Laboratorio interdisciplinare A

Climate Monitoring

Un sistema di monitoraggio di parametri climatici

CATTANEO LUCA

FICARA PAOLO  
MAURI ANDREA

# Introduzione

Climate monitoring è un progetto sviluppato nell’ambito del progetto di Laboratorio A per il corso di laurea in Informatica dell’Università degli Studi dell’Insubria.

Il progetto è stato sviluppato in Java 17, usa un’interfaccia grafica costruita con Java Swing, libreria inclusa in NetBeans apposita per lo sviluppo del design delle applicazioni desktop, ed è stato sviluppato e testato su Windows 11.

# Librerie esterne utilizzate

Durante lo sviluppo di questo progetto è stata utilizzata una libreria di terze parti in aggiunta, chiamata “codice-fiscale-java-master”, sviluppata dagli autori del progetto e inclusa nel progetto, ai fini di denominare il codice fiscale avendo tutte le generalità necessarie fornite dall’utente in fase di registrazione.

# Struttura generale del sistema di classi

Nel progetto si possono suddividere le classi in due macro-rami: classi adibite per la gestione dei dati e la rappresentazione degli oggetti in oggetto e classi impiegate alla gestione dell’interfaccia grafica.

* Gestione dei dati e rappresentazioni oggetti
  + - DatiCondivisi
    - FileManager
    - Forecast
    - InterestingAreas
    - MonitoringStation
    - User
* Gestione dell’interfaccia grafica
  + - AddNotes
    - CreateMonitoringStation
    - Login
    - Menu(main)
    - MenuOperatore
    - Register
    - SearchResult

Verranno ora presentate le classi adibite alla gestione dei dati dettagliamente, non verranno invece affrontate nel particolare le classi per la gestione dell’interfaccia grafica essendo che non sono espressamente richieste nei requisiti del progetto.

# Classi per la gestione dei dati

InterestingAreas

InterestingAreas è la classe che rappresenta le aree di interesse ai quali si riferisce un utente.

L’oggetto che descrive l’area di interesse viene identificato con i seguenti attributi:

* Id: identificativo numerico univoco rappresentato dal tipo String
* name: nome dell’area di interesse rappresentato dal tipo String
* countryCode: il codice del paese dell’area di interesse rappresentato dal tipo String
* lat: la latitudine dell’area di interesse rappresentata dal tipo String
* lon: la longitudine dell’area di interesse rappresentata dal tipo String

Per ogni attributo sono presenti get e set, inoltre è presente un metodo “contains” che verifica se il nome dell’area di interesse contiene una determinata stringa.

MonitoringStation

MonitoringStation è la classe che rappresenta le stazioni di monitoraggio che un operatore può creare.

L’oggetto che descrive la stazione di monitoraggio viene identificato con i seguenti attributi:

* name: il nome della stazione di monitoraggio rappresentato dal tipo String
* address: l’indirizzo della stazione di monitoraggio rappresentato dal tipo String
* interestingAreas: le aree di interesse della stazione di monitoraggio rappresentato da un vettore di String

Per ogni attributo sono presenti get e set.

User

User è la classe che rappresenta un operatore.

L’oggetto che descrive un operatore viene identificato con i seguenti attributi:

* name: il nome dell’operatore rappresentato dal tipo String
* surname: il cognome dell’operatore rappresentato dal tipo String
* cf: il codice fiscale dell’operatore rappresentato dal tipo String
* mail: l’indirizzo mail dell’operatore rappresentato dal tipo String
* nick: il nickname dell’operatore rappresentato dal tipo String
* password: la password dell’operatore rappresentato dal tipo String
* station: la stazione di monitoraggio dell’operatore rappresentato dal tipo String

Per ogni attributo sono presenti get e set, inoltre è presente un metodo statico che crea un oggetto User a partire da una stringa CSV contenente le generalità prese dal file dove sono memorizzati gli operatori.

Forecast

Forecast è la classe che rappresenta una previsione metereologica fatta da un operatore.

