Manuale Tecnico

CLIMATE MONITORING

Iuri Antico *matricola*: 753144

Beatrice Balzarini *matricola*: 752257

Michael Bernasconi *matricola*: 752259

Gabriele Borgia *matricola*: 753262

26 luglio 2023

Indice

1 Introduzione

1.1 Librerie esterne utilizzate

1.1.1 Apache Commons CSV

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

1.1.2 Apache Commons CLI

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

2 Struttura generale del sistema di classi

2.1 cli

2.1.1 App

Classe che contiene il seguente metodo.

- **void start(CommandLine line)**
  Metodo che preleva gli parametri forniti dall'applicazione e avvia il comando corretto all'interno di essa.

  La complessità del metodo è **O(1)**.

2.1.2 ComandoCentri

Classe che contiene il seguente metodo.

- **void start(Terminal term)**
  Metodo che implementa l'interfaccia View.

  La complessità del metodo è **Θ(n)**.

2.1.3 ComandoMisurazioni

Classe che contiene il seguente metodo.

- **void start(Terminal term)**
  Metodo che implementa l'interfaccia View.

  La complessità del metodo è $\mathbf{O(n)}$.

2.1.4 MostraAree

Classe che contiene il seguente metodo.

- **void start(Terminal term)**
  Metodo che implementa l'interfaccia View.

  La complessità del metodo è $\mathbf{\Theta(n + m + k)}$.

2.1.5 MostraCentri

Classe che contiene il seguente metodo.

- **void start(Terminal term)**
  Metodo che implementa l'interfaccia View.

  La complessità del metodo è $\mathbf{O(n)}$.

2.1.6 MostraMisurazioni

Classe che contiene il seguente metodo.

- **void start(Terminal term)**
  Metodo che implementa l'interfaccia View.

  La complessità del metodo è $\mathbf{\Theta(n + m)}$.

2.1.7 Query

Classe che contiene il seguente metodo.

- **void start(Terminal term)**
  Metodo che implementa l'interfaccia View.

  La complessità del metodo è $\mathbf{O(n^2)}$.

2.1.8 Registrazione

Classe che contiene il seguente metodo.

- **void start(Terminal term)**
  Metodo che implementa l'interfaccia View.

  La complessità del metodo è $\mathbf{O(1)}$.

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 chiudere un file.

  La complessità del metodo è **O(1)**.

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

  La complessità del metodo è **O(1)**.

- **DataTable buildObject (CSVRecord r)**
  Metodo astratto che riceve in input come parametro un record e crea l'oggetto associato all'implementatore.

  La complessità del metodo è **O(1)**.

- **Result<String> getProperty(String key)**
  Metodo che si occupa di ottenere una proprietà nel file (*.CSV.DAT) associato ad una tabella (*.CSV).
  Riceve in input come parametro una stringa, che è la chiave per ottenere la proprietà del file.
  Nel caso in cui l'operazione di ricerca venga eseguita correttamente il metodo restituisce un Result valido con la stringa della proprietà richiesta.
  Nel caso in cui l'operazione non venga eseguita in maniera corretta, il metodo restituisce una stringa associata ad un codice di errore.

  La complessità del metodo è $\mathbf{\Theta(n)}$.

- **Result<Object> setProperty(String key, String val)**
  Metodo che si occupa di impostare una proprietà nel file (*.CSV.DAT) associato ad una tabella (*.CSV).
  Il metodo riceve in input due parametri: una stringa che indica la proprietà da impostare e la chiave del file.
  Nel caso in cui l'operazione non venga eseguita correttamente, il metodo restituisce un Result di Object come errore.

La complessità del metodo è **Θ(n)**.

2.2.3 GestoreArea

La classe GestoreArea estende 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.
  Riceve in input come parametro l'ID dell'area geografica.
  Nel caso in cui l'area esista il metodo restituisce un Result di AreaGeografica, altrimenti una stringa di errore.

  La complessità del metodo è **O(n)**.

2.2.4 GestoreCentro

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

- **Result<CentroMonitoraggio> getCentro(String nome)**
  Metodo che ricerca e restituisce l'oggetto CentroMonitoraggio il cui nome corrisponde alla stringa fornita come parametro .

  La complessità del metodo è **O(n)**.

