Министерство образования и науки Российской Федерации

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Санкт-Петербургский государственный университет промышленных технологий и дизайна»

Институт информационных технологий и автоматизации

Кафедра: Интеллектуальных систем и защиты информации

Направление подготовки: 10.03.01 Информационная безопасность

Профиль подготовки: Информационная безопасность в коммерческих структурах

ОТЧЕТ

о выполнении лабораторной работы №1

по дисциплине: «Информатика»

Преподаватель: Бусыгин К. Н.

Обучающийся: Голубева А.П., Шимпф Д.Д.

Курс: 1 Учебная группа: МДА-7,9

Санкт-Петербург
2020

1. Породить 10 потоков, каждый из которых выводит в консоль сообщение «hello from thread N», где N – номер потока (номер потока передать ему через конструктор). Посмотреть, что будет, если перед выходом из главного потока не ожидать завершения порожденных потоков, и если ожидать.

Код Java:

```
public class n1 implements Runnable{
    public void run(){
        System.out.println("Hello from " + Thread.currentThread().getName());
    }
}
class TenThreads{
    public static void main(final String[] args) throws InterruptedException{
        System.out.println("Hello from main Thread");
        Thread[] tr = new Thread[10];
        for(int i = 0; i < 10; i++) tr[i] = new Thread(new n1());
        for(int i = 0; i < 10; i++) tr[i].start();
        for(int i = 0; i < 10; i++) tr[i].interrupt();
        for(int i = 0; i < 10; i++) tr[i].join();
    }
}
```

Пример работы:

```
Hello from main Thread
Hello from Thread-8
Hello from Thread-5
Hello from Thread-1
Hello from Thread-0
Hello from Thread-6
Hello from Thread-9
Hello from Thread-2
Hello from Thread-7
Hello from Thread-4
Hello from Thread-3
```

2. Породить 10 потоков, каждый из которых каждую секунду (Thread.sleep()) выводит в консоль свой номер и время, прошедшее с момента его запуска в секундах.

Код Java:

```
public class n2 extends Thread{
    public void run(){
        long t = System.currentTimeMillis();
        try{
            while(!isInterrupted()){
                Thread.sleep( l: 1000);
                System.out.println(getName() + " has been starting since " + (t - System.currentTimeMillis()));
            }
        }
        catch(InterruptedException e){
            System.out.println(getName() + " interrupted at " + (t - System.currentTimeMillis()));
        }
    }
}
class SecondTask{
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) new n2().start();
    }
}
```

3. Для потоков из п. 2 отменить все потоки через 5 секунд из главного потока через разделяемую память.

Код Java:

```
public class n3 extends Thread{
    public void run(){
        long t = System.currentTimeMillis();
        try{
            while(!isInterrupted()){
                Thread.sleep( l: 1000);
                System.out.println(getName() + " thread has been starting already " + (t - System.currentTimeMillis()));
            }
        }
        catch(InterruptedException e){
            System.out.println(getName() + " has interrupted.");
        }
    }
}
class Thirds{
    public static void main(String[] args)throws InterruptedException{
        n3[] stop = new n3[10];
        for(int i = 0; i < 10; i++) stop[i] = new n3();
        for(int i = 0; i < 10; i++) stop[i].start();
        Thread.sleep( l: 5000);
        for(int i = 0; i < 10; i++) stop[i].interrupt();
        for(int i = 0; i < 10; i++) stop[i].join();
        System.out.println("All done.");
    }
}
```

Пример работы:

```
Thread-9 thread has been starting already -1001
Thread-1 thread has been starting already -1001
Thread-3 thread has been starting already -1001
Thread-4 thread has been starting already -1001
Thread-6 thread has been starting already -1001
Thread-2 thread has been starting already -1001
Thread-8 thread has been starting already -1001
Thread-5 thread has been starting already -1001
Thread-0 thread has been starting already -1001
Thread-7 thread has been starting already -1001
Thread-3 thread has been starting already -2053
Thread-1 thread has been starting already -2053
Thread-9 thread has been starting already -2052
Thread-4 thread has been starting already -2053
Thread-8 thread has been starting already -2053
Thread-6 thread has been starting already -2054
Thread-2 thread has been starting already -2054
Thread-0 thread has been starting already -2055
Thread-5 thread has been starting already -2055
Thread-7 thread has been starting already -2055
Thread-9 thread has been starting already -3052
Thread-3 thread has been starting already -3053
Thread-4 thread has been starting already -3053
Thread-1 thread has been starting already -3053
Thread-6 thread has been starting already -3054
Thread-8 thread has been starting already -3053
Thread-2 thread has been starting already -3054
Thread-5 thread has been starting already -3055
Thread-7 thread has been starting already -3055
Thread-0 thread has been starting already -3055
Thread-4 thread has been starting already -4053
Thread-3 thread has been starting already -4053
Thread-9 thread has been starting already -4052
Thread-1 thread has been starting already -4053
Thread-8 thread has been starting already -4054
Thread-6 thread has been starting already -4055
Thread-5 thread has been starting already -4056
Thread-7 thread has been starting already -4056
Thread-0 thread has been starting already -4056
Thread-2 thread has been starting already -4056
Thread-3 has interrupted.
Thread-8 has interrupted.
Thread-6 has interrupted.
Thread-9 has interrupted.
Thread-1 has interrupted.
Thread-4 has interrupted.
Thread-7 has interrupted.
Thread-5 has interrupted.
Thread-0 has interrupted.
Thread-2 has interrupted.
All done.
```

