**import** java.util.ArrayList;

/\*\*

\* Classe pubblica con metodi statici di utilità generale per:

\* la ricerca delle parole "EXIT" o "ESCI" o delle lettere "EE", scritte in maiuscolo o minuscolo, in una stringa di testo dove sono presenti degli spazi,

\* la ricerca delle parole "RESTART" o "RIAVVIA" o delle lettere "RR", scritte in maiuscolo o minuscolo, in una stringa di testo dove sono presenti degli spazi,

\* la ricerca della parola "cancella\_Campo", in una stringa di testo dove sono presenti degli spazi,

\* la stampa dei messaggi a video utilizzando semplicemente il metodo "stampa()" al posto del metodo statico della classe System "System.out.print()",

\* l'eliminazione di tutti gli spazi in una stringa di testo,

\* l'eliminazione degli spazi ridondanti in una stringa di testo (gli spazi iniziali, quelli finali e quelli doppi tra le parole),

\* la ricerca di un numero intero in una stringa di testo dove sono presenti degli spazi (questo metodo può essere utilizzato in alternativa alla gestione di alcune eccezzioni della classe "Integer" nella trasformazione di una stringa in un numero),

\* la ricerca di un numero intero o decimale in una stringa di testo dove sono presenti degli spazi (questo metodo può essere utilizzato in alternativa alla gestione di alcune eccezioni della classe "Double" nella trasformazione di una stringa in un numero),

\* la verifica di una stringa di testo del campo "cognome" / "nome" / "nome istituto scolastico" / "località", riferito a una persona / una scuola / un paese (o città)...,

\* la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "cognome" / "nome" / "nome istituto scolastico" / "località", riferito a una persona / una scuola / un paese (o città)...,

\* la verifica di una stringa di testo del campo "classe" di una scuola,

\* la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "classe" di una scuola,

\* la verifica di una stringa di testo del campo "codiceAlfanumerico" di una persona o articolo,

\* la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "codice alfanumerico" di una persona o articolo,

\* la verifica di una stringa di testo del campo "indirizzo" di una persona o abitazione,

\* la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "indirizzo" di una persona o abitazione,

\* la verifica di una stringa di testo del campo "data" nel formato 'GG.MM.AAAA',

\* la restituzione degli eventuali caratteri con consentiti presenti in una stringa di testo del campo "data" nel formato 'GG.MM.AAAA',

\* la verifica di una stringa di testo del campo "provincia" in sigla (2 lettere tranne Roma),

\* la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "provincia" in sigla (2 lettere maiuscole tranne Roma),

\* la verifica di una stringa di testo di un campo numerico, come, a esempio, il numero di telefono di una persona,

\* la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo di un campo numerico, come, a esempio, il numero di telefono di una persona,

\* la verifica di una stringa di testo del campo "indirizzo e-mail" di una persona o azienda,

\* la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "indirizzo e-mail" di una persona o azienda,

\* la verifica di una stringa di testo del campo "testo generico" di una nota o una descrizione,

\* la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "testo generico" di una nota o una descrizione,

\* ordinare una serie di stringhe in un array (o in un array dinamico) dalla A alla Z,

\* ordinare una serie di stringhe in un array (o in un array dinamico) dalla Z alla A,

\* **@author** Abramo Gerardo Giordano

\* **@version** 1.1.4 (28 Aprile 2019)

\*/

**public** **class** **AGG** {

/\*\*

\* Metodo pubblico statico per la ricerca delle parole "EXIT" o "ESCI" o delle lettere "EE", scritte in maiuscolo o minuscolo, in una stringa di testo dopo sono presenti degli spazi.

\* **@param** stringaInput (stringa di testo).

\* **@return** condizioneUscitaProgramma (variabile booleana).

\*/

**public** **static** **boolean** esci(**String** stringaInput) {

**int** j = 0;

**boolean** condizioneUscitaProgramma = **false**;

**boolean** condizioneCicloDoWhile = **false**;

**boolean** trovatoPrimoCarattereDiversoDaSpaziatura = **false**;

**boolean** trovatoSpaziaturaDopoCarattere = **false**;

**do** {

condizioneCicloDoWhile = **true**;

// Questa condizione si verifica quando l'utente inserisce una stringa vuota, e termina subito l'esecuzione del metodo restituendo per la variabile booleana "condizioneUscita" un valore pari a FALSE.

**if** (stringaInput.length()==0)

condizioneCicloDoWhile = **false**;

**else**

**for** (**int** i = 0; i< stringaInput.length(); i++) {

// Queste condizioni, eliminano gli eventuali spazi che (nella stringa) precedono la parola ricercata. Appena viene trovato un carattere diverso da quello di spaziatura, viene assegnata alla variabile "trovatoPrimoCarattereDiversoDaspaziatura" un valore booleano pari a TRUE, causando il salto di queste istruzioni di selezione nelle successive iterazioni del ciclo FOR.

**if** ((trovatoPrimoCarattereDiversoDaSpaziatura==**false**)&&(stringaInput.charAt(i)!=' ')) {

trovatoPrimoCarattereDiversoDaSpaziatura = **true**;

**break**;

}

**if** ((trovatoPrimoCarattereDiversoDaSpaziatura==**false**)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

**break**;

}

// Questa condizione si verifica quando (dopo aver eliminato gli eventuali caratteri di spaziatura che precedono la parola da ricercare) compare una sequenza di più caratteri consecutivi che formano una parola. Vengono contate le lettere della parola fino a quando non viene trovato (eventualmente) un altro carattere di spaziatura.

**if** ((trovatoPrimoCarattereDiversoDaSpaziatura==**true** )&&(trovatoSpaziaturaDopoCarattere==**false**)&&(stringaInput.charAt(i)!=' '))

j++;

**else**

trovatoSpaziaturaDopoCarattere = **true**;

// Queste condizioni (dopo aver spostato la prima parola nella prima parte della stringa) verificano che non vi siano caratteri diversi da quello di spaziatura (che vengolo eliminati) dopo la parola.

**if** ((trovatoSpaziaturaDopoCarattere==**true**)&&(i==j)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

**break**;

} **else** **if** ((trovatoSpaziaturaDopoCarattere==**true**)&&(i==j)&&(stringaInput.charAt(i)!=' ')) {

condizioneCicloDoWhile = **false**;

**break**;

}

// Questa condizione si verifica quando (dopo aver isolato nella stringa solo la prima parola o lettera trovata e aver eliminato gli eventuali spazi) l'indice "i" del ciclo FOR giunge all'ultima lettera della parola (posizione dell'ultima lettera -1). Il metodo verifica che la parola o la lettera rimanente nella stringa corrisponda alle parole ricercate "EXIT" o "E" o "ESCI" e in caso affermativo assegna alla variabile booleana "condizioneUscitaProgramma" un valore pari a TRUE.

**if** (i==stringaInput.length()-1) {

stringaInput = stringaInput.toUpperCase();

**if** ((stringaInput.equalsIgnoreCase("EE"))||stringaInput.equalsIgnoreCase("EXIT")||(stringaInput.equalsIgnoreCase("ESCI"))) {

condizioneUscitaProgramma = **true**;

condizioneCicloDoWhile = **false**;

} **else**

condizioneCicloDoWhile = **false**;

}

}

} **while** (condizioneCicloDoWhile==**true**);

**return** condizioneUscitaProgramma;

}

/\*\*

\* Metodo pubblico statico per la ricerca delle parole "RESTART" o "RIAVVIA" o delle lettere "RR", scritte in maiuscolo o minuscolo, in una stringa di testo dopo sono presenti degli spazi.

\* **@param** stringaInput (stringa di testo).

\* **@return** condizioneRiavvioProgramma (variabile booleana).

\*/

**public** **static** **boolean** riavvia(**String** stringaInput) {

**int** j = 0;

**boolean** condizioneRiavvioProgramma = **false**;

**boolean** condizioneCicloDoWhile = **false**;

**boolean** trovatoPrimoCarattereDiversoDaSpaziatura = **false**;

**boolean** trovatoSpaziaturaDopoCarattere = **false**;

**do** {

condizioneCicloDoWhile = **true**;

// Questa condizione si verifica quando l'utente inserisce una stringa vuota, e termina subito l'esecuzione del metodo restituendo per la variabile booleana "condizioneUscita" un valore pari a FALSE.

**if** (stringaInput.length()==0)

condizioneCicloDoWhile = **false**;

**else**

**for** (**int** i = 0; i< stringaInput.length(); i++) {

// Queste condizioni, eliminano gli eventuali spazi che (nella stringa) precedono la parola ricercata. Appena viene trovato un carattere diverso da quello di spaziatura, viene assegnata alla variabile "trovatoPrimoCarattereDiversoDaspaziatura" un valore booleano pari a TRUE, causando il salto di queste istruzioni di selezione nelle successive iterazioni del ciclo FOR.

**if** ((trovatoPrimoCarattereDiversoDaSpaziatura==**false**)&&(stringaInput.charAt(i)!=' ')) {

trovatoPrimoCarattereDiversoDaSpaziatura = **true**;

**break**;

}

**if** ((trovatoPrimoCarattereDiversoDaSpaziatura==**false**)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

**break**;

}

// Questa condizione si verifica quando (dopo aver eliminato gli eventuali caratteri di spaziatura che precedono la parola da ricercare) compare una sequenza di più caratteri consecutivi che formano una parola. Vengono contate le lettere della parola fino a quando non viene trovato (eventualmente) un altro carattere di spaziatura.

**if** ((trovatoPrimoCarattereDiversoDaSpaziatura==**true** )&&(trovatoSpaziaturaDopoCarattere==**false**)&&(stringaInput.charAt(i)!=' '))

j++;

**else**

trovatoSpaziaturaDopoCarattere = **true**;

// Queste condizioni (dopo aver spostato la prima parola nella prima parte della stringa) verificano che non vi siano caratteri diversi da quello di spaziatura (che vengolo eliminati) dopo la parola.

**if** ((trovatoSpaziaturaDopoCarattere==**true**)&&(i==j)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

**break**;

} **else** **if** ((trovatoSpaziaturaDopoCarattere==**true**)&&(i==j)&&(stringaInput.charAt(i)!=' ')) {

condizioneCicloDoWhile = **false**;

**break**;

}

// Questa condizione si verifica quando (dopo aver isolato nella stringa solo la prima parola o lettera trovata e aver eliminato gli eventuali spazi) l'indice "i" del ciclo FOR giunge all'ultima lettera della parola (posizione dell'ultima lettera -1). Il metodo verifica che la parola o la lettera rimanente nella stringa corrisponda alle parole ricercate "RESTART" o "R" o "RIAVVIA" e in caso affermativo assegna alla variabile booleana "condizioneRiavvioProgramma" un valore pari a TRUE.

**if** (i==stringaInput.length()-1) {

stringaInput = stringaInput.toUpperCase();

**if** ((stringaInput.equalsIgnoreCase("RR"))||stringaInput.equalsIgnoreCase("RESTART")||(stringaInput.equalsIgnoreCase("RIAVVIA"))) {

condizioneRiavvioProgramma = **true**;

condizioneCicloDoWhile = **false**;

} **else**

condizioneCicloDoWhile = **false**;

}

}

} **while** (condizioneCicloDoWhile==**true**);

**return** condizioneRiavvioProgramma;

}

/\*\*

\* Metodo pubblico statico per la ricerca della parola "cancella\_Campo", in una stringa di testo dopo sono presenti degli spazi.

\* **@param** stringaInput (stringa di testo).

\* **@return** condizioneCancellaCampo (variabile booleana).

\*/

**public** **static** **boolean** cancella\_Campo(**String** stringaInput) {

**int** j = 0;

**boolean** condizioneCancellaCampo = **false**;

**boolean** condizioneCicloDoWhile = **false**;

**boolean** trovatoPrimoCarattereDiversoDaSpaziatura = **false**;

**boolean** trovatoSpaziaturaDopoCarattere = **false**;

**do** {

condizioneCicloDoWhile = **true**;

// Questa condizione si verifica quando l'utente inserisce una stringa vuota, e termina subito l'esecuzione del metodo restituendo per la variabile booleana "condizioneUscita" un valore pari a FALSE.

**if** (stringaInput.length()==0)

condizioneCicloDoWhile = **false**;

**else**

**for** (**int** i = 0; i< stringaInput.length(); i++) {

// Queste condizioni, eliminano gli eventuali spazi che (nella stringa) precedono la parola ricercata. Appena viene trovato un carattere diverso da quello di spaziatura, viene assegnata alla variabile "trovatoPrimoCarattereDiversoDaspaziatura" un valore booleano pari a TRUE, causando il salto di queste istruzioni di selezione nelle successive iterazioni del ciclo FOR.

**if** ((trovatoPrimoCarattereDiversoDaSpaziatura==**false**)&&(stringaInput.charAt(i)!=' ')) {

trovatoPrimoCarattereDiversoDaSpaziatura = **true**;

**break**;

}

**if** ((trovatoPrimoCarattereDiversoDaSpaziatura==**false**)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

**break**;

}

// Questa condizione si verifica quando (dopo aver eliminato gli eventuali caratteri di spaziatura che precedono la parola da ricercare) compare una sequenza di più caratteri consecutivi che formano una parola. Vengono contate le lettere della parola fino a quando non viene trovato (eventualmente) un altro carattere di spaziatura.

**if** ((trovatoPrimoCarattereDiversoDaSpaziatura==**true** )&&(trovatoSpaziaturaDopoCarattere==**false**)&&(stringaInput.charAt(i)!=' '))

j++;

**else**

trovatoSpaziaturaDopoCarattere = **true**;

// Queste condizioni (dopo aver spostato la prima parola nella prima parte della stringa) verificano che non vi siano caratteri diversi da quello di spaziatura (che vengolo eliminati) dopo la parola.

**if** ((trovatoSpaziaturaDopoCarattere==**true**)&&(i==j)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

**break**;

} **else** **if** ((trovatoSpaziaturaDopoCarattere==**true**)&&(i==j)&&(stringaInput.charAt(i)!=' ')) {

condizioneCicloDoWhile = **false**;

**break**;

}

// Questa condizione si verifica quando (dopo aver isolato nella stringa solo la prima parola o lettera trovata e aver eliminato gli eventuali spazi) l'indice "i" del ciclo FOR giunge all'ultima lettera della parola (posizione dell'ultima lettera -1). Il metodo verifica che la parola o la lettera rimanente nella stringa corrisponda alla parola ricercata "cancella\_Campo" e in caso affermativo assegna alla variabile booleana "condizioneCancellaCampo" un valore pari a TRUE.

**if** (i==stringaInput.length()-1) {

**if** (stringaInput.equals("cancella\_Campo")) {

condizioneCancellaCampo = **true**;

condizioneCicloDoWhile = **false**;

} **else**

condizioneCicloDoWhile = **false**;

}

}

} **while** (condizioneCicloDoWhile==**true**);

**return** condizioneCancellaCampo;

}

/\*\*

\* Metodo pubblico statico per la stampa a video.

\* **@param** stringaInput (stringa di testo).

\* **@return** Nessun valore.

\*/

**public** **static** **void** stampa(**String** stringaInput) {

**System**.***out***.print(stringaInput);

}

/\*\*

\* Metodo pubblico statico per l'eliminazione degli spazi in una stringa di testo.

\* **@param** stringaInput (stringa di testo).

\* **@return** stringaInput (stringa di testo).

\*/

**public** **static** **String** eliminaSpazi(**String** stringaInput) {

**boolean** trovatoCarattereDiSpaziatura = **false**;

**do** {

trovatoCarattereDiSpaziatura = **false**;

**for** (**int** i = 0; i<stringaInput.length(); i++) {

**if** (stringaInput.charAt(i)==' ') {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

trovatoCarattereDiSpaziatura = **true**;

**break**;

}

}

} **while** (trovatoCarattereDiSpaziatura==**true**);

**return** stringaInput;

}

/\*\*

\* Metodo pubblico statico per l'eliminazione degli spazi ridondanti in una stringa di testo. Elimina gli spazi doppi, e quelli all'inizio e alla fine di una stringa di testo dove è presente almeno un carattere diverso da quello di spaziatura.

\* **@param** stringaInput (stringa di testo).

\* **@return** stringaInput (stringa di testo).

\*/

**public** **static** **String** eliminaSpaziInEccesso(**String** stringaInput) {

**boolean** trovatoCarattereDiSpaziatura = **false**;

**boolean** trovatoCarattereDiversoDaSpazio = **false**;

**do** {

trovatoCarattereDiSpaziatura = **false**;

**for** (**int** i = 0; i<stringaInput.length()-1; i++)

**if** ((stringaInput.charAt(i)==' ')&&(stringaInput.charAt(i+1)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1, stringaInput.length());

trovatoCarattereDiSpaziatura = **true**;

**break**;

}

} **while** (trovatoCarattereDiSpaziatura==**true**);

**do** {

trovatoCarattereDiSpaziatura = **false**;

**for** (**int** i = 0; i<stringaInput.length(); i++) {

// Questa condizione si verifica quando vengono trovati dei caratteri di spaziatura all'inizio di una stringa di testo e li elimina.

**if** ((trovatoCarattereDiversoDaSpazio==**false**)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1, stringaInput.length());

trovatoCarattereDiSpaziatura = **true**;

**break**;

// Questa condizione si verifica quando viene trovato per la prima volta un carattere diverso da quello di spaziatura.

} **else** **if** ((trovatoCarattereDiversoDaSpazio==**false**)&&(stringaInput.charAt(0)!=' ')) {

trovatoCarattereDiversoDaSpazio = **true**;

// Questa condizione si verifica quando viene trovato un carattere di spaziatura alla fine di una stringa di testo e lo elimina.

} **else** **if** (stringaInput.charAt(stringaInput.length()-1)==' ') {

stringaInput = stringaInput.substring(0, stringaInput.length()-1);

trovatoCarattereDiSpaziatura = **true**;

**break**;

}

}

} **while** (trovatoCarattereDiSpaziatura==**true**);

**return** stringaInput;

}

/\*\*

\* Metodo pubblico statico per la ricerca di un solo numero INTERO in una stringa di testo dove sono presenti degli spazi.

