CLIMATE MONITORING
Manuale Tecnico

Iuri Antico *matricola*: 753144

Michael Bernasconi *matricola*: 752259

Gabriele Borgia *matricola*: 753262

Beatrice Balzarini *matricola*: 752257

21 maggio 2023

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Librerie esterne utilizzate

1.1.1 Apache Commons CSV

E' una libreria di Apache che fornisce i metodi per la gestione dei file (*.CSV). In particolare è stata usata nel progetto per la lettura e scrittura dei dati da memorizzare.

1.1.2 Apache Commons CLI

E' una libreria di Apache che fornisce i metodi per la gestione della linea di comando all'interno del terminale.

Capitolo 2

Struttura generale del sistema di classi

2.1 cli

2.1.1 Help

2.1.2 Login

2.1.3 MainMenu

2.1.4 MostraMisurazioni

2.1.5 Registrazione

2.2 gestori

All'interno del package Gestori si trovano una serie di classi finalizzate alle operazioni di lettura e scrittura su File contenenti dati utili al monitoraggio di parametri climatici sul territorio italiano. Ad ogni record memorizzato su file viene associato un indice (ID) univoco.

2.2.1 DataBase

La classe DataBase si occupa di creare (per ogni classe che estende Gestore) un oggetto in grado di richiamare le funzioni associate.

2.2.2 Gestore

La classe astratta Gestore contiene metodi relativi la gestione dei file contenenti i dati d'interesse. In particolare:

- **void close()** Metodo che si occupa di aprire un file.
- **void reload()** Metodo che si occupa di ricaricare (chiudere e riaprire un file)

- **DataTable buildObject (CSVRecord r)** Metodo astratto che si occupa di prendere in input come parametro formale un record e crea l'oggetto associato all'implementatore.
- **Result<String>getProperty(String key)** Metodo che si occupa di prendere una proprietà nel file (*.CSV.DAT) associato ad una tabella (*.CSV). Il parametro formale di questo metodo è una stringa che è la chiave per prendere la proprietà del file. Nel caso in cui l'operazione di ricerca è corretta il metodo restituisce un Result valido con la stringa della proprietà che hanno chiesto. Altrimenti nel caso in cui l'operazione non è corretta il metodo restituisce una stringa associata ad un codice di errore.
- **Result<Object>setProperty(String s, String k)** Metodo che si occupa di impostare una proprietà nel file (*.CSV.DAT) associato ad una tabella (*.CSV). Il metodo ha due parametri formali: una stringa che è la proprietà da impostare e l'altra è la chiave del file. Nel caso in cui l'operazione non è corretta il metodo restituisce un Result di Object come errore.

2.2.3 GestoreArea

la classe GestoreArea estende la classe Gestore e ne eredita tutti i metodi implementa l'interfaccia CercaAree e contiene i seguenti metodi:

- **Result<AreaGeografica>getArea(long geoID)** Metodo che ricerca una determinata area geografica in base al suo ID. In questo metodo c'è un solo parametro formale che il numero di geoID. Nel caso in cui l'area esiste il metodo restituisce un Result di AreaGeografica sennò restituisce una stringa di errore.

2.2.4 GestoreCentro

la classe GestoreCentro estende la classe Gestore, eredita tutti i metodi e contiene i seguenti:

- **Result<CentroMonitoraggio>getCentro(String nome)** Metodo che ricerca un determinato centro di monitoraggio in base al nome e restituisce il centro di monitoraggio corrispondente al nome di tipo String fornito come parametro formale.
- **boolean addCentro(CentroMonitoraggio cm)** Metodo che crea un nuovo record relativo a un determinato centro di monitoraggio cioè il parametro formale del metodo e lo memorizza nel file CentriMonitoraggio (.*CSV). Il metodo restituisce una variabile booleana per capire se l'operazione è andata a buon fine.

