Manuale Tecnico

Università degli Studi dell'Insubria – Laurea Triennale in Informatica
Anno accademico 2022/2023 Progetto del Laboratorio A: CLIMATE MONITORING
Sviluppato da:

- Diagne mohamadou fadall , matricola 754545 , Varese
- Di Tullio Edoardo , matricola 753918 , Varese
- Moranzoni Samuele , matricola 754159 , Varese
- Zaninello Simone , matricola 753751 , Varese

Climate Monitoring è un progetto sviluppato nell'ambito del progetto di Laboratorio A dell'anno 2022-2023 per il corso di laurea in Informatica dell'Università degli Studi dell'Insubria. Il progetto è sviluppato in Java 17 ed è stato sviluppato e testato sul sistema operativo Windows 11.

Sommario

Struttura generale del sistema di classi e relative relazioni

L'applicazione segue un'organizzazione a package (package ClimateMonitoring) per migliorare la modularità e la manutenibilità del codice.
Le classi create e utilizzate per la realizzazione del software sono:

- AreeInteresse
- CentroMonitoraggio
- Operatore
- SistemaAccesso
- SistemaRegistrazione
- SistemaRicerca
- SistemaSelezione
- SistemaInserimento
- SistemaAggiunta
- ClimateMonitor(main)

Analizziamole nello specifico:

- ***classe AreeInteresse:*** Rappresenta un'area di interesse con attributi come nome, nazione e coordinate. Presenta metodi getter e fornisce un metodo "toCsvString()" per convertire i dati in formato CSV.
- ***classe CentroMonitoraggio***: Rappresenta un centro di monitoraggio con nome, indirizzo fisico e elenco di aree di interesse.Difatti come concordato da specifiche , l'oggetto CentroMonitoraggio contiene oggetti di tipo AreeInteresse .Presenta metodi getter relativi ai 3 attributi.Inoltre fornisce un metodo "toCsvString()"per convertire i dati in formato CSV,analogamente al metodo di AreeInteresse.
- ***ClimateMonitor(main):*** la classe Main é la classe principale.Contiene il punto d'ingresso principale dell'applicazione. Gestisce l'interazione utente attraverso un menu e coordina le diverse funzionalità.
  L'interazione utente è favorita dalla classe Scanner e in particolare dal suo metodo "nextint()" che permette all'utente di fornire valore numerici per comunicare con l'applicazione.
  L'applicazione integra(tramite istanziazione) i sistemi di selezione (SistemaSelezione), ricerca (SistemaRicerca) e inserimento (SistemaInserimento) per gestire le operazioni principali riguardanti i dati climatici, le aree di interesse e i centri di monitoraggio.

L'area operatori è gestita attraverso il sistema di registrazione (SistemaRegistrazione) e accesso (SistemaAccesso).
La struttura principale è un ciclo while che continua fino a quando l'utente non decide di uscire dall'applicazione. All'interno di questo ciclo, vengono presentate le principali opzioni di scelte all'utente, consentendo di navigare tra diverse funzionalità.
Inizialmente viene stampata a schermo un menu' che illustra le 3 scelte che un utente puo' adoperare,tramite input numerico e gestita dal sistema tramite l'istruzione condizionale switch con relativi case :