- **boolean addCentro(CentroMonitoraggio cm)**
  Metodo che crea un nuovo record relativo a un determinato centro di monitoraggio fornito come parametro e lo memorizza nel file CentriMonitoraggio (.*CSV).
  Il metodo restituisce una variabile booleana che indica se l'operazione è stata eseguita correttamente.

  La complessità del metodo è **O(1)**.

2.2.5 GestoreDato

La classe GestoreDato estende 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, fornito come parametro.
  Restituisce un Result di DatoGeografico se l'operazione viene eseguita correttamente.
  In caso contrario il metodo restituisce un Result che ne indica l'errore.

  La complessità del metodo è **O(n)**.

- **Result <Object> addDato(DatoGeografico dato)**
  Metodo che crea un nuovo record relativo a un determinato dato geografico (fornito come parametro) e lo memorizza nel file *ParametriClimatici.CSV*.
  Il metodo restituisce un Result di DatoGeografico se l'operazione viene eseguita correttamente.
  In caso contrario il metodo restituisce un Result che ne indica l'errore.

  La complessità del metodo è **O(1)**.

2.2.6 GestoreMisurazioni

La classe GestoreMisurazioni estende Gestore, eredita tutti i metodi e contiene i seguenti:

- **Result<Object> addMisurazione(Misurazione mis)**
  Metodo che inserisce una nuova misurazione basandosi sul parametro fornito in input.
  Il metodo restituisce un Result di Misurazione se l'operazione viene eseguita correttamente.
  In caso contrario il metodo restituisce un Result che ne indica l'errore.

  La complessità del metodo è

- **Result <Filtratore> getMisurazioni()**
  Metodo che raccoglie e memorizza i record relativi alle misurazioni presenti nel file *ParametriClimatici (*.CSV)*.

  La complessità del metodo è **Θ(n)**.

2.2.7 GestoreOperatore

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

- **Result<Operatore> registrazione(Operatore op, String pwd)**
  Metodo non implementato, in quanto l'operatore richiede un centro di monitoraggio che il GestoreCentro fallisce nel restituire, impedendo la corretta registrazione dell'utente.

  La complessità del metodo è **O(1)**.

- **Result <Operatore> login(String uid, String pwd)**
  Metodo che permette ad un operatore di effettuare il login.
  I parametri forniti in input sono l'ID utente e la password. In caso questi ultimi corrispondano ad un operatore esistente, il metodo permette all'operatore di effettuare l'accesso.
  In caso contrario restituisce una stringa relativa a un codice di errore.

  La complessità del metodo è **O(n)**.

2.3 Magazzeno

All'interno del package Magazzeno si trovano una serie di classi finalizzate a memorizzare informazioni oggetto di operazioni di lettura o scrittura su file.(*CSV).

2.3.1 AreaGeografica

La classe AreaGeografica implementa l'interfaccia DataTable, e contiene i seguenti metodi:

- **getter**
  getGeoID(), getLatitudine(), getLongitudine(), getStato(), getDenominazione().

  La complessità dei metodi è **O(1)**.

- **String toString()**
  Metodo che permette di stampare l'oggetto AreaGeografica, mostrando i campi denominazione, stato, latitudine e longitudine.

  La complessità del metodo è **O(1)**.

- **boolean equals(Object obj)**
  Metodo che permette di confrontare un oggetto qualsiasi con un oggetto di tipo AreaGeografica.
  Riceve in input come parametro un Object.
  Restituisce *true* se l'oggetto di tipo Object è un istanza di AreaGeografica.
  In caso contrario il metodo restituisce *false*.

  La complessità del metodo è **O(1)**.

2.3.2 CentroMonitoraggio

La classe CentroMonitoraggio implementa le due interfacce Convertable e Datatable e contiene i seguenti metodi:

- **getter**
  getNome(), getIndirizzo(), getAree().

  La complessità dei metodi è **O(1)**.

- **String toString()**
  Metodo che permette di stampare l'oggetto CentroMonitoraggio con nome, indirizzo e aree associate.

  La complessità del metodo è **O(1)**.

- **boolean equals(Object obj)**
  Metodo che permette di confrontare un oggetto qualsiasi con un oggetto di tipo CentroMonitoraggio.
  Restituisce *true* se l'oggetto è un' istanza di CentroMonitoraggio.
  In caso contrario il metodo restituisce *false*.

