MANUALE TECNICO

CLIMATE MONITORING

Università degli Studi dell'Insubria-Laurea triennale in Informatica
Progetto Laboratorio A- Climate Monitoring
sviluppato da:
Riccardo Massini matricola 753291
Massimiliano De Lorenzo matricola 754160

Indice generale

INTRODUZIONE....3
STRUTTURA GENERALE DELLE CLASSI....3
LIBRERIE E PACKAGE UTILIZZATI....4
ECCEZIONI UTILIZZATE:....4
PACKAGE GESTIONEFILE:....5
GestisciOperatori....5
GestisciPaesi....7
Eccezioni:....8
GestisciCentri....9
Eccezioni:....10
GestisciParametri:....10
Eccezioni:....11
PACKAGE OGGETTI....12
OperatoriClimatici....12
Paese:....12
CentroMonitoraggio:....13
ParametriClimatici:....14
PACKAGE CONTROLLOECCEZIONI....15
GestioneInput....15
Eccezioni:....15
PACKAGE CLIMATEMONITORING....16
ClimateMonitor....16
DISCUSSIONE COMPLESSITÀ....17
SITOGRAFIA....17

INTRODUZIONE

Climate Monitor è un progetto sviluppato per il corso “Laboratorio A” del corso di laurea Informatica dell’Universià degli studi dell’Insubria.
Il progetto è stato sviluppato in Java 17 ed è stato progettato e testato sul sistema operativo Windows 11.

STRUTTURA GENERALE DELLE CLASSI

Il progetto è strutturato in 9 classi, ovvero le core classes, che si occupano dell’elaborazioni dei dati e della gestione dei file per salvare le informazioni inserite dall’utente finale.
Core classes:

- ClimateMonitor (main)
- OperatoriClimatici
- Paese
- CentroMonitoraggio
- ParametriClimatici
- GestisciOperatori
- GestisciPaesi
- GestisciCentri
- GestisciParametri

Nella pagina successiva presenteremo tutte le core classes nel dettaglio:

LIBRERIE E PACKAGE UTILIZZATI

- Java.io: questa libreria definisce i concetti base per gestire Input/Output da qualsiasi sorgente e verso qualsiasi destinazione. Questo package fornisce un modo più semplice per aggiungere funzionalità basandosi sul concetto di stream mettendo a disposizione delle classe già formate da utilizzare come ad esempio la classe File o la classe FileWriter.

- Java.util: questa libreria è molto utile, mette a disposizione circa 30 classi e una decina di interfacce che implementano le strutture dati più comuni. Per struttura dati si intende una struttura logica atta a contenere un certo numero di dati, nel quale è possibile inserire informazioni, cancellarle, modificarle, ordinarle o cercarle. Noi nel nostro programma abbiamo utilizzato un ArrayList di dati messo a disposizione appunto da java.util.ArrayList.

- Java.text: questa libreria fornisce classi e interfacce per la gestione di testo, date e numeri. Nel programma è utile per gestire la data di inserimento dei parametri climatici grazie alla classe SimpleDataFormat.

- Package oggetti: è il package che contiene tutte le mie classi che contengono gli attributi necessari per costruire un nuovo oggetto.

- Package gestionefile: è il package che contiene tutte le classi per la gestione dei file e delle strutture dati, queste classi importeranno il package oggetti.

- Package controlloeccezioni: è il package che contiene la classe che gestisce la InputMismatchException.

- Package climatemonitoring: è il package dove è contenuto il main per eseguire il programma, la classe main dovrà importare il package gestionefile.

ECCEZIONI UTILIZZATE:

- FileNotFoundException : questa è l'eccezione più utilizzata nel programma dato che abbiamo lavorato con dei file. Questa eccezione segnala che un tentativo di aprire il file indicato da un percorso specificato non è riuscito.

FileNotFoundException ovviamente è una classe che possiede i seguenti metodi:

- FileNotFoundException() : è il costruttore vuoto della classe.
- FileNotFoundException() : è il costruttore che “costruisce” un oggetto di tipo FileNotFoundException con il messaggio di errore passato come parametro.

Abbiamo utilizzato questa eccezione con la clausola throw.

- InputMismatchException : questa è l’eccezione utilizzata per gestire un input scorretto da parte dell’utente, come per esempio l’inserimento di una stringa invece che di un intero. Questa classe appartiene al package java.util ed è formata dai seguenti metodi:
  - InputMismatchException() : è il costruttore vuoto della classe.
  - InputMismatchException(String s) : è il costruttore che “costruisce” un oggetto di tipo InputMismatchException con il messaggio di errore passato come parametro.

