NON SPEGNERE IL PC A FINE ESAME

Corso di Sistemi Operativi e Reti

Prova scritta di FEBBRAIO 2018

ISTRUZIONI

1. **Rinomina** la cartella chiamata "CognomeNomeMatricola" che hai trovato sul Desktop e in cui hai trovato questa traccia, sostituendo "Cognome" "Nome" e "Matricola" con i tuoi dati personali;
2. **Carica** tutto il materiale didattico che vorrai usare sul Desktop; puoi farlo solo nei primi 5 minuti della prova;
3. **Svolgi** il compito; lascia tutto il sorgente che hai prodotto nella cartella di cui al punto 1;
4. **Quando programmi in Java**, metti tutte le tue classi in un package con un nome che segue lo schema CognomeNomeMatricola.

5. **Quando hai finito** lascia la postazione facendo logout, dopo aver cancellato tutte le dispense e il materiale didattico, e

lascia il PC acceso

SALVA SPESSO il tuo lavoro

NON SPEGNERE IL PC A FINE ESAME

ESERCIZIO 1 (Programmazione multithread. Punti: 0-20)

Uno `SharedInteger` è una classe thread-safe che incapsula la gestione di un numero intero. Una istanza di SharedInteger deve possedere i seguenti metodi:

`int get()`. Restituisce il valore corrente dello shared integer.
`void set(int i)`. Imposta il valore corrente dello shared integer a **`i`**.

`void inc(SharedInteger I)`. Aumenta il valore di `this` del valore di `I`.
`void inc(int i)`. Aumenta il valore di `this` del valore di `i`

`int waitForAtLeast(int soglia)`. Si pone in attesa bloccante finchè il valore di `this` non è almeno pari a `soglia`, e infine restituisce il valore corrente di `this`.

`void setInTheFuture(SharedInteger I, int soglia, int valore)`

si pone in attesa bloccante finchè `I` non raggiunge almeno il valore `soglia`, dopodichè imposta `this` a `valore`.

I metodi di classe devono essere thread-safe, pertanto devono essere obbligatoriamente eliminate: possibili situazioni di race condition; possibili situazioni di deadlock.

NON SPEGNERE IL PC A FINE ESAME

NON SPEGNERE IL PC A FINE ESAME

CI SONO DEI PUNTI AMBIGUI NELLA TRACCIA? COMPLETA TU

È parte integrante di questo esercizio completare le specifiche date nei punti non esplicitamente definiti, introducendo o estendendo tutte le strutture dati laddove si ritenga necessario, e risolvendo eventuali ambiguità.

POSSO CAMBIARE IL PROTOTIPO DEI METODI RICHIESTI? NO

Non è consentito modificare il prototipo dei metodi se questo è stato fornito. Potete aggiungere qualsivoglia campo e metodo di servizio, e qualsivoglia classe ausiliaria, ma NON variare l'interfaccia dei metodi pubblici già specificati.

CHE LINGUAGGIO DEVO USARE? JAVA 7 O SUCCESSIVO

Il linguaggio da utilizzare per l'implementazione è Java. *È* consentito usare qualsiasi funzione di libreria di Java 8 o successivi.

MA IL MAIN() LO DEVO SCRIVERE? E I THREAD DI PROVA? SOLO PER FARE IL TUO DEBUG

Non è esplicitamente richiesto di scrivere un `main()` o di implementare esplicitamente del codice di prova, anche se lo si suggerisce vivamente per testare il proprio codice prima della consegna.

NON SPEGNERE IL PC A FINE ESAME

ESERCIZIO 2 (Linguaggi di scripting. Punti 0-10)

Lo scopo dell'esercizio è di creare uno script Perl, che chiameremo `translate.pl`, in grado di tradurre una o più parole da una lingua ad un'altra.

Si scriva quindi un programma che potrà essere invocato nel seguente modo:

```
translate.pl OPTION [file_dictionary] [string]
```

Come si può notare il parametro `OPTION` è obbligatorio, mentre i parametri `file_dictionary` e il parametro `string` sono opzionali (sono obbligatori solo per alcuni comandi, si veda sotto).

I possibili valori di `OPTION` e i task da eseguire per ciascuno di essi sono:

- **`-t`**

  Quando si usa l'opzione `-t` il parametro `file_dictionary` è obbligatorio mentre il parametro `string` rimane opzionale. Bisogna effettuare la traduzione di una o più parole separate da virgola "," passate come parametro (nel caso sia presente il parametro `string`) oppure sarà necessario ricevere l'input delle parole da tradurre tramite standard input.

  Nota che il mach all'interno del file dictionary dovrà essere ***case insensitive*** (non si fa distinzione tra lettere maiuscole e minuscole).

  `Esempio:`

  Lo script potrà essere lanciato nel seguente modo:

  ```
  ./translate.pl -t it_en "abbellire,riflesso,zuppa"
  ```

  e dovrà produrre in output:

  ```
  La traduzione di abbellire è:  BEAUTIFY;  EMBELLISH;  PRETTIFY;  POSH UP
  La traduzione di riflesso è:  REFLECTION;  REFLEXION;  REFLEX;  EFFECT
  La traduzione di zuppa è:  SOUP
  ```

  alternativamente sarà possibile eseguire lo script nel seguente modo:

  ```
  ./translate.pl -t it_en
  ```

NON SPEGNERE IL PC A FINE ESAME

lo script continuerà chiedendo all'utente di inserire in input una serie di parole separate da virgola

```
Inserire le parole da tradurre separate da virgola (,)
```

ed infine dovrà produrre in output:

```
La traduzione di abbellire è:  BEAUTIFY;  EMBELLISH;  PRETTIFY;  POSH UP
La traduzione di riflesso è:  REFLECTION;  REFLEXION;  REFLEX;  EFFECT
La traduzione di zuppa è:  SOUP
```

- **-h**

Stampa in output la cronologia di tutte le parole tradotte dalla prima all'ultima esecuzione dello script in ordine di comparsa.

Hint: è possibile (ma non obbligatorio) creare un file history in cui si salvano tutte le parole ricercate durante l'esecuzione dello script. Ogni parola potrebbe essere scritta in una nuova riga.

NON SPEGNERE IL PC A FINE ESAME