Prova pratica Sistemi Operativi sessione invernale 2020-2021
Prof. Emanuele Lattanzi

1. Specifiche generali

Scrivere un programma multithread che simuli il funzionamento di un sistema Internet of Things (IoT) per la raccolta di informazioni ambientali e l'aggregazione su cloud. In particolare, il sistema sarà costituito da un numero *`nSensors`* di sensori in grado di accedere all'ambiente e misurare temperatura e luminosità. Le condizioni ambientali saranno controllate da un thread (*`WeatherConditioner`*) che avrà il compito di modificare periodicamente i valori di temperatura e luminosità presenti nell'ambiente secondo la relazione descritta in seguito.

I sensori, dopo aver effettuato la misurazione, invieranno i dati in cloud dove saranno memorizzati utilizzando dei buffer circolari a capacità prefissata agendo, in questo modo, come dei produttori. Saranno, inoltre, presenti *`nUsers`* utenti che sottometteranno richieste per ottenere i valori medi di temperatura e luminosità registrati in cloud. Ogni lettura da parte di un utente sarà ottenuta mediando 4 valori cariati nel buffer in ordine FIFO. La lettura comporterà la rimozione dei 4 valori letti come avviene per un normale consumatore. Avendo il buffer una capacità limitata i consumatori dovranno attendere la presenza di dati nel buffer ed i produttori dovranno attendere in caso di mancanza di spazio. I sensori in attesa di scrivere e gli utenti in attesa di leggere saranno gestiti in ordine FIFO.

La simulazione terminerà quando tutti gli utenti avranno effettuato 100 letture ciascuno.

2. Specifiche dell'oggetto `Cloud`

- L'oggetto `Cloud` sarà un oggetto condiviso tra i thread che rappresentano i sensori ed i thread che rappresentano gli utenti.
- L'oggetto `Cloud` dovrà contenere i buffer circolari di capacità fissa (30 posizioni) utilizzati per contenere i valori di temperatura e luminosità ricavati dall'ambiente.
- L'oggetto `Cloud` esporrà il metodo pubblici *`readAverageTemp(…)`* e *`readAverageLight()`*, invocati dagli utenti per ottenere la media di 4 valori consecutivi presenti nel buffer e consumare i dati. Esporrà, inoltre, il metodo *`writeData(…)`*, invocato dai sensori per inserire i dati nei buffer circolari. Come già detto, facendo uso di buffer a capacità limitata, i metodo *`readAverageTemp(…)`* e *`readAverageLight()`* dovranno sospendere gli utenti in attesa dell'arrivo dei dati mentre, il metodo `writeData()` dovrà sospendere i sensori in attesa di spazio nel buffer.

3. Specifiche dell'oggetto `Environment`

- L'oggetto `Environment` sarà un oggetto condiviso tra i thread che rappresentano i sensori ed il singolo thread che rappresenta il `WeatherConditioner`.
- L'oggetto `Environment` dovrà contenere i valori correnti di temperatura e luminosità.
- L'oggetto `Environment` esporrà i metodi pubblici *`measureParameters(…)`* e *`updateParameters(…)`*, invocati, rispettivamente, dai sensori per misurare la temperatura e luminosità e dal thread *`WehaterConditioner`* per modificarne i valori correnti

4. Specifiche del thread `Sensor`:

- Ogni sensore dovrà essere emulato attraverso l'utilizzo di un singolo thread.
- Ogni sensore, sarà caratterizzato da un errore di lettura, espresso in percentuale, nel range intero [-10, 10] scelto casualmente all'atto della creazione del sensore. Questo significa, per esempio, che se un sensore è stato creato con un errore del -2%, ogni lettura che effettuerà sarà riportata in cloud con un valore decrementato del 2% rispetto al valore reale presente nell'ambiente.
- Il comportamento di ogni entità sensore può essere descritto dai seguenti passi:

  Fino alla fine della simulazione:
  1. Misura i parametri ambientali invocando il metodo `measureParameters()` dell'oggetto `Environment`.
  2. Applica l'errore percentuale
  3. Invia i dati in cloud invocando il metodo `writeData(…)` dell'oggetto `Cloud`
  4. Attende 400 millisecondi.
  5. Torna al punto 1

5. Specifiche del thread `User`:

- Ogni utente dovrà essere emulato attraverso l'utilizzo di un singolo thread.
- Il comportamento di un utente può essere descritto dai seguenti passi:

  Per 100 volte:
  1. Attende un tempo estratto casualmente nell'intervallo [0,99] millisecondi.
  2. Effettua la richiesta per ottenere la temperatura media invocando il metodo `readAverageTemp(…)` dell'oggetto `Cloud`.
  3. Effettua la richiesta per ottenere la luminosità media invocando il metodo `readAverageLight(…)` dell'oggetto `Cloud`.
  4. Torna al punto 1

6. Specifiche del thread `WeatherConditioner`:

- Il thread `WeatherConditioner` è un thread che ha il compito di modificare i parametri ambientali all'interno dell'oggetto Environment.
- Questo thread, per tutta la durata della simulazione, ogni 400 millisecondi aggiornerà i valori di temperatura e luminosità secondo le seguenti relazioni (i pedici *t0* e *t1* identificano, rispettivamente, il valore passato ed il valore futuro):
  - $Luminosità_{(t1)} = Luminosità_{(t0)} + 1000$; *con luminosità iniziale = 0*
  - $Temperatura_{(t1)} = 10+0.00022* Luminosità_{(t1)}$

7. Specifiche di sincronizzazione ed implementazione:

- Il progetto deve essere sviluppato in ambiente Linux/Windows/macOS utilizzando **esclusivamente il linguaggio java su piattaforma Netbeans.**
- Il progetto deve essere esente da deadlock.
- La muta esclusione, l'accesso a variabili e a metodi condivisi e l'attesa dei processi dovranno essere ottenute utilizzando le primitive di sincronizzazione messe a disposizione

dal linguaggio java (ad esempio: package java.util.concurrent.) ad esclusione del costrutto monitor (per chiarezza non sarà possibile utilizzare il costrutto *synchronized*).