- ***case 1*** : menu' di selezione delle aree geografiche,in cui avviene la ricerca e visualizzazione dei dati sulle aree geografiche , definita dalla ricerca per denominazione (il sistema prende in input una stringa di caratteri e restituisce le aree nel cui nome compare la stringa di caratteri) e per Stato di appartenenza o dalla ricerca per coordinate geografiche (il sistema prende in input una latitudine e longitudine e restituisce il nome dell'area corrispondente alle coordinate geografiche/delle aree corrispondenti con coordinate più vicine). Successivamente l'utente puo' confermare l'area di interesse di cui visualizzare i dati o altrimenti uscire dal menu' di selezione dei dati.Il sistema gestisce l'ulteriore richiesta dell'utente tramite un altra istruzione switch-case composta da case 1 (con la visualizzazione dei dati di interesse tramite l'istanziazione della classe SistemaSelezione e il suo metodo "visualizzaAreaGeografica()") e default (implica l'uscita dal menu di selezione dei dati senza visualizzare i dati)
- ***case 2 :*** menu dell' area operatore, destinato agli utenti autorizzati.Anche in questo caso le varie scelte adoperabili dall'operatore sono gestite tramite un'istruzione switch-case:
  -l'utente puo' registrarsi(case 1) , contesto gestito tramite la classe SistemaRegistrazione e il suo metodo "Registrazione()"
  -l'utente puo' accedere (case 2),contesto gestito tramite la classe SistemaAccesso, in questo caso si provvede a verificare tramite un ciclo while che l'utente inserisca un userid consistente e quindi non nullo.Le stringhe di tipo userid e password sono passate come parametro al metodo "accesso() " dell'oggetto di tipo SistemaAccesso che verifica l'effettiva corrispondenza con un profilo operatore già registrato in ""OperatoriRegistrati.dati.txt" e stampa a schermo "accesso eseguito con successo". Ora ad una stringa chiamata login viene assegnato l'output del metodo "accesso()" con parametri userid e password, da implementazione il metodo accesso restituisce la stringa userid se un'autenticazione avviene correttamente altrimenti una stringa vuota.Questo passaggio è fondamentale per determinare se un utente che ha tentato l'accesso e quindi l'autenticazione puo' accedere all'area riservata. E' presente infatti il metodo "equals()" che

confronta la stringa login con una stringa vuota.Se la stringa login non é vuota(restituita indirettamente dal metodo accesso ,quindi garanzia di una corretta autenticazione) si puo' accedere all'area riservata.Alternativamente inserendo un valore diverso da 1 e 2 si esce dall'area di autenticazione dell'operatore(default).
All'interno dell'area riservata troviamo , a livello logico, scelte analoghe al resto del codice: un ciclo while per permettere di rimanere all'interno dell' area riservata (a meno che' non ci sia esplicita richiesta di uscita da parte dell'operatore) e un'altra istruzione switch-case per gestire le funzionalità:

Aggiunta di Parametri di Monitoraggio (case 1):

Se l'utente seleziona l'opzione per aggiungere parametri di monitoraggio, il programma richiama il metodo "inserisciParametriClimatici()" dell'oggetto sisIns di tipo SistemaInserimento per consentire all'utente di inserire i dati necessari.Dopo l'inserimento dei dati, viene visualizzato un messaggio confermando che i parametri sono stati aggiunti correttamente.

Creazione di Centri di Monitoraggio (case 2):

Se l'utente sceglie di creare un nuovo centro di monitoraggio, il programma richiama il metodo "insCentroAree() " dell'oggetto sisIns di tipo SistemaInserimento per consentire all'utente di inserire le informazioni necessarie per il nuovo centro.
Una volta completata la creazione del centro, il nome del centro viene aggiunto al file "OperatoriRegistrati.dati.txt" per tener traccia dei centri di monitoraggio registrati relativi agli operatori.Questo avviene tramite l'invocazione di un oggetto di tipo SistemaAggiunto e del suo metodo relativo "AggiungiAreaaa()"

Aggiunta di un'Area di Interesse (case 3):

Se l'utente sceglie di aggiungere un'area di interesse, il programma visualizza un messaggio che indica che l'operazione è in corso.Successivamente, il programma richiama il metodo" insArea()" dell'oggetto sisIns di tipo SistemaInserimento per consentire all'utente di inserire i dettagli relativi all'area di interesse.Una volta completata l'aggiunta dell'area di interesse, viene visualizzato un messaggio confermando che l'operazione è stata eseguita con successo.
Uscita dall'Area riservata (default):
Se l'utente seleziona un'opzione diversa da 1, 2 o 3, l'applicazione esce dall'area riservata(ponendo il booleano esecuzione4 uguale a false e uscendo dunque dal ciclo while relativa all'area riservata)

Uscita dall'area operatore(default): digitando un numero diverso da 1, 2 e 3 si esce dall'area operatore,ponendo la variabile booleana ***esecuzione3*** pari a false.

- ***default:uscita dal programma***
  L'utente fornendo in input un valore diverso da 1 e 2 esce dall'applicazione ,viene assegnata alla variabile booleana ***esecuzione*** il valore false e quindi impostata l'uscita dal ciclo while (esecuzione),occorre così il culmine dell'esecuzione dell'applicazione.