  La complessità del metodo è **O(1)**.

- **String toCSV()**
  Metodo che permette di creare una stringa che rappresenta CentroMonitoraggio nel formato (*.CSV) adoperato all'interno del programma.

  La complessità del metodo è **O(1)**.

2.3.3 DatoGeografico

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

- **void setDato(TipoDatoGeografico tipo, byte dato)**
  Metodo che imposta un dato geografico.
  Riceve come parametri il tipo di dato da impostare e il suo valore.

  La complessità del metodo è **O(1)**.

- **byte getDato(TipoDatoGeografico tipo)**
  Metodo che riceve come parametro in input un oggetto TipoDatoGeografico e restituisce il tipo di dato geografico associato.

  La complessità del metodo è **O(1)**.

- **String getNota(TipoDatoGeografico key)**
  Metodo che preleva la nota del dato geografico che esegue il metodo.
  Restituisce la nota associata alla chiave fornita come parametro.

  La complessità del metodo è **O(1)**.

- **boolean setNota(TipoDatoGeografico key, String nota)**
  Metodo che permette di impostare le note relative al dato geografico.
  Riceve come parametri una chiave che stabilisce come inserire la nota all'interno del dato geografico e la nota da inserire.
  La funzione restituisce *true* se l'operazione viene eseguita correttamente, *false* altrimenti.

  La complessità del metodo è **O(1)**.

- **boolean equals(Object obj)**
  Metodo che riceve in input un oggetto generico.
  Restituisce *true* se l'oggetto che esegue il metodo è un'istanza di DatoGeografico, altrimenti *false*.

  La complessità del metodo è **O(1)**.

- **String toString()**
  Metodo che restituisce una stringa che rappresenta il dato geografico.

  La complessità del metodo è **O(1)**.

- **boolean noteEquals(DatoGeografico dato)**
  Metodo che riceve in input un oggetto di tipo DatoGeografico da cui estrae la relativa nota e la confronta con quella che esegue il metodo.
  Restituisce *true* se le due note sono uguali, *false* altrimenti.

  La complessità del metodo è **O(1)**.

- **boolean datoEquals(DatoGeografico dato)**
  Metodo che riceve in input come parametro un oggetto di tipo DatoGeografico.
  Restituisce *true* se l'uguaglianza dei valori dati è verificata, *false* altrimenti.

  La complessità del metodo è **O(1)**.

2.3.4 Filtratore

La classe Filtratore implementa le interfacce: Iterable, CercaAree, MediaAree e contiene i seguenti metodi:

- **Filtratore filtra(DataTable... dts)**
  Metodo che permette di filtrare le DataTable.
  Riceve come parametro una serie elementi di tipo Datatable.
  Restituisce le DataTable filtrate.

  La complessità del metodo è **O(n·m)**.

- **Filtratore filtraOperatore(Operatore... ops)**
  Metodo che permette di filtrare gli operatori.
  Riceve come parametro una serie di elementi di tipo Operatore.
  Restituisce gli operatori filtrati.

  La complessità del metodo è **O(n·m)**.

- **Filtratore filtraCentro(CentroMonitoraggio... cms)**
  Metodo che permette di filtrare i centri di monitoraggio.
  Riceve come parametro una serie di elementi di tipo CentroMonitoraggio.
  Restituisce i centri di monitoraggio filtrati.

  La complessità del metodo è **O(n·m)**.

- **Filtratore filtraAree(AreaGeografica... ags)**
  Metodo che permette di filtrare le aree geografiche.
  Riceve come parametro una serie di elementi di tipo AreaGeografica.
  Restituisce le aree geografiche filtrate.

  La complessità del metodo è **O(n·m)**.

- **Filtratore filtraNote(String... note)**
  Metodo che permette di filtrare le note.
  Riceve come parametro una serie di elementi di tipo String.
  Restituisce le note filtrate.

  La complessità del metodo è **O(n·m)**.

- **Filtratore filtraDato(DatoGeografico... dati)**
  Metodo che permette di filtrare i dati geografici.
  Riceve come parametro una serie di elementi di tipo DatoGeografico.
  Restituisce i dati geografici filtrati.