2.2.5 GestoreDato

la classe GestoreDato estende la classe Gestore, eredita tutti i metodi e contiene i seguenti:

- **Result <DatoGeografico >getDato(long rid)** Metodo che ricerca un determinato dato geografico in base al suo ID che è il parametro formale del metodo. Il metodo restituisce un Result di DatoGeografico se l'operazione è andata a buon fine. Invece Se l'operazione non è stata eseguita correttamente restituisce un Result di stringa per indicare l'errore.
- **Result <Object>addDato(DatoGeografico dato)** Metodo che crea un nuovo record relativo a un determinato dato geografico che è il parametro formale e lo memorizza nel file ParametriClimatici.CSV. Il metodo restituisce un Result di Object se l'operazione non è andata a buon fine.

2.2.6 GestoreMisurazioni

la classe GestoMisurazioni estende la classe Gestore, eredita tutti i metodi e contiene i seguenti:

- **Result<Object>addMisurazione(Misurazione mis)** Metodo che serve per aggiungere una Misurazione. Prende in input un parametro formale di tipo Misurazioni e restituisce un Result di Object in caso di errore.
- **Result <Filtratore>getMisurazioni()** Metodo che prende le Misurazioni e memorizza i record relativi alle misurazioni presenti nel file ParametriClimatici (*.CSV).

2.2.7 GestoreOperatore

la classe GestoreOperatore estende la classe Gestore, eredita tutti i metodi e contiene i seguenti:

- **Result<Operatore>registrazione(Operatore op, String pwd)** Metodo che permette a un operatore di registrarsi. Il metodo riceve in input come parametri formali, un oggetto di tipo Operatore e una password di tipo String. Il Metodo restituisce un Result di Operatore se l'operazione che deve svolgere il metodo è corretta. Altrimenti resituisce una stringa con il codice di errore.
- **Result <Operatore>login(String uid, String pwd)** Metodo che permette ad un operatore di effettuare il login. I parametri formali presi in input sono userid e la password. In caso che userid e password corrispondano ad un operatore creato, il metodo restituisce l'accesso dell'Operatore. Altrimenti il metodo restituisce una stringa con un codice di errore.

2.3 Magazzeno

All'interno del package Magazzeno si trovano una serie di classi che servono per memorizzare informazioni che vengono lette o che dovranno essere scritte su file.(*CSV).

2.3.1 AreaGeografica

la classe AreaGeografica implementa l'interfaccia DataTable, che definisce tutti i metodi e contiene i seguenti:
I metodi getter sono: getGeoID(), getLatitudine(), getLongitudine(), getStato(), getDenominazione().

- **String toString()** Metodo che permette di stampare l'oggetto AreaGeografica, con i campi denominazione, stato, latitudine e longitudine.
- **boolean equals(Object obj)** Metodo che permette di confrontare un oggetto qualsiasi con un oggetto di tipo AreaGeografia. Il metodo prende in input come parametro formale un Object. La funzione restituisce true se oggetto di tipo Object è un istanza di un' AreaGeografica. Altrimenti restituisce false.

2.3.2 CentroMonitoraggio

la classe CentroMonitoraggio implementa due interfacce: Convertable e Datatable che definisce tutti i metodi e contiene i seguenti: i metodi getter sono: getNome(), getIndirizzo(), getAree().

- **String toString()** Metodo che permette di stampare l'oggetto CentroMonitoraggio con i campi nome, indirizzo e le aree associate ad esso.
- **boolean equals(Object obj)** Metodo che permette di confrontare un oggetto qualsiasi con un oggetto di tipo CentroMonitoraggio. La funzione restituisce true se l'oggetto è un istanza di CentroMonitoraggio. Altrimento restituisce false.
- **String toCSV()** Metodo che permette di creare una stringa che rappresenta CentroMonitoraggio nel formato (*.CSV) adoperato all'interno del programma.

2.3.3 DatoGeografico

La classe DatoGeografico implementa l'interfaccia DataTable che definisce tutti i metodi e contiene i seguenti:

- **void setDato(TipoDatoGeografico tipo, byte dato)** Metodo che imposta il DatoGeografico. La funzione prende come parametri formali: il dato da impostare all'interno del DatoGeografico e il valore del dato.
- **byte getDato(TipoDatoGeografico tipo)** Metodo che prende come input un oggetto TipoDatoGeografico e restituisce il tipo di dato geografico associato.
- **String getNota(TipoDatoGeografico key)** Metodo che restituisce la nota del dato geografico. La nota che restituisce è dettata dal parametro formale del metodo di tipo TipoDatoGeografico.

- **boolean setNota(TipoDatoGeografico key, String nota)** Metodo che permette di impostare le note relative al DatoGeografico. Prende come parametri formali due input: uno è la chiave che stabilisce il come inserire la nota all'interno del dato geografico. L'altro paramento è la nota da inserire. La funzione restituisce true se l'esecuzione del metodo va a buon fine. Altrimenti false.
- **boolean equals(Object obj)** Metodo che prende in input un oggetto generico e se è un'istanza di DatoGeografico il metodo restituisce true. Altrimenti restituisce false.
- **String toString()** Metodo che restituisce una stringa per rappresentare il DatoGeografico
- **boolean noteEquals(DatoGeografico dato)** Metodo che prende in input un dato di tipo DatoGeografico da cui estrae la nota e la confronta con quella presente. Il metodo restituisce true se le due note sono uguali. Altrimenti restituisce false.
- **boolean datoEquals(DatoGeografico dato)** Metodo che prende in input come parametro formale un dato di tipo DatoGeografico. La funzione restituisce true se l'uguaglianza dei valori dati è verificata. Altrimenti flase.

2.3.4 Filtratore

La classe Filtratore implementa le interfacce :Iterable<Misurazione>, CercaAree, MediaAree, che definisce tutti i metodi e contiene i seguenti:

- **Filtratore filtra(DataTable... dts)** Metodo che permette di filtrare lr DataTable. Il metodo ha come parametro formale un numero variabili di elementi di tipo Datatable. La funzione restituisce le DataTable filtrate.
- **Filtratore filtraOperatore(Operatore... ops)** Metodo che permette di filtrare gli operatori. Il metodo ha come parametro formale un numero variabili di elementi di tipo Operatore. La funzione restituisce gli operatori filtrati.
- **Filtratore filtraCentro(CentroMonitoraggio... cms)** Metodo che permette di filtrare i centri di monitoraggio. Il metodo ha come parametro formale un numero variabili di elementi di tipo CentroMonitoraggio. La funzione restituisce i centri di monitoraggi filtrati.
- **Filtratore filtraAree(AreaGeografica... ags)** Metodo che permette di filtrare le Areegeografiche. Il metodo ha come parametro formale un numero variabili di elementi di tipo AreaGeografica. La funzione restituisce le aree geografiche filtrate.
- **Filtratore filtraNote(String... note)** Metodo che permette di filtrare le note. Il metodo ha come parametro formale un numero variabili di elementi di tipo String. La funzione restituisce le note filtrate.

- **Filtratore filtraDato(DatoGeografico... dati)** Metodo che permette di filtrare i dati di DatoGeografico. Il metodo ha come parametro formale un numero variabile di elementi di tipo DatoGeografico. La funzione restituisce i dati filtrati.
- **String toString()** Metodo che stampa l'oggetto filtratore.
- **DatoGeografico visualizzaAreaGeografica(AreaGeografica area)** Metodo che prende in input come parametro formale un oggetto di tipo AreaGeografica. Il metodo conta quante volte appare un valore dell'Area Geografica. Il metodo restituisce un oggetto di tipo DatoGeografico con il suo valore associato.
- **Iterator<Misurazione> iterator()** Metodo che restituisce un Iterator di Misurazione. contenuto...

2.3.5 Indirizzo

La classe Indirizzo definisce l'indirizzo dei centri di monitoraggio. Contiene i seguenti metodi:
i metodi getter sono: getNomeVia(), getCivico(), getCap(), getComune(), getProvincia().

