Assignment 2: Busy-wait and synchronization of Java threads

I. Introduction

In this class, we will synchronize java threads using 2 methods you are familiar with:

- Busy-wait
- signals (*wait*(), *notify*(), *notifyAll*()).
- Key-word *synchronized*

You may want to improve or refresh your knowledge about these methods in the following link:

http://docs.oracle.com/javase/tutorial/essential/concurrency/sync.html

II. Solve the following tasks:

A. A simple version of the Writer/Reader problem

You are provided with class Queue which only buffers one integer variable *n*. This class has two methods:

- void read(): prints the current value of *n*.
- void write(int x): the value of *x* is assigned to *n*.

```
class Queue{

    int n;

   public void read() {
        System.out.println(n);
    }

    public void write(int x) {

        n = x;
    }
}
```

Your Queue must be shared by 1 consumer and 1 writer. That is to say, you have to create one thread Writer which updates the value of *n*, and one thread Reader which displays *n* on the screen.

```
class Writer implements Runnable {
    Queue q;

    Writer(Queue q){
        this.q=q;
        new Thread(this, "Writer").start();
    }

    public void run(){

        int i=0;
        for(int j=0;j<10;j++){
            try {
                Thread.sleep((long) (Math.random()*500));
            } catch (InterruptedException e) {

                e.printStackTrace();
            }
            q.write(i++);
        }
    }
}
```

```
class Reader implements Runnable{
    Queue q;

    Reader (Queue q){
        this.q=q;
        new Thread(this,"Reader").start();
    }

    public void run(){
        for(int i=0;i<10;i++){
            try {
                Thread.sleep((long) (Math.random()*500));
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            q.read();
        }
    }
}
```

```
public class Main {

    public static void main(String args[]){
        Queue queue;

        queue=new Queue();

        new Writer(queue);
        new Reader(queue);
    }
}
```

Run the code above…

You have to improve the code so that threads work in turns: write, read, write, read,…

In order to do so:

1. **Use condition synchronization by means of busy wait.**
2. **Use the *synchronized* keyword and signals.**

You may want to read

http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#notify%28%29

3. Add one extra Reader

Each value of *n* must be read (printed) by the 2 readers. The *read()* method must show which reader is printing the value.

B. The Ornamental Garden

There is an Ornamental Garden with 2 entrances accessible through a turnstile each. The manager of the Garden wants to limit the access up to 40 people, so a global counter needs to be implemented which sums up the individual counter from each turnstile.

You can download from Campus Virtual an Applet which simulates the problem of the Ornamental Garden, with 2 turnstiles to grant access into the garden.

Since the global counter is shared, we need to grant mutual exclusion when accessing to it.

1. Identify the critical section and implement the Dekker algorithm.

2. Now solve the problem by using the *synchronized* key-word.

III. Evaluation

We will use **2 lab sessions** to solve this assignment. Please upload your final code through Campus Virtual.

Create a new package for each task and use appropriate names, such as “Ass1A”, “Ass1B”,…

Remember you will be evaluated using this rubric:

Respect to the global interests of the group	The group is not well organized. Members did not share the work to carry on. ***0 points***	Some problems arised, and many were solved. ***1 points***	No problems have affected the performance of the group, or were successfully solved, ***2 points***	
Correctness of the assignment	It was not solved correctly at all. ***0 points***	Only a few points were right ***1 points***	Mostly correct ***3 points***	Perfect ***4 points***
Correctness of answers to questions about the code (Algorithms and structures used)	No answers/wrong answers ***0 points***	A few correct, or correct but only member answered ***1 points***	Both members gave mostly correct answers. ***2 points***	Both members answered correctly to all questions. ***4 points***