  La complessità del metodo è $\mathbf{O(n \cdot m)}$.

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

  La complessità del metodo è $\mathbf{\Theta(n)}$.

- **DatoGeografico visualizzaAreaGeografica(AreaGeografica area)**
  Metodo che riceve in input come parametro un oggetto di tipo AreaGeografica.
  Conta quante volte appare un valore dell'Area Geografica.
  Restituisce un oggetto di tipo DatoGeografico e il valore associato.

  La complessità del metodo è $\mathbf{\Theta(n)}$.

- **Iterator<Misurazione> iterator()**

  Metodo che restituisce un Iterator di Misurazione. contenuto...

  La complessità del metodo è .

2.3.5 Indirizzo

La classe Indirizzo definisce l'indirizzo dei centri di monitoraggio.
Contiene i seguenti metodi:

- **getter**
  getNomeVia(), getCivico(), getCap(), getComune(), getProvincia().

  La complessità dei metodi è $\mathbf{O(1)}$.

- **String toString()**
  Metodo che restituisce una stringa relativa all'indirizzo che lo esegue.

La complessità del metodo è **O(1)**.

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

  La complessità del metodo è **O(1)**.

2.3.6 ListaAree

La classe ListaAree implementa le interfacce Iterable, CercaAree e Convertable. Contiene i seguenti metodi:

- **boolean isEmpty()**
  Metodo che restituisce **true** se la ListaAree che esegue è vuota, *false* altrimenti.

  La complessità del metodo è **O(1)**.

- **AreaGeografica get(int k)**
  Metodo che riceve in input un intero k che indica la posizione di un AreaGeografica all'interno della ListaAree.
  Restituisce l'area geografica che è in posizione k.

  La complessità del metodo è **O(n)**.

- **void add(AreaGeografica e, int k)**
  Metodo che aggiunge l'AreaGeografica in posizione k.
  Riceve in input un oggetto di tipo AreaGeografica e un intero k.

  La complessità del metodo è **O(n)**.

- **AreaGeografica getFirst()**
  Metodo che restituisce il primo elemento presente nella ListaAree che esegue il metodo.

  La complessità del metodo è **O(1)**.

- **AreaGeografica getLast()**
  Metodo che restituisce l'ultimo elemento presente nella ListaAree che esegue.

  La complessità del metodo è **O(1)**.

- **int size()**
  Metodo che restituisce un intero che indica la dimensione della ListaAree che esegue.

  La complessità del metodo è **O(n)**.

- **void addFirst(AreaGeografica e)**
  Metodo che aggiunge in prima posizione della ListaAree che esegue un'area geografica, fornita in input come parametro.

  La complessità del metodo è **O(1)**.

- **Iterator<AreaGeografica> iterator()**
  Metodo che restituisce un Iterator di AreaGeografica.

  La complessità del metodo è **O(1)**.

- **ListaAree cercaAreaGeografica(String denominazione, String stato)**
  Metodo che ricerca le AreeGeografiche mediante i parametri denominazione e stato.
  Restituisce il risultato della ricerca in una ListaAree.

  La complessità del metodo è **O(n)**.

- **Result<AreaGeografica> cercaAreeGeografiche(double latitudine, double longitudine)**
  Metodo che ricerca le AreeGeografiche mediante i parametri forniti in input, latitudine e longitudine.
  Restituisce un Result di AreaGeografica.

  La complessità del metodo è **O(n)**.

- **String toString()**
  Metodo che restituisce una stringa che rappresenta l'area geografica che

esegue.

La complessità del metodo è **O(n)**.

- **String toCsv()**
  Metodo che permette di creare una stringa per descrivere l'area geografica che esegue nel formato (*.CSV) adoperato all'interno del programma.

  La complessità del metodo è **O(n)**.

- **Result<AreaGeografica> getArea(long geoId)**
  Metodo recupera l'area tramite il suo geoID.

  La complessità del metodo è **O(n)**.

2.3.7 Misurazioni

La classe Misurazioni implementa le interfacce Convertable e DataTable e contiene i seguenti metodi:

- **getter**
  getRid(), getDato(), getTime(), getTimeString(), getOperatore(), getCentro(), getArea().

  La complessità dei metodi è **O(1)**.

