Centri Vaccinali

MANUALE TECNICO

CANALI LUCA - 744802;
IVAN TONOLLI - 744364

UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA

Sommario

INTRODUZIONE

Centri vaccinali è un progetto sviluppato per Laboratorio A inerente al corso di laurea di Informatica dell'Università degli studi dell'Insubria. Il progetto è sviluppato in Java 16, l'esecuzione avviene tramite terminale ed è stato sviluppato e testato su Windows 10.

• CLASSI ESTERNE

Per lo sviluppo del progetto sono state utilizzate diverse classi esterne, in particolar modo classi per la gestione, la creazione, la scrittura, la lettura dei file di testo e quindi anche per gestire la relazione tra i dati scritti nei file e gli oggetti da creare. Tali classi non verranno approfondite in questo manuale ma solo menzionate:

- File
- FileWriter
- FileReader
- BufferedWriter
- BufferedReader
- Scanner
- ArrayList (utilizzata per code di oggetti utili al corretto funzionamento del programma)

• CLASSI PRINCIPALI

Ecco un elenco delle principali classi implementate. Verranno approfondite in seguito:

- GestoreFile
- CentroVaccinale
- CittadinoVaccinato
- CittadinoRegistrato
- Cittadino
- EventoAvverso

Di seguito verranno approfondite le classi implementate. Non è stata implementata un'interfaccia grafica in quanto non espressamente richiesto dalle specifiche del progetto.

GestoreFile

Questa classe è forse la principale di tutto il progetto. Grazie agli oggetti istanziati da questa classe è possibile eseguire tutte le operazioni di lettura, scrittura e gestione dei file txt. Oltre a ciò, la classe GestoreFile si occupa di gestire le corrispondenze tra file di testo e oggetti grazie all'uso di diversi metodi che saranno approfonditi di seguito.

• STRUTTURA DI UN OGGETTO <*GestoreFile*>

Un oggetto di questa classe ha come campi due informazioni molto importanti:

- *Il campo **path**:* Il nome della directory nella quale verranno salvati i file di testo
- *Il campo **file***: Il nome del file di testo da creare e su cui lavorare

• I METODI <*VerificaPath()*> E <*VerificaTxt()*>

Prima di effettuare qualsiasi operazione sui file di testo presenti nella cartella "data" bisogna verificare prima di tutto se la directory esiste all'interno della cartella del progetto, e se non esiste bisogna crearla. Questo è esattamente il compito del metodo *VerificaPath()* che, semplicemente, prende il primo campo dell'oggetto che esegue il metodo e verifica l'esistenza della directory

Il metodo *VerificaTxt()* funziona in modo analogo: esso sfrutta il secondo parametro dell'oggetto che esegue il metodo (che è semplicemente una Stringa di caratteri) e verifica se esiste, all'interno della directory specificata nel primo parametro, un file .txt dal nome uguale.

NOTA: il secondo parametro rappresenta SOLO il nome del file senza estensione .txt, in quanto vengono aggiunte in automatico dai vari metodi.

• IL METODO <*ScriviTxt()*>

Metodo che riceve come parametro un oggetto *String*. In particolar modo questo metodo si occupa della scrittura della Stringa sul file di testo specificato come secondo parametro dell'oggetto che chiama il metodo. Questo metodo viene impiegato soprattutto per la registrazione e l'inserimento di dati all'interno del nostro sistema.

Questo metodo sfrutta le classi *File*, *FileWriter e BufferedWriter*.

• I METODI <*SetCentri()*> E <*SetVaccinati*>

Per facilitare alcune operazioni è necessario creare oggetti dai dati presenti nei file. Per fare ciò i metodi *SetCentri()* e *SetVaccinati()* leggono riga per riga, tramite uno stream, i relativi file estraendo i dati necessari alla creazione di oggetti di tipo ***CentroVaccinale/ Cittadino***

vaccinato/CittadinoRegistrato già registrati al sistema, in modo tale da importare le nostre istanze all'interno delle relative liste. Difatti, possiamo immaginare una singola riga di un file come un vero e proprio oggetto che viene istanziato nel momento in cui l'esecuzione del programma inizia. Il metodo si interrompe appena incontra una riga vuota.

