_Toc164936459)

[5 Примеры использования потоков 9](#_Toc164936460)

[Список использованных источников 10](#_Toc164936461)

# 1 Постановка задачи

Все варианты заданий подразумевают использование промежуточных буферов, представляющих собой динамические массивы. Максимальный размер буферов - N чисел. N определяется для каждого варианта. Потоки, помещающие числа в буферы, следят за переполнение буферов. Потоки, извлекающие числа из буферов, могут производить данную операцию в произвольный момент времени вне зависимости от того, заполнен ли буфер полностью или нет.

Выбранный язык программирования должен поддерживать объектно-ориентированную парадигму. Выбор остального окружения производится студентом самостоятельно.

Индивидуальное задание:

Значения констант и реализуемые потоками функции:

N = 2;

Первый поток – генерирует в первый буфер 30 случайных чисел из интервала от -1 до 1;

Второй поток – извлекает числа из буфера, начиная с первого и возводит их в квадрат. Результат выводится на экран.

# 2 UML-диаграмма классов

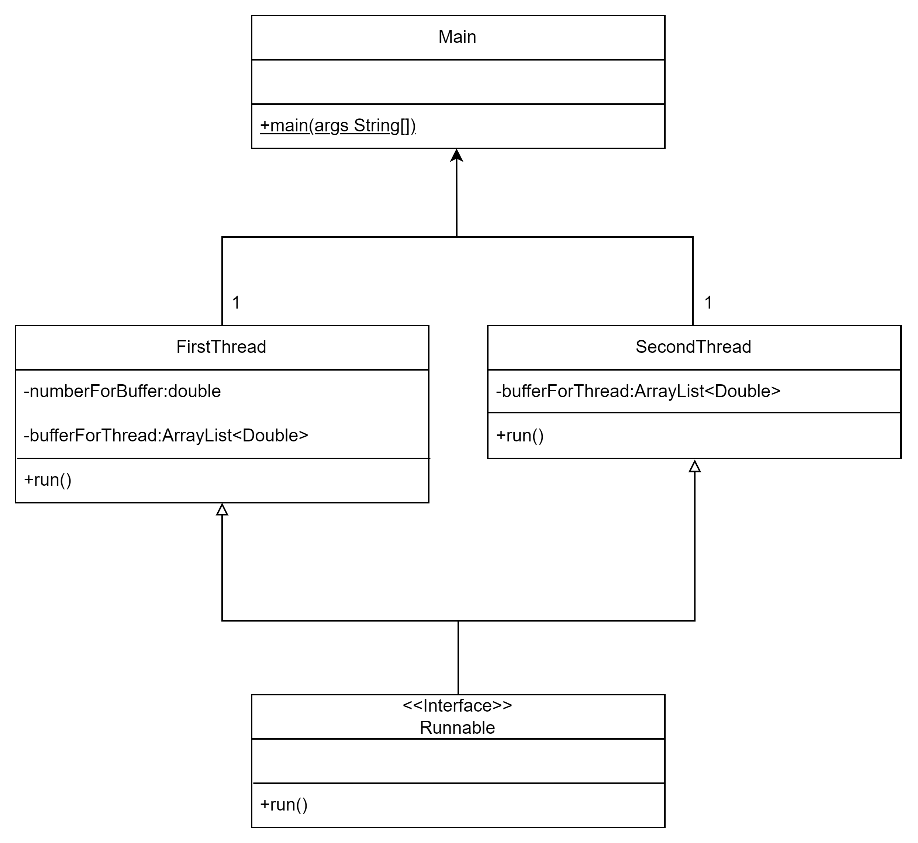


Рисунок 1 – Диаграмма классов программы

1. Main – начальный класс, точка входа в программу.
2. FirstThread – класс, наследующий интерфейс Runnable и реализующий первый поток, добавляющий числа в буфер.
   1. private double numberForBuffer – поле класса double, реализующее рандомное число для буфера.
   2. private ArrayList<Double> bufferForThread – поле класса ArrayList<>, реализующее буфер, в который поток добавляет числа.
   3. public void run() – метод интерфейса Runnable, реализующий работу потока и генерирующий числа в диапазоне от -1 до 1 для буфера buffer.
3. SecondThread – класс, наследующий интерфейс Runnable и реализующий второй поток, извлекающий числа из буфера, возводящий их в квадрат и выводящий результат на экран.
   1. private double numberFromBuffer – поле класса double, реализующее извлекаемое из буфера число.
   2. private ArrayList<Double> bufferForThread – поле класса ArrayList<>, реализующее буфер, из которого поток извлекает числа.
   3. public void run() – метод интерфейса Runnable, реализующий работу потока и извлекающий из буфера число для возведения в квадрат и вывода результата на экран.
4. Runnable – интерфейс, функционал которого реализуется классами FirstThread и SecondThread.
   1. public abstract void run() – метод, реализуемый классами-наследниками.

