Sommario

Introduzione

Metodi di libreria usati in più parti del codice

Per gli stream dei dati

Vengono utilizzate le classi:

Classe BufferedReader

Permette di leggere stream di dati da un file di testo creando un oggetto di questa clase e passando come parametro una stringa contenente il nome del file da aprire.

```
BufferedReader br = new BufferedReader(new
FileReader("OperatoriRegistrati.txt"));
```

In quasi tutti i metodi che utilizzano tale classe viene scritto un ciclo while che permette di leggere il file di testo un intera riga alla volta

```
while((line=br.readLine())!=null) {
    String[] words = line.split(" ");
    if(words[0].equals(id))
    return "nome utente già in uso,riprovare";
        }
```

Il ciclo while ha come espressione booleana una variabile di tipo string a cui viene assegnata la n^esima riga letta dal file con il metodo readLine(), l'espressione risulterà false al termine del file txt che avrà una "riga nulla".

Classe BufferedWriter

```
BufferedWriter bf =new BufferedWriter(new
FileWriter("OperatoriRegistrati.txt",true));//tenere true per
non sovrascrivere il file
```

Assieme alla creazione di un oggetto di questa classe viene utilizzato solo il metodo write (String a) che scrive nel testo le stringhe passate come parametro e il metodo nextLine () che mette il puntatore su cui la classe deve scrivere in una nuova linea di testo.

- Per leggere gli input da tastiera dell'utente

Classe Scanner

Utilizzato creando un oggetto di tipo Scanner

E associando a una variabile di tipo String il metodo della classe Scanner nextline() che legge la prossima riga inserita dall'utente.(viene sempre accompagnato prima da una stampa a schermo che indica all'utente che informazione il programma vuole ottenere)

```
Scanner sc = new Scanner(System.in);
            System.out.print("Id >");
            String id=sc.nextLine();
```

- Porzione di codice utile al funzionamento per la ricerca dei dati

Viene utilizzato la seguente riga di codice

```
String[] words = line.split(" ");
```

dove line è una variabile di tipo stringa, per creare un array di stringhe in cui ogni posizione viene creata suddividendo una striga ad ogni carattere passato come parametro, in questo caso uno spazio.

Esempio: "nuovo array esempio" → viene creato un array avente in posizione [0] la stringa "nuovo", [1] "array" , [2] "esempio"

Successivamente è possibile andare a prelevare singolarmente ogni parola della linea

```
while((line=br.readLine())!=null) {
    String[] words = line.split(" ");
    if(words[0].equals(id))
    return "nome utente già in uso,riprovare";
        }
```

Funzionamento generale per la ricerca dei dati

I dati immagazzinati in file txt vengono gestiti e associati fra loro secondo una logica posizionale, seguendo uno schema simile ad una matrice. All' interno dei file i dati vengono scritti dal programma in modo che ogni riga di testo rappresenti un informazione, e ogni colonna rappresenta i dati che la compongono.

INFORMAZIONE 1	Dato1.1	Dato1.2	Dato1.3	Dato1.4
INFORMAZIONE 2	Dato2.1	Dato2.2	Dato2.3	Dato2.4
INFORMAZIONE 3	Dato3.1	Dato3.2	Dato3.3	Dato3.4
INFORMAZIONE 4	Dato4.1	Dato4.2	Dato4.3	Dato4.4

- *Ad esempio nel file OperatoriRegistrati.txt che tiene traccia degli account registrati degli operatori, ogni linea del testo rappresenta un account diverso registrato alla piattaforma e le colonne, separate da uno spazio, saranno i dati che compongono l'account:*

  ID PASSWORD Centro di monitoraggio riferito

```
1 andrea utzeri CentroA
2 paolo risi CentroB
3 samanta c CentroB
4 user123 1234
5 antonino password CentroB
6
```

Durante ad esempio la fase di inserimento di parametri climatici in cui si controlla che l'utente sia collegato al giusto centro di monitoraggio per il quale vuole inserire nuovi parametri, con il metodo:

```
String[] words = line.split(" ");
```

Ogni linea che viene letta crea un array momentaneo in cui ogni posizione viene creata dividendo la linea ad ogni spazio e viene verificato che nella posizione [2] dell'array risultante ci sia il nome del centro di monitoraggio corretto inserito dall'utente.

Questo approccio viene utilizzato per tutte le funzioni del programma e vengono trattare più dettagliatamente nelle sezioni dedicate alle documentazioni dei singoli metodi.

Struttura generale delle classi

Main:

La classe main è la classe **ClimateMonitor**, contenente il codice per un "interfaccia". Il programma funziona grazie all'utilizzo principale di un ciclo while al cui interno è innestato uno switch che permette all'utente di inserire i comandi utili al funzionamento dell'applicazione.

Variabili:

- *Cond*

  La classe dispone di una variabile booleana *cond* inizializzata inizialmente a true, corrispondente alla variabile che permette di avviare il ciclo while, viene ridefinita a false solo nel caso in cui l'utente digiti il comando per la chiusura del programma, che corrisponde al *case 0 dello switch.*

```
case "0":
  cond=false;
  break;
```

- *id*

Una variabile di tipo String che tiene traccia se l'utente che sta utilizzando l'applicazione ha effettuato il login. La variabile è inizialmente nulla e ridefinita nel *case 3* dello switch che gestisce il comando di login (il codice viene trattato più nel dettaglio nell'apposita sezione).

La variabile viene utilizzata in espressione booleane all'interno dei case che gestiscono operazioni eseguibili solo da utenti loggati.

```
if (id != null){
```

- *sc*

Oggetto della classe Scanner utilizzato per leggere gli input da tastiera inseriti dall'utente.

Metodi:

L'applicazione funziona senza l'utilizzo di oggetti che memorizzano le informazioni, ma sono gestite interamente dal sistema di gestione dei file txt commentato nel paragrafo apposito. La classe metodi contiene esclusivamente metodi statici che definiscono le principali funzioni.

Public static String registrazione ()

Il metodo gestisce la registrazione all'interno della piattaforma.

Il metodo restituisce una stringa corrispondente solo al return di una stringa che comunica a schermo se c'è stato un errore nella registrazione o è avvenuta con successo.

Dopo che l'utente inserisce id e password viene letto il file OperatoriRegistrati e si effettua un controllo che nella posizione [0] di ogni linea del file non ci sia un elemento uguale all id inserito dall'utente, questo permette quindi di verificare che l'id utente non sia già stato utilizzato.

```
while((line=br.readLine())!=null) {
    String[] words = line.split(" ");
    if(words[0].equals(id))
    return "nome utente già in uso,riprovare";
        }
```

Se la procedura va a buon fine vengono scritti nel file id e password distaccati da uno spazio " ", per il corretto funzionamento del codice.

Public static String login()

Il metodo dopo aver letto id e password inseriti, verifica linea per linea che il file OperatoriRegistrati contenga una stringa uguale a

id+" "+password

Se l'utente è correttamente registrato alla piattaforma il contenuto di quella stringa nel file è garantito dal metodo registrazione ().

```
while((line=br.readLine())!=null) {
  if(line.contains(id+" "+password)) {
    System.out.println("Login effettuato con successo");
    return id;}
}
```

Il metodo restituisce una Stringa corrispondente all' id usato per il login per permettere di dare lo stesso valore alla variabile id nel main, che quindi consentirà di avere l'espressione booleana vera per l'utilizzo di tutti i case dello switch. Viene memorizzato quel valore non solo se si sta operando a login effettuato o no ma per tenere traccia quale account sta operando con il programma, che servirà per i metodi commentati a seguire.