Un'operazione rilevante di questi metodi è ***<String.split()>*** con una complessità stimata di $O(n)$, difatti n rappresenta la lunghezza della stringa che può variare di poche unità, mentre la complessità generale stimata del metodo è sempre $O(n)$ dove *n* rappresenta il numero di righe non vuote del file.

CentroVaccinale

Un oggetto centro vaccinale contiene i seguenti campi:

- $nome \rightarrow nome\ univoco\ del\ centro\ vaccinale$
- $qualificatore \rightarrow indica\ il\ tipo\ di\ indirizzo\ (via, piazza, viale)$
- $via \rightarrow indica\ il\ nome\ della\ via\ del\ centro$
- $numero\ civico$
- $comune \rightarrow comune\ in\ cui\ si\ registra\ il\ centro$
- $provincia \rightarrow indica\ la\ sigla\ della\ provincia$
- $cap \rightarrow codice\ postale\ del\ centro$
- $tipo \rightarrow tipologia\ del\ centro\ (hub, ospedaliero\ \dots)$
- $listaVaccinati \rightarrow la\ lista\ dei\ vaccinati\ di\ quel\ centro$
- $listaCentri \rightarrow Array\ \boldsymbol{static}\ che\ serve\ per\ memorizzare\ i\ centri\ che\ si\ registrano$

Questa classe ci permette di rappresentare i centri vaccinali tramite oggetti.

Metodi:

- $registraCentroVaccinale()$

Questo metodo richiama la classe $GestoreFile$ ed in particolare i metodi:

- $verificaPath()$
- $verificaTxt()$
- $scriviTxt()$

Vengono svolte tutte le operazioni da effettuare quando si tenta di registrare un Centro Vaccinale al sistema, in particolare:

- Si verifica che la directory “data” esista all’interno della cartella del progetto, se non esiste significa che stiamo andando ad effettuare la prima operazione in assoluto all’interno del sistema o più semplicemente che la cartella è stata cancellata
- Si verifica che il file *Centri_Vaccinali.txt* esista. Se non esiste significa che stiamo andando a registrare per la prima volta un Centro Vaccinale a sistema quindi si crea la cartella
- Una volta effettuata la verifica dei file e la loro creazione si può procedere alla scrittura effettiva del File specificato dall’oggetto *GestoreFile.* All’interno del File vengono scritti tutti i dati del centro che esegue il metodo.
- Infine, si aggiunge il Centro appena registrato all’interno della coda dei Centri in modo tale da semplificare funzioni future di ricerca. Si è evitato di implementare i metodi di ricerca direttamente sui File di testo.

- $registraVaccinato()$

Questo metodo è analogo al precedente, il funzionamento è identico; cambiano i file su cui si lavora. Con questo metodo possiamo registrare a sistema un Cittadino Vaccinato verificando che il

file $< nome.centro > \_Vaccinati.txt$ esista. Si procederà ancora una volta alla scrittura dei dati dell'oggetto $CittadinoVaccinato$ all'interno del txt corrispondente

- $centroEsiste()$

Questo metodo è tanto importante quanto semplice: viene verificato, dato come parametro una stringa, che esista un centro vaccinale con nome corrispondente a quella stringa. Viene restituito un *boolean true* appena l'elemento viene trovato. Questo metodo viene impiegato tutte le volte che si sta tentando un'operazione di gestione o modifica di un oggetto CentroVaccinale da parte di un'altra entità (che può essere per esempio un oggetto *Cittadino)*. Grazie a questo metodo e al suo uso si evitano errori o interruzioni improvvise del software.

La complessità di questo metodo varia in base alla posizione dell'oggetto da verificare all'interno del vettore.

- CASO PEGGIORE: bisogna scorrere tutta la lista, l'elemento è in ultima posizione $O(n)$
- CASO MEDIO: l'elemento da ricercare si trova a metà vettore $O(\frac{n}{2})$
- CASO MIGLIORE: l'elemento è in prima posizione $O(1)$

Dove *n* è la lunghezza del vettore.