4. Реализовать программу, моделирующую задачу производителей-потребителей с десятью производителями и десятью потребителями без использования средств синхронизации. Какие ошибки здесь могут быть?
   Если создать счетчики произведенного и потребленного, то в результате работы программы значения будут не совпадать, что вызвано одновременным доступом нескольких потоков к критической секции.

5. Модифицировать программу из п. 4 с использованием синхронизации с помощью семафоров.
   Код Java:

```
import java.util.Random;
import java.util.concurrent.*;
import java.util.*;

public class n4{
    static Stack<Integer> St = new Stack<>();
}
class Manufacturer extends Thread{
    String n;
    Semaphore a;
    Semaphore full;
    Semaphore empty;
    Manufacturer(String name, Semaphore a, Semaphore full, Semaphore empty){
        this.n = name;
        this.a = a;
        this.full = full;
        this.empty = empty;
    }
    public void run(){
        int n = 0;
        try{
            while (n < 1000){
                Random rnd = new Random();
                full.acquire();
                a.acquire();
                int s = 0;
                n4.St.push(rnd.nextInt( bound: 100));
                for (int i = 1; i <= 10000; i++)
                    for(int j = 1; j <= 10000; j++)
                        s++;
                n++;
```

```
                    ManufacturCons.manufactur++;
                    a.release();
                    empty.release();
                    for (int i = 1; i <= 10000; i++)
                        for(int j = 1; j <= 10000; j++)
                            s++;
            }
        }
        catch(InterruptedException e){}
    }
}
class Consumer extends Thread {
    String name;
    Semaphore a;
    Semaphore full;
    Semaphore empty;
    Consumer(String name, Semaphore a, Semaphore full, Semaphore empty){
        this.name = name;
        this.a = a;
        this.full = full;
        this.empty = empty;
    }
    public void run() {
        try {
            while (!n4.St.empty()) {
                empty.acquire();
                a.acquire();
                int s = 0;
                n4.St.pop();
                for (int i = 1; i <= 10000; i++)
                    for (int j = 1; j <= 10000; j++)
                        s++;
```

```
                    ManufacturCons.cons++;
                    a.release();
                    full.release();
                    for (int i = 1; i <= 10000; i++)
                        for (int j = 1; j <= 10000; j++)
                            s++;
                }
            } catch
        (InterruptedException e) {
        }
    }
}
class ManufacturCons{
    static int cons = 0;
    static int manufactur = 0;
    public static void main(String[] args)throws InterruptedException{
        Semaphore a = new Semaphore( permits: 1);
        Semaphore full = new Semaphore( permits: 100);
        Semaphore empty = new Semaphore( permits: 100);
        empty.drainPermits();
        Manufacturer[] manufacturers = new Manufacturer[10];
        Consumer[] consumers = new Consumer[10];
        for(int i = 0; i < 10; i++)
            manufacturers[i] = new Manufacturer( name: "Manufacturer " + i, a, full, empty);
        for(int i = 0; i < 10; i++)
            consumers[i] = new Consumer( name: "Consumer " + i, a, full, empty);
        for(int i = 0; i < 10; i++)
            manufacturers[i].start();
        for(int i = 0; i < 10; i++)
            consumers[i].start();
        for(int i = 0; i < 10; i++)
            manufacturers[i].join();
        for(int i = 0; i < 10; i++)
            consumers[i].join();
        int itog = manufactur - cons;
        System.out.println("Result: "+ itog);
    }
}
```

6. Модифицировать программу из п. 4 с использованием синхронизации с помощью блокировок.

