Università degli studi – Insubria
Corso di informatica

MANUALE TECNICO

Prepared for

Laboratorio interdisciplinare A

Prepared by

Andrea Pigliafreddo
Matricola 746578
31 gennaio 2022

SOMMARIO

INTRODUZIONE

L'applicazione "Laboratorio interdisciplinare A" è un progetto sviluppato per il corso di laurea in Informatica dell'Università degli Studi dell'Insubria.

Il progetto è sviluppato in Java 15 ed è stato sviluppato e testato sul sistema operativo Windows 10 e MacOS Monterey 12.2 .

LIBRERIE ESTERNE UTILIZZATE

Per lo sviluppo di questo progetto sono state utilizzate alcune librerie esterne:

- **Java.util**

  Questa libreria contiene il framework delle raccolte, le classi di raccolta legacy, il modello di eventi, le strutture di data e ora, l'internazionalizzazione e le classi di utilità varie (un tokenizzatore di stringhe, un generatore di numeri casuali e un array di bit). In particolare abbiamo utilizzato le librerie java.util.ArrayList e java.util.List per leggere il file che abbiamo interpretato come liste di array. Inoltre abbiamo utilizzato la lista java.util.Scanner per andare a leggere ciò che l'operatore digita

- **Java.io**

  La libreria java.io fornisce un insieme di flussi di input e un insieme di flussi di output utilizzati per leggere e scrivere dati su file o altre sorgenti di input e output.

- **Java.text**

  Il pacchetto java.text è costituito da classi e interfacce utili per scrivere programmi internazionalizzati che gestiscono le usanze locali, come la formattazione di data e ora e l'alfabetizzazione delle stringhe. In particolare abbiamo utilizzato la libreria java.text.ParseException per costruire un'eccezione con il messaggio di dettaglio, cioè una stringa che descrive questa particolare eccezione, e l'offset specificati.

- **Java.lang.RuntimeException**

  RuntimeException è la superclasse di quelle eccezioni che possono essere generate durante il normale funzionamento della Java Virtual Machine. Un metodo non è tenuto a dichiarare nella sua clausola "throws" eventuali sottoclassi di RuntimeException che potrebbero essere generate durante l'esecuzione del metodo ma non rilevate.

STRUTTURA GENERALE DEL SISTEMA DI CLASSI

Presentiamo ora le classi principali che sono state utilizzate per realizzare questo progetto.

CLASSI PRINCIPALI

- CENTRI VACCINALI
- CENTRO
- CITTADINO
- CONTROLLI
- EndofFile
- EVENTI

Andiamo ora ad analizzare nello specifico queste classi principali.

CLASSI

- *CENTRI VACCINALI*

La classe "**centri vaccinali**" è la prima parte del programma con cui l'utente si relaziona ed è formata da un menù principale con il quale l'utente inizia ad utilizzare l'applicazione e seleziona le prime operazioni che vuole svolgere.
In questa classe è presente il metodo "main", che è il primo metodo utilizzato nel programma ed è la base per l'esecuzione del programma stesso.
Abbiamo utilizzato un ciclo "do while" per creare il menù nel quale l'operatore sceglie di accedere al programma come un operatore oppure come un cittadino, oppure se vuole chiudere il programma; abbiamo utilizzato il metodo "toLowerCase" per memorizzare la scelta dell'utente in formato minuscolo così da evitare problemi quando andremo a svolgere i vari "case" all'interno dell'istruzione "switch".
Quindi abbiamo utilizzato l'istruzione "switch" per permetterci di svolgere operazioni differenti in base alla scelta dell'utente.
Nel caso in cui l'utente acceda come operatore, si accede al primo "case" e tramite un ciclo "do while" l'utente può svolgere diverse funzioni: effettuare la registrazione di un nuovo centro vaccinale, effettuare il login, uscire dal programma. Abbiamo utilizzato un'altra istruzione "switch" per permettere all'utente di scegliere l'operazione desiderata ed una volta scelta tramite l'istruzione "break" il programma esce dallo "switch" e prosegue. Se l'istruzione scelta non corrisponde ad una delle tre opzioni proposte si genera un messaggio di errore.
Nel caso in cui l'utente acceda come cittadino, tramite un ciclo "do while" può scegliere tra diverse operazioni: cerca e visualizza le informazioni di un centro vaccinale, registrazione presso un centro vaccinale, effettua il login, esci. Come nel blocco precedente, abbiamo utilizzato un'istruzione "switch" per permettere all'utente di scegliere l'operazione desiderata ed una volta scelta tramite l'istruzione "break" il programma esce dallo "switch" e prosegue. Se l'istruzione scelta non corrisponde ad una delle tre opzioni proposte si genera un messaggio di errore.
Nel caso in cui l'utente seleziona l'opzione "fine" nel terzo "case" dello "switch" il programma genera in output "Fine" e il programma finisce.

