TECNICHE DI PROGRAMMAZIONE

27/05/19

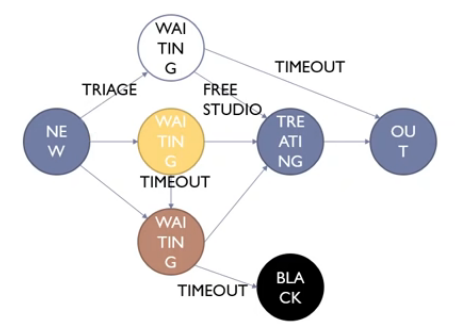
Simulazione ad eventi: continuazione esercizio “Emergency”.

Le ultime due esercitazioni in laboratorio saranno molto importanti in quanto verranno svolte come se fossimo all’esame e ci verranno dati dei problemi in cui lavoreremo su dei Dataset nuovi. Questi dataset saranno gli stessi che saranno presenti all’esame.

Ritorniamo all’esercizio “Emergency”. La difficoltà di questo esercizio è che noi non basta tener traccia di quante persone ci sono nel nostro sistema, cioè non mi basta avere solo dei conteggi. Dobbiamo seguire ogni singola persona. Bisogna modellare lo stato del sistema in modo puntuale e non solo più in modo statistico come visto in precedenza.

Questo ci obbligherà a gestire una doppia struttura basi. Da un lato ho la mia coda degli eventi che va per conto suo e che mi gestisce ciò che avviene nel sistema, ma dall’altro lato ho questi eventi che sono associati a dei pazienti dei quali devo rappresentare delle informazioni. Ho quindi 2 oggetti diversi: l’Evento e il Paziente. Volendo essere pignoli ci potrebbe servire anche un terzo oggetto che è il Medico in quanto potrebbe interessarci l’informazione sui diversi studi medici.

Andiamo quindi ad importare il progetto da GitHub e eseguiamo il file .sql su Heidi come sempre. Abbiamo 2 classi nel Model, Evento e Paziente, in cui la prima è già in è parte compilata sulla base del grafico in figura (vedi sotto), mentre la seconda è ancora da gestire.



Nella classe evento abbiamo una marcatura temporale (LocalTime), il tipo di evento e il riferimento al paziente. È presente il costruttore e i getter per ogni variabile. Ricordarsi sempre il comparatore sulla data che ci servirà per la coda prioritaria.

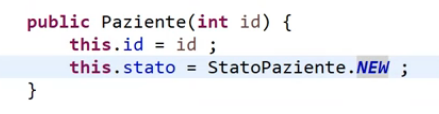
Per quanto riguarda il paziente, quali sono le informazioni che ci servono?

Ci servir à un modo per distinguerli, lo stato in cui si trova (il codice del colore). Questi ultimi possiamo copiarli dal diagramma sopra (new, waiting\_white, waiting\_yellow ecc…).

Immagine che contiene screenshot

Descrizione generata automaticamente

Quando creo un nuovo paziente, dovrà essere creato in uno stato “new”, quindi il costruttore prenderà dall’esterno l’identificatore del paziente e metterà lo stato uguale a “new”.



Dobbiamo inoltre gestire la priorità dei pazienti in quanto il testo ci dice che a parità di codice, passa prima chi è arrivato prima. Quindi nel caso in cui avessimo 2 codici rossi, ma uno dei due era un codice verde che poi è diventato giallo e poi rosso, passerà prima dell’altro che è arrivato e ha subito ricevuto il codice rosso. Il primo infatti è nella sala da aspetto da più tempo.

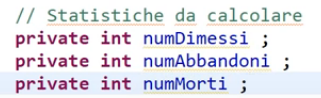
Bisogna quindi memorizzare anche il tempo in cui è arrivato ogni paziente. Nella classe “Paziente”, aggiungiamo una variabile LocalTime denominata “oraArrivo”. Questa ora di arrivo non è l’ora vera (non devo scrivere “now”) ma è l’ora simulata. Questa ora la sa solo chi crea l’evento, quindi dovrà essere un parametro del costruttore.

Immagine che contiene screenshot

Descrizione generata automaticamente

Ci spostiamo ora nella classe “Simulazione” che è già inizializzata con alcuni parametri. Per quanto riguarda il Modello del Mondo, esso dovrà contenere (come già anticipato) le informazioni sui pazienti e sul numero di studi liberi.

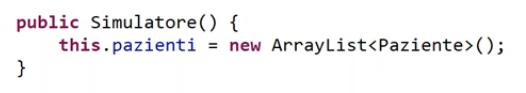
Come output devo avere il numero di pazienti dimessi, il numero di pazienti che hanno abbandonato e il numero di pazienti morti.



Ricordiamoci di aggiungere i Getter e Setter della classe Paziente.