- **String toString()** Metodo che restituisce una stringa contenente l'indirizzo.
- **String toCsv()** Metodo che permette di creare una stringa per descrivere l'indirizzo nel formato (*.CSV) adoperato all'interno del programma.

2.3.6 ListaAree

La classe ListaAree implementa le interfacce :Iterable<AreaGeografica>, CercaAree, Convertable che definisce tutti i metodi e contiene i seguenti:

- **boolean isEmpty()** Metodo restituisce true che indica se la listaAree è vuota. Altrimenti resituisce false.
- **AreaGeografica get(int k)** Metodo che prende in input un intero k che indica la posizione di un AreaGeografica all'interno della ListaAree. Il metodo restituisce l'AreaGeografica che è in posizione k.
- **void add(AreaGeografica e, int k)** Metodo che ha due parametri formali: AreaGeografica e un intero k. il metodo aggiunge l'AreaGeografica in posizione k.
- **AreaGeografica getFirst()** Metodo che restituisce il primo elemento nella ListaAree ch esegue il metodo
- **AreaGeografica getLast()** Metodo che restituisce l'ultimo elemento nella ListaAree che esegue il metodo.
- **int size()** Metodo che restituisce un intero che indica la dimensione della ListaAree.

- **void addFirst(AreaGeografica e)** Metodo che aggiunge un'area geografica che è il parametro formale alla ListaAree in prima posizione.
- **Iterator<AreaGeografica> iterator()** Metodo che restituisce un Iterator di AreaGeografica.
- **ListaAree cercaAreaGeografica(String denominazione, String stato)** Metodo che cerca le AreeGeografiche tramite i parametri formali cioè denominazione e stato. Il metodo restituisce una ListaAree dove tutte le AreeGeografiche all'interno contengo le informazioni dei parametri formali.
- **Result<AreaGeografica> cercaAreeGeografiche(double latitudine, double longitudine)** Metodo che permette di cercare le AreeGeografiche tramite i parametri formali dati in input, che sono due double che rappresentano latitudine e longitudine di un' AreaGeografica. Il metodo restituisce un Result di AreaGeografica.
- **String toString()** Metodo che restituisce una stringa che rappresenta l'AreaGeografica.
- **String toCsv()** Metodo che permette di creare una stringa per descrivere l'AreaGeografica nel formato (*.CSV) adoperato all'interno del programma.

2.3.7 Misurazioni

La classe Misurazioni implementa le interfacce: Convertable, DataTable che definisce tutti i metodi e contiene i seguenti:
Metodi getter: getRid(), getDato(), getTime(), getTimeString(), getOperatore(), getCentro(), getArea().

- **String toString()** Metodo che stampa l'oggetto Misurazione.

2.3.8 Operatore

La classe Operatore definisce i seguenti metodi:
Metodi getter: getCf(), getCentro(), getCognome(), getNome(), getEmail(), getUid().

- **Result<Object> inserisciParametri(AreaGeografica area, DatoGeografico dato, LocalDateTime tempo)** Metodo che prende in input come parametri formali degli oggetti di tipo: AreaGeografica, DatoGeografico, LocalDateTime. Il metodo consente di inserire i dati climatici di una determinata area nel database. Il metodo restituisce un Result di Object.
- **String toString()** Metodo che stampa l'oggetto Operatore.
- **String toCsv()** Metodo che permette di creare una stringa per descrivere l'Operatore nel formato (*.CSV) adoperato all'interno del programma.
- **boolean equals(Object obj)** Metodo che permette di confrontare un oggetto qualsiasi con un oggetto di tipo Operatore. Il metodo prende in input come

parametro formale un Object. La funzione restituisce true se oggetto di tipo Object è un istanza di Operatore. Altrimenti restituisce false.

2.4 Utils

2.4.1 listacustom

Package che contiene le classi che servono da supporto alla classe ListaAree.

2.4.1.1 CollezioniIterator

classe che implementa l'interfaccia Iterator, permette a ListaAree di svolegere l'istruzione "for-each loop".

- **E next()** Metodo che restituisce l'elemento corrente e scorre a quello successivo.
- **boolean hasNext()** Metodo che restituisce true se il nodo che esegue il metodo ha un successore, false altrimenti.

2.4.1.2 Nodo

Classe che rappresenta i nodi della lista, i metodi presenti servono per la gestione degli elementi della lista.
I metodi setter : setDato, setNext;
I metodi getter : getDato, getNext;

2.4.2 result

All'interno del package result sono presenti una serie di classi che servono per gestire i risultati di alcuni metodi dell'applicazione.

2.4.2.1 Panic

Classe che estende Error, serve a gestire degli errori che non è possibile catturare, lanciati dalla classe Result.

2.4.2.2 Result

Classe che si occupa della gestione dei risultati in alcuni metodi che potrebbero lanciare errori nell'applicazione.
I metodi getter: getError(), getMessage(), getFullMessage().

- **boolean isValid()** Metodo che restituisce true se il Result è valido. Altrimenti false.
- **boolen isError()** Metodo che restituisce true se il Result lancia un errore. Altrimenti false.

- **void ifValid(BiConsumer<T, Integer >fn)** Metodo che esegue la funzione data come parametro se il Result è valido.
- **void ifError(BiConsumer<T, Integer>fn)** Metodo che esegue la funzione data come parametro se il Result genera errore.
- **T get()** Metodo che restituisce il contenuto di Result.
- **T getOr(T other)** Metodo che restituisce il contenuto di Result se questo non è nullo. Altrimenti restituisce il paramentro formale other.
- **T getOrElse(Supplier <T>fn)** Metodo che restituisce il contenuto di Result se questo non è nullo. Altrimenti esegue la funzione fn data come parametro e restituisce il risultato di quest'ultima.
- **T except()** Metodo che restituisce il contenuto di Result senza eseguire nessun controllo.
- **void panic()** Metodo che lancia un errore non catturabile.

2.4.3 terminal

2.4.3.1 Screen

Classe che contiene il seguente metodo:

- **void show(View v)** Metodo che serve a mostrare View. Pulisce il terminale prima e dopo l'esecuzione dell'applicazione.

2.4.3.2 Terminal

2.4.3.3 View

Interfaccia che contiene il seguente metodo:

- **abstract void start(Terminal term)** Metodo astratto che serve per avviare una schermata dell'applicazione attraverso il terminale.

2.4.4 CercaAree

L'implementazione di questa interfaccia consente la ricerca di aree geografiche.

- **ListaAree cercaAreaGeografica (String denominazione, String stato)** Metodo che ricerca per denominazione e per stato di appartenenza, restituisce una lista di aree geografiche.
- **Result<AreaGeografica>cercaAreeGeografiche(double latitudine, double longitudine)** Metodo che ricerca per coordinate geografiche (prende in input una latitudine e longitudine) e restituisce un result di tipo AreaGeografica.

2.4.5 Convertable

L'implementazione di questa interfaccia consente a un oggetto di essere convertito nel formato (*.CSV).

- **String toCsv()** Metodo che converte l'oggetto nel formato (*.CSV). Restituisce una stringa.

2.4.6 DataTable

L'implementazione di questa interfaccia permette a due record di essere confrontati.

- **boolean equals(Object obj)** Metodo che confronta l'oggetto che richiama la funzione con l'oggetto fornito come parametro, restituisce true se sono uguali false altrimenti.

2.4.7 MediaAree

La classe che implementa questa interfaccia permette di visualizzare le informazioni relative ad un'area geografica.

- **DatoGeografico visualizzaAreaGeografica (AreaGeografica area)** Metodo che restituisce un nuovo dato geografico che rappresenta un prospetto riassuntivo dei parametri climatici associati all'area geografica fornita come input.

2.4.8 DatoGeografico

Enumerativo che rappresenta il tipo di un dato geografico.

2.5 Main