- *CENTRO*

La classe "**Centro**" è la classe utilizzata per gestire i centri vaccinali ed implementa le interfacce "Controlli" e "Serializable".

- Utilizziamo il metodo boolean "**RegistraCentroVaccinale**" ci permette di registrare un nuov centro vaccinale, una volta inserito il nome avviene un controllo che verifica che non vi sia un centro già registrato con quel nome, in

caso esista rifiuta la registrazione, altrimenti permette di inserire il resto dei dati, all'interno del metodo vengono richiamati altri metodi interni alla classe, tra cui il metodo "SetPsw", il metodo "InserimentoIndirizzo" e il metodo "salvaCentro".

- Il metodo "**SetPsw**" richiede come parametro una stringa che sarà la prima password inserita, la quale verrà confermata all'interno del metodo. Esso è composto internamente da un Try_Catch all'interno del quale le due password vengono scoposte in due array di caratteri e confrontate lettera per lettera, in caso le password non coincidano, il codice richiede la ripetizione del processo.

- Il metodo successivo "**login**" permette all'utente registrato in precedenza di accedere, inserendo il nome del centro e la password, all'inizio del metodo viene richiamata una funzione chiamata "leggifile" la quale prende in ingresso come parametro il nome del file da leggere, in questo caso "CentriVaccinali.dati.txt" e lo converte in un ArrayList, che utilizzerà per controllare che i dati inseriti dall'utente corrispondano. Utilizziamo la variabile booleana "**ceckNome**" per controllare che il nome inserito corrisponda a quello di un centro registrato e la variabile booleana "**ceckpsw**" usata per controllare che la password inserita corrispondi alla password associata al centro, queste verifiche avvengono tramite un ciclo "for" e alcune istruzioni "if". Se nome e password sono corretti si esegue l'accesso, altrimenti si ripete l'operazione. Dopo aver effettuato l'accesso ci si ritrova in uno switch case che compone un menù. nel caso in cui l'operazione vada a buon fine l'utente viene registrato e tramite un "break" si esce dallo "switch" e si continua il programma, in caso contrario l'inserimento non è valido e si passa all'istruzione successiva.

- Il metodo "leggifile" prende in ingresso come parametro il nome del file da leggere, e lo converte in un ArrayList.

- Il metodo "**InserimentoIndirizzo**" permette di inserire l'indirizzo di ubicazione del centro vaccinale. Il metodo inizia inserendo i vari dati dell'indirizzo, via/viale/piazza, nome dell'indirizzo, numero civico, comune, e tramite un ciclo while si controlla se la sigla della provincia venga inserita rispettando il pattern, in caso contrario l'inserimento non è valido. Infine si inserisce il cap e con l'istruzione "return" viene fornito il risultato finale dell'indirizzo.

- Il metodo "**ShowDati**" ha la funzione di mostrare a terminale i dati di un centro vaccinale. Esso utilizza inizialmente il metodo "Nel caso in cui i dati inseriti non esistono, viene mandato in output il messaggio "Centro vaccinale non registrato".

- Il metodo "InfoCentro" permette di cercare i dati di un centro inserendo il nome del centro di interesse e richiamando la funzione "ShowDati"

- il metodo “**registraVaccinato**” permette di registrare nel sistema un vaccinato inserendo il codice fiscale e applicandovi i dovuti controlli sul pattern e sull’esistenza o meno di una vaccinazione a suo nome, in caso non rispetti il pattern il codice richiede la ripetizione dell’inserimento, mentre se vi è già una vaccinazione intestata, la aggiorna. Inoltre permette di specificare il tipo di vaccino somministrato tramite l’istruzione “switch case”, scegliendo tra le diverse opzioni proposte. Permette anche di inserire la data della vaccinazione ed esegue un controllo tramite un ciclo “while”; nel caso di inserimento errato si ripete l’operazione. Inseriamo anche il numero della dose utilizzando nuovamente un ciclo “while” e l’istruzione “switch”. In fine possiamo inserire il nome del centro vaccinale e tramite alcune istruzioni “if” si controlla se il centro di vaccinazione esiste, in tal caso richiama la funzione “salvaVaccinato”, in caso contrario invece rifiuta l’operazione.

- Il metodo “**salvaVaccinato**” esegue una funzione di salvataggio dati all’interno del file "Cittadini_Vaccinati.txt" utilizzando un ciclo “while” e una istruzione “if else”.

- Il metodo “controlloQualificatore” permette di controllare se il qualificatore inserito è corretto , tramite un ciclo “while” e una istruzione “if else”.

- Il metodo “**controlloTipologia**” ha la funzione di controllare che la tipologia di centro vaccinale corrisponda con una di quelle proposte, utilizzando un ciclo “while” e un ciclo “if else”.

- il metodo “salvaCentro” serve a salvare i dati del cittadino all’interno del file "CentriVaccinali.txt". Il metodo inizialmente controlla se il file è esistente, in tal caso, i dati da salvare vengono uniti e stampati sul file. Per realizzare questi passaggi utilizziamo rispettivamente un ciclo “while” e l’istruzione “if else”.

- Il metodo “**serializza**” ha la funzione di serializzare l’array passato come parametro.

- il metodo “**deserializza**” svolge la funzione di deserializzare un array contenente il nome dei centri vaccinali registrati

- *CITTADINI*

La classe “**Cittadini**” è la classe che gestisce tutti i metodi e le operazioni disponibili per l’utente di tipo cittadino e implementa l’interfaccia “Controlli”.

- Il metodo “**registrazione**” permette di registrare un cittadino precedentemente vaccinato. Permette di inserire il proprio nome e cognome e tramite un ciclo “while” controlla che l’operazione sia valida. Poi permette di inserire il codice fiscale e utilizzando istruzioni “if” e “while” controlla il corretto inserimento ; se l’utente è già registrato, con una istruzione “if” e l’istruzione “switch” permette di effettuare il login oppure di creare un nuovo account. Attraverso una ulteriore istruzione “if” si controlla l’esistenza di una vaccinazione intestata al codice fiscale inserito e nel caso in cui non venga trovata alcuna vaccinazione l’operazione viene rifiutata. In seguito l’utente deve inserire il proprio indirizzo, data di nascita e password; in queste fasi verranno richiamati i metodi “SetPsw” e i metodi della classe Controlli “controlloData” , “controllocf”, “controllocf2” e “controllocf2vac”. La gestione di questi dati è resa possibile utilizzando cicli “while” e istruzioni “if”. Se i dati sono stati registrati con successo i dati vengono passati al costruttore “datinuovo” che crea un’sitanza della classe Cittadino i quali dati vengono salvati nel file “Cittadini_Registrati.txt” tramite la funzione “salva”.

- Il metodo “**SetPsw**” richiede come parametro una stringa che sarà la prima password inserita, la quale verrà confermata all’interno del metodo. Esso è composto internamente da un Try Catch all’interno del quale le due password vengono scoposte in due array di caratteri e confrontate lettera per lettera, in caso le password non coincidano, il codice richiede la ripetizione del processo.

- Il metodo “**datinuovo**” è un metodo costruttore della classe, che raccoglie tutti i dati dell’utente all’interno dell’istanza “user” e in seguito utilizzando

- il metodo “salva” serve a salvare i dati del cittadino all’interno del file "Cittadini_Registrati.dati.txt". Il metodo inizialmente controlla se il file è esistente, in tal caso, i dati da salvare vengono uniti e stampati sul file. Per realizzare questi passaggi utilizziamo rispettivamente un ciclo “while” e l’istruzione “if else”.

- Il metodo “**leggifile**” permette di leggere il file passato come parametro e salvare i dati in una lista di array di stringhe, ognuno contenente i dati di un cittadino; utilizza una istruzione “if” e un ciclo “while”.

- Il metodo “**ShowDati**” è un metodo selettore che stampa a terminale i dati del cittadino passato come parametro, esso effettua una ricerca all’interno del file “Cittadini_Registrati.dati.txt” utilizzando il metodo “leggifile”

- Il metodo “**login**” permette l’accesso ad un utente registrato. All’inizio del metodo viene chiesto di inserire il codice fiscale, che viene controllato con un ciclo “while” e l’istruzione “if” e in caso di errore permette di inserire nuovamente il codice. Successivamente si chiede di inserire la password che viene analogamente controllata con un ciclo “while” e l’istruzione “if”. Poi la variabile booleana “ceckcf” controlla che venga trovato il codice fiscale all'interno della lista di stringhe ottenuta dalla conversione. Invece la variabile “ceckpsw” controlla che la password inserita corrispondi alla password associata al codice fiscale. Se il codice fiscale non viene trovato nel file significa che non è stata effettuata alcuna registrazione con questo codice fiscale. Se codice fiscale e password coincidono l’accesso è effettuato. Se il codice fiscale viene trovato ma la password non coincide a quella associata a questo codice fiscale si ha in output un messaggio di errore password. Tutte queste operazioni sono gestite da istruzion “if”. Infine con un ciclo “while” seguito dall’istruzione “switch” viene creato un menù per la segnalazione degli eventi avversi, visibile solo dopo aver effettuato l'accesso.

- *CONTROLLI*

La classe “**Controlli**” è utilizzata sia dalla classe Cittadini sia dalla classe Centri per effettuare controlli durante le varie operazioni da svolgere.

- Il metodo “**controllocf**” è utilizzato per controllare che il codice fiscale rispetti il pattern; per svolgere questa funzione utilizza una istruzione “if else”.

- Il metodo “controllocf2” è utilizzato per controllare che il codice fiscale non sia già presente all'interno del file "Cittadini_Registrati.dati.txt". Questo metodo utilizza istruzioni “if else” e cicli “while” per controllare che esista il file dei registrati e confronta il codice fiscale con quelli letti dal file.

- Il metodo “**controlloCentro**” controlla che il nome del centro non sia già presente all'interno del file "CentriVaccinali.txtt". La struttura è medesima a quella del metodo precedente, utilizza cioè istruzioni “if else” e cicli “while”.

- Il metodo “**controllocf2vac**” controlla se il codice fiscale sia già presente all'interno del file "Cittadini_Vaccinati.txt". Per svolgere l’operazione utilizziamo sempre istruzioni “if else” e cicli “while”.

- Il metodo “**controlladata**” che controlla che la data rispetti il pattern "dd/MM/yyyy". Utilizziamo diverse istruzioni “if else” per controllare se la data di nascita sia precedente confronto a quella inserita e per controllare se il formato con cui è stata inserita rispetta il pattern stabilito.

- Il metodo “**cercaCentro**” controlla l'esistenza del Centro di Vaccinazione. Utilizziamo istruzioni “if else” e cicli “while” per controllare che esista il file dei centri registrati e se il nome esiste tra quelli dei centri registrati.

- Il metodo “**GetVac**” mostra il centro di vaccinazione del codice fiscale inserito come parametro. Questo metodo utilizza istruzioni “if else” e cicli “while” per svolgere le sue funzioni.

- il metodo “**controlloCommento**” controlla, tramite una istruzione “if else” che il commento non sia più lungo di 256 caratteri.

- *EndofFile*

La classe “**EndofFile**” è utilizzata per la serializzazione ed implementa l’interfaccia “Serializable”.

- *EVENTI*

La classe “**Eventi**” è la classe che si occupa della segnalazione degli eventi avversi ed implementa l’interfaccia “Controlli”.

- Il metodo “**newSegnalazione**” che ha il compito di salvare una nuova segnalazione sotto forma di stringa, la quale controlla l'esistenza del file delle segnalazioni del centro di vaccinazione in cui è stato vaccinato l'utente e vi ci scrive la nuova segnalazione, in caso contrario, lo crea e vi ci scrive la segnalazione. Inizialmente deserializza l’array contenente i nomi dei centri di vaccinazione che hanno un file per le segnalazioni, se il centro in cui è stato vaccinato l’utente non è tra questi lo aggiunge e crea il file scrivendovi sopra la segnalazione, altrimenti salva direttamente la nuova segnalazione all’interno del file delle segnalazioni del centro.

- Il metodo “**Interface**” rappresenta l’interfaccia utente di segnalazione eventi avversi. Tramite l’utilizzo dello “Scanner” e di istruzioni “if” questa interfaccia di segnalazione assegna ad ogni patologia un determinato grado di severità, infine richiama il metodo “newSegnalazione.

BIBLIOGRAFIA E CITAZIONI

- Definizione java.util tratto da https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html
- Definizione java.io tratto da http://www.upv.es/~jgonsol/tutorial/tools/packages/java.io.html
- Definizione java.text tratto da https://docstore.mik.ua/orelly/java-ent/jnut/ch22_01.htm
- Definizione java.lang.RuntimeException tratto da https://javadoc.scijava.org/Java6/java/lang/RuntimeException.html