  Abbiamo utilizzato questa classe con la clausola try-catch per gestire ogni input scorretto.

PACKAGE GESTIONEFILE:

GestisciOperatori

GestisciOperatori è una delle due classi principali di tutto il package, si occupa di collegare tutte le altre classi e di elaborare i loro dati.
Si occupa di gestire tutto gli operatori climatici con i loro relativi centri, le loro aree di interesse e i successivi parametri climatici inseriti.

Librerie importate:

- java.io
- java.util
- package oggetti

Attributi e funzionamento generale della classe:

La classe ha come attributi delle strutture dati di tipo OperatoriClimatici, CentroMonitoraggio e ParametriClimatici che mi servono per gestire gli operatori climatici in fase di registrazione e dopo il login.
Inoltre abbiamo degli oggetti di tipo Gestiscicentri, GestisciParametri e GestioneInput che permettono di utilizzare dei metodi relativi alla creazione e gestione dei centri e all'inserimento e salvataggio dei parametri climatici dopo il login di un operatore.

Metodi
La nostra classe ha dei metodi specifici per la gestione della registrazione degli operatori e del login:

- void scrivi(ArrayList<OperatoriClimatici> ar)
    - Questo metodo mi permette di scrivere sul file testuale una struttura dati, in questo caso l'ArrayList di operatori climatici.

- ArrayList<OperatoriClimatici> leggi()
    - Questo metodo è utilizzato per leggere ogni riga di un file testuale e riorganizzare i dati letti in una struttura dati, nel nostro caso l'ArrayList di operatori climatici. Questi dati vengono organizzati ed elaborati grazie alla classe StringTokenizer.

- void registrazione()
    - Questo metodo permette di registrarsi all'applicazione come operatore climatico, inserendo tutti gli attributi necessari. L'applicazione ovviamente controlla se i dati inseriti sono validi, nel caso in cui i dati non siano validi, il programma richiede l'inserimento di eventuali dati.

- void login(ArrayList<OperatoriClimatici> ar)
    - Questo metodo permette di effettuare il login da parte di un operatore già registrato, inoltre una volta effettuato il login l'operatore deve creare il centro se non ne ha ancora uno e successivamente può decidere se inserire eventuali parametri climatici per un area di interesse a scelta monitorata dal suo centro.

- void trovaOperatore()
    - Questo metodo richiama il metodo login eseguendolo e passando come parametro l'attributo ArrayList di operatori della classe.

- boolean controlloID(int userID)
    - Questo metodo controlla se l'id passato come parametro è già presente nel file, se è già presente allora il metodo ritorna false, altrimenti true.

- boolean controlloMail(String mail)
    - Questo metodo controlla se la mail passata come parametro è valida, quindi ritorna true se la mail contiene "@" altrimenti false.

- boolean checkCodiceFiscale(String codiceFiscale)
    - Questo metodo controlla se il codice fiscale passato come parametro è valido, quindi restituisce true se è di 16 caratteri, 7 numeri e 9 cifre, false altrimenti.

Eccezioni:
In questa classe abbiamo usato l'eccezione FileNotFoundException e la InputMismatchException.

GestisciPaesi

GestisciPaesi è la classe che si occupa dell'elaborazione dei dati riguardanti i Paesi e le aree di interesse, contiene i metodi per la ricerca dei Paesi per nome e il metodo della ricerca per coordinate.

Librerie importate:
- Java.io
- Java.util
- Package oggetti

Attributi e funzionamento generale della classe:
La classe ha come attributi una struttura dati di tipo Pese che utilizziamo per gestire tutti i paesi del mondo e il relativo file dove sono scritti.
Inoltre abbiamo un oggetto di tipo Scanner che permette di leggere su console un intero e un oggetto di tipo GestioneInput per le eccezioni.

Metodi
La nostra classe ha dei metodi specifici per le ricerche che potrebbero effettuare i cittadini che utilizzano l'applicazione:

- ArrayList<Paese> leggiPaesi()
    - Questo metodo è utilizzato per leggere ogni riga di un file testuale e riorganizzare i dati letti in una struttura dati, nel nostro caso l'ArrayList di Paesi. Questi dati vengono organizzati ed elaborati grazie alla classe StringTokenizer.

- ArrayList<Paese> ricercaNome(String scelta)
    - Questo metodo legge il file CoordinateMonitoraggio.txt e cerca la stringa passata come parametro in tutto il file di testo e ritorna una struttura dati contenenti i nomi dei Paesi trovati.