- ***classe Operatore:*** Rappresenta un operatore con attributi come nome, cognome, codice fiscale, email, userid, password e centro di monitoraggio.Presenta metodi getter e setter per ogni attributo.
- ***classe SistemaAccesso:*** Gestisce l'accesso degli operatori attraverso la verifica delle credenziali memorizzate nel file "OperatoriRegistrati.dati.txt" tramite il metodo "accesso()" (che restituisce una stringa relativa allo userid se l'operatore sta accedendo al sistema con le credenziali corrette , una stringa vuota se l'autenticazione non é avvenuta correttamente e cioè l'operatore ha sbagliato ad inserire le credenziali o non è registrato)
- ***classe SistemaInserimento:*** Gestisce l'inserimento di nuovi dati, inclusi centri di monitoraggio(metodo" insCentroAree() ", crea una tupla per ogni centro di monitoraggio con parametri inseriti dall'utente e la salva nel file CentroMonitoraggio.dati), aree di interesse(con il metodo "insArea()" che salva le aree di interesse con parametri forniti in input dall'utente nel file "CoordinateMonitoraggio.dati.csv") e parametri climatici(con il metodo "insParametriClimatici()" che salva nel file "ParametriClimatici.dati.csv" i parametri climatici con informazioni fornite in input dall'utente)
- ***classe SistemaRegistrazione:*** Si occupa della registrazione di nuovi operatori tramite il metodo "Registrazione()", richiedendo e memorizzando informazioni personali che vengono salvate nel file "OperatoriRegistrati.dati.txt" . Il sistema verifica che non siano fornite in input dall'utente informazioni personali inconsistenti e nulle.
- ***classe SistemaRicerca:*** Gestisce la ricerca di aree geografiche basata su criteri come nome, stato o coordinate.Presenta il metodo "cercaAreaGeografica()" che restituisce in output una lista di aree i cui parametri corrispondono a quelli forniti in input dall'utente. Sfrutta un oggetto di tipo BufferReader per leggere le aree presenti in "CoordinateMonitoraggio.dati.csv" e verificare la corrispondenza tra i parametri delle aree registrate e i parametri forniti in input dall'utente.
- ***classe SistemaSelezione:*** Si occupa della visualizzazione dei dati di parametri climatici per un'area di interesse specifica.Utilizza il metodo

“visualizzaAreaGeografica” che ,tramite un oggetto di tipo BufferReader, legge i parametri relativi ad un’area di interesse.

- ***classe SistemaAggiunta:*** Permette di aggiungere centri di monitoraggio per uno specifico operatore registrato nel file "OperatoriRegistrati.dati.txt" ,utilizza un metodo AggiungiAreaaa(**String riferimento,String nuovastringa**) che ricerca nel file relativo agli operatori registrati la **stringa riferimento** e aggiunge ,nella stessa riga , la stringa **nuovastringa** separata da uno spazio vuoto.

Scelte architetturali

L'applicazione segue un'architettura basata su classi e utilizza il paradigma di programmazione orientata agli oggetti (OOP). Le funzionalità principali sono suddivise in classi separate per garantire una struttura chiara e modulare e favorire la riutilizzabilità del codice.L’applicazione segue inoltre un organizzazione a package(package ClimateMonitoring).

Strutture dati utilizzate

1)List: Utilizzata in diverse classi per gestire elenchi dinamici di aree di interesse, operatori, parametri climatici:

- ***List<String> risultati*** (SistemaRicerca.cercaAreaGeografica): memorizza i nomi delle aree geografiche che soddisfano la query di ricerca.
- ***List<String> lines*** (SistemaSelezione.visualizzaAreaGeografica): memorizza i dati climatici dell'area geografica specificata.
- ***List <String> areeDiInteresse***(SistemaInserimento.insCentroAree): memorizza le aree di interesse di uno specifico centro di monitoraggio

2)Array:

- ***String[] campi*** (SistemaRicerca.cercaAreaGeografica, SistemaSelezione.visualizzaAreaGeografica): utilizzato per dividere una riga di testo in campi separati da punto e virgola.In particolare per dividere il nome dell’ area e il nome di stato di un’ area di interesse presente in "CoordinateMonitoraggio.dati.csv"

- ***String[] credentials*** (SistemaAccesso.accesso): utilizzato per dividere una riga di testo relativo ad un operatore registrato in più stringhe singole separate ,in particolare per ottenere password e userid di ogni operatore registrato.
- ***String [] parti*** (SistemaRicerca.cercaAreaGeografica):utilizzata per dividere una query di ricerca relativa a latitudine e longitudine in 2 stringhe separate di latitudine e longitudine.
- ***String [] fields*** (SistemaSelezione.visualizzaAreaGeografica): utilizzato per separare le informazioni riguardo ai parametri di un'area di interesse.