- ## Public static String registraCentroAree ()

Con questo metodo, attivato dal case 5 dello switch del main, il programma andrà a registrare nel file di testo dedicato CentroMonitoraggio, i nomi dei centri di monitoraggio e le aree di interesse che monitora ogni centro. Per farlo vengono sempre usate le stesse linee di codice che creano gli oggetti per gli stream e che chiedono all'utente di inserire il nome del centro di monitoraggio e le aree da associargli che verranno salvate nelle variabili *nome* e *localita*.

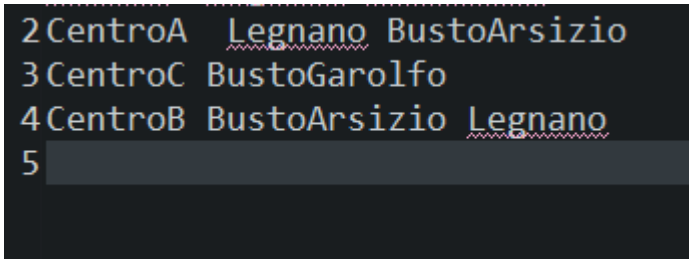
```
while(cond) {
    System.out.print("Area di interesse >");
    localitaTmp=sc.nextLine();
    br = new BufferedReader(new FileReader("CoordinateMonitoraggio.txt"));
    while((line=br.readLine())!=null) {
          String[] words = line.split(" ");
          if((words[0].toLowerCase()).equals((localitaTmp.toLowerCase()).replaceAll("\\s",""))){
            areaRegistrata=false;
            break;
          }

        }
    if(areaRegistrata) {
        System.out.print("L'area inserita non è registrata nelle aree esistenti");
        return;
    }

    localita=(localita+localitaTmp.replaceAll("\\s","")+" ").toLowerCase();
    System.out.print("Premere invio per inserire un'altra area o digitare 0 per terminare");
    if(sc.nextLine().equals("0"))
        cond=false;
}
```

Viene iterato un while che permette di far scegliere aggiunge più aree all'utente, innestato ad esso un ciclo while per verificare che la località data in input dall'utente sia presente nel file delle coordinate geografiche. Viene creata una variabile localita e localitaTmp, la prima per tenere salvato in una variabile tutte le aree che l'utente sta inserendo, la

seconda che memorizza solo ad una ad una ogni area che l'utente inserisce ad ogni iterazione del ciclo while esterno.

Lo spazio fra una località e l'altra permette di andare a separare in colonne diverse ogni area di interesse, permettendo quindi di ricercare un certo nome associato ad un certo centro semplicemente scandendo l'array, e applicato il metodo replaceAll per sostituire eliminare tutti gli spazi (l'utente deve avere la possibilità di inserire ad esempio Busto Arsizio, che verrà però immagazzinato in un singolo slot dell'array e quindi trasformato in BustoArsizio). Viene utilizzato inoltre un metodo toLowerCase per evitare problemi di case sensitive fra i dati inseriti dall'utente e i dati memorizzati nei file :

**Esempio*: *Quando un operatore vorrà inserire dei parametri climatici per una certa zona di interesse, verrà prima effettuato un controllo che andrà a scandire ogni posizione dell'array creato dalla linea corrispondente al centro di monitoraggio a cui appartiene l'utente, e verificato che quella zona sia di competenza di quel centro.*



In questo caso un utente del Centro A che vuole inserire parametri riferiti a Legnano, andrà verificato che ci sia una linea avente in posizione [0]=CentroA e in qualsiasi altra posizione [n]=Legnano.

Prima di andare a scrivere nel file i dati raccolti, viene effettuato un ultimo controllo che verifica che non esista già un centro già registrato avente lo stesso nome (quindi che venga erroneamente creato un duplicato dello stesso). Per farlo viene creato l'array linea per linea e verificato che in nessuna in posizione [0] ci sia già il nome.

Il metodo dopo l'ultimo controllo scrive nel file il nome del centro con la rimozione degli spazi, stesso motivo appena descritto e concatenato alla località.

➢ Public static void associaCentro(String id)

Il metodo associa un utente già registrato ad un centro di monitoraggio anch'esso già registrato nei file.

