МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ   
ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЁВА (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» (САМАРСКИЙ УНИВЕРСИТЕТ)**

Институт информатики и кибернетики  
Кафедра программных систем

Дисциплина  
**Спецкурс по языку Ява**

**ОТЧЕТ**по лабораторной работе № 5

Студент: Кочетков Е.В.  
Группа: 6202-020302D  
  
Преподаватель: Ледкова Т.А  
  
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара 2024

**Задание**

**Задание 1.**

-------ReadThread.java-------

**import** java.util.Random;  
  
**public class** ReadThread **extends** Thread {  
  
 **private final** Int object;  
  
  
 **public** ReadThread(Int \_obj) {  
 **if** (\_obj != **null**) {  
 object = \_obj;  
 }  
 **else throw new** IllegalArgumentException();  
 }  
 @Override  
 **public void** run() {  
 **for** (**int** i = 0; i < object.getLen(); i++) {  
 **if** (Thread.*currentThread*().getState() == State.*WAITING*) {  
 System.*out*.println(Thread.*currentThread*().getName() + "is waiting");  
 }  
 System.*out*.printf("Read: " + object.getElement(i) + " from position " + i + "\n");  
 }  
 }  
 @Override  
 **public void** start() {  
 System.*out*.println(getClass().getName() + " started");  
 **super**.start();  
 }  
  
  
}

-------WriteThread.java-------

import java.util.Random;  
public class WriteThread extends Thread {  
  
 private final Int object;  
 private final Random rand = new Random();  
  
  
 public WriteThread(Int \_obj) {  
 if (\_obj != null) {  
 object = \_obj;  
 }  
 else throw new IllegalArgumentException();  
 }  
  
 @Override  
 public void run() {  
 for (int i = 0; i < object.getLen(); i++) {  
 int l = rand.nextInt(100);  
 object.setElement(i, l);  
 if (Thread.*currentThread*().getState() == State.*WAITING*) {  
 System.*out*.println(Thread.*currentThread*().getName() + "is waiting");  
 }  
 System.*out*.printf("Write: " + l + " to position " + i + "\n");  
 }  
 }  
  
 @Override  
 public void start() {  
 System.*out*.println(getClass().getName() + " started");  
 super.start();  
 }  
}

**Задание 2.**

-------IntSynchronizer.java-------

**public class** IntSynchronizer {  
 **private final** Int obj;  
 **private volatile int** current = 0;  
 **private final** Object isLocked = **new** Object();  
 **private boolean** set = **false**;  
  
 **public** IntSynchronizer(Int i) {  
 **this**.obj = i;  
 }  
  
 **public int** read() **throws** InterruptedException {  
 **int** val;  
 **synchronized**(isLocked) {  
 **if** (!canRead()) **throw new** InterruptedException();  
 **while** (!set)  
 isLocked.wait();  
 val = obj.getElement(current++);  
 System.*out*.printf("Read: " + val + " from position " + (current - 1) + "\n");  
 set = **false**;  
 isLocked.notifyAll();  
 }  
 **return** val;  
 }  
  
 **public void** write(**int** val) **throws** InterruptedException {  
 **synchronized**(isLocked) {  
 **if** (!canWrite()) **throw new** InterruptedException();  
 **while** (set)  
 isLocked.wait();  
 obj.setElement(current, val);  
 System.*out*.printf("Write: " + val + " to position " + current + "\n");  
 set = **true**;  
 isLocked.notifyAll();  
 }  
 }  
  
 **public boolean** canRead() {  
 **return** current < obj.getLen();  
 }  
  
 **public boolean** canWrite() {  
 **return** (!set && current < obj.getLen()) || (set && current < obj.getLen() - 1);  
 }  
}

-------SynchronizedInt.java-------

**import** java.io.OutputStream;  
**import** java.io.Writer;  
  
**public class** SynchronizedInt **implements** Int {  
 **private final** Int packaged;  
  
 **public** SynchronizedInt(Int \_package) {  
 **this**.packaged = \_package;  
 }  
  
 @Override  
 **public synchronized int** getElement(**int** n) {  
 **return** packaged.getElement(n);  
 }  
  
 @Override  
 **public synchronized void** setElement(**int** n, **int** k) {  
 packaged.setElement(n, k);  
 }  
  
 @Override  
 **public synchronized** String getName() {  
 **return** packaged.getName();  
 }  
  
 @Override  
 **public synchronized int** getIntroduce() {  
 **return** packaged.getIntroduce();  
 }  
  
 @Override  
 **public synchronized int** calcPages() **throws** PagesLessZeroException {  
 **return** packaged.calcPages();  
 }  
  
 @Override  
 **public synchronized** String toString() {  
 **return** packaged.toString();  
 }  
  
 @Override  
 **public synchronized boolean** equals(Object o) {  
 **return** packaged.equals(o);  
 }  
  