File e gestione

I file con la quale lavora l'applicazione ClimateMonitoring sono:

- ***OperatoriRegistrati.dati.txt:*** Memorizza le informazioni sugli operatori registrati con un formato tabellare.Questo file è utilizzata nella classe SistemaRegistrazione(in scrittura) , SistemaAccesso(in lettura) e SistemaAccesso(in lettura)
- ***CentroMonitoraggio.dati.txt:*** Contiene i dati relativi ai centri di monitoraggio in formato tabellare.Viene utilizzato in scrittura nella classe SistemaInserimento.
- ***CoordinateMonitoraggio.dati.csv:*** Memorizza le informazioni sulle aree di interesse, incluse coordinate, in formato CSV.E' usato in lettura nella classe SistemaRicerca e in scrittura nella classe SistemaInserimento.
- ***ParametriClimatici.dati.csv:*** Contiene i parametri climatici registrati con informazioni su centro di monitoraggio, area di interesse e data di rilevazione in formato CSV.E' utilizzato in lettura nella classe SistemaSelezione e in scrittura nella classe SistemaInserimento.

Il relative path di ogni file è indicato da : "data\\nomefile.estensione"

Scelte algoritmiche

SistemaAccesso. accesso():

La complessità complessiva dell'algoritmo è principalmente dominata dalla lettura del file e dalla scansione delle righe, quindi può essere approssimata a O(N*M) dove N è il numero di righe presenti nel file e M é la lunghezza media di una riga.

SistemaAggiunta.aggiungiCentro():

La complessità totale dell'algoritmo è dominata dalla lettura e dalla scrittura del file, quindi può essere approssimata a O(N*M), dove N è il numero di righe nel file e M è la lunghezza media di ogni riga.

SistemaRegistrazione.Registrazione():

La complessità totale dell'algoritmo è dominata dal numero di iterazioni del ciclo while, che è determinato dall'interazione con l'utente. Poiché ogni iterazione del ciclo richiede un tempo costante O(1), la complessità complessiva dell'algoritmo è O(k), dove k è il numero di iterazioni del ciclo while necessarie per completare la registrazione.

SistemaRicerca.Cercaareageografica():

La complessità totale del metodo dipende principalmente dal numero di righe nel file CSV e dal numero di risultati che soddisfano le condizioni specificate. Se indichiamo con n il numero di righe nel file e con k il numero di risultati, la complessità totale del metodo cercaAreaGeografica() può essere approssimata come O(n+k).

SistemaSelezione.VisualizzaAreeGeografiche():

La complessità totale del metodo dipende principalmente dal numero di righe nel file CSV. Se indichiamo con n il numero di righe nel file, la complessità totale del metodo visualizzaAreaGeografica() può essere approssimata come O(n).

SistemaSelezione.printFirstAndLastEight():

La complessità complessiva di printFirstAndLastEight è determinata principalmente dal numero di elementi da stampare, con un massimo di O(8) e un minimo di O(1) se la lista contiene meno di otto elementi.

SistemaInserimento.insCentroAree():

La complessità di questo metodo è O(n), dove n é il numero di iterazioni del ciclo while.Il resto delle operazioni nel metodo è principalmente costante rispetto al numero di aree di interesse inserite, quindi trascurabile rispetto alla complessità del ciclo while.

SistemaInserimento.insArea():

Tutte le operazioni all'interno del metodo sono essenzialmente costanti e non dipendono dalla dimensione dei dati di input, la complessità algoritmica di insArea() può essere approssimata come O(1), cioè costante.

SistemaInserimento.inserisciParametriClimatici():

La complessità algoritmica complessiva del metodo può essere approssimata come O(1).