Viene preso come parametro una stringa corrispondente al valore che ha in quel momento la variabile id del main (vedasi la documentazione della classe Main). Nel caso la variabile sia null e che quindi non si sia accora effettuato il login, non sarà possibile utilizzare il case che utilizza questo metodo.

```
case "6":
  if(id != null){
    Metodi.associaCentro(id);
    break;
  }
  else{
    System.out.println("Non hai effettuato il login");
    break;
  }
```

Il metodo nelle prime linee di codice gestisce come negli altri metodi la memorizzazione in una variabile dell'input da tastiera dato dall'utente, corrispondente al nome del centro di monitoraggio che si vuole associare al proprio account. Viene quindi eseguito un primo controllo nel primo ciclo while per verificare che il centro inserito esista. Vengono utilizzate una variabile booleana e una variabile stringa, la prima per leggere riga per riga il file CentroMonitoraggio e verificare con un if innestato se contiene il nome inserito. La seconda variabile è un boolean che gestisce la condizione di entrata nell' if della riga 201. Se la variabile rimane true, entrando nell'if verrà interrotto il metodo con un return e stampato all'utente che il centro non è registrato.

```
String controllo;
boolean condizioneuscita = true;

while((controllo=br.readLine())!=null) {
  if(controllo.contains(nomeCentro))
    condizioneuscita = false;
    break;
}

if(condizioneuscita){
  System.out.println("Questo centro non e' registrato.");
  return;
}
```

Il metodo crea successivamente una linkedlist che prende i valori corrisponendi al file OperatoriRegistrati, disponendo ogni linea di testo in ogni posizione della lista.

Due array di tipo stringa serviranno per effettuare un controllo se un utente è già associato ad un centro di monitoraggio, ed a salvare nella stessa linea di testo il nome del centro in cui vuole associarsi nel caso la prima ipotesi non fosse vera.

```
  String[] parole = new String[3];
  String[] toCopy = nuovotesto.get(j).split(" ");  //

  for(int k= 0; k < toCopy.length; k++){
    parole[k] = toCopy[k];
  }

  if(parole[0].equals(id)) {

    if(parole[2] != null){
      System.out.println("Sei già associato ad un centro di monitoraggio");
      return;
    }else{
      nuovotesto.set(j, nuovotesto.get(j) + " " + nomeCentro.replaceAll("\\s",""));
      break;
    }

  }
}
```

Viene poi riscritto l'intero file operatori registrati con l'eventuale modifica che all'utente che si è associato ad un nuovo centro.

```
  BufferedWriter scrittore = new BufferedWriter(new FileWriter("OperatoriRegistrati.txt"));

  for(String f: nuovotesto){
    scrittore.write(f);
    scrittore.newLine();
  }
  scrittore.close();
  System.out.println("Associazione effettuata con successo");
```

- ## Public static void inserisciParametriClimatici(String id)

Il metodo permette ad un utente che ha effettuato il login di inserire dei parametri climatici per una certa zona, memorizzati nel file ParametriClimatici.

Viene preso come parametro una stringa corrispondente al valore che ha in quel momento la variabile id del main (vedasi la documentazione della calsse Main). Nel caso la variabile sia null e che quindi non si sia accora effettuato il login, non sarà possibile utilizzare il case che utilizza questo metodo.

```
case "7":
    if(id !=null) {
        Metodi.inserisciParametriClimatici(id);
        break;
    }
    else {
        System.out.print("Non hai effettuato il login");
        break;
    }
```

Il parametro stringa viene utilizzato all'inizio del codice per effettuare un controllo che verifica che il centro di monitoraggio per la quale si intende inserire i parametri sia associato all'account.

```
boolean cond=true;
String line;
String nomeCentro=null;
while((line=br.readLine())!=null) {
    String[] words = line.split(" ");
    if(words[0].equals(id)) {                    //Controllo associazione al centro
        if(words.length>2) {
            nomeCentro=words[2];
            cond=false;
            break;
            }
        }
    }
if(cond) {
    System.out.println("Non sei registrato ad alcun centro di monitoraggio");
    return;
}
```

Viene quindi verificata se nella riga con in posizione [0] id ha in posizione [2] un valore non nullo, viene quindi cambiata la condizione per l'if di uscita dal metodo e salvato il centro nella variabile **nomeCentro**, che serve per effettuare altri controlli commentati di seguito.