Код Java:

```
import java.util.Random;
import java.util.concurrent.locks.*;
import java.util.*;

public class n5{
        static Stack<Integer> shelf = new Stack<>();
}
class Prod extends Thread{
    String n;
    ReentrantLock l;
    Condition full;
    Condition empty;
    Prod(String n, ReentrantLock l, Condition full, Condition empty){
        this.n = n;
        this.l = l;
        this.full = full;
        this.empty = empty;
    }
    public void run(){
        int n = 0;
        int product = 0;
        try{
            while (n < 1000){
                int no = 0;
                Random rnd = new Random();
                l.lock();
                while(n5.shelf.search(product) == 100) full.await();
                product = rnd.nextInt( bound: 100);
                n5.shelf.push(product);
                MiC.b++;
                if(n5.shelf.search(product) == 1) empty.signalAll();
                for (int i = 1; i <= 10000; i++)
                   for(int j = 1; j <= 10000; j++)
                      no++;
                   n++;
                   MiC.m++;
                   l.unlock();
                   for (int i = 1; i <= 10000; i++)
                       for(int j = 1; j <= 10000; j++)
                           no++;
            }
        }
        catch(InterruptedException e){}
    }
}
class Cons extends Thread{
    String name;
    ReentrantLock l;
    Condition full;
    Condition empty;
    Cons(String name, ReentrantLock l, Condition full, Condition empty){
        this.name = name;
        this.l = l;
        this.full = full;
        this.empty = empty;
    }
    public void run(){
        int n = 0;
        try{
            while(n < 1000){
                l.lock();
                int no = 0;
                while(n5.shelf.empty()) empty.await();
                n5.shelf.pop();
                MiC.b--;
```

```
                if(MiC.b != 100) full.signalAll();
                for (int i = 1; i <= 10000; i++)
                    for(int j = 1; j <= 10000; j++)
                        no++;
                    n++;
                    MiC.c++;
                    l.unlock();
                    for (int i = 1; i <= 10000; i++)
                        for(int j = 1; j <= 10000; j++)
                            no++;
            }
        }
        catch(InterruptedException e){
            System.out.println(e.getMessage());
        }
    }
}
class MiC{
    static int c = 0;
    static int m = 0;
    static int b = 0;
    public static void main(String[] args)throws InterruptedException{
        ReentrantLock l = new ReentrantLock();
        Condition full = l.newCondition();
        Condition empty = l.newCondition();
        Prod[] manufacturers = new Prod[10];
        Cons[] consumers = new Cons[10];
        for(int i = 0; i < 10; i++) manufacturers[i] = new Prod( n: "Proizvoditel' " + i, l, full, empty);
        for(int i = 0; i < 10; i++) consumers[i] = new Cons( name: "Potrebitel' " + i, l, full, empty);
        for(int i = 0; i < 10; i++) manufacturers[i].start();
        for(int i = 0; i < 10; i++) consumers[i].start();
        for(int i = 0; i < 10; i++) manufacturers[i].join();
        for(int i = 0; i < 10; i++) consumers[i].join();
        int itog = m - c;
        System.out.println("Result: "+ itog);
    }
}
```

7. Вычислить число p с точностью 10^{-9} с помощью формулы Лейбница, пользуясь пятью потоками.

Код Java:

```
public class n6 {
    static double itog = 1;
    static int a = 5;
    public static void main(String[] args) throws InterruptedException{
        double pi = 4;
        double toch = Math.pow(10, -9);
        double b = 1;
        Calculator[] threadCalculators = new Calculator[a];
        for (int i = 0; i < a; i++) threadCalculators[i] = new Calculator(b, toch,  control: i + 1);
        long time = System.currentTimeMillis();
        for (int i = 0; i < a; i++) threadCalculators[i].start();
        for (int i = 0; i < a; i++) {
            threadCalculators[i].join();
            itog += threadCalculators[i].itog;
        }
        pi *= itog;
        time = System.currentTimeMillis() - time;
        System.out.println(pi);
        System.out.println(time);
    }
}
class Calculator extends Thread{
    double b;
    double toch;
    int control;
    double itog;
    Calculator(double b, double toch, int control){
        this.b = b;
        this.toch = toch;
        this.control = control;
    }
    public void run(){
        while (Math.abs(b) > toch){
            b = Math.pow(-1, control) / (2 * control + 1);
            itog += b;
            control += n6.a;
        }
    }
}
```

Пример работы:

```
3.1415926555892373
6533
```

8. Разобрать задачу обедающих философов (Таненбаум). Реализовать ее для 7 философов с использованием блокировок.