- void ricercaN(String scelta)
    - Questo metodo richiama il metodo precedente "ricercaNome()" e lo esegue, successivamente stampa tutti i paesi trovati.

- Paese ricercaNomeCC(String scelta1, String scelta2)
    - Questo metodo legge il file CoordinateMonitoraggio.txt e cerca un solo Paese dati i parametri scelta1 che corrisponde al nome del Paese in codice ascii e scelta2 che corrisponde al codice o country code del Paese. Il metodo ritorna l'oggetto di tipo Paese trovato. Se non viene trovato nessun paese il metodo ritorna un oggetto vuoto di tipo Paese.

- Paese ricercaCoo()
    - Questo metodo legge il file CoordinateMonitoraggio.txt e cerca un Paese in base alle coordinate inserite in input (latitudine e longitudine) in un range di -90 e 90 per la latitudine e -180 e 180 per la longitudine. Il metodo implementa un algoritmo di ricerca delle coordinate più vicine e ritorna un oggetto di tipo Paese dell'area più vicina cercata.

- void ricercaC()
    - Questo metodo richiama il metodo precedente "ricercaCoo()" e lo esegue. Successivamente stampa l'oggetto di tipo Paese trovato grazie al metodo toString.

Eccezioni:

In questa classe abbiamo usato l'eccezione FileNotFoundException e la InputMismatchException.

GestisciCentri

GestisciCentri è la classe che si occupa di registrare un centro di monitoraggio per un operatore e di salvare tutti i centri registrati in un file testuale.

Librerie utilizzate:

- Java.io
- Java.util
- Package oggetti

Attributi e funzionamento generale della classe:
La classe ha come attributi una struttura dati di tipo CentroMonitoraggio che utilizziamo per gestire i centri registrati relativi ad un operatore, un file testuale per salvare i relativi dati e un oggetto GestioneInput per gestire le eccezioni.

Metodi:

- public GestisciCentri()
    - Questo è il costruttore senza parametri della classe che contiene un controllo (condizione), infatti se il file non esiste lo crea.

- void scrivi(ArrayList<CentroMonitoraggio> ar)
    - Questo metodo permette di scrivere su file di testo un ArrayList di oggetti di tipo CentroMonitoraggio, scrivendoli secondo il toString implementato nella classe CentroMonitoraggio.

- ArrayList<CentroMonitoraggio> leggi()
    - Questo metodo permette di leggere il file testuale organizzando ed elaborando i dati letti in un ArrayList di oggetti di tipo CentroMonitoraggio.

- void registraCentroAree()
    - Questo metodo permette ad un operatore registrato di creare il suo centro di monitoraggio e quindi di registrarlo nel file CentroMonitoraggio.dati.txt.

- CentroMonitoraggio trovaCentro(String nomeC)

    - Questo metodo permette la ricerca di un centro creato da un determinato operatore climatico e ritorna il centro di monitoraggio trovato.

- void stampaCentro(String nomeC)
    - Questo metodo stampa il centro di monitoraggio trovato dal metodo precedente.

- Paese trovaArea(String nomeP, String code)
    - Questo metodo ritorna un oggetto di tipo Paese, ovvero l'area di ricerca trovata relativa ad un certo centro di monitoraggio a sua volta collegato ad un operatore.

Eccezioni:

In questa classe abbiamo usato l'eccezione FileNotFoundException e la InputMismatchException.

GestisciParametri:

GestisciParametri è la classe che gestisce la registrazione e l'elaborazione dei parametri climatici relativi ad una determinata area di interesse di un centro di monitoraggio, inoltre la classe offre un metodo per visualizzare un resoconto dei parametri climatici relativi ad un'area di interesse mostrando la media generale.

Librerie utilizzate:

- java.io
- Java.util
- Java.text
- Package oggetti

Attributi e funzionamento generale della classe:
La classe ha come attributi una struttura dati di tipo ParametriClimatici che utilizziamo per gestire i parametri climatici inseriti relativi ad un'area di interesse, un file testuale per salvare i dati delle rilevazioni effettuate da un operatore e un oggetto GestioneInput per gestire le eccezioni.

Metodi:

- public GestisciParametri()
    - Questo è il costruttore senza parametri della classe che permette di creare un nuovo file “ParametriClimatici.dati.txt” se non esiste già.

- scrivi(ArrayList<ParametriClimatici> ar)
    - Questo è il metodo che permette di scrivere una struttura dati su un file testuale.
- ArrayList<ParametriClimatici> leggi()
    - Questo è il metodo che legge il file “ParametriClimatici.dati.txt” organizzando ed elaborando i dati in un ArrayList di parametri climatici.

