Министерство образования и науки Украины

Харьковский национальный университет радиоэлектроники

Кафедра Системотехники

Отчет по лабораторной работе №6

с курса: «Программирование под платформу .NET»

на тему: «Многопоточность и параллельные вычисления»

Выполнил:

Ст. гр. ИТКНу-17-1

Гончаренко В.А.

Проверила:

Жернова П.Е

Харьков 2019

Цель: Научиться создавать программы с использованием многопоточности. Получение навыков организации параллельных вычислений.

Ход выполнения работы

Задание 1.

Код программы:

public class MultiThread {

public static void main(String[] args) throws Exception{

BufferedReader in = new BufferedReader(new FileReader("input.txt"));

List<String> collect = in.lines().collect(Collectors.toList());

String stringToSearch = "r";

int onePart = collect.size() / 3;

new Thread(new CustomRunnable(0, onePart, collect, stringToSearch)).start();

new Thread(new CustomRunnable(onePart, onePart + onePart, collect, stringToSearch)).start();

new Thread(new CustomRunnable(2 \* onePart, collect.size() - 1, collect, stringToSearch)).start();

}

}

class CustomRunnable implements Runnable{

private int start;

private int end;

private List<String> list;

private String search;

public CustomRunnable(int start, int end, List<String> list, String search) {

this.start = start;

this.end = end;

this.list = list;

this.search = search;

}

@Override

public void run() {

for (int i = start; i <= end; i++) {

if (list.get(i).equals(search)) {

System.out.println("Searched string : " + search + ", index : " + i);

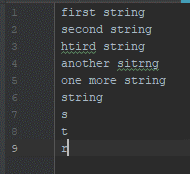
}

}

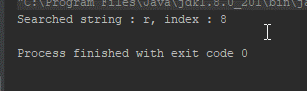
}

}

Файл:



Результат:

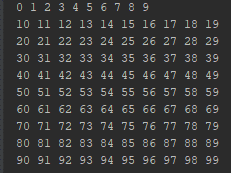


Задание 2.

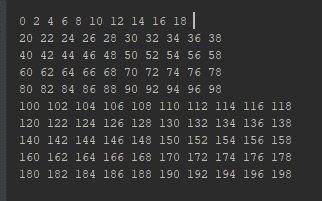
Код программы:

public class Second {  
 public static void main(String[] args) throws ExecutionException, InterruptedException {  
 int width = 10;  
 int height = 10;  
 int[] vector = new int[] {2,2,2,2,2,2,2,2,2,2};  
 int[][] matrix = new int[width][height];  
 for (int i = 0; i < width; i++) {  
 for (int j = 0; j < height; j++) {  
 matrix[i][j] = i \* height + j;  
 }  
 }  
 *print*(width, height, matrix);  
 System.*out*.println();  
  
 List<FutureTask<int[]>> tasks = new ArrayList<>();  
 for (int i = 0; i < matrix.length; i++) {  
 tasks.add(new FutureTask<>(new RowCalculator(matrix[i], vector[i])));  
 new Thread(tasks.get(i)).start();  
 }  
 for (int i = 0; i < matrix.length; i++) {  
 matrix[i] = tasks.get(i).get();  
 }  
 *print*(width, height, matrix);  
 }  
  
 private static void print(int width, int height, int[][] matrix) {  
 for (int i = 0; i < width; i++) {  
 for (int j = 0; j < height; j++) {  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 }  
}  
  
class RowCalculator implements Callable<int[]> {  
 private int[] arr;  
 private int multiply;  
  
 public RowCalculator(int[] arr, int multiply) {  
 this.arr = arr;  
 this.multiply = multiply;  
 }  
  
 @Override  
 public int[] call() throws Exception {  
 for (int i = 0; i < arr.length; i++) {  
 arr[i] = arr[i] \* multiply;  
 }  
 return arr;  
 }  
}

Входные данные:



Результат:

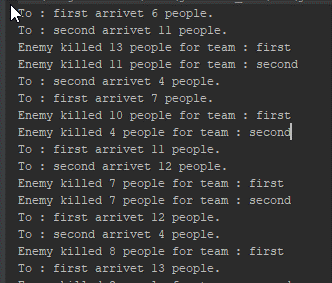


Задание 3.