8. Specifiche dati in input:

- Il programma dovrà ricevere in input, tramite riga di comando o tramite file di specifica, i seguenti parametri di input:
  - Numero di sensori (`nSensors`).
  - Numero di utenti (`nUsers`).

9. Specifiche dati di output:

- Al termine della simulazione il programma dovrà scrivere, su di un file di testo oppure sulla console, **la media e la deviazione standard** del tempo necessari per effettuare la lettura della temperatura e della luminosità da parte degli utenti (le due letture successive saranno considerate come una sola azione e quindi misurate insieme). Per rilevare i tempi d'attesa è consigliato l'utilizzo del metodo `currentTimeMillis()` della classe `System`.

10. Documentazione

- Il progetto Netbeans deve essere corredato dalla seguente documentazione:
  - Eventuali file di configurazione dell'applicativo
  - Relazione in formato PDF contenente:
    - Identificazione studente/i: Nome cognome e matricola.
    - Descrizione della progettazione: si descrivano i motivi che hanno portato alle attuali scelte progettuali. Ci si può avvalere anche di schemi a blocchi o figure.
    - Descrizione dell'implementazione: si descrivano le scelte implementative anche di natura tecnica.
    - Contributo degli studenti (solo nel caso di progetto svolto in coppia): per ogni studente si dovrà descrivere il ruolo avuto nel gruppo ed il relativo contributo specificando, ad esempio, quale parte del progetto/documentazione è stata implementata/redatta.
    - Testing e commenti ai risultati ottenuti: **la relazione dovrà contenere** una sezione dedicata al test del programma che riporti almeno:
      1. **Un grafico** che mostri l'andamento del tempo medio (assieme alla deviazione standard) impiegato dagli utenti per ottenere le letture della temperatura e della luminosità al variare del numero di sensori (`nSensors`).
      2. **Un grafico** che mostri l'andamento del tempo medio (assieme alla deviazione standard) impiegato dagli utenti per ottenere le letture della temperatura e della luminosità al variare del numero di utenti (`nUsers`).
      3. **Due grafici** che mostrino la relazione che sussiste tra la luminosità e la temperatura misurate da un sensore, scelto dallo studente, in una simulazione dove, nel primo grafico `nSensors` = 100 e `nUsers` = 4 e nel secondo dove `nSensors` = 5 e `nUsers` = 4.
      4. **Dei commenti critici ai grafici** sopra descritti che ne giustifichino l'andamento. *SUGGERIMENTO: per ottenere i grafici 1) e 2) fare un set*

di simulazioni tenendo fisso il valore di una delle due variabili e far variare la seconda. Assicurarsi di inizializzare le variabili che vengono tenute fisse ad un valore significativo per lo scopo della simulazione. Come tipo di grafico si consiglia l'uso di uno scatter plot (dispersione). Per commentare gli andamenti dei grafici si consiglia di fare uso di uno strumento di regressione (come la linea di tendenza in Microsoft Excel)

- La relazione **NON deve contenere l'intero listato del codice**.
- La relazione **NON deve superare le 10 pagine** a pena bocciatura del progetto.

11. Modalità di presentazione del progetto:

- Il presente progetto può essere presentato individualmente o in coppia, fino ad **UNA SETTIMMANA prima dell'appello** d'esame in cui s'intende sostenere la prova orale. Per le date di consegna si faccia riferimento al calendario esami.
- La consegna del progetto dovrà avvenire esclusivamente attraverso la piattaforma blended.uniurb.it accendendo alla sezione "**Consegna progetto d'esame sessione estiva 2020-2021**". Gli studenti che non avessero accesso all'anno accademico 2020-2021 sulla piattaforma blended.uniurb.it potranno richiedere l'attivazione alla segreteria inviando una mail a: anya.pellegrin@uniurb.it
- La consegna dei progetti fatti in coppia avverrà da parte di uno solo degli studenti.
- Il progetto deve essere caricato sulla piattaforma blended.uniurb.it sotto forma di file compresso in modalità tar.gz o zip contenente il progetto Netbeans e la relazione in formato PDF.
- Progetti consegnati fuori tempo limite o consegnati al difuori delle piattaforma blended (per email ad esempio) non verranno accettati.
- Progetti consegnati senza relazione oppure accompagnati da relazione che non soddisfi le specifiche richieste non saranno ritenuti sufficienti.

12. Divieto di plagio:

- Ogni progetto, elaborato autonomamente e personalmente da ciascuno studente o coppia di studenti, deve essere originale. Codice, o porzione di codice che verrà identificato come "copiato" da altre fonti comporterà l'automatica valutazione insufficiente del progetto. Fanno doverosamente eccezione a questa norma i riferimenti a package e librerie reperite nei repository pubblici per i quali sarà comunque necessario citarne la fonte d'origine in un'apposita sezione bibliografica della relazione.