Il metodo chiede all'utente per quale area di interesse intende inserire i parametri climatici e viene gestito un ulteriore controllo che verifica che nel file CentroMonitoraggio ci sia in corrispondenza di una linea contenente **nomeCentro** l'area inserita.

```
BufferedReader lettore = new BufferedReader(new FileReader("CentroMonitoraggio.txt"));
    boolean uscita = true;

    while((line=lettore.readLine())!=null){
      if(line.contains(nomeCentro)){
          line=line.toLowerCase();
          if(line.contains(area.toLowerCase().replaceAll("\\s","")))
          uscita = false;
        break;
      }
    }

    lettore.close();

    if(uscita){
      System.out.println("L'area che hai inserito non e' di competenza del tuo centro di monitoraggio");
      return;
```

Importante specificare che alla variabile area,utilizzata per controllare che l'area inserita sia di competenza del centro di monitoraggio dell'utente, a cui viene applicato il metodo replaceAll, poiché all'interno del file CentroMonitoraggio i nomi delle zone di interesse occupano una singola posizione dell'array e quindi prive di spazio(si rimanda alla sezione sul metodo registraCentroAree per ulteriori dettagli.

```
Scanner sc = new Scanner(System.in);
System.out.println("Inserire l'area di interesse");
String area=sc.nextLine();
String tmch=area.replaceAll("\\n","-");
String areaVisualizzata=tmch;
/* questa variabile permette di avere area visualizzata con i giusti spazi quando l'utente guarderà
  i parametri, con la variabile area invece si fa la ricerca all'internodei file di sitema, ad esempio
  inserendo Busto Arsizio, useremo area per cercare bustoarsizio all'interno dei file di sistema
  mentre una volta verificato che sia effettivamente un area di competenza del centro andremo a salvare nel file
  dei parametri climatici la sua versione non strecchata*/


BufferedReader lettore = new BufferedReader(new FileReader("CentroMonitoraggio.txt"));
    boolean uscita = true;

    while((line=lettore.readLine())!=null){
```

le prossime linee di codice faranno in modo di raccogliere tutte le variabili contenente i valori dei parametri climatici inseriti dall'utente

```
boolean tmp=true;
while(tmp) {
    System.out.println("Inserire punteggio(1/5) relativo alla velocità del vento (km/h), suddivisa in fasce");
    vento=sc.nextLine();
    if(vento.equals("1")||vento.equals("2")||vento.equals("3")||vento.equals("4")||vento.equals("5")){
        tmp=false;
    }
    else System.out.println("il valore deve essere compreso fra 1 e 5");
}
```

Il ciclo while obbliga l'utente a inserire necessariamente un valore compreso da 1 a 5, verrà quindi riproposto di reinserire il valore in caso venga inserito qualsiasi carattere non corrispondente a quei valori.

Il procedimento è identico per ogni variabile relativo ad un parametro climatico , effettuando quindi il medesimo while, che per ridondanza non verranno riportati tutti nel manuale tecnico.

Nota: la variabile tmp viene ridefinita a true prima di ogni ciclo while altrimenti una volta usciti da un ciclo while non si potrà entrare in quello seguente.

