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 №8

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](#_Toc163117882)

[2 Характеристика паттерна 4](#_Toc163117883)

[3 Шаги реализация паттерна 5](#_Toc163117884)

[4 UML-диаграмма классов 6](#_Toc163117885)

[5 Результаты тестирования 8](#_Toc163117886)

[6 Исходный код 9](#_Toc163117887)

[Список использованных источников 12](#_Toc163117888)

# 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

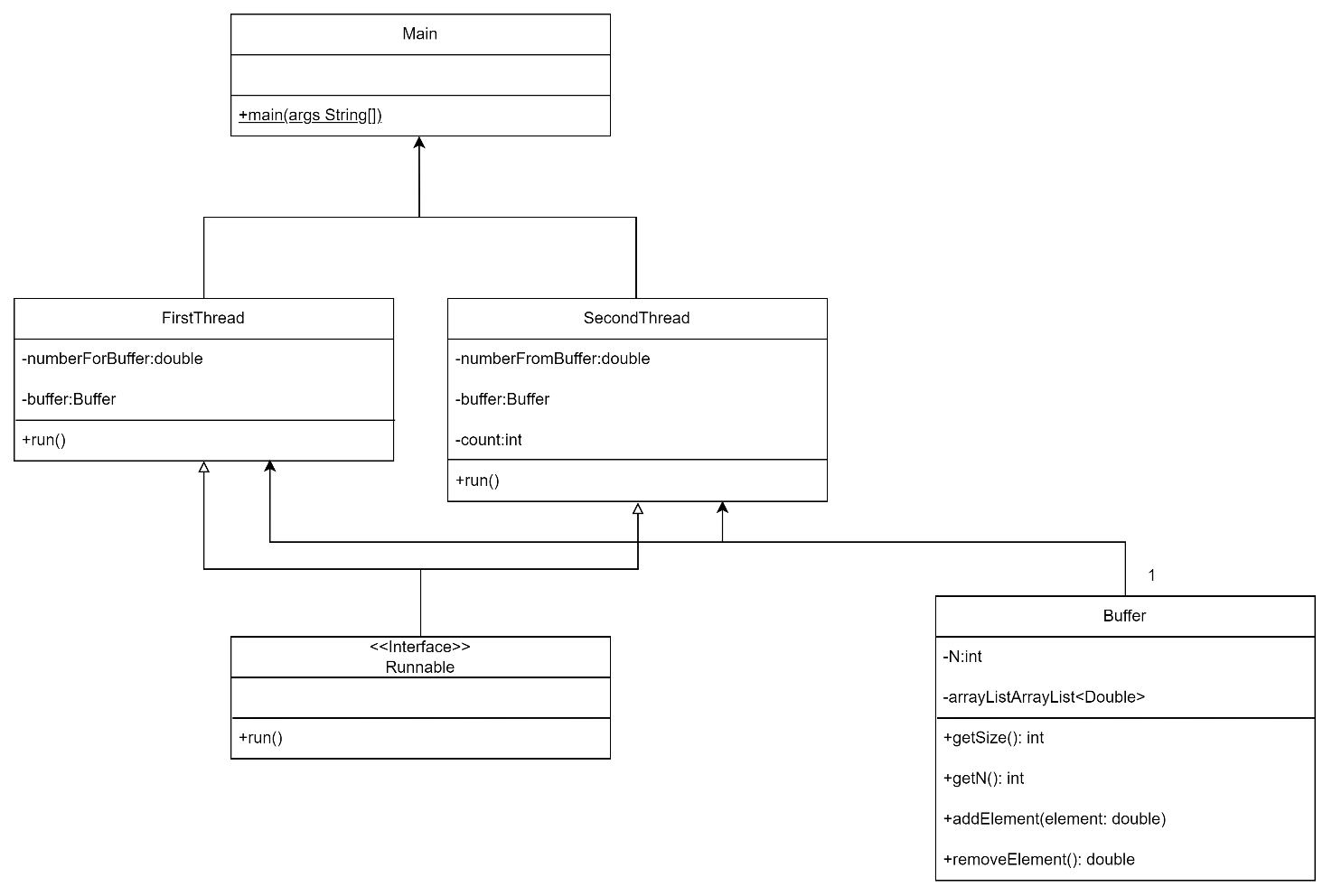


Рисунок 1 – Диаграмма классов программы

1. Main – начальный класс, точка входа в программу.
2. Menu – класс, реализующий интерфейс программы в консоли.
   1. public static void mainMenu() – метод, отображающий главное меню программы и реализующий взаимодействие с объектом класса DataAnalysis.
3. 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 Buffer buffer – поле класса Buffer, 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.
4. 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 Buffer buffer – поле класса Buffer, 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.
5. Buffer – класс, реализующий буфер с которым взаимодействуют потоки.
   1. private int N – поле класса int, реализующее ограничение вместимости буфера.
   2. private ArrayList<Double> arrayList – поле класса ArrayList<>, реализующее динамический массив в который добавляются числа.
   3. public int getSize() – метод, возвращающий текущее количество элементов в массиве arrayList.
   4. public int getN() – метод, возвращающий ограничение вместимости буфера в виде поля N.
   5. public void addElement(double element) – метод, добавляющий в массив arrayList новый элемент.
   6. public double removeElement() – метод, возвращающий первый элемент массива arrayList и удаляющий его из массива.
6. Runnable – интерфейс, функционал которого реализуется классами FirstThread и SecondThread.
   1. public abstract void run() – метод, реализуемый классами-наследниками.

# 3 Результаты тестирования работы программы

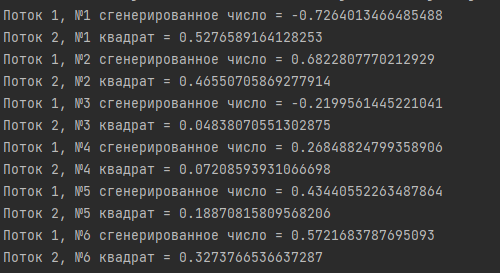