L’oggetto viene identificato con i seguenti attributi:

* idCitta: l'ID della città rappresentato dal tipo String
* NomeStazione: il nome della stazione rappresentato dal tipo String
* data: la data della previsione rappresentato dal tipo Date
* ora: l'ora della previsione rappresentato dal tipo Date
* vento: i dati relativi al vento rappresentato da un vettore di String
* umidita: i dati relativi all'umidità rappresentato da un vettore di String
* pressione: i dati relativi alla pressione rappresentato da un vettore di String
* temperatura: i dati relativi alla temperatura rappresentato da un vettore di String
* precipitazioni: i dati relativi alle precipitazioni rappresentato da un vettore di String
* altitudine: i dati relativi all’latitudine dei ghiacciai rappresentato da un vettore di String
* massa: i dati relativi alla massa dei ghiacciai rappresentato da un vettore di String

La classe fornisce inoltre get e set per ogni attributo e un metodo “toCSV” che restituisce una rappresentazione in formato CSV della previsione metereologica.

FileManager

FileManager è una classe prettamente di metodi statici ed è una delle classi fulcro del progetto in quanto garantisce la lettura e scrittura su file. Sfrutta la classe “java.nio”, un set di API per la lettura e scrittura su file, in quanto fornisce metodi con prestazioni molto efficienti per appunto le due funzioni principali delle operazioni su file soprattutto per file di notevoli dimensioni. Le operazioni di lettura e scrittura da parte di questa classe sono solitamente eseguite utilizzando un buffer per memorizzare temporaneamente i dati da leggere o scrivere.

Di seguito i metodi forniti dalla classe FileManager:

* **readUser**

prende come parametro un oggetto di tipo Path che identifica il percorso del file da leggere e restituisce una lista di User che rappresenta una lista di tutti gli operatori che si sono registrati all’interno dell’applicazione. Sfrutta il metodo “readAllLines” della java.nio;

* **readAreas**

prende come parametro un oggetto di tipo Path che identifica il percorso del file da leggere e restituisce una lista di InterestingAreas che rappresenta una lista di tutti le aree di interesse che sono state registrate all’interno dell’applicazione. Sfrutta il metodo “readAllLines” della java.nio;

* **readStation**

prende come parametro un oggetto di tipo Path che identifica il percorso del file da leggere e restituisce una lista di MonitoringStation che rappresenta una lista di tutte le stazioni di monitoraggio che sono state registrate/create all’interno dell’applicazione. Sfrutta il metodo “readAllLines” della java.nio;

* **readForecast**

prende come parametro un oggetto di tipo Path che identifica il percorso del file da leggere e restituisce una lista di Forecast che rappresenta una lista di tutte le rilevazioni metereologiche che sono state registrate all’interno dell’applicazione. Sfrutta il metodo “readAllLines” della java.nio;

* **write**

scrive il contenuto specificato come parametro nel file specificato anch’esso come parametro. Per evitare problemi di charset si è usato il classico “UTF-8”. Invece per la scrittura in sé viene sfruttato un buffer di tipo BufferedWriter eseguendo una scrittura in append sul file.

Per evitare problemi nella lettura legati ad eccezioni di tipo IOException viene controllato se la linea a cui ci si sta riferendo non sia vuota.

DatiCondivisi

DatiCondivisi è una classe Singleton che memorizza i vari contenuti dei file utili del progetto in delle liste apposite. Di per certo è la classe più importante del progetto in quanto offre i metodi principali per i vari calcoli delle distanze tra le città ed inoltre fornisce tutti i dati utili.

Possiede sei attributi:

* instance: identifica l’istanza della classe
* monitoringStations: una lista di MonitoringStation che identifica le stazioni di monitoraggio dove fare le rilevazioni
* users: una lista di User che identifica gli operatori registrati
* areas: una lista di InterestingAreas che identifica tutte le aree di interesse disponibili
* forecasts: una lista di Forecast che identifica tutte le rilevazioni fatte fino a quel momento
* operatore: l’operatore attualmente in sessione

Sono presenti i metodi get per tutti gli attributi, solo per l’attributo operatore è presente il set dato che l’operatore in sessione può continuamente cambiare.

Inoltre la classe fornisce altri metodi: