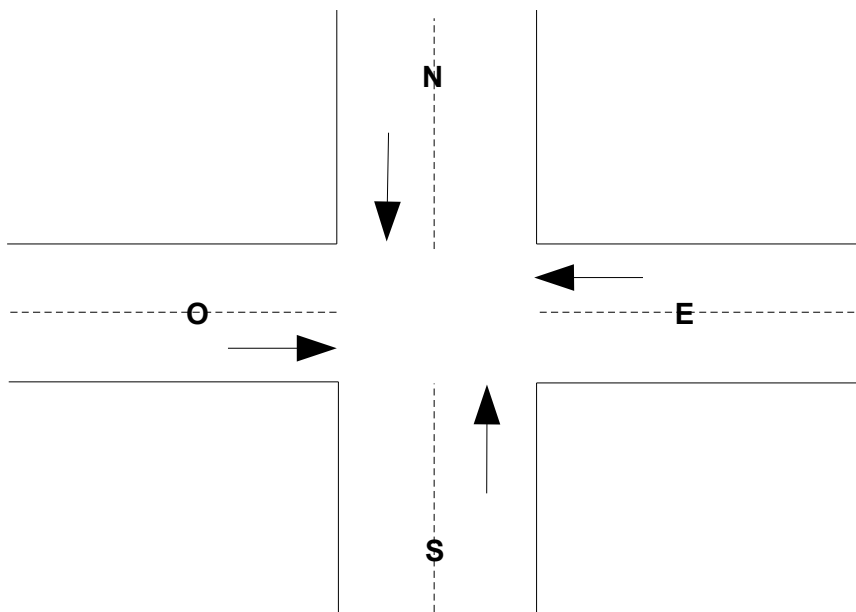


**SISTEMI OPERATIVI IIN/IEL/IDT**  
**INFORMATICA INDUSTRIALE E SISTEMI OPERATIVI IDI**  
**SISTEMI DI ELABORAZIONE P.O.**  
**prova scritta del 05.09.2006**

Nome \_\_\_\_\_ Cognome \_\_\_\_\_  
Matricola \_\_\_\_\_ Corso di laurea IIN IEL IDT IDI

Si consideri l'incrocio riportato in figura:



Le auto si muovono secondo le direttrici N-S/S-N e O-E/E-O. Le auto non possono cambiare direttrice di marcia (in pratica non possono svoltare all'incrocio, ma solo procedere nella loro direzione).

L'accesso all'incrocio, in un certo istante, deve essere possibile solo alle auto che si muovono lungo direttrici compatibili (N-S e S-N possono transitare insieme inibendo l'accesso alle vetture lungo la direttrice ortogonale). Pertanto, prima di accedere l'incrocio le auto devono verificare la disponibilità dello stesso. Nel caso l'incrocio non sia disponibile le auto si sospendono. Al loro risveglio non è richiesto di preservare l'ordinamento nel loro attraversamento dell'incrocio.

Scrivere un programma Java che sincronizzi l'accesso all'incrocio utilizzando i metodi sincronizzati come primitiva di sincronizzazione, e tralasciando le problematiche relative a starvation e deadlock.

Discutere le modifiche alla soluzione precedente eventualmente necessarie al fine di garantire l'assenza di starvation e deadlock.

## Soluzione

```
package incrocio;
```

```
public class Main {  
    private static final int autoNS = 5;  
    private static final int autoOE = 5;  
  
    public static void main( String[] args ) {  
        Incrocio in = new Incrocio();  
        int id = 0;  
        Auto auto;  
        for (int i=0;i<autoNS;i++) { // crea e avvia le auto lungo la direttrice verticale  
            auto = new Auto( id++, 0, in );  
            auto.start();  
            auto = new Auto( id++, 1, in );  
            auto.start();  
        }  
        for (int i=0;i<autoOE;i++) { // crea e avvia le auto lungo la direttrice orizzontale  
            auto = new Auto( id++, 2, in );  
            auto.start();  
            auto = new Auto( id++, 3, in );  
            auto.start();  
        }  
    }  
}
```

```
public class Incrocio {  
    private int nV; // auto presenti nell'incrocio lungo la direttrice verticale  
    private int nO; // auto presenti nell'incrocio lungo la direttrice orizzontale  
  
    public Incrocio() {  
        nV = 0; // incrocio vuoto lungo la direttrice verticale  
        nO = 0; // incrocio vuoto lungo la direttrice orizzontale  
    }  
  
    // acquisizione dell'incrocio lungo la direttrice verticale  
    public synchronized void acquireV() {  
        // la prima auto lungo la direttrice verticale inibisce la direttrice opposta o si blocca  
        while ( nO > 0 ) {  
            try {  
                wait();  
            }  
            catch ( InterruptedException ie ) {  
                ie.printStackTrace();  
            }  
        }  
    }  
}
```

```

        nV ++;

        System.out.println( "Nell'incrocio ci sono " + nV + " auto in direzione verticale" );
    }

    // acquisizione dell'incrocio lungo la direttrice orizzontale
    public synchronized void acquireO() {
        // la prima auto lungo la direttrice orizzontale inibisce la direttrice opposta o si blocca
        while ( nV > 0 ) {
            try {
                wait();
            }
            catch ( InterruptedException ie ) {
                ie.printStackTrace();
            }
        }
        nO ++;

        System.out.println( "Nell'incrocio ci sono " + nO + " auto in direzione orizzontale" );
    }

    // rilascio dell'incrocio lungo la direttrice verticale
    public synchronized void releaseV() {
        // l'ultima auto lungo la direttrice verticale libera la direttrice opposta
        nV --;
        if ( nV == 0 )
            notifyAll();

        System.out.println( "Nell'incrocio ci sono " + nV + " auto in direzione verticale" );
    }

    // rilascio dell'incrocio lungo la direttrice orizzontale
    public synchronized void releaseO() {
        // l'ultima auto lungo la direttrice orizzontale libera la direttrice opposta
        nO --;
        if ( nO == 0 )
            notifyAll();

        System.out.println( "Nell'incrocio ci sono " + nO + " auto in direzione orizzontale" );
    }
}

public class Auto extends Thread {
    private int id; // numero identificativo dell'auto
    private int dir; // direzione di moto (0 (N-S), 1 (S-N), 2 (O-E) e 3 (E-O))
    private Incrocio in;

```

```

public Auto( int id, int dir, Incrocio in ) {
    this.id = id;
    this.dir = dir;
    this.in = in;
}

public void run() {
    System.out.println( "Avviata auto " + id + " in direzione " + dir );
    while ( true ) {
        // acquisizione incrocio
        if ( dir < 2 )
            in.acquireV();
        else
            in.acquireO();

        move( 2 );    // attraversamento dell'incrocio

        System.out.println( "l'auto " + id + "(" + dir + ") ha attraversato l'incrocio" );

        // rilascio incrocio
        if ( dir < 2 )
            in.releaseV();
        else
            in.releaseO();

        // l'auto si ripresenta all'incrocio dopo un certo tempo nella stessa direzione
        move( 5 );
    }
}

public static void move( int sec ) {
    try{    // simula il tempo necessario all'auto per muoversi
        Thread.sleep( (int)(1000*Math.random()*sec) );
    }
    catch ( InterruptedException ie ) {
        ie.printStackTrace();
    }
}
}

```