Код Java:

```
import java.util.concurrent.locks.*;

public class n7 extends Thread{
    private int i;
    private Forks f;
    private static int d = 0;
    n7(int i, Forks f){
        this.i = i;
        this.f = f;
    }
    @Override
    public void run(){
        while(true){
            this.think();
            if(this.f.tryForks(i)){
                announce( action: 1);
                kushou();
                this.f.putForks(i);
                announce( action: 2);
            }
            this.think();
        }
    }
    private void announce(int action){
        switch (action){
            case 1: System.out.println("Philosof " + i + " dostal vilki");
            break;
            case 2: System.out.println("Philosof " + i + " poglotil' pishchu");
            break;
        }
    }
```

```
    private void think() {
        for (int i = 0; i < 10000; i++)
            for (int j = 0; j < 10000; j++)
                d++;
    }
    private void kushou(){
        int eaten = 0;
        for(int i = 0; i < 10000; i++)
            for(int j = 0; j < 10000; j++)
                eaten++;
    }
}
class Forks{
    private int n;
    private boolean[] f;
    private static ReentrantLock lock = new ReentrantLock();
    Forks(int n){
        this.n = n;
        this.f = new boolean[n];
        for(int i = 0; i < n; i++){
            this.f[i] = true;
        }
    }
    boolean tryForks(int i){
        boolean check = false;
        lock.lock();
        if(this.f[i] && this.f[(this.n + i - 1) % this.n]){
            check = true;
            this.f[i] = this.f[(this.n + i - 1)% this.n] = false;
        }
        lock.unlock();
        return check;
    }
    void putForks(int i){
        lock.lock();
        this.f[i] = this.f[(this.n + i - 1) % this.n] = true;
        lock.unlock();
    }
}
class Phils{
    public static void main(String[] args){
        int numberOfThinkers = 7;
        Forks f = new Forks(numberOfThinkers);
        n7[] c = new n7[numberOfThinkers];
        for(int i = 0; i < numberOfThinkers; i++)
            c[i] = new n7(i, f);
        for(int i = 0; i < numberOfThinkers; i++)
            c[i].start();
    }
}
```

9. Разобрать задачу читателей и писателей (Таненбаум). Реализовать ее для 5 писателей и 10 читателей с помощью блокировок.

Код Java:

```
import java.util.concurrent.locks.*;
import java.util.Random;

public class n8 {
    static int written = 0;
    static int read = 0;
    public static void main(String[] args)throws InterruptedException{
        ReadWriteLock lock = new ReentrantReadWriteLock();
        Writer[] writer = new Writer[5];
        Reader[] reader = new Reader[10];
        for(int i = 0; i < 5; i++)
            writer[i] = new Writer(lock, "Writer " + i);
        for(int i = 0; i < 10; i++)
            reader[i] = new Reader(lock, "Reader " + i);
        for(int i = 0; i < 5; i++)
            writer[i].start();
        for(int i = 0; i < 10; i++)
            reader[i].start();
        for(int i = 0; i < 5; i++)
            writer[i].join();
        for(int i = 0; i < 10; i++)
            reader[i].join();
        System.out.println("control: " + (written - read));
    }
}
class Book{
    static int content;
}
class Writer extends Thread{
    ReadWriteLock l;
    String name;
    int written = 0;
    Writer(ReadWriteLock l, String name){
        this.l = l;
        this.name = name;
    }
    public void run(){
        int n = 0;
        Random r = new Random(100);
        while(n < 10000){
            l.writeLock().lock();
            writing(r);
            l.writeLock().unlock();
            n++;
            written++;
        }
        n8.written += written;
    }
    private void writing(Random r) {
```

```
        Book.content = r.nextInt(100);
        for (int i = 0; i < 10000; i++)
            for (int j = 0; j < 10000; j++) {}
    }
}
class Reader extends Thread{
    ReadWriteLock l;
    String name;
    int c;
    int read = 0;
    Reader(ReadWriteLock l, String name){
        this.l = l;
        this.name = name;
    }
    public void run(){
        int n = 0;
        while(n < 5000){
            l.readLock().lock();
            reading();
            l.readLock().unlock();
            n++;
            read++;
        }
        n8.read += read;
    }
    private void reading(){
        c = Book.content;
        for(int i = 0; i < 10000; i++)
            for(int j = 0; j < 10000; j++){}
    }
}
```

10. Предположим, что некий университет в США решил продемонстрировать свою политкорректность, применив известную доктрину Верховного суда США «Отделенный, но равный – по сути неравный» не только к расовой, но и к половой принадлежности, положив конец устоявшейся практике раздельных душевых для мужчин и женщин в своем кампусе. Но уступая традиции, университет вынес решение, что если в душевой находится женщина, то туда может зайти другая женщина, но не мужчина, и наоборот. Символ на сдвижном индикаторе на двери душевой показывает три возможных состояния:
    а) свободно;
    б) душевая занята женщинами;
    в) душевая занята мужчинами.
    Спроектируйте и реализуйте класс ShowerRoom на 10 душевых мест, в котором есть методы: для женщины, желающей войти, – womanWantsToEnter; для мужчины, желающего войти, – manWantsToEnter; для женщины, выходящей из душевой, – womanLeaves; для мужчины, выходящего из душевой, – manLeaves. Синхронизация с помощью блокировок. Проверить работу программы для душевой, 20-ти мужчин и 20-ти женщин, где мужчины и женщины представлены в виде потоков, которые некоторое время заняты своими делами,

затем хотят в течение некоторого времени помыться в одной на всех душевой, и так в бесконечном цикле.

Код Java:

```
import java.util.concurrent.locks.*;
import java.util.*;
import java.util.Random;

public class n9 {
    static int[] menItog = new int[20];
    static int[] womenItog = new int[20];
    static int persIn = 0;
    static boolean Full = (persIn == 10);
    static boolean wshow = false;
    static boolean mshow = false;
    public static void main(String[] args)throws InterruptedException{
        ReentrantLock lock = new ReentrantLock(true);
        Condition men = lock.newCondition();
        Condition women = lock.newCondition();
        Arrays.fill(menItog, 0);
        Arrays.fill(womenItog, 0);
        Man[] man = new Man[20];
        Woman[] woman = new Woman[20];
        for(int i = 0; i < 20; i++)
            man[i] = new Man(i, lock, men, women, Full, wshow, mshow);
        for(int i = 0; i < 20; i++)
            woman[i] = new Woman(i, lock, men, women, Full, wshow, mshow);
        for(int i = 0; i < 20; i++)
            man[i].start();
        for(int i = 0; i < 20; i++)
            woman[i].start();
        for(int i = 0; i < 20; i++)
            man[i].join();
        for(int i = 0; i < 20; i++)
            woman[i].join();
        System.out.println(Arrays.toString(menItog) + " muzhchin");
        System.out.println(Arrays.toString(womenItog) + " zhenshchin");
    }
}
class Shower{
    static Stack<Integer> showerStack = new Stack<>();
}
class Man extends Thread{
    int n;
    ReentrantLock lock;
    Condition men;
    Condition women;
    boolean Full;
    boolean wshow;
    boolean mshow;
    Man(int n, ReentrantLock lock, Condition men, Condition women, boolean
Full, boolean wshow, boolean mshow){
        this.n = n;
```

```
        this.lock = lock;
        this.men = men;
        this.women = women;
        this.Full = Full;
        this.wshow = wshow;
        this.mshow = mshow;
    }
    public void run(){
        int b = 0;
        try{
            while (b < 10000){
                Thread.sleep(10);
                Random r = new Random();
                lock.lock();
                while(Full || wshow)
                    women.await();
                if(mshow == false)
                    mshow = true;
                Shower.showerStack.push(r.nextInt(100));
                n9.persIn++;
                lock.unlock();
                lock.lock();
                n9.menItog[n]++;
                Shower.showerStack.pop();
                if(Shower.showerStack.empty()){
                    mshow = false;
                    men.signalAll();
                }
                lock.unlock();
                System.out.println(Arrays.toString(n9.menItog)+ " muzhchin");
                b++;
            }
        }
        catch(InterruptedException e){}
    }
}
class Woman extends Thread{
    int n;
    ReentrantLock lock;
    Condition men;
    Condition women;
    boolean Full;
    boolean wshow;
    boolean mshow;
    Woman(int n, ReentrantLock lock, Condition men, Condition women, boolean
Full, boolean wshow, boolean mshow){
        this.n = n;
        this.lock = lock;
        this.men = men;
        this.women = women;
        this.Full = Full;
        this.wshow = wshow;
        this.mshow = mshow;
```

```
    }
    public void run(){
        int b = 0;
        try{
            while (b < 10000){
                Thread.sleep(10);
                Random rnd = new Random();
                lock.lock();
                while(Full || mshow)
                    men.await();
                if(wshow == false)
                    wshow = true;
                Shower.showerStack.push(rnd.nextInt(100));
                n9.persIn++;
                lock.unlock();
                lock.lock();
                n9.womenItog[n]++;
                Shower.showerStack.pop();
                if(Shower.showerStack.empty()){
                    wshow = false;
                    women.signalAll();
                }
                lock.unlock();
                System.out.println(Arrays.toString(n9.womenItog)+ "
zhenshchin");
                b++;
            }
        }
        catch(InterruptedException e){}
    }
}
```