TECNICHE DI PROGRAMMAZIONE

20/05/19

Simulazione ad eventi “Noleggio Auto”.

La lezione scorsa, abbiamo definito una classe Evento. Essa incorpora un metodo “enum” il quale ha al suo interno delle costanti che rappresentano i diversi tipi di eventi che possono verificarsi. Sulla base di questa classe Evento, costruiamo nella classe Simulatore una coda prioritaria che è quella su cui si reggerà l’intera simulazione.

L’oggetto simulatore cosa farà?

1. Mantenere lo “stato del mondo”.
2. Tenere conto di alcuni parametri di simulazione.
3. Conterrà delle statistiche derivanti dalle diverse simulazioni.

Per descrivere lo stato del mondo inseriamo 2 variabili: il numero di auto totali che possediamo e il numero di auto disponibili per il noleggio.

I parametri di simulazione sono dei dati sulla base dei quali il simulatore decide che cosa fare quando avviene un evento. Avremo quindi 3 variabili: orarioInizio, orarioFine e intervalloArrivoCliente. Possiamo già inizializzare le variabili in quanto sappiamo che l’orario in cui si può noleggiare è 8-20 e che l’intervallo tra un cliente ed un altro è di 10 minuti. Come ultima cosa, aggiungiamo una Lista si “durate” che contiene le diverse durate per cui posso noleggiare un auto.

Per quanto riguarda le statistiche derivanti dalla simulazione, inseriamo che cosa deve calcolare il nostro simulatore. Nel nostro caso, ci dovrà dire il numero di clienti totali e il numero di clienti insoddisfatti.

Sulla base di queste variabili, il nostro simulatore avrà 2 metodi, uno per inizializzare la simulazione (preparare le variabili) e uno per iniziare la simulazione vera e propria. Decidiamo di chiamare questi due metodi “init” e “run”.

Immagine che contiene screenshot

Descrizione generata automaticamente

Il metodo “init” dovrà impostare i valori fondamentali affinché la simulazione possa avvenire correttamente quindi la coda degli eventi, lo stato del mondo e le statistiche andranno impostate ad un valore iniziale. Il numero di auto totali deve essere fornito ogni volta e possiamo passarlo come parametro al metodo “init” così ogni volta che inizializziamo la simulazione possiamo passargli un numero di auto diverse. Inizialmente, il numero di auto disponibili sarà uguale al numero di auto totali. Successivamente, resetto lo stato del mondo, le statistiche raccolte e la coda degli eventi. Non tocco i parametri.

Carichiamo poi gli eventi iniziali. Per far questo mi basta un “for” in cui itero dall’ora di inizio all’ora di fine e incremento quest’ora sommandogli ogni volta l’intervalloArrivoCliente.

La scorsa lezione ci siamo dimenticati di aggiungere il costruttore alla classe Evento quindi lo facciamo ora.

Nel “for”, aggiungiamo alla coda un nuovo oggetto Evento in cui il tempo è l’ora e il tipo di Evento è la costante “CLIENTE\_ARRIVA”.

Bisogna anche inizializzare la Lista delle durate del Noleggio. Non possiamo inizializzarla quando la creiamo, quindi dobbiamo creare un costruttore che ci permetta di farlo.

Immagine che contiene interni

Descrizione generata automaticamente.

Abbiamo aggiunto la durata di 1 ora, 2 ore e 3 ore perché ci veniva dato dal testo. Il metodo “init” terminato è il seguente:

Immagine che contiene screenshot

Descrizione generata automaticamente

Passiamo ora al metodo “run” che ci permette . Esso non sarà altro che un grosso loop che andrà avanti fino a quando la coda degli eventi non è vuota. Se ci sono ancora eventi da simulare nella coda, ne estraggo uno (metodi “pick” e “poll”, il primo guarda il primo evento della coda ma lo lascia li, mentre il secondo lo estrae anche). Dopo aver estratto il primo evento, attraverso uno “switch” (bisogna ricordarsi di aggiungere i “getter” nella classe Evento) decido cosa fare in base ai 2 casi possibili:

1. Cliente che arriva per prendere una nuova auto: in questo caso dovrò incrementare il numero di clienti totali di 1. Successivamente mi interrogo se ci sono auto disponibili da dare a questo nuovo cliente:
2. Non ci sono auto disponibili e quindi incremento di 1 il numero di clienti insoddisfatti.
3. C’è un auto disponibile e quindi noleggio l’auto al cliente. Decremento le auto disponibili e creo l’evento di restituzione. Questo evento non potevo crearlo nel metodo “init” in quanto non potevo sapere se c’erano auto disponibili. Devo creare una durata casuale tra le 3 disponibili attraverso una classe “Random”.