CittadinoVaccinato

Passiamo ora alla classe che ci permette di rappresentare i cittadini vaccinati. Essa è composta da:

- $nome$
- $codicefiscale$
- $data \rightarrow data\ di\ vaccinazione$
- $vaccino \rightarrow tipo\ di\ vaccino$
- $idvaccinazione \rightarrow id\ univoco\ di\ vaccinazione$

Metodi:

  - $idOccupato()$

Questo metodo è l'unico metodo significativo di questa classe. Esso è un metodo istanziato direttamente dalla classe, riceve come parametro uno short e un oggetto di tipo *CentroVaccinale* e permette di verificare se l'*Id* fornito come parametro (short) è già registrato presso il Centro Vaccinale ricercato.

    - $return\ true \rightarrow$ *se l'id è occupato*
    - $return\ false \rightarrow$ *se l'Id è libero*

Questo metodo viene sfruttato durante l'esecuzione del programma nel momento in cui si senta di registrare un vaccinato presso un determinato centro; cioè: quando tentiamo di aggiungere un oggetto $CittadinoVaccinato$ ad un array specifico di un oggetto $CentroVaccinale$, tramite il metodo $< add() >$, si verifica prima che l'id assegnato al Cittadino non sia già "posseduto" da un altro oggetto $CittadinoVaccinato$.

Così facendo si evitano conflitti tra dati ed è possibile identificare ciascun oggetto tramite l'univocità dell'Id.

Complessità:

- CASO PEGGIORE: $O(n)$
- CASO MEDIO: $O(\frac{n}{2})$
- CASO MIGLIORE: $O(1)$

CittadinoRegistrato

È necessario tenere traccia di ogni cittadino che si registra all'interno dei File txt, per fare ciò ci avvaliamo della classe $CittadinoRegistrato$ che ha come campo un $ArrayList$ appartenente all'intera classe (e non al singolo oggetto) che appunto contiene tutti gli oggetti di questo tipo che registriamo nel nostro sistema.

Metodi:

- $login()$

Questo metodo permette l'identificazione di un cittadino che tenta di accedere al sistema. Per accedere al sistema è necessario verificare che lo user-id e la password inseriti esistano all'interno della lista dei Cittadini registrati; o meglio: il metodo verifica che il cittadino che sta tentando l'accesso sia registrato nel sistema andando a scansionare l'ArrayList dei registrati.

Distinguiamo le tre complessità:

- CASO PEGGIORE: $O(n)$
- CASSO MEDIO: $O(\frac{n}{2})$
- CASO MIGLIORE: $O(1)$

Questa classe contiene anche il metodo static $menuOperatore()$ che rappresenta il main della sotto-applicazione $CittadinoRegistrato$.

Cittadino

La classe cittadino serve per creare e manipolare gli oggetti che rappresentano un Cittadino (non vaccinato e non registrato) che accede al sistema. Un cittadino senza registrazione e senza vaccino può accedere al sistema solo per ricercare un centro vaccinale e visualizzare le informazioni di esso come: lista degli eventi avversi aggiunti e informazioni del centro (nome, via, tipo…)

È stato necessario creare questa classe in quanto c'era il bisogno di fare una distinzione tra classi apparentemente molto simili ma che differiscono per poche caratteristiche molto importanti come l'id vaccinazione, ovviamente. In questo modo un oggetto di questa classe non può in alcun modo sfruttare i metodi che le classi $CittadinoVaccinato$ e $CittadinoRegistrato$ usano.

Questa classe contiene anche il main della sotto-applicazione $Cittadino$ che viene richiamata nella $home$

EventoAvverso

Questa classe rappresenta gli oggetti EventoAvverso. Ogni volta che un Cittadino vuole inserire un evento avverso, l'applicazione crea un oggetto che rappresenta l'evento con i seguenti campi:

- $nome$
- $severità$

Ogni evento avverso viene salvato in un file *(nome.centro)_Eventi_Avversi.txt.*

La soluzione più efficiente trovata è stata quella di gestire gli eventi direttamente su File txt senza avvalersi di strutture dati specifiche.

È possibile consultare tutti gli eventi avversi di un determinato centro mediante una ricerca che va a prendere il file degli eventi specifici di quel centro, lo legge tutto e stampa i valori su console.

La complessità di questa operazione è notevole in quanto essa deve per forza scorrere tutte le righe dell'intero File per riportare tutti gli eventi che si sono registrati. La complessità è $O(n)$ dove *n* sono le righe del file di testo che possono essere più o meno elevate.