\* **@param** stringaInput (stringa di testo).

\* **@return** trovatoUnNumeroIntero (variabile booleana).

\*/

**public** **static** **boolean** trovaNumeroIntero (**String** stringaInput) {

**int** j = 0;

**boolean** trovatoUnNumeroIntero = **false**;

**boolean** trovatoUnCarattereNonConsentito = **false**;

**boolean** trovatoSpazioA = **false**;

**boolean** trovatoSpazioB = **false**;

**boolean** trovataUnaCifra = **false**;

**boolean** trovatoUnPrimoCarattereConsentito = **false**;

**boolean** trovatoUnoSpazioDopoUnCarattereValido = **false**;

**do** {

trovatoSpazioA = **false**;

trovatoSpazioB = **false**;

**for** (**int** i = 0; i<stringaInput.length(); i++) {

// Questa condizione verifica che non vengano mai inseriti caratteri non validi (diversi dalla cifre da 0 a 9, dal segno negativo (-) o positivo (+) e dai caratteri di spaziatura ( )).

**if** ((trovatoUnoSpazioDopoUnCarattereValido==**false**)&&(trovatoUnPrimoCarattereConsentito==**false**)&&((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='+')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9'))) {

trovatoUnCarattereNonConsentito = **true**;

trovatoUnNumeroIntero = **false**;

**break**;

// Questa condizione verifica che non vengano inseriti altri segni negativi (-) o positivi (+) dopo il primo carattere valido. In caso contrario, il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroIntero" e termina il ciclo iterativo FOR.

} **else** **if** ((trovatoUnoSpazioDopoUnCarattereValido==**false**)&&(trovatoUnPrimoCarattereConsentito==**true**)&&((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9'))) {

trovatoUnCarattereNonConsentito = **true**;

trovatoUnNumeroIntero = **false**;

**break**;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato come primo carattere valido un segno negativo (-) o positivo (+).

} **else** **if** ((trovatoUnoSpazioDopoUnCarattereValido==**false**)&&(trovatoUnPrimoCarattereConsentito==**false**)&&((stringaInput.charAt(i)=='-')||(stringaInput.charAt(i)=='+'))) {

trovatoUnPrimoCarattereConsentito = **true**;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato come carattere valido una cifra da 0 a 9. Questo ramo della selezione serve per verificare che nella stringa vi sia almeno un cifra.

} **else** **if** ((trovatoUnoSpazioDopoUnCarattereValido==**false**)&&(trovataUnaCifra==**false**)&&((stringaInput.charAt(i)=='0')||(stringaInput.charAt(i)=='1')||(stringaInput.charAt(i)=='2')||(stringaInput.charAt(i)=='3')||(stringaInput.charAt(i)=='4')||(stringaInput.charAt(i)=='5')||(stringaInput.charAt(i)=='6')||(stringaInput.charAt(i)=='7')||(stringaInput.charAt(i)=='8')||(stringaInput.charAt(i)=='9'))) {

trovataUnaCifra = **true**;

trovatoUnNumeroIntero = **true**;

trovatoUnPrimoCarattereConsentito = **true**;

// Questa condizione si verifica eventualmente tutte le volte che viene trovato un carattere di spaziatura prima di un carattere valido e lo elimina.

} **else** **if** ((trovatoUnPrimoCarattereConsentito==**false**)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1, stringaInput.length());

trovatoSpazioA = **true**;

**break**;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato un carattere di spaziatura dopo un carattere valido e lo elimina. Viene settata la variabile "j" = i e la variabile "trovatoUnoSpazioDopoUnCarattereValido" = TRUE per bypassare l'elaborazione dei primi 4 rami della struttura di selezione semplice del metodo.

} **else** **if** ((stringaInput.charAt(i)==' ')&&(j==0)) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1, stringaInput.length());

trovatoSpazioB = **true**;

trovatoUnoSpazioDopoUnCarattereValido = **true**;

j = i;

**break**;

// Questa condizione si verifica eventualmente tutte le volte che viene trovato un secondo carattere di spaziatura dopo un carattere valido ed elimina tutti gli spazi dopo il numero intero.

} **else** **if** ((stringaInput.charAt(i)==' ')&&(i==j)) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1, stringaInput.length());

trovatoSpazioB = **true**;

**break**;

// Questa condizione si verifica quando viene trovato un carattere diverso da quello di spaziatura dopo il numero intero. Il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroDecimale" e interrompe il ciclo iterativo FOR.

} **else** **if** ((stringaInput.charAt(i)!=' ')&&(i==j)) {

trovatoUnCarattereNonConsentito = **true**;

trovatoUnNumeroIntero = **false**;

**break**;

// Questa condizione si verifica se viene trovata almeno una cifra nella stringa di testo. In caso contrario il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroDecimale" e interrompe il ciclo iterativo FOR.

} **else** **if** (trovataUnaCifra!=**true**) {

trovatoUnNumeroIntero = **false**;

}

}

} **while** ((trovatoUnCarattereNonConsentito==**false**)&&((trovatoSpazioA==**true**)||(trovatoSpazioB==**true**)));

**return** trovatoUnNumeroIntero;

}

/\*\*

\* Metodo pubblico statico per la ricerca di un solo numero INTERO o DECIMALE in una stringa di testo dove sono presenti degli spazi.

\* **@param** stringaInput (stringa di testo).

\* **@return** trovatoUnNumeroInteroDecimale (variabile booleana).

\*/

**public** **static** **boolean** trovaNumeroDecimale(**String** stringaInput) {

**int** j = 0;

**boolean** trovatoUnNumeroInteroDecimale = **false**;

**boolean** trovatoUnCarattereNonConsentito = **false**;

**boolean** trovatoSpazioA = **false**;

**boolean** trovatoSpazioB = **false**;

**boolean** trovatoUnPuntoRadice = **false**;

**boolean** trovataUnaCifra = **false**;

**boolean** trovatoUnPrimoCarattereConsentito = **false**;

**boolean** trovatoUnoSpazioDopoUnCarattereValido = **false**;

**do** {

trovatoSpazioA = **false**;

trovatoSpazioB = **false**;

**for** (**int** i = 0; i<stringaInput.length(); i++) {

// Questa condizione verifica che non vengano inseriti caratteri non validi (diversi dalle cifre da 0 a 9, dal segno negativo (-) e positivo (+), dal punto radice (.) e dal carattere di spaziatura( )).

**if** ((trovatoUnoSpazioDopoUnCarattereValido==**false**)&&(trovatoUnPrimoCarattereConsentito==**false**)&&((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='+')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9'))) {

trovatoUnCarattereNonConsentito = **true**;

trovatoUnNumeroInteroDecimale = **false**;

**break**;

// Questa condizione si verifica quando viene trovato un segno negativo (-) o positivo (+) dopo un primo carattere valido; il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroDecimale", interrompendo il ciclo FOR.

} **else** **if** ((trovatoUnoSpazioDopoUnCarattereValido==**false**)&&(trovatoUnPrimoCarattereConsentito==**true**)&&((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9'))) {

trovatoUnCarattereNonConsentito = **true**;

trovatoUnNumeroInteroDecimale = **false**;

**break**;

// Questa condizione si verifica quando viene trovato un secondo punto radice nel numero decimale; il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroDecimale", interrompendo il ciclo FOR.

} **else** **if** ((trovatoUnoSpazioDopoUnCarattereValido==**false**)&&(trovatoUnPrimoCarattereConsentito==**true**)&&(trovatoUnPuntoRadice==**true**)&&((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9'))) {

trovatoUnCarattereNonConsentito = **true**;

trovatoUnNumeroInteroDecimale = **false**;

**break**;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato come primo carattere valido un segno negativo (-) o positivo (+).

} **else** **if** ((trovatoUnoSpazioDopoUnCarattereValido==**false**)&&(trovatoUnPrimoCarattereConsentito==**false**)&&((stringaInput.charAt(i)=='-')||(stringaInput.charAt(i)=='+'))) {

trovatoUnPrimoCarattereConsentito = **true**;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato come primo carattere valido un punto radice (.).

} **else** **if** ((trovatoUnoSpazioDopoUnCarattereValido==**false**)&&(trovatoUnPuntoRadice==**false**)&&(stringaInput.charAt(i)=='.')) {

trovatoUnPuntoRadice = **true**;

trovatoUnPrimoCarattereConsentito = **true**;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato come carattere valido una cifra da 0 a 9. Questo ramo della selezione serve per verificare che nella stringa vi sia almeno un cifra.

} **else** **if** ((trovatoUnoSpazioDopoUnCarattereValido==**false**)&&(trovataUnaCifra==**false**)&&((stringaInput.charAt(i)=='0')||(stringaInput.charAt(i)=='1')||(stringaInput.charAt(i)=='2')||(stringaInput.charAt(i)=='3')||(stringaInput.charAt(i)=='4')||(stringaInput.charAt(i)=='5')||(stringaInput.charAt(i)=='6')||(stringaInput.charAt(i)=='7')||(stringaInput.charAt(i)=='8')||(stringaInput.charAt(i)=='9'))) {

trovataUnaCifra = **true**;

trovatoUnNumeroInteroDecimale = **true**;

trovatoUnPrimoCarattereConsentito = **true**;

// Questa condizione si verifica eventualmente tutte le volte che viene trovato un carattere di spaziatura prima di un carattere valido e lo elimina.

} **else** **if** ((trovatoUnPrimoCarattereConsentito==**false**)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(i+1, stringaInput.length());

trovatoSpazioA = **true**;

**break**;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato un carattere di spaziatura dopo un carattere valido e lo elimina. Viene settata la variabile "j" = i e la variabile "trovatoUnoSpazioDopoUnCarattereValido" = TRUE per bypassare l'elaborazione dei primi 6 rami di selezione del metodo.

} **else** **if** ((stringaInput.charAt(i)==' ')&&(j==0)) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1, stringaInput.length());

trovatoSpazioB = **true**;

trovatoUnoSpazioDopoUnCarattereValido = **true**;

j = i;

**break**;

// Questa condizione si verifica eventualmente tutte le volte che viene trovato un secondo carattere di spaziatura dopo un carattere valido ed elimina tutti gli spazi dopo il numero intero o decimale.

} **else** **if** ((stringaInput.charAt(i)==' ')&&(i==j)) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1,stringaInput.length());

trovatoSpazioB = **true**;

**break**;

// Questa condizione si verifica quando viene trovato un carattere diverso da quello di spaziatura dopo il numero decimale. Il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroDecimale" e interrompe il ciclo iterativo FOR.

} **else** **if** ((stringaInput.charAt(i)!=' ')&&(i==j)) {

trovatoUnCarattereNonConsentito = **true**;

trovatoUnNumeroInteroDecimale = **false**;

**break**;

// Questa condizione si verifica se viene trovata almeno una cifra nella stringa di testo. In caso contrario il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroDecimale" e interrompe il ciclo iterativo FOR.

} **else** **if** (trovataUnaCifra!=**true**) {

trovatoUnNumeroInteroDecimale = **false**;

}

}

} **while** ((trovatoUnCarattereNonConsentito==**false**)&&((trovatoSpazioA==**true**)||(trovatoSpazioB==**true**)));

**return** trovatoUnNumeroInteroDecimale;

}

/\*\*

\* Metodo pubblico statico per la verifica di una stringa di testo del campo "cognome" / "nome" / "nome istituto scolastico" / "località", riferito a una persona / una scuola / un paese (o città)...

\* Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti.

\* Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito.

\* **@param** stringaInput (stringa di testo).

\* **@return** Valore booleano TRUE o FALSE.

\*/

**public** **static** **boolean** verificaCampoNome(**String** stringaInput) {

**boolean** esitoVerifica = **true**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='\'')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='à')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='è')&&(stringaInput.charAt(i)!='é')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='ì')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='ò')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='ù')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='À')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='È')&&(stringaInput.charAt(i)!='É')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='Ì')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='Ò')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='Ù')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

esitoVerifica = **false**;

**break**;

}

**return** esitoVerifica;

}

/\*\*

\* Metodo pubblico statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "cognome" / "nome" / "nome istituto scolastico" / "località", riferito a una persona / una scuola / un paese (o città)...

\* Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito.

\* **@param** stringaInput (stringa di testo).

\* **@return** Stringa.

\*/

**public** **static** **String** restituisciCaratteriNonConsentitiCampoNome(**String** stringaInput) {

**String** elencoCaratteriNonConsentiti = **new** String("");

**boolean** trovatoUnCarattereNonConsentito = **false**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='\'')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='à')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='è')&&(stringaInput.charAt(i)!='é')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='ì')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='ò')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='ù')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='À')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='È')&&(stringaInput.charAt(i)!='É')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='Ì')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='Ò')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='Ù')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

elencoCaratteriNonConsentiti += stringaInput.charAt(i) + " ";

trovatoUnCarattereNonConsentito = **true**;

}

**if** (trovatoUnCarattereNonConsentito==**true**)

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

**return** elencoCaratteriNonConsentiti;

}

/\*\*

\* Metodo pubblico statico per la verifica di una stringa di testo per il campo "classe" di una scuola.

\* Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti.

\* Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito.

\* **@param** stringaInput (stringa di testo).

\* **@return** Valore booleano TRUE o FALSE.

\*/

**public** **static** **boolean** verificaCampoClasseScuola(**String** stringaInput) {

**boolean** esitoVerifica = **true**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='ª')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

esitoVerifica = **false**;

**break**;

}

**return** esitoVerifica;

}

/\*\*

\* Metodo pubblico statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "classe" di una scuola.

\* Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito.

\* **@param** stringaInput (stringa di testo).

\* **@return** Stringa.

\*/

**public** **static** **String** restituisciCaratteriNonConsentitiCampoClasseScuola(**String** stringaInput) {

**String** elencoCaratteriNonConsentiti = **new** String("");

**boolean** trovatoUnCarattereNonConsentito = **false**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='ª')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

elencoCaratteriNonConsentiti += stringaInput.charAt(i) + " ";

trovatoUnCarattereNonConsentito = **true**;

}

**if** (trovatoUnCarattereNonConsentito==**true**)

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

**return** elencoCaratteriNonConsentiti;

}

/\*\*

\* Metodo pubblico statico per la verifica di una stringa di testo del campo "codiceAlfanumerico" di una persona o articolo.

\* Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti.

\* Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito.

\* **@param** stringaInput (stringa di testo).

\* **@return** esitoVerifica (variabile booleana).

\*/

**public** **static** **boolean** verificaCampoCodiceAlfanumerico (**String** stringaInput) {

**boolean** esitoVerifica = **true**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!='\_')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

esitoVerifica = **false**;

**break**;

}

**return** esitoVerifica;

}

/\*\*

\* Metodo pubblico statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "codiceAlfanumerico" di una persona o articolo.

\* Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito.

\* **@param** stringaInput (stringa di testo).

\* **@return** elencoCaratteriNonConsentiti (stringa di testo).

\*/

**public** **static** **String** restituisciCaratteriNonConsentitiCampoCodiceAlfanumerico(**String** stringaInput) {

**String** elencoCaratteriNonConsentiti = **new** String("");

**ArrayList**<String> caratteriNonConsentiti = **new** ArrayList<String>();

**boolean** trovatoUnCarattereNonConsentito = **false**;

**boolean** eliminatoUnCarattereDoppio = **false**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!='\_')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

caratteriNonConsentiti.add(stringaInput.charAt(i) + " ");

trovatoUnCarattereNonConsentito = **true**;

}

**if** (trovatoUnCarattereNonConsentito==**true**) {

**do** {

eliminatoUnCarattereDoppio = **false**;

etichetta:

**for** (**int** i = 0; i<caratteriNonConsentiti.size(); i++)

**for** (**int** j = 0; j<caratteriNonConsentiti.size(); j++)

**if** ((i!=j)&&(caratteriNonConsentiti.get(i).equals(caratteriNonConsentiti.get(j)))) {

caratteriNonConsentiti.remove(j);

eliminatoUnCarattereDoppio = **true**;

**break** etichetta;

}

} **while** (eliminatoUnCarattereDoppio==**true**);

**for** (**int** i = 0; i<caratteriNonConsentiti.size(); i++)

**if** (caratteriNonConsentiti.get(i).equals(" "))

elencoCaratteriNonConsentiti += "carattereDiSpaziatura ";

**else**

elencoCaratteriNonConsentiti += caratteriNonConsentiti.get(i) + " ";

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

}

**return** elencoCaratteriNonConsentiti;

}

/\*\*

\* Metodo pubblico statico per la per la verifica di una stringa di testo del campo "indirizzo" di una persona o abitazione.

\* Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti.

\* Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito.

\* **@param** stringaInput (stringa di testo).

\* **@return** esitoVerifica (variabile booleana).

\*/

**public** **static** **boolean** verificaCampoIndirizzo(**String** stringaInput) {

**boolean** esitoVerifica = **true**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='/')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='\'')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='à')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='è')&&(stringaInput.charAt(i)!='é')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='ì')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='ò')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='ù')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='À')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='È')&&(stringaInput.charAt(i)!='É')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='Ì')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='Ò')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='Ù')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

esitoVerifica = **false**;

**break**;

}

**return** esitoVerifica;

}

/\*\*

\* Metodo pubblico statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "indirizzo" di una persona o abitazione.

\* Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito.

\* **@param** stringaInput (stringa di testo).

\* **@return** elencoCaratteriNonConsentiti (stringa di testo).

\*/

**public** **static** **String** restituisciCaratteriNonConsentitiCampoIndirizzo(**String** stringaInput) {

**String** elencoCaratteriNonConsentiti = **new** String("");

**boolean** trovatoUnCarattereNonConsentito = **false**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='/')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='\'')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='à')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='è')&&(stringaInput.charAt(i)!='é')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='ì')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='ò')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='ù')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='À')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='È')&&(stringaInput.charAt(i)!='É')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='Ì')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='Ò')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='Ù')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

elencoCaratteriNonConsentiti += stringaInput.charAt(i) + " ";

trovatoUnCarattereNonConsentito = **true**;

}

**if** (trovatoUnCarattereNonConsentito==**true**)

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

**return** elencoCaratteriNonConsentiti;

}

/\*\*

\* Metodo pubblico statico per la verifica di una stringa di testo del campo "data" nel formato 'GG.MM.AAAA'.

\* Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti.

\* Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito o se viene trovata una data errata del tipo '41.13.2019' ma scritta con caratteri consentiti.

\* **@param** stringaInput (stringa di testo).

\* **@return** esitoVerifica (variabile booleana).

\*/

**public** **static** **boolean** verificaCampoData(**String** stringaInput) {

**boolean** esitoVerifica = **true**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**if** (stringaInput.length()!=10)

esitoVerifica = **false**;

**else** {

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='/')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')) {

esitoVerifica = **false**;

**break**;

}

**if** (esitoVerifica==**true**) {

**if** ((stringaInput.charAt(0)!='0')&&(stringaInput.charAt(0)!='1')&&(stringaInput.charAt(0)!='2')&&(stringaInput.charAt(0)!='3'))

esitoVerifica = **false**;

**if** ((stringaInput.charAt(1)!='0')&&(stringaInput.charAt(1)!='1')&&(stringaInput.charAt(1)!='2')&&(stringaInput.charAt(1)!='3')&&(stringaInput.charAt(1)!='4')&&(stringaInput.charAt(1)!='5')&&(stringaInput.charAt(1)!='6')&&(stringaInput.charAt(1)!='7')&&(stringaInput.charAt(1)!='8')&&(stringaInput.charAt(1)!='9'))

esitoVerifica = **false**;

**if** ((stringaInput.charAt(2)!='.')&&(stringaInput.charAt(2)!='/'))

esitoVerifica = **false**;

**if** ((stringaInput.charAt(3)!='0')&&(stringaInput.charAt(3)!='1'))

esitoVerifica = **false**;

**if** ((stringaInput.charAt(4)!='0')&&(stringaInput.charAt(4)!='1')&&(stringaInput.charAt(4)!='2')&&(stringaInput.charAt(4)!='3')&&(stringaInput.charAt(4)!='4')&&(stringaInput.charAt(4)!='5')&&(stringaInput.charAt(4)!='6')&&(stringaInput.charAt(4)!='7')&&(stringaInput.charAt(4)!='8')&&(stringaInput.charAt(4)!='9'))

esitoVerifica = **false**;

**if** ((stringaInput.charAt(5)!='.')&&(stringaInput.charAt(5)!='/'))

esitoVerifica = **false**;

**if** ((stringaInput.charAt(6)!='0')&&(stringaInput.charAt(6)!='1')&&(stringaInput.charAt(6)!='2')&&(stringaInput.charAt(6)!='3')&&(stringaInput.charAt(6)!='4')&&(stringaInput.charAt(6)!='5')&&(stringaInput.charAt(6)!='6')&&(stringaInput.charAt(6)!='7')&&(stringaInput.charAt(6)!='8')&&(stringaInput.charAt(6)!='9'))

esitoVerifica = **false**;

**if** ((stringaInput.charAt(7)!='0')&&(stringaInput.charAt(7)!='1')&&(stringaInput.charAt(7)!='2')&&(stringaInput.charAt(7)!='3')&&(stringaInput.charAt(7)!='4')&&(stringaInput.charAt(7)!='5')&&(stringaInput.charAt(7)!='6')&&(stringaInput.charAt(7)!='7')&&(stringaInput.charAt(7)!='8')&&(stringaInput.charAt(7)!='9'))

esitoVerifica = **false**;

**if** ((stringaInput.charAt(8)!='0')&&(stringaInput.charAt(8)!='1')&&(stringaInput.charAt(8)!='2')&&(stringaInput.charAt(8)!='3')&&(stringaInput.charAt(8)!='4')&&(stringaInput.charAt(8)!='5')&&(stringaInput.charAt(8)!='6')&&(stringaInput.charAt(8)!='7')&&(stringaInput.charAt(8)!='8')&&(stringaInput.charAt(8)!='9'))

esitoVerifica = **false**;

**if** ((stringaInput.charAt(9)!='0')&&(stringaInput.charAt(9)!='1')&&(stringaInput.charAt(9)!='2')&&(stringaInput.charAt(9)!='3')&&(stringaInput.charAt(9)!='4')&&(stringaInput.charAt(9)!='5')&&(stringaInput.charAt(9)!='6')&&(stringaInput.charAt(9)!='7')&&(stringaInput.charAt(9)!='8')&&(stringaInput.charAt(9)!='9'))

esitoVerifica = **false**;

**if** ((stringaInput.charAt(2)=='.')&&(stringaInput.charAt(5)!='.'))

esitoVerifica = **false**;

**if** ((stringaInput.charAt(2)=='/')&&(stringaInput.charAt(5)!='/'))

esitoVerifica = **false**;

}

}

**return** esitoVerifica;

}

/\*\*

\* Metodo pubblico statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "data" nel formato'GG.MM.AAAA'.

\* Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito, o se viene trovata una data errata del tipo '41.13.2019' ma scritta con caratteri consentiti.

\* **@param** stringaInput (stringa di testo).

\* **@return** elencoCaratteriNonConsentiti (stringa di testo).

\*/

**public** **static** **String** restituisciCaratteriNonConsentitiCampoData(**String** stringaInput) {

**String** elencoCaratteriNonConsentiti = **new** String("");

**ArrayList**<String> caratteriNonConsentiti = **new** ArrayList<String>();

**boolean** trovatoUnCarattereNonConsentito = **false**;

**boolean** eliminatoUnCarattereDoppio = **false**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='/')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')) {

caratteriNonConsentiti.add(stringaInput.charAt(i) + " ");

trovatoUnCarattereNonConsentito = **true**;

}

**if** (trovatoUnCarattereNonConsentito==**true**) {

**do** {

eliminatoUnCarattereDoppio = **false**;

etichetta:

**for** (**int** i = 0; i<caratteriNonConsentiti.size(); i++)

**for** (**int** j = 0; j<caratteriNonConsentiti.size(); j++)

**if** ((i!=j)&&(caratteriNonConsentiti.get(i).equals(caratteriNonConsentiti.get(j)))) {

caratteriNonConsentiti.remove(j);

eliminatoUnCarattereDoppio = **true**;

**break** etichetta;

}

} **while** (eliminatoUnCarattereDoppio==**true**);

**for** (**int** i = 0; i<caratteriNonConsentiti.size(); i++)

**if** (caratteriNonConsentiti.get(i).equals(" "))

elencoCaratteriNonConsentiti += "carattereDiSpaziatura ";

**else**

elencoCaratteriNonConsentiti += caratteriNonConsentiti.get(i) + " ";

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

}

**return** elencoCaratteriNonConsentiti;

}

/\*\*

\* Metodo pubblico statico per la verifica di una stringa di testo del campo "provincia" in sigla (2 lettere tranne Roma).

\* Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti.

\* Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito o se viene trovata una provincia non esistente del tipo 'ZO' ma scritta con caratteri consentiti.

\* **@param** stringaInput (stringa di testo).

\* **@return** esitoVerifica (variabile booleana).

\*/

**public** **static** **boolean** verificaCampoSiglaProvincia(**String** stringaInput) {

**String** elencoProvincieItaliane[][] = **new** **String**[110][3];

elencoProvincieItaliane[0][0] = "AG";

elencoProvincieItaliane[0][1] = "Agrigento";

elencoProvincieItaliane[0][2] = "Sicilia";

elencoProvincieItaliane[1][0] = "AL";

elencoProvincieItaliane[1][1] = "Alessandria";

elencoProvincieItaliane[1][2] = "Piemonte";

elencoProvincieItaliane[2][0] = "AN";

elencoProvincieItaliane[2][1] = "Ancona";

elencoProvincieItaliane[2][2] = "Marche";

elencoProvincieItaliane[3][0] = "AO";

elencoProvincieItaliane[3][1] = "Aosta";

elencoProvincieItaliane[3][2] = "Valle d'Aosta";

elencoProvincieItaliane[4][0] = "AQ";

elencoProvincieItaliane[4][1] = "L'Aquila";

elencoProvincieItaliane[4][2] = "Abruzzo";

elencoProvincieItaliane[5][0] = "AR";

elencoProvincieItaliane[5][1] = "Arezzo";

elencoProvincieItaliane[5][2] = "Toscana";

elencoProvincieItaliane[6][0] = "AP";

elencoProvincieItaliane[6][1] = "Ascoli Piceno";

elencoProvincieItaliane[6][2] = "Marche";

elencoProvincieItaliane[7][0] = "AT";

elencoProvincieItaliane[7][1] = "Asti";

elencoProvincieItaliane[7][2] = "Piemonte";

elencoProvincieItaliane[8][0] = "AV";

elencoProvincieItaliane[8][1] = "Avellino";

elencoProvincieItaliane[8][2] = "Campania";

elencoProvincieItaliane[9][0] = "BA";

elencoProvincieItaliane[9][1] = "Bari";

elencoProvincieItaliane[9][2] = "Puglia";

elencoProvincieItaliane[10][0] = "BT";

elencoProvincieItaliane[10][1] = "Barletta Andria Trani";

elencoProvincieItaliane[10][2] = "Puglia";

elencoProvincieItaliane[11][0] = "BL";

elencoProvincieItaliane[11][1] = "Belluno";

elencoProvincieItaliane[11][2] = "Veneto";

elencoProvincieItaliane[12][0] = "BN";

elencoProvincieItaliane[12][1] = "Benevento";

elencoProvincieItaliane[12][2] = "Campania";

elencoProvincieItaliane[13][0] = "BG";

elencoProvincieItaliane[13][1] = "Bergamo";

elencoProvincieItaliane[13][2] = "Lombardia";

elencoProvincieItaliane[14][0] = "BI";

elencoProvincieItaliane[14][1] = "Biella";

elencoProvincieItaliane[14][2] = "Piemonte";

elencoProvincieItaliane[15][0] = "BO";

elencoProvincieItaliane[15][1] = "Bologna";

elencoProvincieItaliane[15][2] = "Emilia Romagna";

elencoProvincieItaliane[16][0] = "BZ";

elencoProvincieItaliane[16][1] = "Bolzano";

elencoProvincieItaliane[16][2] = "Trentino Alto Adige";

elencoProvincieItaliane[17][0] = "BS";

elencoProvincieItaliane[17][1] = "Brescia";

elencoProvincieItaliane[17][2] = "Lombardia";

elencoProvincieItaliane[18][0] = "BR";

elencoProvincieItaliane[18][1] = "Brindisi";

elencoProvincieItaliane[18][2] = "Puglia";

elencoProvincieItaliane[19][0] = "CA";

elencoProvincieItaliane[19][1] = "Cagliari";

elencoProvincieItaliane[19][2] = "Sardegna";

elencoProvincieItaliane[20][0] = "CL";

elencoProvincieItaliane[20][1] = "Caltanissetta";

elencoProvincieItaliane[20][2] = "Sicilia";

elencoProvincieItaliane[21][0] = "CB";

elencoProvincieItaliane[21][1] = "Campobasso";

elencoProvincieItaliane[21][2] = "Molise";

elencoProvincieItaliane[22][0] = "CI";

elencoProvincieItaliane[22][1] = "Carbonia Iglesias";

elencoProvincieItaliane[22][2] = "Sardegna";

elencoProvincieItaliane[23][0] = "CE";

elencoProvincieItaliane[23][1] = "Caserta";

elencoProvincieItaliane[23][2] = "Campania";

elencoProvincieItaliane[24][0] = "CT";

elencoProvincieItaliane[24][1] = "Catania";

elencoProvincieItaliane[24][2] = "Sicilia";

elencoProvincieItaliane[25][0] = "CZ";

elencoProvincieItaliane[25][1] = "Catanzaro";

elencoProvincieItaliane[25][2] = "Calabria";

elencoProvincieItaliane[26][0] = "CH";

elencoProvincieItaliane[26][1] = "Chieti";

elencoProvincieItaliane[26][2] = "Abruzzo";

elencoProvincieItaliane[27][0] = "CO";

elencoProvincieItaliane[27][1] = "Como";

elencoProvincieItaliane[27][2] = "Lombardia";

elencoProvincieItaliane[28][0] = "CS";

elencoProvincieItaliane[28][1] = "Cosenza";

elencoProvincieItaliane[28][2] = "Calabria";

elencoProvincieItaliane[29][0] = "CR";

elencoProvincieItaliane[29][1] = "Cremona";

elencoProvincieItaliane[29][2] = "Lombardia";

elencoProvincieItaliane[30][0] = "KR";

elencoProvincieItaliane[30][1] = "Crotone";

elencoProvincieItaliane[30][2] = "Calabria";

elencoProvincieItaliane[31][0] = "CN";

elencoProvincieItaliane[31][1] = "Cuneo";

elencoProvincieItaliane[31][2] = "Piemonte";

elencoProvincieItaliane[32][0] = "EN";

elencoProvincieItaliane[32][1] = "Enna";

elencoProvincieItaliane[32][2] = "Sicilia";

elencoProvincieItaliane[33][0] = "FM";

elencoProvincieItaliane[33][1] = "Fermo";

elencoProvincieItaliane[33][2] = "Marche";

elencoProvincieItaliane[34][0] = "FE";

elencoProvincieItaliane[34][1] = "Ferrara";

elencoProvincieItaliane[34][2] = "Emilia Romagna";

elencoProvincieItaliane[35][0] = "FI";

elencoProvincieItaliane[35][1] = "Firenze";

elencoProvincieItaliane[35][2] = "Toscana";

elencoProvincieItaliane[36][0] = "FG";

elencoProvincieItaliane[36][1] = "Foggia";

elencoProvincieItaliane[36][2] = "Puglia";

elencoProvincieItaliane[37][0] = "FC";

elencoProvincieItaliane[37][1] = "Forlì Cesena";

elencoProvincieItaliane[37][2] = "Emilia Romagna";

elencoProvincieItaliane[38][0] = "FR";

elencoProvincieItaliane[38][1] = "Frosinone";

elencoProvincieItaliane[38][2] = "Lazio";

elencoProvincieItaliane[39][0] = "GE";

elencoProvincieItaliane[39][1] = "Genova";

elencoProvincieItaliane[39][2] = "Liguria";

elencoProvincieItaliane[40][0] = "GO";

elencoProvincieItaliane[40][1] = "Gorizia";

elencoProvincieItaliane[40][2] = "Friuli Venezia Giulia";

elencoProvincieItaliane[41][0] = "GR";

elencoProvincieItaliane[41][1] = "Grosseto";

elencoProvincieItaliane[41][2] = "Toscana";

elencoProvincieItaliane[42][0] = "IM";

elencoProvincieItaliane[42][1] = "Imperia";

elencoProvincieItaliane[42][2] = "Liguria";

elencoProvincieItaliane[43][0] = "IS";

elencoProvincieItaliane[43][1] = "Isernia";

elencoProvincieItaliane[43][2] = "Molise";

elencoProvincieItaliane[44][0] = "SP";

elencoProvincieItaliane[44][1] = "La Spezia";

elencoProvincieItaliane[44][2] = "Liguria";

elencoProvincieItaliane[45][0] = "LT";

elencoProvincieItaliane[45][1] = "Latina";

elencoProvincieItaliane[45][2] = "Lazio";

elencoProvincieItaliane[46][0] = "LE";

elencoProvincieItaliane[46][1] = "Lecce";

elencoProvincieItaliane[46][2] = "Puglia";

elencoProvincieItaliane[47][0] = "LC";

elencoProvincieItaliane[47][1] = "Lecco";

elencoProvincieItaliane[47][2] = "Lombardia";

elencoProvincieItaliane[48][0] = "LI";

elencoProvincieItaliane[48][1] = "Livorno";

elencoProvincieItaliane[48][2] = "Toscana";

elencoProvincieItaliane[49][0] = "LO";

elencoProvincieItaliane[49][1] = "Lodi";

elencoProvincieItaliane[49][2] = "Lombardia";

elencoProvincieItaliane[50][0] = "LU";

elencoProvincieItaliane[50][1] = "Lucca";

elencoProvincieItaliane[50][2] = "Toscana";

elencoProvincieItaliane[51][0] = "MC";

elencoProvincieItaliane[51][1] = "Macerata";

elencoProvincieItaliane[51][2] = "Marche";

elencoProvincieItaliane[52][0] = "MN";

elencoProvincieItaliane[52][1] = "Mantova";

elencoProvincieItaliane[52][2] = "Lombardia";

elencoProvincieItaliane[53][0] = "MS";

elencoProvincieItaliane[53][1] = "Massa Carrara";

elencoProvincieItaliane[53][2] = "Toscana";

elencoProvincieItaliane[54][0] = "MT";

elencoProvincieItaliane[54][1] = "Matera";

elencoProvincieItaliane[54][2] = "Basilicata";

elencoProvincieItaliane[55][0] = "VS";

elencoProvincieItaliane[55][1] = "Medio Campidano";

elencoProvincieItaliane[55][2] = "Sardegna";

elencoProvincieItaliane[56][0] = "ME";

elencoProvincieItaliane[56][1] = "Messina";

elencoProvincieItaliane[56][2] = "Sicilia";

elencoProvincieItaliane[57][0] = "MI";

elencoProvincieItaliane[57][1] = "Milano";

elencoProvincieItaliane[57][2] = "Lombardia";

elencoProvincieItaliane[58][0] = "MO";

elencoProvincieItaliane[58][1] = "Modena";

elencoProvincieItaliane[58][2] = "Emilia Romagna";

elencoProvincieItaliane[59][0] = "MB";

elencoProvincieItaliane[59][1] = "Monza Brianza";

elencoProvincieItaliane[59][2] = "Lombardia";

elencoProvincieItaliane[60][0] = "NA";

elencoProvincieItaliane[60][1] = "Napoli";

elencoProvincieItaliane[60][2] = "Campania";

elencoProvincieItaliane[61][0] = "NO";

