ФЕДЕРАЛЬНОЕ Государственное АВТОНОМНОЕ образовательное УЧРЕЖДЕНИЕ ВЫСШЕГО образования

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»**

**(НИУ «БелГУ»)**

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИРОВЫХ ТЕХНОЛОГИЙ

**Кафедра прикладной информатики и информационных технологий**

**Лабораторная работа 5**

**по дисциплине «Разработка программных приложений на языках высокого уровня»**

**студента очной формы обучения**

**направления подготовки 09.03.03. «Прикладная информатика»**

**3 курса группы 12002005**

**Морозова Максима Андреевича**

Научный руководитель:

ст. пр. Лифиренко Максим Вячеславович

|  |  |
| --- | --- |
| ***Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2023 г.  \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись (расшифровка подписи) |  |

**БЕЛГОРОД, 2023**

Задания выполненные в данной работы (Вариант 8):

Содержание работы:

1 Разработать многопоточную программу для указанного варианта.

2 В приложении должен быть главный поток, порожденные потоки выполняющие определенный процесс и поток, выводящий на экран результаты.

8 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.

Результат работы программы, представлен на рисунке 1:

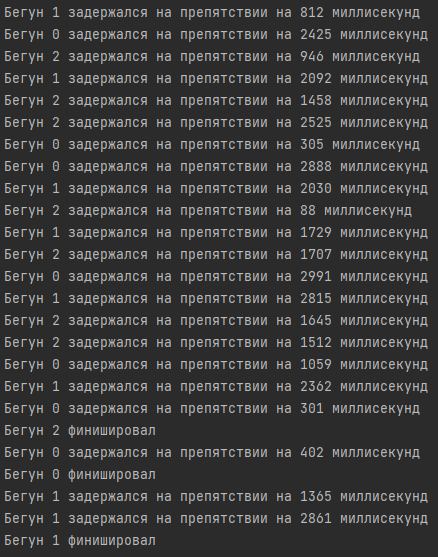


Рисунок 1 – результат работы программы

Листинг программы:  
/\*\*

Класс для запуска игры «Забег с препятствиями»

\* @author Maxim

\* @since 1.0

\*/

/\*\*

Класс для запуска игры "Забег с препятствиями"

\*/

package Lab5;

import java.util.Random;

public class ObstacleRace {

/\*\* Ширина трассы \*/

public static final int WIDTH = 5;

/\*\* Высота трассы \*/

public static final int HEIGHT = 10;

/\*\* Количество бегунов \*/

public static final int NUM\_RUNNERS = 3;

/\*\* Максимальное время задержки на препятствии в миллисекундах \*/

public static final int MAX\_DELAY = 3000;

/\*\*

\* Главный метод, создает трассу, бегунов и запускает их потоки

\* @param args аргументы командной строки (не используются)

\*/

public static void main(String[] args) {

// Создаем трассу с произвольным количеством препятствий

int[][] track = new int[WIDTH][HEIGHT];

Random random = new Random();

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

for (int j = 0; j < HEIGHT; j++) {

if (random.nextBoolean()) {

track[i][j] = 1;

}

}

}

// Создаем бегунов и запускаем их потоки

Runner[] runners = new Runner[NUM\_RUNNERS];

for (int i = 0; i < NUM\_RUNNERS; i++) {

runners[i] = new Runner(i, track);

runners[i].start();

}

// Создаем поток для вывода результатов

ResultPrinter resultPrinter = new ResultPrinter(runners);

resultPrinter.start();

}

}

/\*\*

Класс для представления бегуна

\*/

class Runner extends Thread {

/\*\* Номер бегуна \*/

private int id;

/\*\* Трасса с препятствиями \*/

private int[][] track;

/\*\*

Конструктор бегуна

@param id номер бегуна

@param track трасса с препятствиями

\*/

public Runner(int id, int[][] track) {

this.id = id;

this.track = track;

}

/\*\*

Метод для запуска потока бегуна

\*/

@Override

public void run() {

int position = 0; // Начальная позиция бегуна

while (position < ObstacleRace.HEIGHT) {

// Проверяем наличие препятствия на текущей позиции

if (track[id][position] == 1) {

int delay = new Random().nextInt(ObstacleRace.MAX\_DELAY);

System.out.println("Бегун " + id + " задержался на препятствии на " + delay + " миллисекунд");

try {

Thread.sleep(delay);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

position++; // Перемещаем бегуна на следующую позицию

}

System.out.println("Бегун " + id + " финишировал");

ResultPrinter.addResult(id); // Добавляем результат в общий список результатов

}

}

/\*\*

\* Класс ResultPrinter предназначен для вывода результатов забега на экран.

\*/

public class ResultPrinter extends Thread {

private static int[] results = new int[ObstacleRace.NUM\_RUNNERS]; // Список результатов бегунов

private static int numFinished = 0; // Количество финишировавших бегунов

private Runner[] runners; // Список бегунов

/\*\*

\* Конструктор класса ResultPrinter.

\* @param runners - список бегунов.

\*/

public ResultPrinter(Runner[] runners) {

this.runners = runners;

}

/\*\*

\* Метод addResult добавляет результат финишировавшего бегуна в список результатов и увеличивает количество финишировавших бегунов на 1.

\* @param id - номер финишировавшего бегуна.

\*/

public static synchronized void addResult(int id) {

results[id] = numFinished + 1; // Номер места - порядковый номер финишировавшего бегуна

numFinished++;

}

}