Andiamo poi a predisporre i 3 metodi minimi che dobbiamo avere:

1. Il costruttore del Simulatore: deve creare la lista di pazienti.



1. Il metodo “init”: deve fare 3 cose:
2. Creare i pazienti. Cancelliamo eventualmente quelli precedenti e, mediante un ciclo “for”, creo NP nuovi oggetti “Paziente” (dove NP è il numero inserito dall’utente) e li aggiungo alla lista.
3. Creare gli eventi iniziali. Puliamo la coda degli eventi e genero un evento di arrivo per ogni paziente in cui “p” arriva all’ora “p.oraArrivo” (vedi immagine sotto per capire). Questo evento dobbiamo aggiungerlo alla lista “pazienti”. Dobbiamo anche inizializzare il numero di studi liberi.
4. Resettare le statistiche (o contatori).

Immagine che contiene screenshot

Descrizione generata automaticamente

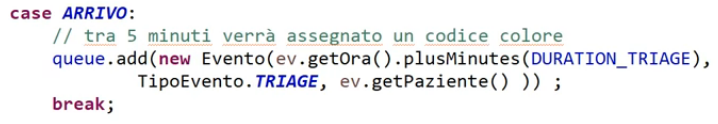
1. Il metodo “run”: avrà un “while” che andrà ad estrarre gli eventi. Dobbiamo però andare ad inserire anche il vincolo sul tempo. Nel caso un paziente si presenti dopo l’ora, morirà direttamente senza ricevere aiuto.

Immagine che contiene screenshot

Descrizione generata automaticamente

Successivamente, mediante uno switch, andiamo a decidere cosa fare in base al tipo di evento:

* ARRIVO: creo un nuovo evento che abbia come ora l’ora in cui è stato creato + 5 minuti.



* TRIAGE: in questo caso dobbiamo andare a gestire il colore dei codici. Per far questo, creiamo tra le variabili interne una nuova variabile denominata “nuovoStatoPaziente” che ci andrà ad indicare il colore. La inizializziamo a “bianco” (il testo vuole così). Fatto questo, vado ad impostare il Timeout per il paziente con il codice bianco. Creo un nuovo evento il cui all’ora aggiungo il tempo di attesa del codice bianco. Lo stesso viene fatto per gli altri colori. Per decidere il colore degli altri pazienti, il testo ci dice che dobbiamo fare a rotazione (bianco, giallo,rosso,bianco…). Creiamo un nuovo metodo “ruotaNuovoStatoPaziente” che ci gestisce questa rotazione dei codici.

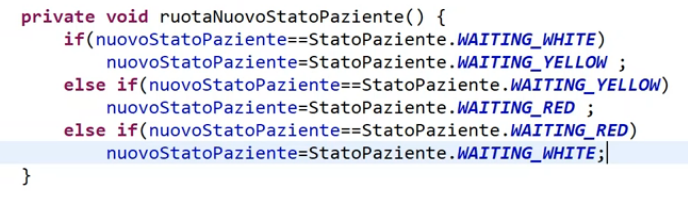


Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente



* TIMEOUT: in questo case decidiamo cosa succede nel caso in cui scada il Timeout di un colore. Nel caso del bianco, il paziente abbandona l’ospedale e quindi andiamo ad aggiornare il numero di abbandoni. Nel caso di giallo e rosso, i pazienti passano al colore successivo, con il nuovo tempo di timeout relativo. Bisogna ricordarsi che se il paziente diventa “black” va aggiornato il numero di morti.

Immagine che contiene screenshot, persona

Descrizione generata automaticamente

Proviamo un attimo a far partire il programma per vedere se funziona. Andiamo ad aggiungere il metodo “toString” alla classe Evento e alla classe Paziente. Creo successivamente la classe di Test (con un main) che conterrà la creazione del simulatore, il metodo init e il metodo run.

Immagine che contiene screenshot

Descrizione generata automaticamente

Il programma funziona ma notiamo che i diversi pazienti o abbandonano l’ospedale nel caso di codice bianco, o muoiono nel caso in cui abbiano il codice giallo (passa a codice rosso e poi a codice nero) o rosso. Nessun paziente viene curato. Noi abbiamo 3 studi medici liberi e dobbiamo chiederci qual è l’evento che fa entrare un paziente in un o studio medico. Quand’è che dobbiamo aggiungere un evento di tipo “Visita”. All’inizio della simulazione, i primi 3 pazienti occuperanno di sicuro i 3 studi medici. Queste persone verranno curate (andremo a gestire questo nel case CURATO) e libereranno uno studio medico. Se si libera uno studio medico, posso immediatamente schedulare un evento VISITA. Nella prossima lezione dovremo quindi gestire le seguenti dinamiche:

Immagine che contiene screenshot

Descrizione generata automaticamente

Sicuramente dovremo andare a modificare il case TRIAGE perché in precedenza noi abbiamo gestito solo l’assegnazione di un colore e il TIMEOUT. Ora dobbiamo considerare anche il caso in cui uno studio medico sia libero e quindi il paziente possa entrare per farsi curare.

FINE