- **String toString()**
  Metodo che restituisce una stringa che rappresenta la misurazione che esegue.

  La complessità dei metodi è **O(1)**.

2.3.8 Operatore

La classe Operatore definisce i seguenti metodi:

- **getter**
  getCf(), getCentro(), getCognome(), getNome(), getEmail(), getUid().

  La complessità dei metodi è **O(1)**.

- **Result<Object> inserisciParametri(AreaGeografica area, DatoGeografico dato, LocalDateTime tempo)**
  Metodo che consente di inserire i dati climatici di una determinata area nel database i cui dati vengono forniti in input come parametri.
  Restituisce un Result di Object.

  La complessità del metodo è **O(1)**.

- **String toString()**
  Metodo che restituisce una stringa che rappresenta l'operatore che esegue.

  La complessità del metodo è **O(1)**.

- **String toCsv()**
  Metodo che crea una stringa relativa all'operatore che esegue nel formato (*.CSV) adoperato all'interno del programma.

  La complessità del metodo è **O(1)**.

- **boolean equals(Object obj)**
  Metodo che confronta un oggetto qualsiasi con un oggetto di tipo Operatore.
  Riceve in input come parametro un Object.
  Restituisce *true* se l'oggetto di tipo Object è un istanza di Operatore, *false* altrimenti.

  La complessità del metodo è **O(1)**.

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 permette a ListaAree di svolgere l'istruzione "for-each loop". Implementa l'interfaccia Iterator.

- **E next()**
  Metodo che restituisce l'elemento corrente e scorre a quello successivo.

  La complessità del metodo è **O(1)**.

- **boolean hasNext()**
  Metodo che restituisce *true* se il nodo che esegue il metodo ha un successore, *false* altrimenti.

  La complessità del metodo è **O(1)**.

2.4.1.2 Nodo

Classe che rappresenta i nodi della lista i cui elementi sono gestiti dai seguenti metodi:

- **setter**
  setDato(), setNext().

  La complessità dei metodi è **O(1)**.

- **getter**
  getDato(), getNext().

  La complessità dei metodi è **O(1)**.

2.4.2 result

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

2.4.2.1 Panic

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

2.4.2.2 Result

Classe che si occupa della gestione dei risultati in alcuni metodi che potrebbero lanciare errori nell'applicazione.

- **getter**
  getError(), getMessage(), getFullMessage().

  La complessità dei metodi è **O(1)**.

- **boolean isValid()**
  Metodo che restituisce *true* se il Result è valido, *false* altrimenti.

  La complessità del metodo è **O(1)**.

- **boolen isError()**
  Metodo che restituisce *true* se il Result lancia un errore, *false* altrimenti.

  La complessità del metodo è **O(1)**.

- **void ifValid(BiConsumer<T, Integer > fn)**
  Metodo che esegue la funzione data come parametro se il Result è valido.

  La complessità del metodo è **O(1)**.

- **void ifError(BiConsumer<T, Integer> fn)**
  Metodo che esegue la funzione data come parametro se il Result genera errore.

  La complessità del metodo è **O(1)**.

- **T get()**
  Metodo che restituisce il contenuto di Result.

  La complessità del metodo è **O(1)**.

- **T getOr(T other)**
  Metodo che restituisce il contenuto di Result se questo non è nullo.
  In caso contrario restituisce il parametro other.

  La complessità del metodo è **O(1)**.

- **T getOrElse(Supplier <T> fn)**
  Metodo che restituisce il contenuto di Result se questo non è nullo.
  In caso contrario esegue la funzione fornita come parametro e restituisce il risultato di quest'ultima.

  La complessità del metodo è **O(1)**.

- **T except()**
  Metodo che restituisce il contenuto di Result senza eseguire nessun controllo.

  La complessità del metodo è **O(1)**.

- **void panic()**
  Metodo che lancia un errore non catturabile.

  La complessità del metodo è **O(1)**.

2.4.3 terminal

2.4.3.1 Screen

Classe che contiene il seguente metodo:

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

La complessità del metodo è **O(1)**.

2.4.3.2 Terminal

Classe Involucro che racchiude System.in() e System.out(), aggiungendo varie funzionalità.

- **void clear()**
  Metodo che pulisce la console con il codice di uscita ANSI.

  La complessità del metodo è **O(1)**.