Рисунок 2 – результат работы программы

# 4 Исходный код

Код класса Main

|  |
| --- |
| public class Main {  public static void main(String[] args) {  Buffer buffer = new Buffer();  FirstThread firstThread = new FirstThread(buffer);  SecondThread secondThread = new SecondThread(buffer);  Thread thread1 = new Thread(firstThread);  Thread thread2 = new Thread(secondThread);  thread1.start();  thread2.start();  }  } |

Код класса FirstThread

import java.util.concurrent.ThreadLocalRandom;

public class FirstThread implements Runnable{

double numberForBuffer;

Buffer buffer;

public FirstThread(Buffer buffer) {

this.buffer = buffer;

}

@Override

public void run() {

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

synchronized (buffer)

{

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

if (buffer.getSize() >= buffer.getN()) {

i--;

continue;

}

buffer.addElement(numberForBuffer);

System.out.println("Поток 1, №" + (i+1) + " сгенерированное число = " + numberForBuffer);

}

}

}

}

Код класса SecondThread

public class SecondThread implements Runnable{

Buffer buffer;

double numberFromBuffer;

int count = 1;

public SecondThread(Buffer buffer) {

this.buffer = buffer;

}

@Override

public void run() {

while(true) {

if (buffer.getSize() != 0) {

numberFromBuffer = buffer.removeElement();

System.out.println("Поток 2, №" + count++ + " квадрат = " + numberFromBuffer \* numberFromBuffer);

}

}

}

}

Код класса Buffer

import java.util.ArrayList;

public class Buffer {

private int N = 2;

private ArrayList<Double> arrayList = new ArrayList<Double>();

public int getSize()

{

return arrayList.size();

}

public int getN() {

return N;

}

public void addElement(double element)

{

arrayList.add(element);

}

public double removeElement()

{

double returnElement = arrayList.get(0);

arrayList.remove(0);

return returnElement;

}

}

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

# 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.