- void inserisciParametriClimatici(Paese paese, String nomeC)
    - Questo metodo permette l’inserimento di parametri climatici per una determinata area di interesse, richiamando infine il metodo scrivi() per salvare le nuove rilevazioni sul file.

- ArrayList<ParametriClimatici> ricercaParametriArea(String nomeP, String codice)
    - Questo metodo riceve dei valori come parametro che corrispondo al nome dell’area e al suo codice e controlla se quell’area contiene dei parametri climatici registrati, se così fosse aggiunge ad un ArrayList tutte le rilevazioni salvandole e calcolando successivamente la media.

- Void visualizzaAreaGeografica()
    - Questo metodo esegue il metodo precedente ricercaParametriArea stampando a schermo le rilevazioni effettuate su una determinata area e il relativo resoconto dei parametri climatici.

Eccezioni:

In questa classe abbiamo usato l’eccezione FileNotFoundException e la InputMismatchException.

PACKAGE OGGETTI

OperatoriClimatici

OperatoriClimatici è la classe che rappresenta l'istanza di ogni singolo operatore climatico, questa è dotata di costruttori, metodi get e set per estrapolare gli attributi privati dalla classe e poterli utilizzare in altre classi e infine di un metodo toString per poter stampare nel file tutti gli attributi di un operatore.

Librerie utilizzate:

- Nessuna libreria utilizzata in questa classe

Attributi e funzionamento generale della classe:
La classe ha come attributi tutte le caratteristiche che deve avere un operatore climatico come il nome, il codice fiscale, la mail, l'userID, la password e il centro di appartenenza. Tutti gli attributi sono di tipo stringa ad eccezione dell'userID che è una variabile di tipo intero.

Costruttore:
Serve per creare e istanziare l'oggetto di tipo OperatoriClimatici.

Metodi get e set:
Abbiamo implementato i metodi get per ogni attributo della classe dato che andranno poi gestiti nella classe gestisciOperatori e un metodo set per il centro di monitoraggio che servirà al programma per aggiungere l'attributo "centroM", una volta inserito, all'oggetto OperatoriClimatici già esistente.

Metodo toString:
Abbiamo infine il metodo toString che serve per organizzare a piacimento la scrittura su file, infatti quando il programma scrive sul file testuale andrà a scrivere ogni attributo dell'oggetto secondo il toString implementato.

Paese:

Paese è la classe che rappresenta l'istanza di ogni singolo Paese, questa è dotata di costruttori, metodi get e set per estrapolare gli attributi privati dalla classe e poterli

utilizzare in altre classi e infine di un metodo toString per poter stampare nel file tutti gli attributi di ogni Paese.

Librerie utilizzate:

- Nessuna libreria utilizzata in questa classe

Attributi e funzionamento generale della classe:
La classe ha come attributi tutte le caratteristiche che deve avere un Paese come il nome in codice ascii, il codice dello Stato, la latitudine e la longitudine. Il nome e il codice sono delle stringhe mentre le coordinate sono degli attributi double.

Costruttore:
Serve per creare e istanziare l'oggetto di tipo Paese.

Metodi get e set:
Abbiamo implementato i metodi get per ogni attributo privato della classe dato che andranno poi gestiti nella classe gestisciPaesi.

Metodo toString:
Abbiamo infine il metodo toString che serve per organizzare a piacimento la scrittura su file, infatti quando il programma scrive sul file testuale andrà a scrivere ogni attributo dell'oggetto secondo il toString implementato.

CentroMonitoraggio:

CentroMonitoraggio è la classe che rappresenta l'istanza di ogni singolo centro di monitoraggio, questa è dotata di costruttori, metodi get e convert per convertire il metodo toString dell'ArrayList in un metodo più comodo da gestire per la stampa a schermo.

Librerie utilizzate:

- Nessuna libreria utilizzata in questa classe

Attributi e funzionamento generale della classe:
La classe ha come attributi tutte le caratteristiche che deve avere un centro di monitoraggio come il nome, la via, il numero civico, il cap, il comune, la provincia e tutte le aree di interesse relative. Il nome, la via, il comune e la provincia sono delle

stringhe. Il numero civico e il cap sono interi mentre le aree di interesse sono contenuto in una struttura dati (ArrayList di Paesi).

Costruttore:
Serve per creare e istanziare l'oggetto di tipo CentroMonitoraggio.

Metodi get e set:
Abbiamo implementato i metodi get per ogni attributo privato della classe dato che andranno poi gestiti nella classe gestisciCentri.