Код программы:

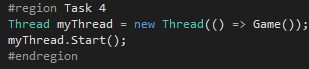
public class Third {  
 public static void main(String[] args) {  
 Team t1 = new Team(100, null, "first");  
 Team t2 = new Team(100, t1, "second");  
 t1.setEnemy(t2);  
 *initTeam*(t1);  
 *initTeam*(t2);  
 }  
  
 private static void initTeam(Team team) {  
 new Thread(() -> {  
 int i = 5;  
 while (i >= 0) {  
 --i;  
 try {  
 team.arrivet(new Random().nextInt(15));  
 Thread.*sleep*(new Random().nextInt(500));  
 team.killed(new Random().nextInt(15));  
 Thread.*sleep*(new Random().nextInt(500));  
 } catch (InterruptedException e) {  
 }  
 }  
 }).start();  
 }  
}  
  
class Team {  
 private String name;  
 private int fighters;  
 private Team enemy;  
  
 public Team(int fighters, Team enemy, String name) {  
 this.fighters = fighters;  
 this.enemy = enemy;  
 this.name = name;  
 }  
  
 public void killed(int count) {  
 System.*out*.println("Enemy killed " + count + " people for team : " + name);  
 this.fighters -= count;  
 }  
  
 public void arrivet(int count) {  
 System.*out*.println("To : " + name + " arrivet " + count + " people.");  
 this.fighters += count;  
 }  
  
 public void setEnemy (Team enemy) {  
 this.enemy = enemy;  
 }  
}

Результат:



Задание 4. Создать игру, где будут 2-3 барана и волк. Если координат волка совпадают с координатами барана, то баран исчезает. Если координаты баранов совпадают появляется новый баран. Все движутся хаотично.

Код программы:



public class Fourth {

public static void main(String[] args) {

Wolf wolf = new Wolf(3, 3);

List<Sheep> sheep = new ArrayList<>();

sheep.add(new Sheep(1,2));

sheep.add(new Sheep(3,4));

Field field = new Field(wolf, sheep);

for (Sheep sheep1 : sheep) {

new Thread(() -> {

while(true) {

sheep1.move();

try {

Thread.sleep(500);

} catch (InterruptedException e){};

}

}).start();

}

new Thread(() -> {

while (true) {

wolf.move();

try {

Thread.sleep(500);

} catch (InterruptedException e) {

}

}

}).start();

new Thread(() -> {

while (true) {

try {

Thread.sleep(500);

} catch (InterruptedException e) {

}

field.play();

}

}).start();

}

}

class Hero {

protected int x;

protected int y;

public void move () {

checkForBorderValues();

switch (new Random().nextInt(4)) {

case 0: x+=1;break;

case 1: x-=1;break;

case 2: y+=1;break;

case 3: y-=1;break;

}

}

private void checkForBorderValues() {

if (x == 6) {

x -= 1;

}

if (y == 6) {

y -= 1;

}

if (x == 0) {

x += 1;

}

if (y ==0) {

y += 1;

}

}

}

class Wolf extends Hero{

private char symbol = 'W';

public Wolf(int x, int y) {

this.x = x;

this.y = y;

}

public char getSymbol() {

return this.symbol;

}

}

class Sheep extends Hero {

private char symbol = 'S';

public Sheep(int x, int y) {

this.x = x;

this.y = y;

}

public char getSymbol() {

return this.symbol;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Sheep sheep = (Sheep) o;

return symbol == sheep.symbol;

}

@Override

public int hashCode() {

return Objects.hash(symbol);

}

}

class Field {

private static final int FIELD\_WIDTH = 7;

private static final int FIELD\_HEIGHT = 7;

private Wolf wolf;

private List<Sheep> sheeps;

private static final char SHEEP = '@';

private static final char[][] field = new char[FIELD\_HEIGHT][FIELD\_WIDTH];

public Field(Wolf wolf, List<Sheep> sheep) {

for (int i = 0; i < FIELD\_WIDTH; i++) {

for (int j = 0; j < FIELD\_HEIGHT; j++) {

field[i][j] = '.';

}

}

this.wolf = wolf;

this.sheeps = sheep;

}

public void play() {

for (int i = 0; i < FIELD\_WIDTH; i++) {

for (int j = 0; j < FIELD\_HEIGHT; j++) {

field[i][j] = '.';

}

}

field[wolf.x][wolf.y] = wolf.getSymbol();

sheeps.forEach(sheep -> field[sheep.x][sheep.y] = sheep.getSymbol());

for (int i = 0; i < sheeps.size(); i++) {

if (wolf.x == sheeps.get(i).x && wolf.y == sheeps.get(i).y) {

sheeps.remove(i);

}

}

for (int i = 0; i < 5; i++) {

System.out.println();

}

for (int i = 0; i < FIELD\_WIDTH; i++) {

for (int j = 0; j < FIELD\_HEIGHT; j++) {

System.out.print(field[i][j] + " ");

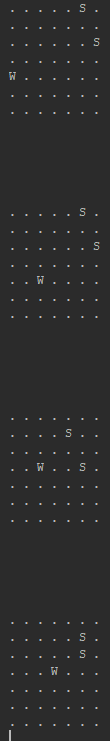
}

System.out.println();

}

}

Результат:



Задание 5.

