ФЕДЕРАЛЬНОЕ Государственное АВТОНОМНОЕ образовательное УЧРЕЖДЕНИЕ ВЫСШЕГО образования

**БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»**

**(НИУ «БелГУ»)**

**ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ**

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

**Отчет**

по дисциплине «Разработка программных приложений на языках высокого уровня»

специальности 09.03.03 «Прикладная информатика»

**Лабораторная работа № 5**

студента очного отделения

3 курса 12002005 группы

Огородникова Льва Олеговича

Проверил:

старший преподаватель

Лифиренко Максим Вячеславович

БЕЛГОРОД 2023

**Цель работы:** получить практические навыки создания многопоточных приложений.

**Вариант №15**

Бег с препятствиями. Создается условная карта трассы в виде матрицы, ширина которой соответствует количеству бегунов, а высота – фиксирована, содержащей произвольное количество единиц (препятствий) в произвольных ячейках. Стартующие бегуны (процессы, потоки) перемещаются по трассе и при встрече с препятствием задерживаются на фиксированное время. По достижении финиша бегуны сообщают свой номер.

**Листинг**

import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
  
*/\*\*  
 Класс Race представляет гонку бегунов на треке с препятствиями.  
 \*/*public class Race implements Runnable  
{  
 private static int[][] *track*;  
 private int id;  
 private int position;  
 private int time;  
 private static int *finishLine*;  
 private static List<Race> *finishers* = new ArrayList<>();  
   
 */\*\*  
 \* Конструктор класса Race, создающий нового бегуна с заданным идентификатором.  
 \** ***@param*** *id идентификатор бегуна  
 \*/* public Race(int id)  
 {  
 this.id = id;  
 this.position = 0;  
 this.time = 0;  
 }  
  
 */\*\*  
 \* Метод run() реализует логику бега бегуна на треке с препятствиями.  
 \* Пока позиция бегуна меньше конца трека, бегун продолжает движение по треку,  
 \* учитывая препятствия на пути.  
 \*/* public void run()  
 {  
 while (position < *finishLine*)  
 {  
 try  
 {  
 int delay = 0;  
 for (int i = time; i < *finishLine* && *track*[i][id] == 1; i++)  
 {  
 delay++;  
 }  
 Thread.*sleep*(delay \* 1000); //задержка на препятствии  
 position++;  
 time += delay + 1;  
 }  
 catch (InterruptedException e)  
 {  
 e.printStackTrace();  
 }  
 }  
 System.*out*.println("Бегун " + (id + 1) + " финишировал за " + (time - 1) + " секунд(ы)!");  
 *finishers*.add(this);  
 }  
  
 */\*\*  
 \* Метод main() является точкой входа в программу.  
 \* Создаётся трек с препятствиями и список бегунов, запускаются потоки бегунов на выполнение.  
 \* По завершении выполнения всех потоков, выводятся результаты гонки.  
 \** ***@param*** *args массив строковых аргументов командной строки  
 \*/* public static void main(String[] args)  
 {  
 int width = 4; // Количество бегунов  
 int height = 6; // Длинна трека  
 *finishLine* = height - 1; // Конец трека  
  
 *track* = new int[height][width];  
 for (int i = 0; i < height; i++)  
 {  
 for (int j = 0; j < width; j++)  
 {  
 *track*[i][j] = (int) (Math.*random*() \* 2); // случайный 0 или 1  
 }  
 }  
  
 System.*out*.println("Трек:");  
 for (int i = 0; i < height; i++)  
 {  
 for (int j = 0; j < width; j++)  
 {  
 System.*out*.print(*track*[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 System.*out*.println();  
  
 List<Thread> runners = new ArrayList<>();  
 for (int i = 0; i < width; i++)  
 {  
 runners.add(new Thread(new Race(i)));  
 }  
  
 for (Thread runner : runners)  
 {  
 runner.start();  
 }  
  
 for (Thread runner : runners)  
 {  
 try  
 {  
 runner.join();  
 } catch (InterruptedException e)  
 {  
 e.printStackTrace();  
 }  
 }  
  
 System.*out*.println("\nГонка окончена! Результаты:");  
 Collections.*sort*(*finishers*, (r1, r2) -> r1.time - r2.time);  
 for (int i = 0; i < *finishers*.size(); i++)  
 {  
 System.*out*.println("Место в гонке " + (i + 1) + ": Бегун " + (*finishers*.get(i).id + 1) + " завершил забег за " + (*finishers*.get(i).time - 1) + " секунд(ы)");  
 }  
 }  
}

На рисунке 1 представлен результат работы программы.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Результат работы программы

**Вывод:** были получены практические навыки создания многопоточных приложений.