Metodo toString:
Abbiamo infine il metodo toString che serve per organizzare a piacimento la scrittura su file, infatti quando il programma scrive sul file testuale andrà a scrivere ogni attributo dell'oggetto secondo il toString implementato.
Metodo convert:
Questo metodo trasforma il metodo toString di arrayList e ritorna una stringa senza considerare le parentesi quadre che creano problemi in fase di lettura.

ParametriClimatici:

ParametriClimatici è la classe che rappresenta l'istanza dei parametri climatici, questa è dotata di costruttori, metodi get e il metodo toString.

Librerie utilizzate:

- Nessuna libreria utilizzata in questa classe

Attributi e funzionamento generale della classe:
La classe ha come attributi tutti i parametri climatici che andranno salvati durante le rilevazioni come gli attributi intero vento, umidità, pressione, temperatura, precipitazioni, altitudine, massa e tutte le note relative ad ogni parametro climatico che hanno lunghezza massima di 256 caratteri.

Costruttore:
Serve per creare e istanziare l'oggetto di tipo ParametriClimatici.

Metodi get e set:
Abbiamo implementato i metodi get per ogni attributo privato della classe dato che andranno poi gestiti nella classe gestisciParametri.

Metodo toString:
Abbiamo infine il metodo toString che serve per organizzare a piacimento la scrittura su file, infatti quando il programma scrive sul file testuale andrà a scrivere ogni attributo dell'oggetto secondo il toString implementato.

PACKAGE CONTROLLOECCEZIONI

GestioneInput

GestioneInput è l'unica classe contenuta nel package controlloeccezioni e si occupa di gestire la InputMismatchException per tutte le altre classi.

Librerie utilizzate:

- Java.util

Metodi

- Int inserisciIntero()
    - Questo metodo gestisce l'inserimento di un intero da console, invocando l'eccezione InputMismatchException se l'input inserito non è un valore intero.

- Double inserisciDouble()
    - Questo metodo gestisce l'inserimento di un double da console, invocando l'eccezione InputMismatchException se l'input inserito non è un valore double.

Eccezioni:

L'eccezione utilizzata in questa è classe è la InputMismatchException per gestire l'inserimento dei valori corretti su console.

PACKAGE CLIMATEMONITORING

ClimateMonitor

ClimateMonitor è la classe dove è contenuto il main, serve per eseguire e provare il programma. La classe inoltre è dotata di metodi per la creazione di menù.

Librerie utilizzate:

- Java.io
- Java.util
- Package gestionefile

Attributi e funzionamento generale della classe:
La classe ClimateMonitor è la nostra Main Class è ha lo scopo di utilizzare i metodi delle altre classi per eseguire il codice. Gli attributi che possiede sono un oggetto di tipo GestisciOperatori, uno di tipo GestisciPaesi, uno di tipo GestisciParametri e due oggetti di tipo Scanner per la lettura di stringhe e interi su console.

Tutti i menù:

- Menu1: è il menù che permette di scegliere se utilizzare l'applicazione come operatore, come cittadino oppure uscire.
- Menu2: è il menù per il cittadino, permette di scegliere che tipo di ricerca usare o di tornare al menù precedente.
- Menù3: è il menù operatore che permette di registrarsi come operatore climatico, di effettuare il login se si è già registrati oppure di tornare indietro.

Ogni menù è dotato anche dell'opzione "pulisci schermo", dato che stiamo eseguendo il codice su console e senza interfaccia grafica è molto più ordinato pulire la console (ovvero cancellare tutto quello che è presente), così da evitare che l'esecuzione del programma si vada a concatenare formando una lunga sequenza di menù già utilizzati.

Metodo Main:

- Static void main(String[] args)
    - Questo è il metodo main che eseguirà tutto il programma eseguendo soltando il metodo menù1.

DISCUSSIONE COMPLESSITÀ

Come struttura dati nel nostro programma abbiamo utilizzato una classe offerta da java, la classe ArrayList offerta dal package java.util.
Questa struttura dati è particolarmente comoda e semplice da utilizzare ma vediamo nel dettaglio il tempo di esecuzione e la complessità di questa struttura dati.

Tre casi (esempio Ricerca):

- Caso migliore: O(1)
- Caso peggiore: O(n)
- Caso medio: O(n/2)

Mentre la complessità dell'inserimento in un file non dipende dal numero di inserimenti. Quindi la sua complessità è costante.

SITOGRAFIA

Abbiamo utilizzato il sito di oracle per capire come sfruttare al meglio le classi utilizzate.

- https://www.oracle.com/it/java/