# 3 Результаты тестирования работы программы

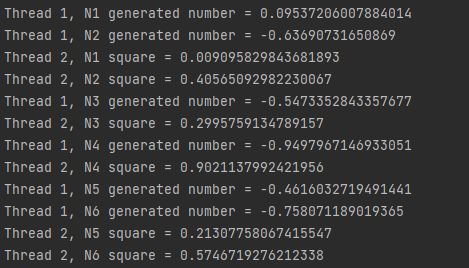


Рисунок 2 – результат работы программы

# 4 Исходный код

Код класса Main

|  |
| --- |
| import java.util.ArrayList;  public class Main {  public static void main(String[] args) {  ArrayList<Double> bufferForThread = new ArrayList<>(2);  FirstThread firstThread = new FirstThread(bufferForThread);  Thread thread1 = new Thread(firstThread);  thread1.start();  SecondThread secondThread = new SecondThread(bufferForThread);  Thread thread2 = new Thread(secondThread);  thread2.start();  }  } |

Код класса FirstThread

import java.util.ArrayList;

import java.util.concurrent.ThreadLocalRandom;

public class FirstThread implements Runnable{

private double numberForBuffer;

private ArrayList<Double> bufferForThread;

public FirstThread(ArrayList<Double> bufferForThread) {

this.bufferForThread = bufferForThread;

}

@Override

public void run() {

for (int i = 0; i < 30; i++) {

numberForBuffer = ThreadLocalRandom.current().nextDouble(-1, 1);

synchronized (bufferForThread)

{

while (bufferForThread.size() >= 2)

try {

bufferForThread.wait();

} catch (InterruptedException e) {}

bufferForThread.add(numberForBuffer);

System.out.println("Thread 1, N" + (i+1) + " generated number = " + numberForBuffer);

bufferForThread.notify();

}

}

Thread.currentThread().interrupt();

}

}

Код класса SecondThread

import java.util.ArrayList;

public class SecondThread implements Runnable{

private ArrayList<Double> bufferForThread;

public SecondThread(ArrayList<Double> bufferForThread) {

this.bufferForThread = bufferForThread;

}

@Override

public void run() {

for (int i = 0; i < 30; i++) {

synchronized (bufferForThread) {

while(bufferForThread.isEmpty()) {

try {

bufferForThread.wait();

} catch (InterruptedException e) {}

}

System.out.println("Thread 2, N" + (i+1) + " square = " + Math.pow(bufferForThread.get(0), 2) );

bufferForThread.remove(0);

bufferForThread.notify();

}

}

Thread.currentThread().interrupt();

}

}

Ссылка на полный проект <https://github.com/CrusaderKleptoman/trainThread.git>

# 5 Примеры использования потоков

GIMPS (Great Internet Mersenne Prime Search) — широкомасштабный проект добровольных вычислений по поиску простых чисел Мерсенна.

Как следует из названия, целью проекта GIMPS является поиск новых простых чисел Мерсенна. Самое большое известное на данный момент простое число было найдено в рамках проекта GIMPS 7 декабря 2018 года и состоит из 24 862 048 десятичных цифр. Более того, 15 предыдущих рекордов также были установлены участниками GIMPS. Причина кроется в наличии эффективного (детерминированного) критерия их простоты, носящего имя Люка — Лемера. Для поиска простых чисел Мерсенна сервер GIMPS раздаёт клиентам простые «экспоненты» p для проверки числа на простоту тестом Люка — Лемера.

# Список использованных источников

1. 1. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. Н.Новгород: ННГУ, 2000, 121 с. (2 изд. 2003).
2. ИНТУИТ. Параллельное программирование URL: https://www.intuit.ru/studies/courses/1110/153/info
3. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. — СПб: БХВПетербург, 2002. — 608 с.