elencoProvincieItaliane[61][1] = "Novara";

elencoProvincieItaliane[61][2] = "Piemonte";

elencoProvincieItaliane[62][0] = "NU";

elencoProvincieItaliane[62][1] = "Nuoro";

elencoProvincieItaliane[62][2] = "Sardegna";

elencoProvincieItaliane[63][0] = "OG";

elencoProvincieItaliane[63][1] = "Ogliastra";

elencoProvincieItaliane[63][2] = "Sardegna";

elencoProvincieItaliane[64][0] = "OT";

elencoProvincieItaliane[64][1] = "Olbia Tempio";

elencoProvincieItaliane[64][2] = "Sardegna";

elencoProvincieItaliane[65][0] = "OR";

elencoProvincieItaliane[65][1] = "Oristano";

elencoProvincieItaliane[65][2] = "Sardegna";

elencoProvincieItaliane[66][0] = "PD";

elencoProvincieItaliane[66][1] = "Padova";

elencoProvincieItaliane[66][2] = "Veneto";

elencoProvincieItaliane[67][0] = "PA";

elencoProvincieItaliane[67][1] = "Palermo";

elencoProvincieItaliane[67][2] = "Sicilia";

elencoProvincieItaliane[68][0] = "PR";

elencoProvincieItaliane[68][1] = "Parma";

elencoProvincieItaliane[68][2] = "Emilia Romagna";

elencoProvincieItaliane[69][0] = "PV";

elencoProvincieItaliane[69][1] = "Pavia";

elencoProvincieItaliane[69][2] = "Lombardia";

elencoProvincieItaliane[70][0] = "PG";

elencoProvincieItaliane[70][1] = "Perugia";

elencoProvincieItaliane[70][2] = "Umbria";

elencoProvincieItaliane[71][0] = "PU";

elencoProvincieItaliane[71][1] = "Pesaro Urbino";

elencoProvincieItaliane[71][2] = "Marche";

elencoProvincieItaliane[72][0] = "PE";

elencoProvincieItaliane[72][1] = "Pescara";

elencoProvincieItaliane[72][2] = "Abruzzo";

elencoProvincieItaliane[73][0] = "PC";

elencoProvincieItaliane[73][1] = "Piacenza";

elencoProvincieItaliane[73][2] = "Emilia Romagna";

elencoProvincieItaliane[74][0] = "PI";

elencoProvincieItaliane[74][1] = "Pisa";

elencoProvincieItaliane[74][2] = "Toscana";

elencoProvincieItaliane[75][0] = "PT";

elencoProvincieItaliane[75][1] = "Pistoia";

elencoProvincieItaliane[75][2] = "Toscana";

elencoProvincieItaliane[76][0] = "PN";

elencoProvincieItaliane[76][1] = "Pordenone";

elencoProvincieItaliane[76][2] = "Friuli Venezia Giulia";

elencoProvincieItaliane[77][0] = "PZ";

elencoProvincieItaliane[77][1] = "Potenza";

elencoProvincieItaliane[77][2] = "Basilicata";

elencoProvincieItaliane[78][0] = "PO";

elencoProvincieItaliane[78][1] = "Prato";

elencoProvincieItaliane[78][2] = "Toscana";

elencoProvincieItaliane[79][0] = "RG";

elencoProvincieItaliane[79][1] = "Ragusa";

elencoProvincieItaliane[79][2] = "Sicilia";

elencoProvincieItaliane[80][0] = "RA";

elencoProvincieItaliane[80][1] = "Ravenna";

elencoProvincieItaliane[80][2] = "Emilia Romagna";

elencoProvincieItaliane[81][0] = "RC";

elencoProvincieItaliane[81][1] = "Reggio Calabria";

elencoProvincieItaliane[81][2] = "Calabria";

elencoProvincieItaliane[82][0] = "RE";

elencoProvincieItaliane[82][1] = "Reggio Emilia";

elencoProvincieItaliane[82][2] = "Emilia Romagna";

elencoProvincieItaliane[83][0] = "RI";

elencoProvincieItaliane[83][1] = "Rieti";

elencoProvincieItaliane[83][2] = "Lazio";

elencoProvincieItaliane[84][0] = "RN";

elencoProvincieItaliane[84][1] = "Rimini";

elencoProvincieItaliane[84][2] = "Emilia Romagna";

elencoProvincieItaliane[85][0] = "Roma";

elencoProvincieItaliane[85][1] = "Roma";

elencoProvincieItaliane[85][2] = "Lazio";

elencoProvincieItaliane[86][0] = "RO";

elencoProvincieItaliane[86][1] = "Rovigo";

elencoProvincieItaliane[86][2] = "Veneto";

elencoProvincieItaliane[87][0] = "SA";

elencoProvincieItaliane[87][1] = "Salerno";

elencoProvincieItaliane[87][2] = "Campania";

elencoProvincieItaliane[88][0] = "SS";

elencoProvincieItaliane[88][1] = "Sassari";

elencoProvincieItaliane[88][2] = "Sardegna";

elencoProvincieItaliane[89][0] = "SV";

elencoProvincieItaliane[89][1] = "Savona";

elencoProvincieItaliane[89][2] = "Liguria";

elencoProvincieItaliane[90][0] = "SI";

elencoProvincieItaliane[90][1] = "Siena";

elencoProvincieItaliane[90][2] = "Toscana";

elencoProvincieItaliane[91][0] = "SR";

elencoProvincieItaliane[91][1] = "Siracusa";

elencoProvincieItaliane[91][2] = "Sicilia";

elencoProvincieItaliane[92][0] = "SO";

elencoProvincieItaliane[92][1] = "Sondrio";

elencoProvincieItaliane[92][2] = "Lombardia";

elencoProvincieItaliane[93][0] = "TA";

elencoProvincieItaliane[93][1] = "Taranto";

elencoProvincieItaliane[93][2] = "Puglia";

elencoProvincieItaliane[94][0] = "TE";

elencoProvincieItaliane[94][1] = "Teramo";

elencoProvincieItaliane[94][2] = "Abruzzo";

elencoProvincieItaliane[95][0] = "TR";

elencoProvincieItaliane[95][1] = "Terni";

elencoProvincieItaliane[95][2] = "Umbria";

elencoProvincieItaliane[96][0] = "TO";

elencoProvincieItaliane[96][1] = "Torino";

elencoProvincieItaliane[96][2] = "Piemonte";

elencoProvincieItaliane[97][0] = "TP";

elencoProvincieItaliane[97][1] = "Trapani";

elencoProvincieItaliane[97][2] = "Sicilia";

elencoProvincieItaliane[98][0] = "TN";

elencoProvincieItaliane[98][1] = "Trento";

elencoProvincieItaliane[98][2] = "Trentino Alto Adige";

elencoProvincieItaliane[99][0] = "TV";

elencoProvincieItaliane[99][1] = "Treviso";

elencoProvincieItaliane[99][2] = "Veneto";

elencoProvincieItaliane[100][0] = "TS";

elencoProvincieItaliane[100][1] = "Trieste";

elencoProvincieItaliane[100][2] = "Friuli Venezia Giulia";

elencoProvincieItaliane[101][0] = "UD";

elencoProvincieItaliane[101][1] = "Udine";

elencoProvincieItaliane[101][2] = "Friuli Venezia Giulia";

elencoProvincieItaliane[102][0] = "VA";

elencoProvincieItaliane[102][1] = "Varese";

elencoProvincieItaliane[102][2] = "Lombardia";

elencoProvincieItaliane[103][0] = "VE";

elencoProvincieItaliane[103][1] = "Venezia";

elencoProvincieItaliane[103][2] = "Veneto";

elencoProvincieItaliane[104][0] = "VB";

elencoProvincieItaliane[104][1] = "Verbania";

elencoProvincieItaliane[104][2] = "Piemonte";

elencoProvincieItaliane[105][0] = "VC";

elencoProvincieItaliane[105][1] = "Vercelli";

elencoProvincieItaliane[105][2] = "Piemonte";

elencoProvincieItaliane[106][0] = "VR";

elencoProvincieItaliane[106][1] = "Verona";

elencoProvincieItaliane[106][2] = "Veneto";

elencoProvincieItaliane[107][0] = "VV";

elencoProvincieItaliane[107][1] = "Vibo Valentia";

elencoProvincieItaliane[107][2] = "Calabria";

elencoProvincieItaliane[108][0] = "VI";

elencoProvincieItaliane[108][1] = "Vicenza";

elencoProvincieItaliane[108][2] = "Veneto";

elencoProvincieItaliane[109][0] = "VT";

elencoProvincieItaliane[109][1] = "Viterbo";

elencoProvincieItaliane[109][2] = "Lazio";

**boolean** esitoVerifica = **true**;

**boolean** trovataSiglaProvincia = **false**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='Z')) {

esitoVerifica = **false**;

**break**;

}

**if** (esitoVerifica==**true**) {

**for** (**int** i = 0; i<elencoProvincieItaliane.length; i++)

**if** (stringaInput.equalsIgnoreCase(elencoProvincieItaliane[i][0])) {

trovataSiglaProvincia = **true**;

**break**;

}

**if** (trovataSiglaProvincia==**false**)

esitoVerifica = **false**;

}

**return** esitoVerifica;

}

/\*\*

\* Metodo pubblico statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "provincia" in sigla (2 lettere maiuscole tranne Roma).

\* Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito, o se viene trovata una provincia inesistente del tipo 'ZO' ma scritta con caratteri consentiti.

\* **@param** stringaInput (stringa di testo).

\* **@return** elencoCaratteriNonConsentiti (stringa di testo).

\*/

**public** **static** **String** restituisciCaratteriNonConsentitiCampoSiglaProvincia(**String** stringaInput) {

**String** elencoCaratteriNonConsentiti = **new** String("");

**ArrayList**<String> caratteriNonConsentiti = **new** ArrayList<String>();

**boolean** trovatoUnCarattereNonConsentito = **false**;

**boolean** eliminatoUnCarattereDoppio = **false**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='Z')) {

caratteriNonConsentiti.add(stringaInput.charAt(i) + " ");

trovatoUnCarattereNonConsentito = **true**;

}

**if** (trovatoUnCarattereNonConsentito==**true**) {

**do** {

eliminatoUnCarattereDoppio = **false**;

etichetta:

**for** (**int** i = 0; i<caratteriNonConsentiti.size(); i++)

**for** (**int** j = 0; j<caratteriNonConsentiti.size(); j++)

**if** ((i!=j)&&(caratteriNonConsentiti.get(i).equals(caratteriNonConsentiti.get(j)))) {

caratteriNonConsentiti.remove(j);

eliminatoUnCarattereDoppio = **true**;

**break** etichetta;

}

} **while** (eliminatoUnCarattereDoppio==**true**);

**for** (**int** i = 0; i<caratteriNonConsentiti.size(); i++)

**if** (caratteriNonConsentiti.get(i).equals(" "))

elencoCaratteriNonConsentiti += "carattereDiSpaziatura ";

**else**

elencoCaratteriNonConsentiti += caratteriNonConsentiti.get(i) + " ";

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

}

**return** elencoCaratteriNonConsentiti;

}

/\*\*

\* Metodo pubblico statico per la verifica di una stringa di testo per un campo numerico, come, a esempio, il numero di telefono di una persona.

\* Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti.

\* Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito.

\* **@param** stringaInput (stringa di testo).

\* **@return** esitoVerifica (variabile booleana).

\*/

**public** **static** **boolean** verificaCampoNumerico(**String** stringaInput) {

**boolean** esitoVerifica = **true**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')) {

esitoVerifica = **false**;

**break**;

}

**return** esitoVerifica;

}

/\*\*

\* Metodo pubblico statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo per un campo numerico, come, a esempio, il numero di telefono di una persona.

\* Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito.

\* **@param** stringaInput (stringa di testo).

\* **@return** elencoCaratteriNonConsentiti (stringa di testo).

\*/

**public** **static** **String** restituisciCaratteriNonConsentitiCampoNumerico(**String** stringaInput) {

**String** elencoCaratteriNonConsentiti = **new** String("");

**ArrayList**<String> caratteriNonConsentiti = **new** ArrayList<String>();

**boolean** trovatoUnCarattereNonConsentito = **false**;

**boolean** eliminatoUnCarattereDoppio = **false**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')) {

caratteriNonConsentiti.add(stringaInput.charAt(i) + " ");

trovatoUnCarattereNonConsentito = **true**;

}

**if** (trovatoUnCarattereNonConsentito==**true**) {

**do** {

eliminatoUnCarattereDoppio = **false**;

etichetta:

**for** (**int** i = 0; i<caratteriNonConsentiti.size(); i++)

**for** (**int** j = 0; j<caratteriNonConsentiti.size(); j++)

**if** ((i!=j)&&(caratteriNonConsentiti.get(i).equals(caratteriNonConsentiti.get(j)))) {

caratteriNonConsentiti.remove(j);

eliminatoUnCarattereDoppio = **true**;

**break** etichetta;

}

} **while** (eliminatoUnCarattereDoppio==**true**);

**for** (**int** i = 0; i<caratteriNonConsentiti.size(); i++)

**if** (caratteriNonConsentiti.get(i).equals(" "))

elencoCaratteriNonConsentiti += "carattereDiSpaziatura ";

**else**

elencoCaratteriNonConsentiti += caratteriNonConsentiti.get(i) + " ";

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

}

**return** elencoCaratteriNonConsentiti;

}

/\*\*

\* Metodo pubblico statico per la verifica di una stringa di testo del campo "indirizzo e-mail" di una persona o azienda.

\* Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti.

\* Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito o se viene trovato un indirizzo e-mail errato del tipo "marco@rossi@gmail.com" ma scritto con caratteri consentiti.

\* **@param** stringaInput (stringa di testo).

\* **@return** esitoVerifica (variabile booleana).

\*/

**public** **static** **boolean** verificaCampoIndirizzoE\_mail(**String** stringaInput) {

**int** numeroCaratteriChiocciola = 0;

**boolean** esitoVerifica = **true**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++) {

**if** ((stringaInput.charAt(i)!='@')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')) {

esitoVerifica = **false**;

**break**;

}

**if** (stringaInput.charAt(i)=='@')

numeroCaratteriChiocciola++;

}

**if** (numeroCaratteriChiocciola!=1)

esitoVerifica = **false**;

**return** esitoVerifica;

}

/\*\*

\* Metodo pubblico statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "indirizzo e-mail" di una persona o azienda.

\* Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito o se viene trovato un indirizzo e-mail errato del tipo "marco@rossi@gmail.com" ma scritto con caratteri consentiti.

\* **@param** stringaInput (stringa di testo).

\* **@return** elencoCaratteriNonConsentiti (stringa di testo).

\*/

**public** **static** **String** restituisciCaratteriNonConsentitiCampoIndirizzoE\_mail(**String** stringaInput) {

**String** elencoCaratteriNonConsentiti = **new** String("");

**ArrayList**<String> caratteriNonConsentiti = **new** ArrayList<String>();

**boolean** trovatoUnCarattereNonConsentito = **false**;

**boolean** eliminatoUnCarattereDoppio = **false**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!='@')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')) {

caratteriNonConsentiti.add(stringaInput.charAt(i) + " ");

trovatoUnCarattereNonConsentito = **true**;

}

**if** (trovatoUnCarattereNonConsentito==**true**) {

**do** {

eliminatoUnCarattereDoppio = **false**;

etichetta:

**for** (**int** i = 0; i<caratteriNonConsentiti.size(); i++)

**for** (**int** j = 0; j<caratteriNonConsentiti.size(); j++)

**if** ((i!=j)&&(caratteriNonConsentiti.get(i).equals(caratteriNonConsentiti.get(j)))) {

caratteriNonConsentiti.remove(j);

eliminatoUnCarattereDoppio = **true**;

**break** etichetta;

}

} **while** (eliminatoUnCarattereDoppio==**true**);

**for** (**int** i = 0; i<caratteriNonConsentiti.size(); i++)

**if** (caratteriNonConsentiti.get(i).equals(" "))

elencoCaratteriNonConsentiti += "carattereDiSpaziatura ";

**else**

elencoCaratteriNonConsentiti += caratteriNonConsentiti.get(i) + " ";

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

}

**return** elencoCaratteriNonConsentiti;

}

/\*\*

\* Metodo pubblico statico per la verifica di una stringa di testo del campo "testo generico" di una nota o una descrizione.

\* Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti.

\* Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito.

\* **@param** stringaInput (stringa di testo).

\* **@return** esitoVerifica (variabile booleana).

\*/

**public** **static** **boolean** verificaCampoTesto(**String** stringaInput) {

**boolean** esitoVerifica = **true**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!='ª')&&(stringaInput.charAt(i)!='°')&&(stringaInput.charAt(i)!='@')&&(stringaInput.charAt(i)!='\_')&&(stringaInput.charAt(i)!='+')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='\*')&&(stringaInput.charAt(i)!='/')&&(stringaInput.charAt(i)!='^')&&(stringaInput.charAt(i)!='&')&&(stringaInput.charAt(i)!='%')&&(stringaInput.charAt(i)!='$')&&(stringaInput.charAt(i)!='£')&&(stringaInput.charAt(i)!='€')&&(stringaInput.charAt(i)!='!')&&(stringaInput.charAt(i)!='?')&&(stringaInput.charAt(i)!='\"')&&(stringaInput.charAt(i)!='\'')&&(stringaInput.charAt(i)!=',')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!=';')&&(stringaInput.charAt(i)!=':')&&(stringaInput.charAt(i)!='(')&&(stringaInput.charAt(i)!=')')&&(stringaInput.charAt(i)!='[')&&(stringaInput.charAt(i)!=']')&&(stringaInput.charAt(i)!='{')&&(stringaInput.charAt(i)!='}')&&(stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='à')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='è')&&(stringaInput.charAt(i)!='é')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='ì')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='ò')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='ù')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='À')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='È')&&(stringaInput.charAt(i)!='É')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='Ì')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='Ò')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='Ù')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

esitoVerifica = **false**;

**break**;

}

**return** esitoVerifica;

}

/\*\*

\* Metodo pubblico statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "testo generico" di una nota o una descrizione.

\* Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito.

\* **@param** stringaInput (stringa di testo).

\* **@return** elencoCaratteriNonConsentiti (stringa di testo).

\*/

**public** **static** **String** restituisciCaratteriNonConsentitiCampoTesto(**String** stringaInput) {

**String** elencoCaratteriNonConsentiti = **new** String("");

**boolean** trovatoUnCarattereNonConsentito = **false**;

stringaInput = *eliminaSpaziInEccesso*(stringaInput);

**for** (**int** i = 0; i<stringaInput.length(); i++)

**if** ((stringaInput.charAt(i)!='ª')&&(stringaInput.charAt(i)!='°')&&(stringaInput.charAt(i)!='@')&&(stringaInput.charAt(i)!='\_')&&(stringaInput.charAt(i)!='+')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='\*')&&(stringaInput.charAt(i)!='/')&&(stringaInput.charAt(i)!='^')&&(stringaInput.charAt(i)!='&')&&(stringaInput.charAt(i)!='%')&&(stringaInput.charAt(i)!='$')&&(stringaInput.charAt(i)!='£')&&(stringaInput.charAt(i)!='€')&&(stringaInput.charAt(i)!='!')&&(stringaInput.charAt(i)!='?')&&(stringaInput.charAt(i)!='\"')&&(stringaInput.charAt(i)!='\'')&&(stringaInput.charAt(i)!=',')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!=';')&&(stringaInput.charAt(i)!=':')&&(stringaInput.charAt(i)!='(')&&(stringaInput.charAt(i)!=')')&&(stringaInput.charAt(i)!='[')&&(stringaInput.charAt(i)!=']')&&(stringaInput.charAt(i)!='{')&&(stringaInput.charAt(i)!='}')&&(stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='à')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='è')&&(stringaInput.charAt(i)!='é')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='ì')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='ò')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='ù')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='À')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='È')&&(stringaInput.charAt(i)!='É')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='Ì')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='Ò')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='Ù')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

elencoCaratteriNonConsentiti += stringaInput.charAt(i) + " ";

trovatoUnCarattereNonConsentito = **true**;

}

**if** (trovatoUnCarattereNonConsentito==**true**)

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

**return** elencoCaratteriNonConsentiti;

}

/\*\*

\* Metodo pubblico statico per l'ordinamento di una serie di stringhe (contenute in un array dinamico) dinamico dalla A alla Z.

\* **@param** nomi (array dinamico di stringhe).

\* **@return** void (nessun valore).

\*/

**public** **static** **void** ordinaStringheAZ(**ArrayList**<String> nomi) {

**String** stringa01 = **new** String();

**String** stringa02 = **new** String();

**int** lunghezzaMaxNome = 0;

**int** variazioneLunghezzaNome = 0;

**boolean** effettuatoScambio = **false**;

// Questo ciclo FOR cerca all'interno dell'array dinamico "nomi" il nome con lunghezza maggiore (in caratteri) e assegna il valore numerico della lunghezza del nome alla variabile "lunghezzaMaxNome".

**for** (**int** i = 0; i<nomi.size(); i++)

**if** (nomi.get(i).length()>lunghezzaMaxNome)

lunghezzaMaxNome = nomi.get(i).length();

// Questo ciclo FOR aggiunge una serie di caratteri di spaziatura alla fine di ogni nome contenuto nell'array dinamico "nomi" in nodo da ottenere soltanto nomi della stessa lunghezza.

**for** (**int** i = 0; i<nomi.size(); i++)

**if** (nomi.get(i).length()<lunghezzaMaxNome) {

stringa01 = nomi.get(i);

variazioneLunghezzaNome = lunghezzaMaxNome-stringa01.length();

**for** (**int** j = 0; j<(variazioneLunghezzaNome); j++)

stringa01 += " ";

nomi.remove(i);

nomi.add(i, stringa01);

}

{ // Questo blocco di elaborazione ordina le stringhe all'interno dell'array dinamico "nomi" dalla A alla Z.

**do** {

effettuatoScambio = **false**;

**for** (**int** i = 0; i<nomi.size()-1; i++)

**for** (**int** j = 0; j<lunghezzaMaxNome; j++)

**if** ((nomi.get(i).toLowerCase().substring(0, j).equalsIgnoreCase(nomi.get(i+1).toLowerCase().substring(0, j)))&&(((**int**) nomi.get(i).toLowerCase().charAt(j))>((**int**) nomi.get(i+1).toLowerCase().charAt(j)))) {

stringa01 = nomi.get(i);

stringa02 = nomi.get(i+1);

nomi.remove(i);

nomi.add(i, stringa02);

nomi.remove(i+1);

nomi.add(i+1, stringa01);

effettuatoScambio = **true**;

}

} **while** (effettuatoScambio==**true**);

}

}

/\*\*

\* Metodo pubblico statico per l'ordinamento di una serie di stringhe (contenute in un array dinamico) dalla Z alla A.

\* **@param** nomi (array dinamico di stringhe).

\* **@return** void (nessun valore).

\*/

**public** **static** **void** ordinaStringheZA(**ArrayList**<String> nomi) {

**String** stringa01 = **new** String();

**String** stringa02 = **new** String();

**int** lunghezzaMaxNome = 0;

**int** variazioneLunghezzaNome = 0;

**boolean** effettuatoScambio = **false**;

// Questo ciclo FOR cerca all'interno dell'array dinamico "nomi" il nome con lunghezza maggiore (in caratteri) e assegna il valore numerico della lunghezza del nome alla variabile "lunghezzaMaxNome".

**for** (**int** i = 0; i<nomi.size(); i++)

**if** (nomi.get(i).length()>lunghezzaMaxNome)

lunghezzaMaxNome = nomi.get(i).length();

// Questo ciclo FOR aggiunge una serie di caratteri di spaziatura alla fine di ogni nome contenuto nell'array dinamico "nomi" in nodo da ottenere soltanto nomi della stessa lunghezza.

**for** (**int** i = 0; i<nomi.size(); i++)

**if** (nomi.get(i).length()<lunghezzaMaxNome) {

stringa01 = nomi.get(i);

variazioneLunghezzaNome = lunghezzaMaxNome-stringa01.length();

**for** (**int** j = 0; j<(variazioneLunghezzaNome); j++)

stringa01 += " ";

nomi.remove(i);

nomi.add(i, stringa01);

}

{ // Questo blocco di elaborazione ordina le stringhe all'interno dell'array dinamico "nomi" dalla Z alla A.

**do** {

effettuatoScambio = **false**;

**for** (**int** i = 0; i<nomi.size()-1; i++)

**for** (**int** j = 0; j<lunghezzaMaxNome; j++)

**if** ((nomi.get(i).toLowerCase().substring(0, j).equalsIgnoreCase(nomi.get(i+1).toLowerCase().substring(0, j)))&&(((**int**) nomi.get(i).toLowerCase().charAt(j))<((**int**) nomi.get(i+1).toLowerCase().charAt(j)))) {

stringa01 = nomi.get(i);

stringa02 = nomi.get(i+1);

nomi.remove(i);

nomi.add(i, stringa02);

nomi.remove(i+1);

nomi.add(i+1, stringa01);

effettuatoScambio = **true**;

}

} **while** (effettuatoScambio==**true**);

}

}

/\*\*

\* Metodo pubblico statico per l'ordinamento di una serie di stringhe (contenute in un array) dalla A alla Z.

\* **@param** nomi[] (array di stringhe).

\* **@return** void (nessun valore).

\*/

**public** **static** **void** ordinaStringheAZ(**String** nomi[]) {

**String** stringa01 = **new** String();

**String** stringa02 = **new** String();

**int** lunghezzaMaxNome = 0;

**int** variazioneLunghezzaNome = 0;

**boolean** effettuatoScambio = **false**;

// Questo ciclo FOR cerca all'interno dell'array "nomi" il nome con lunghezza maggiore (in caratteri) e assegna il valore numerico della lunghezza del nome alla variabile "lunghezzaMaxNome".

**for** (**int** i = 0; i<nomi.length; i++)

**if** (nomi[i].length()>lunghezzaMaxNome)

lunghezzaMaxNome = nomi[i].length();

// Questo ciclo FOR aggiunge una serie di caratteri di spaziatura alla fine di ogni nome contenuto nell'array "nomi" in nodo da ottenere soltanto nomi della stessa lunghezza.

**for** (**int** i = 0; i<nomi.length; i++)

**if** (nomi[i].length()<lunghezzaMaxNome) {

stringa01 = nomi[i];

variazioneLunghezzaNome = lunghezzaMaxNome-stringa01.length();

**for** (**int** j = 0; j<(variazioneLunghezzaNome); j++)

stringa01 += " ";

nomi[i] = stringa01;

}

{ // Questo blocco di elaborazione ordina le stringhe all'interno dell'array "nomi" dalla A alla Z.

**do** {

effettuatoScambio = **false**;

**for** (**int** i = 0; i<nomi.length-1; i++)

**for** (**int** j = 0; j<lunghezzaMaxNome; j++)

**if** ((nomi[i].toLowerCase().substring(0, j).equalsIgnoreCase(nomi[i+1].toLowerCase().substring(0, j)))&&(((**int**) nomi[i].toLowerCase().charAt(j))>((**int**) nomi[i+1].toLowerCase().charAt(j)))) {

stringa01 = nomi[i];

stringa02 = nomi[i+1];

nomi[i] = stringa02;

nomi[i+1] = stringa01;

effettuatoScambio = **true**;

}

} **while** (effettuatoScambio==**true**);

}

}

/\*\*

\* Metodo pubblico statico per l'ordinamento di una serie di stringhe (contenute in un array) dalla Z alla A.

\* **@param** nomi[] (array di stringhe).

\* **@return** void (nessun valore).

\*/

**public** **static** **void** ordinaStringheZA(**String** nomi[]) {

**String** stringa01 = **new** String();

**String** stringa02 = **new** String();

**int** lunghezzaMaxNome = 0;

**int** variazioneLunghezzaNome = 0;

**boolean** effettuatoScambio = **false**;

// Questo ciclo FOR cerca all'interno dell'array "nomi" il nome con lunghezza maggiore (in caratteri) e assegna il valore numerico della lunghezza del nome alla variabile "lunghezzaMaxNome".

**for** (**int** i = 0; i<nomi.length; i++)

**if** (nomi[i].length()>lunghezzaMaxNome)

lunghezzaMaxNome = nomi[i].length();

// Questo ciclo FOR aggiunge una serie di caratteri di spaziatura alla fine di ogni nome contenuto nell'array "nomi" in nodo da ottenere soltanto nomi della stessa lunghezza.

**for** (**int** i = 0; i<nomi.length; i++)

**if** (nomi[i].length()<lunghezzaMaxNome) {

stringa01 = nomi[i];

variazioneLunghezzaNome = lunghezzaMaxNome-stringa01.length();

**for** (**int** j = 0; j<(variazioneLunghezzaNome); j++)

stringa01 += " ";

nomi[i] = stringa01;

}

{ // Questo blocco di elaborazione ordina le stringhe all'interno dell'array "nomi" dalla Z alla A.

**do** {

effettuatoScambio = **false**;

**for** (**int** i = 0; i<nomi.length-1; i++)

**for** (**int** j = 0; j<lunghezzaMaxNome; j++)

**if** ((nomi[i].toLowerCase().substring(0, j).equalsIgnoreCase(nomi[i+1].toLowerCase().substring(0, j)))&&(((**int**) nomi[i].toLowerCase().charAt(j))<((**int**) nomi[i+1].toLowerCase().charAt(j)))) {

stringa01 = nomi[i];

stringa02 = nomi[i+1];

nomi[i] = stringa02;

nomi[i+1] = stringa01;

effettuatoScambio = **true**;

}

} **while** (effettuatoScambio==**true**);

}

}

/\*

// Metodo privato statico per la ricerca delle parole "EXIT" o "ESCI" o delle lettere "EE", scritte in maiuscolo o minuscolo, in una stringa di testo dopo sono presenti degli spazi.

private static boolean esci(String stringaInput) {

int j = 0;

boolean condizioneUscitaProgramma = false;

boolean condizioneCicloDoWhile = false;

boolean trovatoPrimoCarattereDiversoDaSpaziatura = false;

boolean trovatoSpaziaturaDopoCarattere = false;

do {

condizioneCicloDoWhile = true;

// Questa condizione si verifica quando l'utente inserisce una stringa vuota, e termina subito l'esecuzione del metodo restituendo per la variabile booleana "condizioneUscita" un valore pari a FALSE.

if (stringaInput.length()==0)

condizioneCicloDoWhile = false;

else

for (int i = 0; i< stringaInput.length(); i++) {

// Queste condizioni, eliminano gli eventuali spazi che (nella stringa) precedono la parola ricercata. Appena viene trovato un carattere diverso da quello di spaziatura, viene assegnata alla variabile "trovatoPrimoCarattereDiversoDaspaziatura" un valore booleano pari a TRUE, causando il salto di queste istruzioni di selezione nelle successive iterazioni del ciclo FOR.

if ((trovatoPrimoCarattereDiversoDaSpaziatura==false)&&(stringaInput.charAt(i)!=' ')) {

trovatoPrimoCarattereDiversoDaSpaziatura = true;

break;

}

if ((trovatoPrimoCarattereDiversoDaSpaziatura==false)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

break;

}

// Questa condizione si verifica quando (dopo aver eliminato gli eventuali caratteri di spaziatura che precedono la parola da ricercare) compare una sequenza di più caratteri consecutivi che formano una parola. Vengono contate le lettere della parola fino a quando non viene trovato (eventualmente) un altro carattere di spaziatura.

if ((trovatoPrimoCarattereDiversoDaSpaziatura==true )&&(trovatoSpaziaturaDopoCarattere==false)&&(stringaInput.charAt(i)!=' '))

j++;

else

trovatoSpaziaturaDopoCarattere = true;

// Queste condizioni (dopo aver spostato la prima parola nella prima parte della stringa) verificano che non vi siano caratteri diversi da quello di spaziatura (che vengolo eliminati) dopo la parola.

if ((trovatoSpaziaturaDopoCarattere==true)&&(i==j)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

break;

} else if ((trovatoSpaziaturaDopoCarattere==true)&&(i==j)&&(stringaInput.charAt(i)!=' ')) {

condizioneCicloDoWhile = false;

break;

}

// Questa condizione si verifica quando (dopo aver isolato nella stringa solo la prima parola o lettera trovata e aver eliminato gli eventuali spazi) l'indice "i" del ciclo FOR giunge all'ultima lettera della parola (posizione dell'ultima lettera -1). Il metodo verifica che la parola o la lettera rimanente nella stringa corrisponda alle parole ricercate "EXIT" o "E" o "ESCI" e in caso affermativo assegna alla variabile booleana "condizioneUscitaProgramma" un valore pari a TRUE.

if (i==stringaInput.length()-1) {

stringaInput = stringaInput.toUpperCase();

if ((stringaInput.equalsIgnoreCase("EE"))||stringaInput.equalsIgnoreCase("EXIT")||(stringaInput.equalsIgnoreCase("ESCI"))) {

condizioneUscitaProgramma = true;

condizioneCicloDoWhile = false;

} else

condizioneCicloDoWhile = false;

}

}

} while (condizioneCicloDoWhile==true);

return condizioneUscitaProgramma;

}

// Metodo privato statico per la ricerca delle parole "RESTART" o "RIAVVIA" o delle lettere "RR", scritte in maiuscolo o minuscolo, in una stringa di testo dopo sono presenti degli spazi.

private static boolean riavvia(String stringaInput) {

int j = 0;

boolean condizioneRiavvioProgramma = false;

boolean condizioneCicloDoWhile = false;

boolean trovatoPrimoCarattereDiversoDaSpaziatura = false;

boolean trovatoSpaziaturaDopoCarattere = false;

do {

condizioneCicloDoWhile = true;

// Questa condizione si verifica quando l'utente inserisce una stringa vuota, e termina subito l'esecuzione del metodo restituendo per la variabile booleana "condizioneUscita" un valore pari a FALSE.

if (stringaInput.length()==0)

condizioneCicloDoWhile = false;

else

for (int i = 0; i< stringaInput.length(); i++) {

// Queste condizioni, eliminano gli eventuali spazi che (nella stringa) precedono la parola ricercata. Appena viene trovato un carattere diverso da quello di spaziatura, viene assegnata alla variabile "trovatoPrimoCarattereDiversoDaspaziatura" un valore booleano pari a TRUE, causando il salto di queste istruzioni di selezione nelle successive iterazioni del ciclo FOR.

if ((trovatoPrimoCarattereDiversoDaSpaziatura==false)&&(stringaInput.charAt(i)!=' ')) {

trovatoPrimoCarattereDiversoDaSpaziatura = true;

break;

}

if ((trovatoPrimoCarattereDiversoDaSpaziatura==false)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

break;

}

// Questa condizione si verifica quando (dopo aver eliminato gli eventuali caratteri di spaziatura che precedono la parola da ricercare) compare una sequenza di più caratteri consecutivi che formano una parola. Vengono contate le lettere della parola fino a quando non viene trovato (eventualmente) un altro carattere di spaziatura.

if ((trovatoPrimoCarattereDiversoDaSpaziatura==true )&&(trovatoSpaziaturaDopoCarattere==false)&&(stringaInput.charAt(i)!=' '))

j++;

else

trovatoSpaziaturaDopoCarattere = true;

// Queste condizioni (dopo aver spostato la prima parola nella prima parte della stringa) verificano che non vi siano caratteri diversi da quello di spaziatura (che vengolo eliminati) dopo la parola.

if ((trovatoSpaziaturaDopoCarattere==true)&&(i==j)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

break;

} else if ((trovatoSpaziaturaDopoCarattere==true)&&(i==j)&&(stringaInput.charAt(i)!=' ')) {

condizioneCicloDoWhile = false;

break;