 @Override  
 **public synchronized int** getLen() {  
 **return** packaged.getLen();  
 }  
  
  
 @Override  
 **public synchronized void** output(OutputStream out) {  
 packaged.output(out);  
 }  
  
 @Override  
 **public synchronized void** write(Writer out) {  
 packaged.write(out);  
 }  
  
 @Override  
 **public synchronized int** hashCode() {  
 **return** packaged.hashCode();  
 }  
}

-------Main.java-------

**import** java.util.Scanner;  
  
**public class** Main {  
 **static** Scanner *scan* = **new** Scanner(System.*in*);  
  
 **public static void** main(String**[]** s) {  
 System.*out*.println("====ЛАБОРАТОРНАЯ РАБОТА №5====");  
 **boolean** end = **false**;  
 **while** (!end) {  
 System.*out*.println("=========================МЕНЮ=========================");  
 System.*out*.println("1. Параллельные запись/чтение массива (Задание 1)");  
 System.*out*.println("2. Последовательные запись/чтение массива (Задание 2)");  
 System.*out*.println("3. Последовательные запись/чтение массива (Задание 3)");  
 System.*out*.println("0. Выход из программы");  
 System.*out*.print("Выберете пункт меню: ");  
 Int obj;  
 // Из-за пользования одним и тем же потоком ввода (System.in), остаётся что-то  
 // на что реагирует .nextLine(). Этот код помогает избежать проблемы  
 **try** {  
 **while** (!*scan*.hasNext()) {  
 Thread.*sleep*(200);  
 }  
 } **catch** (InterruptedException ignored) { }  
 String choice = *scan*.nextLine();  
 **switch** (choice) {  
 **case** "1":  
 obj = *CreateEssays*();  
 System.*out*.println("====УСПЕШНО====\n");  
 Thread readThread = **new** ReadThread(obj);  
 Thread writeThread = **new** WriteThread(obj);  
 writeThread.start();  
 readThread.start();  
 **try** {  
 readThread.join();  
 } **catch** (InterruptedException ignored) {  
 }  
 **try** {  
 writeThread.join();  
 } **catch** (InterruptedException ignored) {  
 }  
 **break**;  
 **case** "2":  
 obj = *CreateEssays*();  
 System.*out*.println("====УСПЕШНО====\n");  
 IntSynchronizer intSynchronizer = **new** IntSynchronizer(obj);  
 Thread read = **new** Thread(**new** ReadSyncThread(intSynchronizer));  
 Thread write = **new** Thread(**new** WriteSyncThread(intSynchronizer));  
 read.start();  
 write.start();  
 **try** {  
 read.join();  
 } **catch** (InterruptedException ignored) {}  
 **try** {  
 write.join();  
 } **catch** (InterruptedException ignored) {}  
 **break**;  
 **case** "3":  
 obj = *CreateEssays*();  
 System.*out*.println("====УСПЕШНО====\n");  
 SynchronizedInt synchronizedInt = IOMethods.*synchronizedInt*(obj);  
 Thread thread1 = **new** ReadThread(synchronizedInt);  
 thread1.setName("ReadThread");  
 Thread thread2 = **new** WriteThread(synchronizedInt);  
 thread2.setName("WriteThread");  
  
 Thread checkThread1 = **new** CheckThread(thread1);  
 Thread checkThread2 = **new** CheckThread(thread2);  
  
 thread1.start();  
 thread2.start();  
  
 checkThread1.start();  
 checkThread2.start();  
  
 **try** {  
 thread2.join();  
 } **catch** (InterruptedException ignored) {}  
 **try** {  
 thread1.join();  
 } **catch** (InterruptedException ignored) {}  
 **try** {  
 checkThread1.join();  
 } **catch** (InterruptedException ignored) {}  
 **try** {  
 checkThread2.join();  
 } **catch** (InterruptedException ignored) {}  
 **break**;  
 **case** "0":  
 System.*out*.println("\n====ЗАВЕРШЕНИЕ РАБОТЫ====");  
 end = **true**;  
 **break**;  
 **default**:  
 System.*out*.println("\n====ОШИБКА====");  
 System.*out*.println("Введено неправильное значение!\n");  
 **break**;  
 }  
  
 }  
 }  
  
 **private static void** PrintOutArray(Int**[]** array) {  
 **for** (**int** i = 0; i < array.length; i++) {  
 System.*out*.println("====ЭЛЕМЕНТ #" + (i + 1) + "====");  
 System.*out*.println(array[i].toString());  
 }  
 }  
  
  
 **private static** Int CreateEssays() {  
 System.*out*.println("\n====СОЗДАНИЕ ОБЪЕКТА====");  
 System.*out*.print("Введите длину массива: ");  
 **int** length = *scan*.nextInt();  
 **return new** Essays(length);  
 }  
  
  
}