- **void printf(String str, Object... args)**
  Metodo che formatta e stampa una stringa nel terminale.
  I parametri formali servono a stampare all'utente una stringa (str) interpolata (args).

  La complessità del metodo è **O(1)**.

- **void printfln(String str, Object... args)**
  Metodo che formatta e stampa una stringa nel terminale.
  I parametri formali sono finalizzati a stampare all'utente una stringa (str) interpolata (args).

  La complessità del metodo è**O(1)**.

- **String readLine()**
  Metodo che permette di leggere una linea dalla console utente.
  Restituisce la stringa inserita dall'utente.

  La complessità del metodo è **O(1)**.

- **String readLine(String str, Object... args)**
  Metodo che stampa una stringa e aspetta una risposta dell'utente.
  I parametri formali sono finalizzati a stampare all'utente una stringa (str) interpolata (args).

  La complessità del metodo è **O(1)**.

- **String readLineOrDefault(String def, String str, Object... args)**
  Metodo che stampa una stringa all'utente e aspetta una risposta, se quest'ultima non viene data (stringa vuota), il metodo restituisce (def).
  I parametri formali sono finalizzati a stampare all'utente una stringa (str) interpolata (args).

  La complessità del metodo è **O(1)**.

- **String readWhile(Predicate<String> fn, String str, Object... args)**
  Stampa una stringa e aspetta una risposta dall'utente, controllando la stringa fornita come parametro.
  I parametri formali sono finalizzati a stampare all'utente una stringa (str) interpolata (args). In particolare fn è una funzione che restituisce un booleano.

  La complessità del metodo è **O(n)**.

- **boolean promptUser(boolean yes, String str, Object... args)**
  Metodo che pone all'utente una domanda con risposta di tipo si/no.
  I parametri formali sono finalizzati a stampare all'utente una stringa (str) interpolata (args). In particolare rappresenta la scelta di default.

  La complessità del metodo è **O(1)**.

2.4.3.3 View

Interfaccia che contiene il seguente metodo:

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

  La complessità del metodo è **O(1)**.

2.4.4 CercaAree

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

- **ListaAree cercaAreaGeografica (String denominazione, String stato)**
  Metodo che ricerca delle aree geografiche mediante denominazione e stato di appartenenza.
  Restituisce un Result di tipo AreaGeografica contenente le aree geografiche corrispondenti al risultato della ricerca.

- **Result<AreaGeografica> cercaAreeGeografiche(double latitudine, double longitudine)**
  Metodo che ricerca delle aree geografiche mediante coordinate (latitudine e longitudine fornite in input come parametro).
  Restituisce un Result di tipo AreaGeografica contenente le aree geografiche corrispondenti al risultato della ricerca.

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 relativa all'oggetto.

2.4.6 DataTable

L'implementazione di questa interfaccia permette di confrontare due record.

- **boolean equals(Object obj)**
  Metodo che confronta l'oggetto che esegue con l'oggetto fornito come parametro.
  Restituisce *true* se i due oggetti 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 in input.

  La complessità del metodo è .

2.4.8 IniFile

Classe che si occupa di leggere un file *.ini* ed eseguirne il parsing.

- **void load(String path) throws IOException**
  Metodo che carica il file *.ini*.

  La complessità del metodo è **Θ(n)**

- **String getString(String section, String key, String defaultvalue)**
  Metodo che preleva un valore identificato in base a chiave e a sezione fornite in input come parametri.

  La complessità del metodo è **O(1)**

- **int getInt(String section, String key, int defaultvalue)**
  Metodo che preleva il valore identificato dalla chiave e dalla sezione date come parametri e lo prova a convertire ad intero.

  La complessità del metodo è **O(1)**

- **double getDouble(String section, String key, double defaultvalue)**
  Metodo che preleva il valore identificato in base a chiave e a sezione fornite in input come parametri e lo converte a *double*.

  La complessità del metodo è **O(1)**

2.4.9 TipoDatoGeografico

Enumerativo che rappresenta il tipo di un dato geografico.

2.5 Main

Entry point dell'applicazione.

La complessità è **O(1)**

Bibliografia

[Ant23] Iuri Antico. LaTeXTemplate. https://github.com/chichibio-savoiardi/LaTeX-Template, 2023.