}

// Questa condizione si verifica quando (dopo aver isolato nella stringa solo la prima parola o lettera trovata e aver eliminato gli eventuali spazi) l'indice "i" del ciclo FOR giunge all'ultima lettera della parola (posizione dell'ultima lettera -1). Il metodo verifica che la parola o la lettera rimanente nella stringa corrisponda alle parole ricercate "RESTART" o "R" o "RIAVVIA" e in caso affermativo assegna alla variabile booleana "condizioneRiavvioProgramma" un valore pari a TRUE.

if (i==stringaInput.length()-1) {

stringaInput = stringaInput.toUpperCase();

if ((stringaInput.equalsIgnoreCase("RR"))||stringaInput.equalsIgnoreCase("RESTART")||(stringaInput.equalsIgnoreCase("RIAVVIA"))) {

condizioneRiavvioProgramma = true;

condizioneCicloDoWhile = false;

} else

condizioneCicloDoWhile = false;

}

}

} while (condizioneCicloDoWhile==true);

return condizioneRiavvioProgramma;

}

// Metodo privato statico per la ricerca della parola "cancella\_Campo", in una stringa di testo dopo sono presenti degli spazi.

private static boolean cancella\_Campo(String stringaInput) {

int j = 0;

boolean condizioneCancellaCampo = false;

boolean condizioneCicloDoWhile = false;

boolean trovatoPrimoCarattereDiversoDaSpaziatura = false;

boolean trovatoSpaziaturaDopoCarattere = false;

do {

condizioneCicloDoWhile = true;

// Questa condizione si verifica quando l'utente inserisce una stringa vuota, e termina subito l'esecuzione del metodo restituendo per la variabile booleana "condizioneUscita" un valore pari a FALSE.

if (stringaInput.length()==0)

condizioneCicloDoWhile = false;

else

for (int i = 0; i< stringaInput.length(); i++) {

// Queste condizioni, eliminano gli eventuali spazi che (nella stringa) precedono la parola ricercata. Appena viene trovato un carattere diverso da quello di spaziatura, viene assegnata alla variabile "trovatoPrimoCarattereDiversoDaspaziatura" un valore booleano pari a TRUE, causando il salto di queste istruzioni di selezione nelle successive iterazioni del ciclo FOR.

if ((trovatoPrimoCarattereDiversoDaSpaziatura==false)&&(stringaInput.charAt(i)!=' ')) {

trovatoPrimoCarattereDiversoDaSpaziatura = true;

break;

}

if ((trovatoPrimoCarattereDiversoDaSpaziatura==false)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

break;

}

// Questa condizione si verifica quando (dopo aver eliminato gli eventuali caratteri di spaziatura che precedono la parola da ricercare) compare una sequenza di più caratteri consecutivi che formano una parola. Vengono contate le lettere della parola fino a quando non viene trovato (eventualmente) un altro carattere di spaziatura.

if ((trovatoPrimoCarattereDiversoDaSpaziatura==true )&&(trovatoSpaziaturaDopoCarattere==false)&&(stringaInput.charAt(i)!=' '))

j++;

else

trovatoSpaziaturaDopoCarattere = true;

// Queste condizioni (dopo aver spostato la prima parola nella prima parte della stringa) verificano che non vi siano caratteri diversi da quello di spaziatura (che vengolo eliminati) dopo la parola.

if ((trovatoSpaziaturaDopoCarattere==true)&&(i==j)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

break;

} else if ((trovatoSpaziaturaDopoCarattere==true)&&(i==j)&&(stringaInput.charAt(i)!=' ')) {

condizioneCicloDoWhile = false;

break;

}

// Questa condizione si verifica quando (dopo aver isolato nella stringa solo la prima parola o lettera trovata e aver eliminato gli eventuali spazi) l'indice "i" del ciclo FOR giunge all'ultima lettera della parola (posizione dell'ultima lettera -1). Il metodo verifica che la parola o la lettera rimanente nella stringa corrisponda alla parola ricercata "cancella\_Campo" e in caso affermativo assegna alla variabile booleana "condizioneCancellaCampo" un valore pari a TRUE.

if (i==stringaInput.length()-1) {

if (stringaInput.equals("cancella\_Campo")) {

condizioneCancellaCampo = true;

condizioneCicloDoWhile = false;

} else

condizioneCicloDoWhile = false;

}

}

} while (condizioneCicloDoWhile==true);

return condizioneCancellaCampo;

}

// Metodo privato statico per la stampa a video.

private static void stampa(String stringaInput) {

System.out.print(stringaInput);

}

// Metodo privato statico per l'eliminazione degli spazi in una stringa di testo.

private static String eliminaSpazi(String stringaInput) {

boolean trovatoCarattereDiSpaziatura = false;

do {

trovatoCarattereDiSpaziatura = false;

for (int i = 0; i<stringaInput.length(); i++) {

if (stringaInput.charAt(i)==' ') {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring((i+1), stringaInput.length());

trovatoCarattereDiSpaziatura = true;

break;

}

}

} while (trovatoCarattereDiSpaziatura==true);

return stringaInput;

}

// Metodo privato statico per l'eliminazione degli spazi ridondanti in una stringa di testo. Elimina gli spazi doppi, e quelli all'inizio e alla fine di una stringa di testo dove è presente almeno un carattere diverso da quello di spaziatura.

private static String eliminaSpaziInEccesso(String stringaInput) {

boolean trovatoCarattereDiSpaziatura = false;

boolean trovatoCarattereDiversoDaSpazio = false;

do {

trovatoCarattereDiSpaziatura = false;

for (int i = 0; i<stringaInput.length()-1; i++)

if ((stringaInput.charAt(i)==' ')&&(stringaInput.charAt(i+1)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1, stringaInput.length());

trovatoCarattereDiSpaziatura = true;

break;

}

} while (trovatoCarattereDiSpaziatura==true);

do {

trovatoCarattereDiSpaziatura = false;

for (int i = 0; i<stringaInput.length(); i++) {

// Questa condizione si verifica quando vengono trovati dei caratteri di spaziatura all'inizio di una stringa di testo e li elimina.

if ((trovatoCarattereDiversoDaSpazio==false)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1, stringaInput.length());

trovatoCarattereDiSpaziatura = true;

break;

// Questa condizione si verifica quando viene trovato per la prima volta un carattere diverso da quello di spaziatura.

} else if ((trovatoCarattereDiversoDaSpazio==false)&&(stringaInput.charAt(0)!=' ')) {

trovatoCarattereDiversoDaSpazio = true;

// Questa condizione si verifica quando viene trovato un carattere di spaziatura alla fine di una stringa di testo e lo elimina.

} else if (stringaInput.charAt(stringaInput.length()-1)==' ') {

stringaInput = stringaInput.substring(0, stringaInput.length()-1);

trovatoCarattereDiSpaziatura = true;

break;

}

}

} while (trovatoCarattereDiSpaziatura==true);

return stringaInput;

}

// Metodo privato statico per la ricerca di un solo numero INTERO in una stringa di testo dove sono presenti degli spazi.

private static boolean trovaNumeroIntero (String stringaInput) {

int j = 0;

boolean trovatoUnNumeroIntero = false;

boolean trovatoUnCarattereNonConsentito = false;

boolean trovatoSpazioA = false;

boolean trovatoSpazioB = false;

boolean trovataUnaCifra = false;

boolean trovatoUnPrimoCarattereConsentito = false;

boolean trovatoUnoSpazioDopoUnCarattereValido = false;

do {

trovatoSpazioA = false;

trovatoSpazioB = false;

for (int i = 0; i<stringaInput.length(); i++) {

// Questa condizione verifica che non vengano mai inseriti caratteri non validi (diversi dalla cifre da 0 a 9, dal segno negativo (-) o positivo (+) e dai caratteri di spaziatura ( )).

if ((trovatoUnoSpazioDopoUnCarattereValido==false)&&(trovatoUnPrimoCarattereConsentito==false)&&((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='+')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9'))) {

trovatoUnCarattereNonConsentito = true;

trovatoUnNumeroIntero = false;

break;

// Questa condizione verifica che non vengano inseriti altri segni negativi (-) o positivi (+) dopo il primo carattere valido. In caso contrario, il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroIntero" e termina il ciclo iterativo FOR.

} else if ((trovatoUnoSpazioDopoUnCarattereValido==false)&&(trovatoUnPrimoCarattereConsentito==true)&&((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9'))) {

trovatoUnCarattereNonConsentito = true;

trovatoUnNumeroIntero = false;

break;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato come primo carattere valido un segno negativo (-) o positivo (+).

} else if ((trovatoUnoSpazioDopoUnCarattereValido==false)&&(trovatoUnPrimoCarattereConsentito==false)&&((stringaInput.charAt(i)=='-')||(stringaInput.charAt(i)=='+'))) {

trovatoUnPrimoCarattereConsentito = true;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato come carattere valido una cifra da 0 a 9. Questo ramo della selezione serve per verificare che nella stringa vi sia almeno un cifra.

} else if ((trovatoUnoSpazioDopoUnCarattereValido==false)&&(trovataUnaCifra==false)&&((stringaInput.charAt(i)=='0')||(stringaInput.charAt(i)=='1')||(stringaInput.charAt(i)=='2')||(stringaInput.charAt(i)=='3')||(stringaInput.charAt(i)=='4')||(stringaInput.charAt(i)=='5')||(stringaInput.charAt(i)=='6')||(stringaInput.charAt(i)=='7')||(stringaInput.charAt(i)=='8')||(stringaInput.charAt(i)=='9'))) {

trovataUnaCifra = true;

trovatoUnNumeroIntero = true;

trovatoUnPrimoCarattereConsentito = true;

// Questa condizione si verifica eventualmente tutte le volte che viene trovato un carattere di spaziatura prima di un carattere valido e lo elimina.

} else if ((trovatoUnPrimoCarattereConsentito==false)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1, stringaInput.length());

trovatoSpazioA = true;

break;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato un carattere di spaziatura dopo un carattere valido e lo elimina. Viene settata la variabile "j" = i e la variabile "trovatoUnoSpazioDopoUnCarattereValido" = TRUE per bypassare l'elaborazione dei primi 4 rami della struttura di selezione semplice del metodo.

} else if ((stringaInput.charAt(i)==' ')&&(j==0)) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1, stringaInput.length());

trovatoSpazioB = true;

trovatoUnoSpazioDopoUnCarattereValido = true;

j = i;

break;

// Questa condizione si verifica eventualmente tutte le volte che viene trovato un secondo carattere di spaziatura dopo un carattere valido ed elimina tutti gli spazi dopo il numero intero.

} else if ((stringaInput.charAt(i)==' ')&&(i==j)) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1, stringaInput.length());

trovatoSpazioB = true;

break;

// Questa condizione si verifica quando viene trovato un carattere diverso da quello di spaziatura dopo il numero intero. Il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroDecimale" e interrompe il ciclo iterativo FOR.

} else if ((stringaInput.charAt(i)!=' ')&&(i==j)) {

trovatoUnCarattereNonConsentito = true;

trovatoUnNumeroIntero = false;

break;

// Questa condizione si verifica se viene trovata almeno una cifra nella stringa di testo. In caso contrario il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroDecimale" e interrompe il ciclo iterativo FOR.

} else if (trovataUnaCifra!=true) {

trovatoUnNumeroIntero = false;

}

}

} while ((trovatoUnCarattereNonConsentito==false)&&((trovatoSpazioA==true)||(trovatoSpazioB==true)));

return trovatoUnNumeroIntero;

}

// Metodo privato statico per la ricerca di un solo numero INTERO o DECIMALE in una stringa di testo dove sono presenti degli spazi.

private static boolean trovaNumeroDecimale(String stringaInput) {

int j = 0;

boolean trovatoUnNumeroInteroDecimale = false;

boolean trovatoUnCarattereNonConsentito = false;

boolean trovatoSpazioA = false;

boolean trovatoSpazioB = false;

boolean trovatoUnPuntoRadice = false;

boolean trovataUnaCifra = false;

boolean trovatoUnPrimoCarattereConsentito = false;

boolean trovatoUnoSpazioDopoUnCarattereValido = false;

do {

trovatoSpazioA = false;

trovatoSpazioB = false;

for (int i = 0; i<stringaInput.length(); i++) {

// Questa condizione verifica che non vengano inseriti caratteri non validi (diversi dalle cifre da 0 a 9, dal segno negativo (-) e positivo (+), dal punto radice (.) e dal carattere di spaziatura( )).

if ((trovatoUnoSpazioDopoUnCarattereValido==false)&&(trovatoUnPrimoCarattereConsentito==false)&&((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='+')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9'))) {

trovatoUnCarattereNonConsentito = true;

trovatoUnNumeroInteroDecimale = false;

break;

// Questa condizione si verifica quando viene trovato un segno negativo (-) o positivo (+) dopo un primo carattere valido; il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroDecimale", interrompendo il ciclo FOR.

} else if ((trovatoUnoSpazioDopoUnCarattereValido==false)&&(trovatoUnPrimoCarattereConsentito==true)&&((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9'))) {

trovatoUnCarattereNonConsentito = true;

trovatoUnNumeroInteroDecimale = false;

break;

// Questa condizione si verifica quando viene trovato un secondo punto radice nel numero decimale; il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroDecimale", interrompendo il ciclo FOR.

} else if ((trovatoUnoSpazioDopoUnCarattereValido==false)&&(trovatoUnPrimoCarattereConsentito==true)&&(trovatoUnPuntoRadice==true)&&((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9'))) {

trovatoUnCarattereNonConsentito = true;

trovatoUnNumeroInteroDecimale = false;

break;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato come primo carattere valido un segno negativo (-) o positivo (+).

} else if ((trovatoUnoSpazioDopoUnCarattereValido==false)&&(trovatoUnPrimoCarattereConsentito==false)&&((stringaInput.charAt(i)=='-')||(stringaInput.charAt(i)=='+'))) {

trovatoUnPrimoCarattereConsentito = true;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato come primo carattere valido un punto radice (.).

} else if ((trovatoUnoSpazioDopoUnCarattereValido==false)&&(trovatoUnPuntoRadice==false)&&(stringaInput.charAt(i)=='.')) {

trovatoUnPuntoRadice = true;

trovatoUnPrimoCarattereConsentito = true;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato come carattere valido una cifra da 0 a 9. Questo ramo della selezione serve per verificare che nella stringa vi sia almeno un cifra.

} else if ((trovatoUnoSpazioDopoUnCarattereValido==false)&&(trovataUnaCifra==false)&&((stringaInput.charAt(i)=='0')||(stringaInput.charAt(i)=='1')||(stringaInput.charAt(i)=='2')||(stringaInput.charAt(i)=='3')||(stringaInput.charAt(i)=='4')||(stringaInput.charAt(i)=='5')||(stringaInput.charAt(i)=='6')||(stringaInput.charAt(i)=='7')||(stringaInput.charAt(i)=='8')||(stringaInput.charAt(i)=='9'))) {

trovataUnaCifra = true;

trovatoUnNumeroInteroDecimale = true;

trovatoUnPrimoCarattereConsentito = true;

// Questa condizione si verifica eventualmente tutte le volte che viene trovato un carattere di spaziatura prima di un carattere valido e lo elimina.

} else if ((trovatoUnPrimoCarattereConsentito==false)&&(stringaInput.charAt(i)==' ')) {

stringaInput = stringaInput.substring(i+1, stringaInput.length());

trovatoSpazioA = true;

break;

// Questa condizione si verifica eventualmente solo una volta, appena viene trovato un carattere di spaziatura dopo un carattere valido e lo elimina. Viene settata la variabile "j" = i e la variabile "trovatoUnoSpazioDopoUnCarattereValido" = TRUE per bypassare l'elaborazione dei primi 6 rami di selezione del metodo.

} else if ((stringaInput.charAt(i)==' ')&&(j==0)) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1, stringaInput.length());

trovatoSpazioB = true;

trovatoUnoSpazioDopoUnCarattereValido = true;

j = i;

break;

// Questa condizione si verifica eventualmente tutte le volte che viene trovato un secondo carattere di spaziatura dopo un carattere valido ed elimina tutti gli spazi dopo il numero intero o decimale.

} else if ((stringaInput.charAt(i)==' ')&&(i==j)) {

stringaInput = stringaInput.substring(0, i) + stringaInput.substring(i+1,stringaInput.length());

trovatoSpazioB = true;

break;

// Questa condizione si verifica quando viene trovato un carattere diverso da quello di spaziatura dopo il numero decimale. Il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroDecimale" e interrompe il ciclo iterativo FOR.

} else if ((stringaInput.charAt(i)!=' ')&&(i==j)) {

trovatoUnCarattereNonConsentito = true;

trovatoUnNumeroInteroDecimale = false;

break;

// Questa condizione si verifica se viene trovata almeno una cifra nella stringa di testo. In caso contrario il programma restituisce un valore booleano FALSE per la variabile "trovatoUnNumeroDecimale" e interrompe il ciclo iterativo FOR.