public class SimpleAndFibonacci {

public static final String FIBOACCI\_TXT = "fiboacci.txt";

public static final String PRIME\_TXT = "prime.txt";

private static boolean isCalculating = true;

public static void main(String[] args) throws Exception {

List<Thread> list = new ArrayList<>();

list.add(new Thread(() -> {

BufferedWriter bufferedWriter = null;

try {

new File(FIBOACCI\_TXT).createNewFile();

bufferedWriter = new BufferedWriter(new FileWriter(FIBOACCI\_TXT));

} catch (IOException e) {

return;

}

long previous = 0, next = 1;

while (isCalculating) {

long summ = previous + next;

previous = next;

next = summ;

try {

Thread.sleep(50);

bufferedWriter.write(summ + System.lineSeparator());

} catch (IOException | InterruptedException e){}

}

try {

bufferedWriter.close();

} catch (IOException e){}

}));

list.add(new Thread(() -> {

BufferedWriter bufferedWriter = null;

try {

new File(PRIME\_TXT).createNewFile();

bufferedWriter = new BufferedWriter(new FileWriter(PRIME\_TXT));

} catch (IOException e) {

return;

}

int start = 0;

while (isCalculating) {

if (checkForPrime(++start)) {

try {

Thread.sleep(50);

bufferedWriter.write(start + System.lineSeparator());

} catch (IOException | InterruptedException e){

}

}

}

try {

bufferedWriter.close();

} catch (IOException e){}

}));

list.get(0).start();

list.get(1).start();

Thread.sleep(800);

isCalculating = false;

if (new File(FIBOACCI\_TXT).canRead()) {

System.out.println("Fibo");

Files.lines(Paths.get(FIBOACCI\_TXT)).forEach(s -> System.out.print(s + " "));

}

System.out.println();

if (new File(PRIME\_TXT).canRead()) {

System.out.println("Prime");

Files.lines(Paths.get(PRIME\_TXT)).forEach(s -> System.out.print(s + " "));

}

}

static boolean checkForPrime(int number) {

int i , lastCheckedValue;

lastCheckedValue = number / 2;

if(number == 0|| number == 1) {

return false;

} else {

for(i = 2; i <= lastCheckedValue; i++){

if( number % i == 0){

return false;

}

}

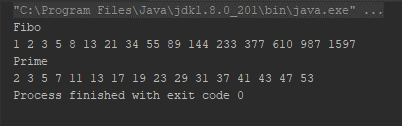
}

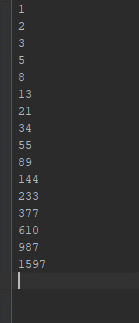
return true;

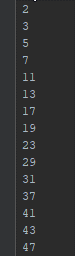
}

}

Результат:







Задание 6.

public class WriteAndRead {

private static File file = new File("test.txt");

public static void main(String[] args) throws Exception {

file.createNewFile();

new Thread(() -> {

try (BufferedReader reader = Files.newBufferedReader(Paths.get("test.txt"))) {

while(true) {

synchronized (file) {

String line;

while((line = reader.readLine()) != null) {

System.out.println(line);

}

}

try {

Thread.sleep(100);

} catch (InterruptedException i){}

}

} catch (IOException ignored) {

}

}).start();

new Thread(() -> {

try (BufferedWriter reader = Files.newBufferedWriter(Paths.get("test.txt"))) {

while(true) {

synchronized (file) {

reader.write(new Random().nextInt(5000) + System.lineSeparator());

reader.flush();

}

try {

Thread.sleep(150);

} catch (InterruptedException i){}

}

} catch (IOException ignored) {

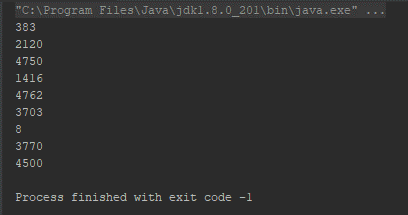
}

}).start();

}

}

Результат:



Задание 7.

public class Last {

private static double value = 1.0;

public static void main(String[] args) {

new Thread(() -> {

for (int x = 0; x < 1000000000; x++) {

value = 1 + sin(x) - 2 \* cos(x) + 4 \* pow(sin(x), 2) - 8 \* pow(cos(x), 2);

}

}).start();

while(true) {

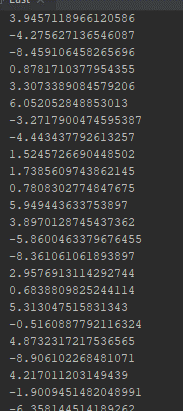
System.out.println(value);

}

}

}

Результат:



Выводы:

В данной лабораторной работе мы рассмотрели работу с многопоточностью и параллельными асинхронными вычислениями. В данной работы были закреплены навыки работы с асинхронным выполнением методов, а также синхронизация потоков.