Il metodo prevede di poter inserire delle note di massimo 256 caratteri dopo i parametri quindi viene inizializzata una variabile che conterrà il valore letto a schermo corrispondente all'eventuale nota messa dall'utente, e un ciclo while che garantisce il limite di caratteri

```
String note="Nessuna nota inserita";
boolean comando=true;
while(comando) {
    System.out.println("Premere 1 per inserire una nota(max 256 caratteri) oppure 0 per non inserire");
    String scelta=sc.nextLine();
    switch(scelta) {

        case "1":

            while(contacaratteri) {
                System.out.println("Inserire le note:");
                note=sc.nextLine();
                if(note.length()>256) {
                    System.out.println("Limite di 256 caratteri superato, rinserire la nota");
                }
                else contacaratteri=false;
            }
            comando=false;
            break;

        case "0":
            comando=false;
            break;
        default:System.out.println("Comando inserito non valido");

    }
}
```

Tutte le variabili(nome centro,area,parametri e note) vengono trascritte nel file di testo.

```
bf.write(nomeCentro+" "+areaVisualizzata+" "+escamotage+"
"+vento+" "+umidita+" "+pressione+" "+temperatura+"
"+pioggia+" "+altitudine+" "+massa+"
"+note.replaceAll("\\s","-"));
        bf.newLine();
        bf.close();
```

Nota: alla variabile note vengono sostituiti gli spazi con il carattere "-" così da permettere di essere organizzati in un unico spazio dell'array e ristampati con più facilità nel metodo visualizzaAreageografica() la quale eseguirà la procedura inversa quindi risostituendo i trattini con gli spazi.

(questo viene fatto perché come spiegato nel paragrafo sul funzionamento generale del codice, gli spazi sono il carattere che separa una linea di testo creandone l'array corrispondente.

- ## Public static void visualizzaAreaGeografica()

Il metodo deve stampare a schermo tutte le informazioni presenti nel file ParametriClimatici relativi ad una determinata area selezionata dall'utente.

Viene inizializzata una variabile booleana a true in modo tale che se scandendo il file di testo nel ciclo while, non si dovesse trovare alcuna linea contenente l'area selezionata, la variabile rimanga true e entri nell'if di uscita dal metodo.

Nota: le informazioni all'interno del file parametriclimatici sono sistemate in maniera del tipo [nome_centro] [area] [data_di_rivelazione] [parametro_1] […] [parametro_7] [note]

(più dettagli nella sezione dedicata al metodo inserisciParametriClimatici())

Il funzionamento del ciclo while permette che ad ogni sua iterazione alla linea selezionata venga applicato il metodo contains(area) come espressione booleana del if, in caso di risultato positivo il metodo ridefinisce la variabile d'uscita a false, divide in un array la linea di testo per permettere la stampa di ogni dato accompagnato ad una stringa che ne spieghi il significato all'utente.

```
while((line = br.readLine()) != null){
    if((line.toLowerCase()).contains(area.toLowerCase())){
    uscita = false;
    words = line.split(" ");
    System.out.println("Parametri registrati da: " + words[0]);
    System.out.println("Data di registrazione dei parametri " + words[2]);
    System.out.println("Punteggio assegnato al vento: " + words[3]);
    System.out.println("Punteggio assegnato all'umidità: " + words[4]);
    System.out.println("Punteggio assegnato alla pressione atmosferica: " + words[5]);
    System.out.println("Punteggio assegnato alla temperatura: " + words[6]);
    System.out.println("Punteggio assegnato alle precipitazioni: " + words[7]);
    System.out.println("Punteggio assegnato all'altitudine dei ghiacciai: " + words[8]);
    System.out.println("Punteggio assegnato alle massa dei ghiacciai: " + words[9]);
    System.out.println("Note: " + words[10].replaceAll("-"," ") + '\n');
  }
}

br.close();

if(uscita)
  System.out.println("Non sono stati registrati parametri per l'area che hai inserito.");
```

Nota2: Le note vengono stampate sostituendo il carattere "-" con uno spazio, consultare il metodo relativo al metodo sull'inserimento dei parametri climatici per maggiori dettagli.