} else if (trovataUnaCifra!=true) {

trovatoUnNumeroInteroDecimale = false;

}

}

} while ((trovatoUnCarattereNonConsentito==false)&&((trovatoSpazioA==true)||(trovatoSpazioB==true)));

return trovatoUnNumeroInteroDecimale;

}

// Metodo privato statico per la verifica di un stringa di testo del campo "cognome" / "nome" / "nome istituto scolastico" / "località", riferito a una persona / una scuola / un paese (o città)... Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti. Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito.

private static boolean verificaCampoNome(String stringaInput) {

boolean esitoVerifica = true;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='\'')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='à')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='è')&&(stringaInput.charAt(i)!='é')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='ì')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='ò')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='ù')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='À')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='È')&&(stringaInput.charAt(i)!='É')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='Ì')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='Ò')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='Ù')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

esitoVerifica = false;

break;

}

return esitoVerifica;

}

// Metodo privato statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "cognome" / "nome" / "nome istituto scolastico" / "località", riferito a una persona / una scuola / un paese (o città)... Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito.

private static String restituisciCaratteriNonConsentitiCampoNome(String stringaInput) {

String elencoCaratteriNonConsentiti = new String("");

boolean trovatoUnCarattereNonConsentito = false;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='\'')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='à')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='è')&&(stringaInput.charAt(i)!='é')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='ì')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='ò')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='ù')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='À')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='È')&&(stringaInput.charAt(i)!='É')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='Ì')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='Ò')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='Ù')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

elencoCaratteriNonConsentiti += stringaInput.charAt(i) + " ";

trovatoUnCarattereNonConsentito = true;

}

if (trovatoUnCarattereNonConsentito==true)

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

return elencoCaratteriNonConsentiti;

}

// Metodo privato statico per la verifica di un stringa di testo del campo "classe" di una scuola. Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti. Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito.

private static boolean verificaCampoClasseScuola(String stringaInput) {

boolean esitoVerifica = true;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='ª')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

esitoVerifica = false;

break;

}

return esitoVerifica;

}

// Metodo privato statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "classe" di una scuola. Questo metodo estituisce una stringa vuota se non viene trovato alcun carattere non consentito.

private static String restituisciCaratteriNonConsentitiCampoClasseScuola(String stringaInput) {

String elencoCaratteriNonConsentiti = new String("");

boolean trovatoUnCarattereNonConsentito = false;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='ª')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

elencoCaratteriNonConsentiti += stringaInput.charAt(i) + " ";

trovatoUnCarattereNonConsentito = true;

}

if (trovatoUnCarattereNonConsentito==true)

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

return elencoCaratteriNonConsentiti;

}

// Metodo privato statico per la verifica di un stringa di testo del campo "codiceAlfanumerico" di una persona o articolo. Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti. Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito.

private static boolean verificaCampoCodiceAlfanumerico (String stringaInput) {

boolean esitoVerifica = true;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!='\_')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

esitoVerifica = false;

break;

}

return esitoVerifica;

}

// Metodo privato statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "codiceAlfanumerico" di una persona o articolo. Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito.

private static String restituisciCaratteriNonConsentitiCampoCodiceAlfanumerico(String stringaInput) {

String elencoCaratteriNonConsentiti = new String("");

ArrayList<String> caratteriNonConsentiti = new ArrayList<String>();

boolean trovatoUnCarattereNonConsentito = false;

boolean eliminatoUnCarattereDoppio = false;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!='\_')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

caratteriNonConsentiti.add(stringaInput.charAt(i) + " ");

trovatoUnCarattereNonConsentito = true;

}

if (trovatoUnCarattereNonConsentito==true) {

do {

eliminatoUnCarattereDoppio = false;

etichetta:

for (int i = 0; i<caratteriNonConsentiti.size(); i++)

for (int j = 0; j<caratteriNonConsentiti.size(); j++)

if ((i!=j)&&(caratteriNonConsentiti.get(i).equals(caratteriNonConsentiti.get(j)))) {

caratteriNonConsentiti.remove(j);

eliminatoUnCarattereDoppio = true;

break etichetta;

}

} while (eliminatoUnCarattereDoppio==true);

for (int i = 0; i<caratteriNonConsentiti.size(); i++)

if (caratteriNonConsentiti.get(i).equals(" "))

elencoCaratteriNonConsentiti += "carattereDiSpaziatura ";

else

elencoCaratteriNonConsentiti += caratteriNonConsentiti.get(i) + " ";

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

}

return elencoCaratteriNonConsentiti;

}

// Metodo privato statico per la per la verifica di una stringa di testo del campo "indirizzo" di una persona o abitazione. Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti. Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito.

private static boolean verificaCampoIndirizzo(String stringaInput) {

boolean esitoVerifica = true;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='/')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='\'')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='à')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='è')&&(stringaInput.charAt(i)!='é')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='ì')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='ò')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='ù')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='À')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='È')&&(stringaInput.charAt(i)!='É')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='Ì')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='Ò')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='Ù')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

esitoVerifica = false;

break;

}

return esitoVerifica;

}

// Metodo privato statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "indirizzo" di una persona o abitazione. Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito.

private static String restituisciCaratteriNonConsentitiCampoIndirizzo(String stringaInput) {

String elencoCaratteriNonConsentiti = new String("");

boolean trovatoUnCarattereNonConsentito = false;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='/')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='\'')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='à')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='è')&&(stringaInput.charAt(i)!='é')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='ì')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='ò')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='ù')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='À')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='È')&&(stringaInput.charAt(i)!='É')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='Ì')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='Ò')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='Ù')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

elencoCaratteriNonConsentiti += stringaInput.charAt(i) + " ";

trovatoUnCarattereNonConsentito = true;

}

if (trovatoUnCarattereNonConsentito==true)

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

return elencoCaratteriNonConsentiti;

}

// Metodo privato statico per la verifica di una stringa di testo del campo "data" nel formato 'GG.MM.AAAA'. Questo metodo estituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti. Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito o se viene trovata una data errata del tipo '41.13.2019' ma scritta con caratteri consentiti.

private static boolean verificaCampoData(String stringaInput) {

boolean esitoVerifica = true;

stringaInput = eliminaSpaziInEccesso(stringaInput);

if (stringaInput.length()!=10)

esitoVerifica = false;

else {

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='/')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')) {

esitoVerifica = false;

break;

}

if (esitoVerifica==true) {

if ((stringaInput.charAt(0)!='0')&&(stringaInput.charAt(0)!='1')&&(stringaInput.charAt(0)!='2')&&(stringaInput.charAt(0)!='3'))

esitoVerifica = false;

if ((stringaInput.charAt(1)!='0')&&(stringaInput.charAt(1)!='1')&&(stringaInput.charAt(1)!='2')&&(stringaInput.charAt(1)!='3')&&(stringaInput.charAt(1)!='4')&&(stringaInput.charAt(1)!='5')&&(stringaInput.charAt(1)!='6')&&(stringaInput.charAt(1)!='7')&&(stringaInput.charAt(1)!='8')&&(stringaInput.charAt(1)!='9'))

esitoVerifica = false;

if ((stringaInput.charAt(2)!='.')&&(stringaInput.charAt(2)!='/'))

esitoVerifica = false;

if ((stringaInput.charAt(3)!='0')&&(stringaInput.charAt(3)!='1'))

esitoVerifica = false;

if ((stringaInput.charAt(4)!='0')&&(stringaInput.charAt(4)!='1')&&(stringaInput.charAt(4)!='2')&&(stringaInput.charAt(4)!='3')&&(stringaInput.charAt(4)!='4')&&(stringaInput.charAt(4)!='5')&&(stringaInput.charAt(4)!='6')&&(stringaInput.charAt(4)!='7')&&(stringaInput.charAt(4)!='8')&&(stringaInput.charAt(4)!='9'))

esitoVerifica = false;

if ((stringaInput.charAt(5)!='.')&&(stringaInput.charAt(5)!='/'))

esitoVerifica = false;

if ((stringaInput.charAt(6)!='0')&&(stringaInput.charAt(6)!='1')&&(stringaInput.charAt(6)!='2')&&(stringaInput.charAt(6)!='3')&&(stringaInput.charAt(6)!='4')&&(stringaInput.charAt(6)!='5')&&(stringaInput.charAt(6)!='6')&&(stringaInput.charAt(6)!='7')&&(stringaInput.charAt(6)!='8')&&(stringaInput.charAt(6)!='9'))

esitoVerifica = false;

if ((stringaInput.charAt(7)!='0')&&(stringaInput.charAt(7)!='1')&&(stringaInput.charAt(7)!='2')&&(stringaInput.charAt(7)!='3')&&(stringaInput.charAt(7)!='4')&&(stringaInput.charAt(7)!='5')&&(stringaInput.charAt(7)!='6')&&(stringaInput.charAt(7)!='7')&&(stringaInput.charAt(7)!='8')&&(stringaInput.charAt(7)!='9'))

esitoVerifica = false;

if ((stringaInput.charAt(8)!='0')&&(stringaInput.charAt(8)!='1')&&(stringaInput.charAt(8)!='2')&&(stringaInput.charAt(8)!='3')&&(stringaInput.charAt(8)!='4')&&(stringaInput.charAt(8)!='5')&&(stringaInput.charAt(8)!='6')&&(stringaInput.charAt(8)!='7')&&(stringaInput.charAt(8)!='8')&&(stringaInput.charAt(8)!='9'))

esitoVerifica = false;

if ((stringaInput.charAt(9)!='0')&&(stringaInput.charAt(9)!='1')&&(stringaInput.charAt(9)!='2')&&(stringaInput.charAt(9)!='3')&&(stringaInput.charAt(9)!='4')&&(stringaInput.charAt(9)!='5')&&(stringaInput.charAt(9)!='6')&&(stringaInput.charAt(9)!='7')&&(stringaInput.charAt(9)!='8')&&(stringaInput.charAt(9)!='9'))

esitoVerifica = false;

if ((stringaInput.charAt(2)=='.')&&(stringaInput.charAt(5)!='.'))

esitoVerifica = false;

if ((stringaInput.charAt(2)=='/')&&(stringaInput.charAt(5)!='/'))

esitoVerifica = false;

}

}

return esitoVerifica;

}

// Metodo privato statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "data" nel formato'GG.MM.AAAA'. Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito, o se viene trovata una data errata del tipo '41.13.2019' ma scritta con caratteri consentiti.

private static String restituisciCaratteriNonConsentitiCampoData(String stringaInput) {

String elencoCaratteriNonConsentiti = new String("");

ArrayList<String> caratteriNonConsentiti = new ArrayList<String>();

boolean trovatoUnCarattereNonConsentito = false;

boolean eliminatoUnCarattereDoppio = false;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='/')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')) {

caratteriNonConsentiti.add(stringaInput.charAt(i) + " ");

trovatoUnCarattereNonConsentito = true;

}

if (trovatoUnCarattereNonConsentito==true) {

do {

eliminatoUnCarattereDoppio = false;

etichetta:

for (int i = 0; i<caratteriNonConsentiti.size(); i++)

for (int j = 0; j<caratteriNonConsentiti.size(); j++)

if ((i!=j)&&(caratteriNonConsentiti.get(i).equals(caratteriNonConsentiti.get(j)))) {

caratteriNonConsentiti.remove(j);

eliminatoUnCarattereDoppio = true;

break etichetta;

}

} while (eliminatoUnCarattereDoppio==true);

for (int i = 0; i<caratteriNonConsentiti.size(); i++)

if (caratteriNonConsentiti.get(i).equals(" "))

elencoCaratteriNonConsentiti += "carattereDiSpaziatura ";

else

elencoCaratteriNonConsentiti += caratteriNonConsentiti.get(i) + " ";

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

}

return elencoCaratteriNonConsentiti;

}

// Metodo privato statico per la verifica di una stringa di testo del campo "provincia" in sigla (2 lettere tranne Roma). Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti. Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito o se viene trovata una provincia non esistente del tipo 'ZO' ma scritta con caratteri consentiti.

private static boolean verificaCampoSiglaProvincia(String stringaInput) {

String elencoProvincieItaliane[][] = new String[110][3];

elencoProvincieItaliane[0][0] = "AG";

elencoProvincieItaliane[0][1] = "Agrigento";

elencoProvincieItaliane[0][2] = "Sicilia";

elencoProvincieItaliane[1][0] = "AL";

elencoProvincieItaliane[1][1] = "Alessandria";

elencoProvincieItaliane[1][2] = "Piemonte";

elencoProvincieItaliane[2][0] = "AN";

elencoProvincieItaliane[2][1] = "Ancona";

elencoProvincieItaliane[2][2] = "Marche";

elencoProvincieItaliane[3][0] = "AO";

elencoProvincieItaliane[3][1] = "Aosta";

elencoProvincieItaliane[3][2] = "Valle d'Aosta";

elencoProvincieItaliane[4][0] = "AQ";

elencoProvincieItaliane[4][1] = "L'Aquila";

elencoProvincieItaliane[4][2] = "Abruzzo";

elencoProvincieItaliane[5][0] = "AR";

elencoProvincieItaliane[5][1] = "Arezzo";

elencoProvincieItaliane[5][2] = "Toscana";

elencoProvincieItaliane[6][0] = "AP";

elencoProvincieItaliane[6][1] = "Ascoli Piceno";

elencoProvincieItaliane[6][2] = "Marche";

elencoProvincieItaliane[7][0] = "AT";

elencoProvincieItaliane[7][1] = "Asti";

elencoProvincieItaliane[7][2] = "Piemonte";

elencoProvincieItaliane[8][0] = "AV";

elencoProvincieItaliane[8][1] = "Avellino";

elencoProvincieItaliane[8][2] = "Campania";

elencoProvincieItaliane[9][0] = "BA";

elencoProvincieItaliane[9][1] = "Bari";

elencoProvincieItaliane[9][2] = "Puglia";

elencoProvincieItaliane[10][0] = "BT";

elencoProvincieItaliane[10][1] = "Barletta Andria Trani";

elencoProvincieItaliane[10][2] = "Puglia";

elencoProvincieItaliane[11][0] = "BL";

elencoProvincieItaliane[11][1] = "Belluno";

elencoProvincieItaliane[11][2] = "Veneto";

elencoProvincieItaliane[12][0] = "BN";

elencoProvincieItaliane[12][1] = "Benevento";

elencoProvincieItaliane[12][2] = "Campania";

elencoProvincieItaliane[13][0] = "BG";

elencoProvincieItaliane[13][1] = "Bergamo";

elencoProvincieItaliane[13][2] = "Lombardia";

elencoProvincieItaliane[14][0] = "BI";

elencoProvincieItaliane[14][1] = "Biella";

elencoProvincieItaliane[14][2] = "Piemonte";

elencoProvincieItaliane[15][0] = "BO";

elencoProvincieItaliane[15][1] = "Bologna";

elencoProvincieItaliane[15][2] = "Emilia Romagna";

elencoProvincieItaliane[16][0] = "BZ";

elencoProvincieItaliane[16][1] = "Bolzano";

elencoProvincieItaliane[16][2] = "Trentino Alto Adige";

elencoProvincieItaliane[17][0] = "BS";

elencoProvincieItaliane[17][1] = "Brescia";

elencoProvincieItaliane[17][2] = "Lombardia";

elencoProvincieItaliane[18][0] = "BR";

elencoProvincieItaliane[18][1] = "Brindisi";

elencoProvincieItaliane[18][2] = "Puglia";

elencoProvincieItaliane[19][0] = "CA";

elencoProvincieItaliane[19][1] = "Cagliari";

elencoProvincieItaliane[19][2] = "Sardegna";

elencoProvincieItaliane[20][0] = "CL";

elencoProvincieItaliane[20][1] = "Caltanissetta";

elencoProvincieItaliane[20][2] = "Sicilia";

elencoProvincieItaliane[21][0] = "CB";

elencoProvincieItaliane[21][1] = "Campobasso";

elencoProvincieItaliane[21][2] = "Molise";

elencoProvincieItaliane[22][0] = "CI";

elencoProvincieItaliane[22][1] = "Carbonia Iglesias";

elencoProvincieItaliane[22][2] = "Sardegna";

elencoProvincieItaliane[23][0] = "CE";

elencoProvincieItaliane[23][1] = "Caserta";

elencoProvincieItaliane[23][2] = "Campania";

elencoProvincieItaliane[24][0] = "CT";

elencoProvincieItaliane[24][1] = "Catania";

elencoProvincieItaliane[24][2] = "Sicilia";

elencoProvincieItaliane[25][0] = "CZ";

elencoProvincieItaliane[25][1] = "Catanzaro";

elencoProvincieItaliane[25][2] = "Calabria";

elencoProvincieItaliane[26][0] = "CH";

elencoProvincieItaliane[26][1] = "Chieti";

elencoProvincieItaliane[26][2] = "Abruzzo";

elencoProvincieItaliane[27][0] = "CO";

elencoProvincieItaliane[27][1] = "Como";

elencoProvincieItaliane[27][2] = "Lombardia";

elencoProvincieItaliane[28][0] = "CS";

elencoProvincieItaliane[28][1] = "Cosenza";

elencoProvincieItaliane[28][2] = "Calabria";

elencoProvincieItaliane[29][0] = "CR";

elencoProvincieItaliane[29][1] = "Cremona";

elencoProvincieItaliane[29][2] = "Lombardia";

elencoProvincieItaliane[30][0] = "KR";

elencoProvincieItaliane[30][1] = "Crotone";

elencoProvincieItaliane[30][2] = "Calabria";

elencoProvincieItaliane[31][0] = "CN";

elencoProvincieItaliane[31][1] = "Cuneo";

elencoProvincieItaliane[31][2] = "Piemonte";

elencoProvincieItaliane[32][0] = "EN";

elencoProvincieItaliane[32][1] = "Enna";

elencoProvincieItaliane[32][2] = "Sicilia";

elencoProvincieItaliane[33][0] = "FM";

elencoProvincieItaliane[33][1] = "Fermo";

elencoProvincieItaliane[33][2] = "Marche";

elencoProvincieItaliane[34][0] = "FE";

elencoProvincieItaliane[34][1] = "Ferrara";

elencoProvincieItaliane[34][2] = "Emilia Romagna";

elencoProvincieItaliane[35][0] = "FI";

elencoProvincieItaliane[35][1] = "Firenze";

elencoProvincieItaliane[35][2] = "Toscana";

elencoProvincieItaliane[36][0] = "FG";

elencoProvincieItaliane[36][1] = "Foggia";

elencoProvincieItaliane[36][2] = "Puglia";

elencoProvincieItaliane[37][0] = "FC";

elencoProvincieItaliane[37][1] = "Forlì Cesena";

elencoProvincieItaliane[37][2] = "Emilia Romagna";

elencoProvincieItaliane[38][0] = "FR";

elencoProvincieItaliane[38][1] = "Frosinone";

elencoProvincieItaliane[38][2] = "Lazio";

elencoProvincieItaliane[39][0] = "GE";

elencoProvincieItaliane[39][1] = "Genova";

elencoProvincieItaliane[39][2] = "Liguria";

elencoProvincieItaliane[40][0] = "GO";

elencoProvincieItaliane[40][1] = "Gorizia";

