[monitor] Coda prioritaria

La verifica consiste nell'implementazione di un monitor `CodeMonitor` per la sincronizzazione di un certo numero di thread `Persona`. I thread `Persona` rappresentano gli utenti di un servizio con uno sportello e **due code**: una **normale** e una **prioritaria**. Le persone, in base al proprio status (**verde**=prioritarie, **blu**=normali), si mettono in fila sulla rispettiva coda.

Gli utenti della coda prioritaria hanno la **precedenza** sugli altri, ma solo se sono arrivati alla fine della coda e sono effettivamente in attesa del servizio (se sono in movimento e si stanno accodando non hanno precedenza!).

Un thread `Persona` si comporta come segue:

```
1.  Thread Persona(boolean priority) {
2.      // si accoda e procede fino alla fine della coda
3.
4.      // è alla fine della coda attende lo sportello
5.      sportello = p.attendiSportello(priority);
6.
7.      // va allo sportello e attende il servizio
8.
9.      // libera lo sportello e esce
10.     p.liberaSportello(sportello);
11. }
```

I metodi del monitor che dovete implementare sono i seguenti:

```
1.  // La persona è la prossima ad essere servita e attende che si liberi
2.  // lo sportello. Se la persona è sulla coda prioritaria (priority è
3.  // true) appena lo sportello si libera la persona può procedere. Se
4.  // la persona è sulla coda normale (priority è false) la persona
5.  // attende che non ci siano altre persone in coda prioritaria già
6.  // in attesa e che lo sportello si liberi. In altre parole, dà
7.  // la precedenza a persone in attesa in coda prioritaria.
8.  synchronized void attendiSportello(boolean priority)
9.          throws InterruptedException {
10.     if (priority) {
11.         // gestione coda prioritaria
12.     } else {
13.         // gestione coda non prioritaria
```

```
14.         }
15.     }
16.
17.     // La persona ha raggiunto lo sportello, ha fruito del servizio e ora
18.     // lo libera
19.     public synchronized void liberaSportello() {
20.     }
```

Scaricate la verifica da qui e modificate il file **CodeMonitor.java**. Quando un thread `Persona` viola le regole delle code diventa rosso e viene stampato un messaggio di errore (**Attenzione**: i test non sono esaustivi, se non dà messaggi di errore non è detto che il codice sia corretto!).

Per compilare ed eseguire il programma, usate i seguenti comandi:

```
> javac Code.java
> java Code
```

NOTA BENE

- Consegnate solo il file **CodeMonitor.java**;
- **Commentate** in maniera appropriata il vostro programma: in particolare, inserite un commento iniziale in cui spiegate l'idea risolutiva. Soluzioni non commentate **NON** saranno valutate.
- **NON COPIATE!!!**

Nascondi soluzione

```
/*
 * COMMENTO GENERALE (necessario per la sufficienza!):
 *
 * - Come si è arrivati a scegliere le strutture dati utilizzate per la
 *   sincronizzazione
 *
 * Per la sincronizzazione utilizziamo:
 *
 * 1. un booleano 'occupato' che tiene traccia dello stato di occupazione
 *    dello sportello. Il booleano è inizializzato a false per indicare
 *    che lo sportello è libero;
 * 2. un booleano 'p_attende' che tiene traccia se una persona è in attesa
 *    sulla coda prioritaria. Il booleano è inizializzato a false per
 *    indicare che nessuna persona inizialmente è in attesa sulla coda
 *    prioritaria.
 *
 * - Come funziona, intuitivamente, la sincronizzazione
```

```
 *
 * la funzione 'attendiSportello' si comporta diversamente a seconda che la
 * coda sia prioritaria o meno:
 *
 * 1. Coda prioritaria: la persona attende solo se lo sportello è
'occupato'.
 *    Tuttavia, è necessario porre 'p_attende' a true prima di entrare nel
 *    nel ciclo while in modo da segnalare una potenziale attesa
prioritaria.
 *    La variabile 'p_attende' viene rimessa a false dopo l'uscita dal
while.
 *    In questo modo le persone in coda non prioritaria potranno dare la
 *    precedenza a quelle in coda prioritaria.
 * 2. Coda non prioritaria: la persona attende se c'è qualcuno già in attesa
 *    in coda prioritaria oppure se lo sportello è occupato. In questo modo
 *    viene data la precedenza alla coda prioritaria.
 *
 * La persona che va allo sportello lo occupa settando 'occupato' a true e
 * poi ponendolo a false nella funzione 'liberaSportello'.
 *
 * - Come sono state utilizzate le wait e le notify / notifyAll
 *
 * La wait viene usate in 'attendiSportello' nel caso che
 *
 * - la persona è sulla coda prioritaria e lo sportello è occupato
 *   ('occupato' è true)
 * - la persona è sulla coda non prioritaria e c'è una persona in attesa
 *   sulla coda prioritaria oppure lo sportello è occupato ('p_attende' ||
 *   'occupato')
 *
 * La notifyAll viene fatta su 'liberaSportello' dopo che lo sportello è
 * stato liberato ('occupato' è posto a false). E' necessaria la notifyAll
 * per svegliare le persone in entrambe le code.
 * Le wait sono racchiuse in un while in modo da ri-verificare la
 * condizione di bloccaggio. Se venisse sbloccato erroneamente un thread
 * questo non sarebbe un problema perché si riblocchererebbe autonomamente.
 *
 */

public class CodeMonitor {
    // indica se lo sportello è occupato
    private boolean    occupato=false;

    // indica se una persona è in attesa sulla coda prioritaria
    // E' sufficiente un booleano perché un solo thread alla volta
    // attende in coda
    private boolean    p_attende=false;

    // La persona è la prossima ad essere servita e attende che si liberi
    // lo sportello. Se la persona è sulla coda prioritaria (priority è
    // true) appena lo sportello si libera la persona può procedere. Se
    // la persona è sulla coda normale (priority è false) la persona
    // attende che non ci siano altre persone in coda prioritaria già
```

```
    // in attesa e che lo sportello si liberi. In altre parole, dà
    // la precedenza a persone in attesa in coda prioritaria.
    synchronized void attendiSportello(boolean priority)
            throws InterruptedException {
        if (priority) {
            // indica che sta per attendere sulla coda prioritaria
            p_attende=true;

            // attende se lo sportello è occupato
            while( occupato )
                wait();

            // indica che non è più in attesa sulla coda prioritaria
            p_attende=false;
        } else {
            // attende se c'è qualcuno in attesa su coda prioritaria
            // o se lo sportello è occupato
            while( p_attende || occupato )
                wait();
        }
        occupato=true;      // occupa lo sportello
    }

    // La persona ha raggiunto lo sportello, ha fruito del servizio e ora
    // lo libera
    public synchronized void liberaSportello() {
        // libera lo sportello
        occupato=false;

        // notifica eventuali thread in attesa dello sportello
        notifyAll();
    }
}
```