

EPAM Systems, RD Dep.
Практическое задание

JAVA.SE.07 Multithreading

REVISION HISTORY					
Ver.	Description of Change	Author	Date	Approved	
				Name	Effective Date
<1.0>	Первая версия	Игорь Блинов	<04.08.2011>		
<2.0>	Вторая версия: задания изменены согласно обновленному содержанию модуля	Ольга Смолякова	<12.11.2014>		

Legal Notice

This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM Systems.

Задание 1. Синхронизация 1

В текстовом (или xml) файле содержится информация о переводах средств со счета на счет. Создайте приложение, позволяющее в параллельном режиме обработать эту информацию (счета с приложения представляются собой объекты). Синхронизируйте код приложения используя ключевое слово `synchronized` (1 вариант) и библиотеку `java.util.concurrent` (2 вариант).

Задание 2. Синхронизация 2

Создать “универсальный” класс, позволяющий получить значение из любого `properties`-файла. Физическое чтение файла должно происходить только один раз. Учтите ситуацию, когда несколько потоков одновременно обращаются к одному и тому же файлу.

Задание 3. Управление многопоточным приложением

Перепишите код приложения для темы `Wait, notify` так, чтобы ситуация, когда все потоки хотят прочитать из очереди, не могла возникнуть.

```
package _java._se._07._waitnotify;
import java.util.ArrayList;
import java.util.List;
public class SharedResource {
    private List<Integer> list;

    public SharedResource() {
        list = new ArrayList<Integer>();
    }

    public void setElement(Integer element) {
        list.add(element);
    }

    public Integer getElement() {
        if (list.size() > 0) {
            return list.remove(0);
        }
        return null;
    }
}

package _java._se._07._waitnotify;
import java.util.Random;

public class UserResourceThread {
    public static void main(String[] args) throws InterruptedException {
        SharedResource res = new SharedResource();
        IntegerSetterGetter t1 = new IntegerSetterGetter("1", res);
```

Legal Notice

This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM Systems.

```
IntegerSetterGetter t2 = new IntegerSetterGetter("2", res);
IntegerSetterGetter t3 = new IntegerSetterGetter("3", res);
IntegerSetterGetter t4 = new IntegerSetterGetter("4", res);
IntegerSetterGetter t5 = new IntegerSetterGetter("5", res);

t1.start();
t2.start();
t3.start();
t4.start();
t5.start();

Thread.sleep(100);

t1.stopThread();
t2.stopThread();
t3.stopThread();
t4.stopThread();
t5.stopThread();

t1.join();
t2.join();
t3.join();
t4.join();
t5.join();

System.out.println("main");
}
}

class IntegerSetterGetter extends Thread {
    private SharedResource resource;
    private boolean run;

    private Random rand = new Random();

    public IntegerSetterGetter(String name, SharedResource resource) {
        super(name);
        this.resource = resource;
        run = true;
    }

    public void stopThread() {
        run = false;
    }

    public void run() {
        int action;

        try {
            while (run) {
                action = rand.nextInt(1000);
                if (action % 2 == 0) {
                    getIntegersFromResource();
                } else {
                    setIntegersIntoResource();
                }
            }
        }
    }
}
```

Legal Notice

This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM Systems.

```
        System.out.println("Поток " + getName() + " завершил
работу.");
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

private void getIntegersFromResource() throws InterruptedException {
    Integer number;

    synchronized (resource) {
        System.out.println("Поток " + getName()
            + " хочет извлечь число.");
        number = resource.getElement();
        while (number == null) {
            System.out.println("Поток " + getName()
                + " ждет пока очередь заполнится.");
            resource.wait();
            System.out
                .println("Поток " + getName() + "
возобновил работу.");
            number = resource.getElement();
        }
        System.out
            .println("Поток " + getName() + " извлек число
" + number);
    }
}

private void setIntegersIntoResource() throws InterruptedException {
    Integer number = rand.nextInt(500);
    synchronized (resource) {
        resource.setElement(number);
        System.out.println("Поток " + getName() + " записал число "
            + number);
        resource.notify();
    }
}
```