**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине: «Визуальные средства разработки программных приложений»

на тему: «Разработка многопоточных приложений»

Выполнил: студент гр. ИТИ-31

Ковшаров Г. Ю.  
Принял: преподаватель

Ястребов А. А.

Гомель 2024

**Цель работы**: изучить создание многопоточных приложений с использованием языка программирования JAVA.

**Задание:**

Разработать приложение, в котором выполняется следующий алгоритм: два параллельных потока циклически выполняют вызов функций (согласно варианта). Каждый поток использует свою функцию. Результаты работы каждый поток помещает в общую очередь конечной длины. Третий поток забирает из очереди результаты работы функций и отображает их на экране в произвольной форме. Схема взаимодействия потоков приведена на рисунке:



Очередь должна быть реализована с использованием объектов синхронизации.

Потоки должны корректно завершаться при завершении приложения.

Для реализации выбрать любые две функции из таблицы согласно варианту.

| № | Приложение | Функции сервера |
| --- | --- | --- |
| 14 | Проверка делимости чисел | Проверка делимости целых чисел на: 2, 3, 5, 10 |

**Ход работы:**

Были сделаны функции для проверки деления на 2 и 5. Консольный вывод после завершения работы приложения представлен на рисунке 1.

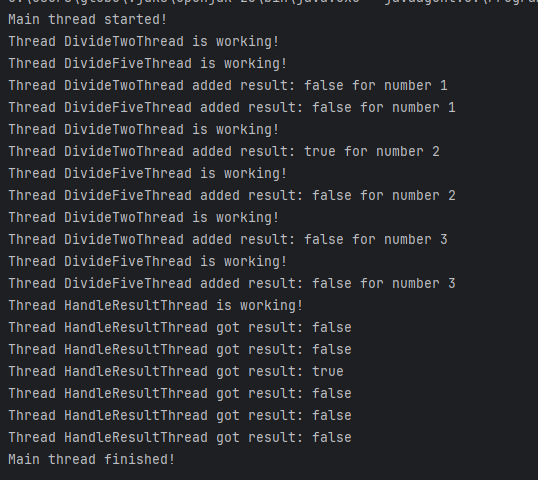


Рисунок 1 – Результат работы приложения

Текст разработанных классов на языке *Java* представлен в приложении А.

**Выводы:** в ходе проделанной работы получены навыки создания многопоточных приложений с использованием языка программирования JAVA.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст классов**

**DivideFiveThread.java:**

package Threads;

import java.util.concurrent.BlockingQueue;

public class DivideFiveThread extends Thread {

private final BlockingQueue<Boolean> queue;

private final int verifyNumber;

public DivideFiveThread(int verifyNumber, BlockingQueue<Boolean> queue) {

this.verifyNumber = verifyNumber;

this.queue = queue;

}

@Override

public void run(){

try{

System.out.println("Thread DivideFiveThread is working!");

boolean isVerify = verifyNumber % 5 == 0;

queue.put(isVerify);

System.out.println("Thread DivideFiveThread added result: " + isVerify + " for number " + verifyNumber);

Thread.sleep(2000);

}

catch(InterruptedException e){

System.out.println("Thread has been interrupted");

}

}

}

**DivideTwoThread.java:**

package Threads;

import java.util.concurrent.BlockingQueue;

public class DivideTwoThread extends Thread {

private final BlockingQueue<Boolean> queue;

private final int verifyNumber;

public DivideTwoThread(int verifyNumber, BlockingQueue<Boolean> queue) {

this.verifyNumber = verifyNumber;

this.queue = queue;

}

@Override

public void run(){

try{

System.out.println("Thread DivideTwoThread is working!");

boolean isVerify = verifyNumber % 2 == 0;

queue.put(isVerify);

System.out.println("Thread DivideTwoThread added result: " + isVerify + " for number " + verifyNumber);

Thread.sleep(2000);

}

catch(InterruptedException e){

System.out.println("Thread has been interrupted");

}

}

}

**DivideFiveThread.java:**

package Threads;

import java.util.concurrent.BlockingQueue;

public class DivideFiveThread extends Thread {

private final BlockingQueue<Boolean> queue;

private final int verifyNumber;

public DivideFiveThread(int verifyNumber, BlockingQueue<Boolean> queue) {

this.verifyNumber = verifyNumber;

this.queue = queue;

}

@Override

public void run(){

try{

System.out.println("Thread DivideFiveThread is working!");

boolean isVerify = verifyNumber % 5 == 0;

queue.put(isVerify);

System.out.println("Thread DivideFiveThread added result: " + isVerify + " for number " + verifyNumber);

Thread.sleep(2000);

}

catch(InterruptedException e){

System.out.println("Thread has been interrupted");

}

}

}

**HandleResultThread.java:**

package Threads;

import java.util.concurrent.BlockingQueue;

public class HandleResultThread extends Thread {

private final BlockingQueue<Boolean> queue;

private final int totalResults;

public HandleResultThread(BlockingQueue<Boolean> queue, int totalResults) {

this.queue = queue;

this.totalResults = totalResults;

}

@Override

public void run() {

try {

System.out.println("Thread HandleResultThread is working!");

// Collect all results from the queue

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

boolean result = queue.take();

System.out.println("Thread HandleResultThread got result: " + result);

}

} catch (InterruptedException e) {

System.out.println("Thread has been interrupted");

}

}

}

**Main.java:**

import Threads.DivideFiveThread;

import Threads.DivideTwoThread;

import Threads.HandleResultThread;

import java.util.concurrent.ArrayBlockingQueue;

import java.util.concurrent.BlockingQueue;

public class Main {

public static void main(String[] args) {

System.out.println("Main thread started!");

int queueLength = 6;

BlockingQueue<Boolean> queue = new ArrayBlockingQueue<Boolean>(queueLength);

try{

for (int i = 0; i < queueLength / 2; i++) {

var divideTwoThread = new DivideTwoThread(i + 1, queue);

var divideFiveThread = new DivideFiveThread(i + 1, queue);

divideTwoThread.start();

divideFiveThread.start();

divideTwoThread.join();

divideFiveThread.join();

}

var resultThread = new HandleResultThread(queue, queueLength);

resultThread.start();

resultThread.join();

}

catch(InterruptedException e){

System.out.print("Process has been interrupted");

}

System.out.println("Main thread finished!");

}

}