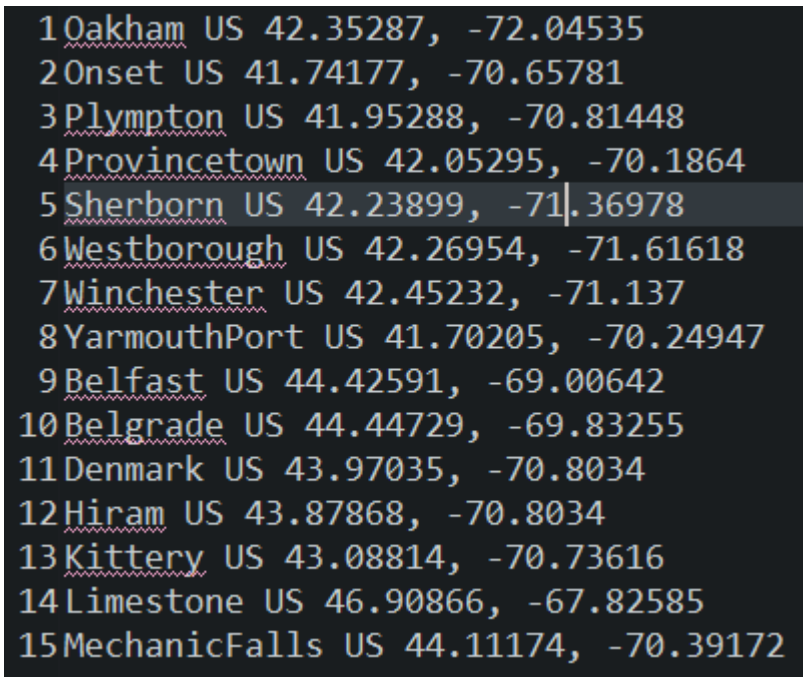
➢ Public stati void cercaAreaGeografica()

Il metodo deve permettere che l'utente possa selezionare se ricercare all'interno del file CoordinateMonitoraggio utilizzando il nome dell'area da ricercare o attraverso coordinate geometriche.

Viene quindi gestito uno switch che avente i due case in base alla modalità scelta:

case "1": Si scandisce con un ciclo while in file di testo ricercando in posizione [0] dell' array risultante di ogni linea, se il valore è uguale al valore in input dato dall'utente, in caso di espressione vera si stampa la linea di testo. Alla variabile che contiene il valore inserito dall'utente e il valore preso all'interno del file vengono applicati il metodo toLowerCase() (se ad esempio l'utente inserisce Varese, permette di trovare una corrispondenza all'interno del file per la parola Varese).

Nota: Si effettua la ricerca nella poszione 0 poiché all'inteno del file i dati sono organizzati nelle seguenti posizioni di un eventuale array risultante



Dove oakham corrisponde alla posizione [0], uso in posizione [1]…

Case "2":

Viene effettuata una ricerca nella medesima maniera del case 1, ma l'espressione booleana per entrare nell'if è dato se la linea presa in esame contiene sia il valore numerico corrispondente alla latitudine che alla longitudine(in questo caso quindi non viene creato un array ma sfruttato il metodo contains(lat)&&contains(lon).

```
switch(scelta){

case "1":
  System.out.print("Località: >");
  String localita=sc.nextLine();


  while((line=br.readLine())!=null) {
    String[] words = line.split(" ");
    if((words[0].toLowerCase()).equals((localita.toLowerCase()).replaceAll("\\s",""))){
      System.out.println(line);
      return;
    }

  }

  System.out.print("Nessun risultato trovato");
  break;

case "2":
  System.out.println("Latitudine: >");
  String lat = sc.nextLine();
  System.out.println("Longitudine: >");
  String lon = sc.nextLine();

  while((line=br.readLine())!=null) {
    if(line.contains(lat)&&line.contains(lon)){
      System.out.println(line);
      return;
    }

  }
```

E’ presente anche un deafult che garantisce di ripetere la scelta da parte dell ‘utente in caso di comando non valido.