

Quarta esercitazione

Java Thread

Stefano Monti
stefano.monti6@unibo.it

Esercizio 1

Si realizzi un programma Java per la gestione e la elaborazione di stringhe.

In particolare, il programma dovrà

- ricevere e memorizzare stringhe in un buffer di **dim. finita N** da un **thread produttore**
- applicare **in sequenza** 3 operazioni (delegate ad altrettanti thread *operatori*) al **primo elemento del buffer**
 - **troncamento** della seconda metà della stringa
 - **append** di una stringa relativa alla data corrente
 - **stampa** a video e **prelievo** dal buffer

2

Note alla soluzione

Vincoli di sincronizzazione

- accesso al buffer in **mutua esclusione**
 - sia per aggiunta stringhe
 - sia per operazioni di troncamento/append/prelievo
- problema di sincronizzazione:
 - produttore non deve scrivere se il buffer è pieno
 - operatori non possono agire su buffer vuoto
- vincolo temporale
 - necessità di **ordinare** le operazioni dei tre operatori

3

Uso dei semafori

- accesso al buffer in mutua esclusione
 - utilizzo di un **semaforo binario**
- problema di sincronizzazione:
 - utilizzo di un semaforo di dimensione N
- vincolo temporale
 - utilizzo di 3 semafori
 - uno per ciascuna operazione
 - inizialmente a 0 --> il termine della operazione precedente sblocca quella seguente

4

```

public class Risorsa {
    final int capienzaMaxBufferStringhe = 10; // N
    // --- Semafori -----
    Semaforo s_aggiuntaStringa,
        s_operazione1, s_operazione2, s_operazione3,
        sM;
    java.util.List bufferStringhe; //Buffer
    int numeroStringhe = 0; // Contatore stringhe presenti nel buffer
    int stringheElaborate = 0;

    public Risorsa (){
        s_aggiuntaStringa = new Semaforo(capienzaMaxBufferStringhe);
        s_operazione1 = new Semaforo(0);
        s_operazione2 = new Semaforo(0);
        s_operazione3 = new Semaforo(0);
        sM = new Semaforo(1);
        bufferStringhe = new java.util.Vector();

    }

    public void nuovaStringa(String string){ }
    public void operazione1(){}
    public void operazione2(){}
    public void operazione3(){}
}

```

semaforo Buffer
semafori ordinamento
semaforo mutua esclusione

5

```

public class Risorsa {
    //...
    public void nuovaStringa(String string){
        s_aggiuntaStringa.p();
        sM.p();
        bufferStringhe.add(string);
        numeroStringhe++;
        if (numeroStringhe == 1)
            s_operazione1.v();
        sM.v();
    }

    public void operazione1(){
        s_operazione1.p();
        sM.p();
        String daElaborare = (String) bufferStringhe.get(0);
        daElaborare = daElaborare.substring(0, daElaborare.length()/2);
        bufferStringhe.set(0, daElaborare);
        s_operazione2.v();
        sM.v();
    }
}

```

ATTENZIONE: è necessaria la condizione *if*?
 Cosa succederebbe se la togliessi?

6

```

public class Risorsa {
    ...
    public void operazione2(){
        s_operazione2.p();
        sM.p();
        String daElaborare = (String) bufferStringhe.get(0);
        daElaborare = daElaborare + (new java.util.Date()).toString();
        bufferStringhe.set(0, daElaborare);
        s_operazione3.v();
        sM.v();
    }

    public void operazione3(){
        s_operazione3.p();
        sM.p();
        stringheElaborate++;
        bufferStringhe.remove(0);
        numeroStringhe--;
        if (numeroStringhe > 0)
            s_operazione1.v();
        s_aggiuntaStringa.v();
        sM.v();
    }
}

```

7

Esercizio 2

Si realizzi un programma Java che simuli la gestione di una cucina di un ristorante.

In particolar modo:

- due thread AiutoCuoco si incaricano di portare sul tavolo di preparazione (risorsa condivisa) gli ingredienti necessari (di due tipi diversi)
- un thread Chef sovrintende alla preparazione del piatto,
 - attende che siano disponibili le quantità necessarie di ingredienti
 - quando disponibili, le preleva dal tavolo e prepara il piatto

8

Esercizio 2- dettagli

- Tavolo ha capacità limitata (diversa) per ingrediente1 e ingrediente2 (risp. M1 e M2)
- Ciascun AiutoCuoco porta sempre una unità di ingrediente
- Per preparare il piatto, Chef ha bisogno di quantità prestabilite per i 2 ingredienti
 - Q1 per ingrediente1
 - Q2 per ingrediente2

9

Esercizio 2 - Sincronizzazione

- Tavolo è una risorsa condivisa con **due** buffer, uno per ciascun ingrediente
 - gestione della capacità: non è possibile altre unità di ingredienteX se il tavolo ha già saturato la capacità per X
- necessità di **mutua esclusione** tra thread che accedono al tavolo
- thread Chef deve attendere che siano disponibili gli ingredienti, e soltanto allora, iniziare la preparazione del piatto (**ordinamento**)

10