Questa classe ha un metodo chiamato “nextInt” che mi restituisce un numero casuale tra 0 e un numero specificato. Mediante il metodo “getTempo” otteniamo l’ora in cui ci troviamo. A questa ci aggiungiamo la durata del noleggio.

Immagine che contiene screenshot

Descrizione generata automaticamente

Ora abbiamo tutto quello che ci serve per creare un nuovo evento.

Immagine che contiene screenshot

Descrizione generata automaticamente

1. Cliente che ritorna per restituire l’auto: in questo caso dovrò solamente incrementare la variabile delle auto disponibili di 1.

Immagine che contiene screenshot

Descrizione generata automaticamente

Ricordiamoci di aggiungere i “getter” e i “setter” della classe “Simulatore”.

Creiamo sempre nel package Model una classe “TestSimulatore” che conterrà il main e che useramo per provare il nostro simulatore. In questa classe creiamo un nuovo simulatore, lo inizializziamo (per esempio con 10 auto) e lo facciamo partire. Ricordiamoci di stampare a video il risultato che vogliamo ottenere.

È molto elevato e quindi dobbiamoImmagine che contiene screenshot

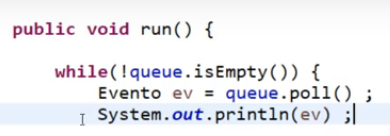
Descrizione generata automaticamente

Notiamo che runnando il programma, il numero di persone insoddisfatte è molto elevato e quindi dobbiamo cambiare il numero di auto disponibili. Se al posto di 10 mattiamo 15, notiamo che il numero di persone insoddisfatte è 0 (al massimo 1).

Se cambiamo l’intervallo di arrivo dei clienti, cambiano sia il numero di clienti totali che il numero di clienti insoddisfatti. La modifica possiamo eseguirla direttamente dal TestSimulatore.



Nel Simulatore, dopo aver creato l’evento “ev” posso anche stamparlo per vedere cosa succede. Questo ci obbliga ad inserire un metodo toString nella classe “Evento”.





Esercizio Noleggio Auto terminato!

Nel prossimo esercizio dobbiamo gestire il comportamento di un pronto soccorso. Ad ogni persona che arriva, il base alla pericolosità del suo incidente, viene assegnato un colore (Bianco, giallo, rosso, nero). Nel caso in cui un paziente di tipo BIANCO non viene servito entro un tot di tempo, torna a casa. Nel caso di paziente di tipo GIALLO o ROSSO, se non viene servito in tempo, passa al colore successivo. Codice NERO vuol dire che il paziente è morto.

Le variabili temporali da gestire sono le seguenti:

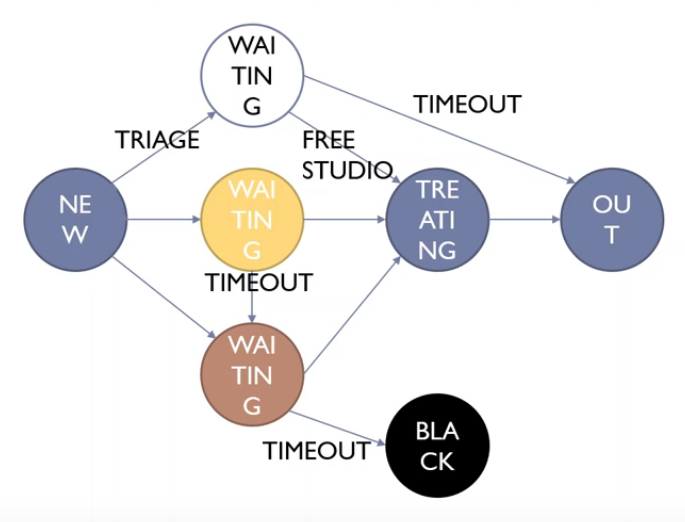
Immagine che contiene screenshot

Descrizione generata automaticamente

Duration\_Triage è il tempo necessario per dare un codice ad un paziente. Gli altri 3 tempi sono il tempo necessario per gestire un paziente bianco, giallo o rosso.

Il modello del Mondo sarà una “collection” di pazienti in cui la cosa importante da sapere su ogni paziente è il suo stato.

Noi dobbiamo analizzare l’evoluzione dello stato di un paziente. La prossima figura ci rappresenta bene tutte le situazioni che possono capitare. I cerchi corrispondono allo Stato del Paziente mentre le frecce mi rappresentano gli Eventi.



Il nostro Simulatore, partendo da dei parametri (numero di studi medici, numero di pazienti, frequenza di arrivo), ipotizzando che le severità dei pazienti vengano assegnate in modo circolare (non casuale) e simulando dalle 8:00 alle 20:00, deve restituirci il numero di pazienti che sono stati curati, il numero di pazienti che hanno abbandonato e il numero di pazienti morti.

FINE