elencoProvincieItaliane[40][2] = "Friuli Venezia Giulia";

elencoProvincieItaliane[41][0] = "GR";

elencoProvincieItaliane[41][1] = "Grosseto";

elencoProvincieItaliane[41][2] = "Toscana";

elencoProvincieItaliane[42][0] = "IM";

elencoProvincieItaliane[42][1] = "Imperia";

elencoProvincieItaliane[42][2] = "Liguria";

elencoProvincieItaliane[43][0] = "IS";

elencoProvincieItaliane[43][1] = "Isernia";

elencoProvincieItaliane[43][2] = "Molise";

elencoProvincieItaliane[44][0] = "SP";

elencoProvincieItaliane[44][1] = "La Spezia";

elencoProvincieItaliane[44][2] = "Liguria";

elencoProvincieItaliane[45][0] = "LT";

elencoProvincieItaliane[45][1] = "Latina";

elencoProvincieItaliane[45][2] = "Lazio";

elencoProvincieItaliane[46][0] = "LE";

elencoProvincieItaliane[46][1] = "Lecce";

elencoProvincieItaliane[46][2] = "Puglia";

elencoProvincieItaliane[47][0] = "LC";

elencoProvincieItaliane[47][1] = "Lecco";

elencoProvincieItaliane[47][2] = "Lombardia";

elencoProvincieItaliane[48][0] = "LI";

elencoProvincieItaliane[48][1] = "Livorno";

elencoProvincieItaliane[48][2] = "Toscana";

elencoProvincieItaliane[49][0] = "LO";

elencoProvincieItaliane[49][1] = "Lodi";

elencoProvincieItaliane[49][2] = "Lombardia";

elencoProvincieItaliane[50][0] = "LU";

elencoProvincieItaliane[50][1] = "Lucca";

elencoProvincieItaliane[50][2] = "Toscana";

elencoProvincieItaliane[51][0] = "MC";

elencoProvincieItaliane[51][1] = "Macerata";

elencoProvincieItaliane[51][2] = "Marche";

elencoProvincieItaliane[52][0] = "MN";

elencoProvincieItaliane[52][1] = "Mantova";

elencoProvincieItaliane[52][2] = "Lombardia";

elencoProvincieItaliane[53][0] = "MS";

elencoProvincieItaliane[53][1] = "Massa Carrara";

elencoProvincieItaliane[53][2] = "Toscana";

elencoProvincieItaliane[54][0] = "MT";

elencoProvincieItaliane[54][1] = "Matera";

elencoProvincieItaliane[54][2] = "Basilicata";

elencoProvincieItaliane[55][0] = "VS";

elencoProvincieItaliane[55][1] = "Medio Campidano";

elencoProvincieItaliane[55][2] = "Sardegna";

elencoProvincieItaliane[56][0] = "ME";

elencoProvincieItaliane[56][1] = "Messina";

elencoProvincieItaliane[56][2] = "Sicilia";

elencoProvincieItaliane[57][0] = "MI";

elencoProvincieItaliane[57][1] = "Milano";

elencoProvincieItaliane[57][2] = "Lombardia";

elencoProvincieItaliane[58][0] = "MO";

elencoProvincieItaliane[58][1] = "Modena";

elencoProvincieItaliane[58][2] = "Emilia Romagna";

elencoProvincieItaliane[59][0] = "MB";

elencoProvincieItaliane[59][1] = "Monza Brianza";

elencoProvincieItaliane[59][2] = "Lombardia";

elencoProvincieItaliane[60][0] = "NA";

elencoProvincieItaliane[60][1] = "Napoli";

elencoProvincieItaliane[60][2] = "Campania";

elencoProvincieItaliane[61][0] = "NO";

elencoProvincieItaliane[61][1] = "Novara";

elencoProvincieItaliane[61][2] = "Piemonte";

elencoProvincieItaliane[62][0] = "NU";

elencoProvincieItaliane[62][1] = "Nuoro";

elencoProvincieItaliane[62][2] = "Sardegna";

elencoProvincieItaliane[63][0] = "OG";

elencoProvincieItaliane[63][1] = "Ogliastra";

elencoProvincieItaliane[63][2] = "Sardegna";

elencoProvincieItaliane[64][0] = "OT";

elencoProvincieItaliane[64][1] = "Olbia Tempio";

elencoProvincieItaliane[64][2] = "Sardegna";

elencoProvincieItaliane[65][0] = "OR";

elencoProvincieItaliane[65][1] = "Oristano";

elencoProvincieItaliane[65][2] = "Sardegna";

elencoProvincieItaliane[66][0] = "PD";

elencoProvincieItaliane[66][1] = "Padova";

elencoProvincieItaliane[66][2] = "Veneto";

elencoProvincieItaliane[67][0] = "PA";

elencoProvincieItaliane[67][1] = "Palermo";

elencoProvincieItaliane[67][2] = "Sicilia";

elencoProvincieItaliane[68][0] = "PR";

elencoProvincieItaliane[68][1] = "Parma";

elencoProvincieItaliane[68][2] = "Emilia Romagna";

elencoProvincieItaliane[69][0] = "PV";

elencoProvincieItaliane[69][1] = "Pavia";

elencoProvincieItaliane[69][2] = "Lombardia";

elencoProvincieItaliane[70][0] = "PG";

elencoProvincieItaliane[70][1] = "Perugia";

elencoProvincieItaliane[70][2] = "Umbria";

elencoProvincieItaliane[71][0] = "PU";

elencoProvincieItaliane[71][1] = "Pesaro Urbino";

elencoProvincieItaliane[71][2] = "Marche";

elencoProvincieItaliane[72][0] = "PE";

elencoProvincieItaliane[72][1] = "Pescara";

elencoProvincieItaliane[72][2] = "Abruzzo";

elencoProvincieItaliane[73][0] = "PC";

elencoProvincieItaliane[73][1] = "Piacenza";

elencoProvincieItaliane[73][2] = "Emilia Romagna";

elencoProvincieItaliane[74][0] = "PI";

elencoProvincieItaliane[74][1] = "Pisa";

elencoProvincieItaliane[74][2] = "Toscana";

elencoProvincieItaliane[75][0] = "PT";

elencoProvincieItaliane[75][1] = "Pistoia";

elencoProvincieItaliane[75][2] = "Toscana";

elencoProvincieItaliane[76][0] = "PN";

elencoProvincieItaliane[76][1] = "Pordenone";

elencoProvincieItaliane[76][2] = "Friuli Venezia Giulia";

elencoProvincieItaliane[77][0] = "PZ";

elencoProvincieItaliane[77][1] = "Potenza";

elencoProvincieItaliane[77][2] = "Basilicata";

elencoProvincieItaliane[78][0] = "PO";

elencoProvincieItaliane[78][1] = "Prato";

elencoProvincieItaliane[78][2] = "Toscana";

elencoProvincieItaliane[79][0] = "RG";

elencoProvincieItaliane[79][1] = "Ragusa";

elencoProvincieItaliane[79][2] = "Sicilia";

elencoProvincieItaliane[80][0] = "RA";

elencoProvincieItaliane[80][1] = "Ravenna";

elencoProvincieItaliane[80][2] = "Emilia Romagna";

elencoProvincieItaliane[81][0] = "RC";

elencoProvincieItaliane[81][1] = "Reggio Calabria";

elencoProvincieItaliane[81][2] = "Calabria";

elencoProvincieItaliane[82][0] = "RE";

elencoProvincieItaliane[82][1] = "Reggio Emilia";

elencoProvincieItaliane[82][2] = "Emilia Romagna";

elencoProvincieItaliane[83][0] = "RI";

elencoProvincieItaliane[83][1] = "Rieti";

elencoProvincieItaliane[83][2] = "Lazio";

elencoProvincieItaliane[84][0] = "RN";

elencoProvincieItaliane[84][1] = "Rimini";

elencoProvincieItaliane[84][2] = "Emilia Romagna";

elencoProvincieItaliane[85][0] = "Roma";

elencoProvincieItaliane[85][1] = "Roma";

elencoProvincieItaliane[85][2] = "Lazio";

elencoProvincieItaliane[86][0] = "RO";

elencoProvincieItaliane[86][1] = "Rovigo";

elencoProvincieItaliane[86][2] = "Veneto";

elencoProvincieItaliane[87][0] = "SA";

elencoProvincieItaliane[87][1] = "Salerno";

elencoProvincieItaliane[87][2] = "Campania";

elencoProvincieItaliane[88][0] = "SS";

elencoProvincieItaliane[88][1] = "Sassari";

elencoProvincieItaliane[88][2] = "Sardegna";

elencoProvincieItaliane[89][0] = "SV";

elencoProvincieItaliane[89][1] = "Savona";

elencoProvincieItaliane[89][2] = "Liguria";

elencoProvincieItaliane[90][0] = "SI";

elencoProvincieItaliane[90][1] = "Siena";

elencoProvincieItaliane[90][2] = "Toscana";

elencoProvincieItaliane[91][0] = "SR";

elencoProvincieItaliane[91][1] = "Siracusa";

elencoProvincieItaliane[91][2] = "Sicilia";

elencoProvincieItaliane[92][0] = "SO";

elencoProvincieItaliane[92][1] = "Sondrio";

elencoProvincieItaliane[92][2] = "Lombardia";

elencoProvincieItaliane[93][0] = "TA";

elencoProvincieItaliane[93][1] = "Taranto";

elencoProvincieItaliane[93][2] = "Puglia";

elencoProvincieItaliane[94][0] = "TE";

elencoProvincieItaliane[94][1] = "Teramo";

elencoProvincieItaliane[94][2] = "Abruzzo";

elencoProvincieItaliane[95][0] = "TR";

elencoProvincieItaliane[95][1] = "Terni";

elencoProvincieItaliane[95][2] = "Umbria";

elencoProvincieItaliane[96][0] = "TO";

elencoProvincieItaliane[96][1] = "Torino";

elencoProvincieItaliane[96][2] = "Piemonte";

elencoProvincieItaliane[97][0] = "TP";

elencoProvincieItaliane[97][1] = "Trapani";

elencoProvincieItaliane[97][2] = "Sicilia";

elencoProvincieItaliane[98][0] = "TN";

elencoProvincieItaliane[98][1] = "Trento";

elencoProvincieItaliane[98][2] = "Trentino Alto Adige";

elencoProvincieItaliane[99][0] = "TV";

elencoProvincieItaliane[99][1] = "Treviso";

elencoProvincieItaliane[99][2] = "Veneto";

elencoProvincieItaliane[100][0] = "TS";

elencoProvincieItaliane[100][1] = "Trieste";

elencoProvincieItaliane[100][2] = "Friuli Venezia Giulia";

elencoProvincieItaliane[101][0] = "UD";

elencoProvincieItaliane[101][1] = "Udine";

elencoProvincieItaliane[101][2] = "Friuli Venezia Giulia";

elencoProvincieItaliane[102][0] = "VA";

elencoProvincieItaliane[102][1] = "Varese";

elencoProvincieItaliane[102][2] = "Lombardia";

elencoProvincieItaliane[103][0] = "VE";

elencoProvincieItaliane[103][1] = "Venezia";

elencoProvincieItaliane[103][2] = "Veneto";

elencoProvincieItaliane[104][0] = "VB";

elencoProvincieItaliane[104][1] = "Verbania";

elencoProvincieItaliane[104][2] = "Piemonte";

elencoProvincieItaliane[105][0] = "VC";

elencoProvincieItaliane[105][1] = "Vercelli";

elencoProvincieItaliane[105][2] = "Piemonte";

elencoProvincieItaliane[106][0] = "VR";

elencoProvincieItaliane[106][1] = "Verona";

elencoProvincieItaliane[106][2] = "Veneto";

elencoProvincieItaliane[107][0] = "VV";

elencoProvincieItaliane[107][1] = "Vibo Valentia";

elencoProvincieItaliane[107][2] = "Calabria";

elencoProvincieItaliane[108][0] = "VI";

elencoProvincieItaliane[108][1] = "Vicenza";

elencoProvincieItaliane[108][2] = "Veneto";

elencoProvincieItaliane[109][0] = "VT";

elencoProvincieItaliane[109][1] = "Viterbo";

elencoProvincieItaliane[109][2] = "Lazio";

boolean esitoVerifica = true;

boolean trovataSiglaProvincia = false;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='Z')) {

esitoVerifica = false;

break;

}

if (esitoVerifica==true) {

for (int i = 0; i<elencoProvincieItaliane.length; i++)

if (stringaInput.equalsIgnoreCase(elencoProvincieItaliane[i][0])) {

trovataSiglaProvincia = true;

break;

}

if (trovataSiglaProvincia==false)

esitoVerifica = false;

}

return esitoVerifica;

}

// Metodo privato statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "provincia" in sigla (2 lettere maiuscole tranne Roma). Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito, o se viene trovata una provincia inesistente del tipo 'ZO' ma scritta con caratteri consentiti.

private static String restituisciCaratteriNonConsentitiCampoSiglaProvincia(String stringaInput) {

String elencoCaratteriNonConsentiti = new String("");

ArrayList<String> caratteriNonConsentiti = new ArrayList<String>();

boolean trovatoUnCarattereNonConsentito = false;

boolean eliminatoUnCarattereDoppio = false;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='Z')) {

caratteriNonConsentiti.add(stringaInput.charAt(i) + " ");

trovatoUnCarattereNonConsentito = true;

}

if (trovatoUnCarattereNonConsentito==true) {

do {

eliminatoUnCarattereDoppio = false;

etichetta:

for (int i = 0; i<caratteriNonConsentiti.size(); i++)

for (int j = 0; j<caratteriNonConsentiti.size(); j++)

if ((i!=j)&&(caratteriNonConsentiti.get(i).equals(caratteriNonConsentiti.get(j)))) {

caratteriNonConsentiti.remove(j);

eliminatoUnCarattereDoppio = true;

break etichetta;

}

} while (eliminatoUnCarattereDoppio==true);

for (int i = 0; i<caratteriNonConsentiti.size(); i++)

if (caratteriNonConsentiti.get(i).equals(" "))

elencoCaratteriNonConsentiti += "carattereDiSpaziatura ";

else

elencoCaratteriNonConsentiti += caratteriNonConsentiti.get(i) + " ";

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

}

return elencoCaratteriNonConsentiti;

}

// Metodo privato statico per la verifica di una stringa di testo per un campo numerico, come, a esempio, il numero di telefono di una persona. Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti. Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito.

private static boolean verificaCampoNumerico(String stringaInput) {

boolean esitoVerifica = true;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')) {

esitoVerifica = false;

break;

}

return esitoVerifica;

}

// Metodo privato statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo per un campo numerico, come, a esempio, il numero di telefono di una persona. Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito.

private static String restituisciCaratteriNonConsentitiCampoNumerico(String stringaInput) {

String elencoCaratteriNonConsentiti = new String("");

ArrayList<String> caratteriNonConsentiti = new ArrayList<String>();

boolean trovatoUnCarattereNonConsentito = false;

boolean eliminatoUnCarattereDoppio = false;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')) {

caratteriNonConsentiti.add(stringaInput.charAt(i) + " ");

trovatoUnCarattereNonConsentito = true;

}

if (trovatoUnCarattereNonConsentito==true) {

do {

eliminatoUnCarattereDoppio = false;

etichetta:

for (int i = 0; i<caratteriNonConsentiti.size(); i++)

for (int j = 0; j<caratteriNonConsentiti.size(); j++)

if ((i!=j)&&(caratteriNonConsentiti.get(i).equals(caratteriNonConsentiti.get(j)))) {

caratteriNonConsentiti.remove(j);

eliminatoUnCarattereDoppio = true;

break etichetta;

}

} while (eliminatoUnCarattereDoppio==true);

for (int i = 0; i<caratteriNonConsentiti.size(); i++)

if (caratteriNonConsentiti.get(i).equals(" "))

elencoCaratteriNonConsentiti += "carattereDiSpaziatura ";

else

elencoCaratteriNonConsentiti += caratteriNonConsentiti.get(i) + " ";

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

}

return elencoCaratteriNonConsentiti;

}

// Metodo privato statico per la verifica di una stringa di testo del campo "indirizzo e-mail" di una persona o azienda. Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti. Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito o se viene trovato un indirizzo e-mail errato del tipo "marco@rossi@gmail.com" ma scritto con caratteri consentiti.

private static boolean verificaCampoIndirizzoE\_mail(String stringaInput) {

int numeroCaratteriChiocciola = 0;

boolean esitoVerifica = true;

stringaInput = eliminaSpaziInEccesso(stringaInput);

stringaInput = stringaInput.toLowerCase();

for (int i = 0; i<stringaInput.length(); i++) {

if ((stringaInput.charAt(i)!='@')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')) {

esitoVerifica = false;

break;

}

if (stringaInput.charAt(i)=='@')

numeroCaratteriChiocciola++;

}

if (numeroCaratteriChiocciola!=1)

esitoVerifica = false;

return esitoVerifica;

}

// Metodo privato statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "indirizzo e-mail" di una persona o azienda. Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito o se viene trovato un indirizzo e-mail errato del tipo "marco@rossi@gmail.com" ma scritto con caratteri consentiti.

private static String restituisciCaratteriNonConsentitiCampoIndirizzoE\_mail(String stringaInput) {

String elencoCaratteriNonConsentiti = new String("");

ArrayList<String> caratteriNonConsentiti = new ArrayList<String>();

boolean trovatoUnCarattereNonConsentito = false;

boolean eliminatoUnCarattereDoppio = false;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!='@')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')) {

caratteriNonConsentiti.add(stringaInput.charAt(i) + " ");

trovatoUnCarattereNonConsentito = true;

}

if (trovatoUnCarattereNonConsentito==true) {

do {

eliminatoUnCarattereDoppio = false;

etichetta:

for (int i = 0; i<caratteriNonConsentiti.size(); i++)

for (int j = 0; j<caratteriNonConsentiti.size(); j++)

if ((i!=j)&&(caratteriNonConsentiti.get(i).equals(caratteriNonConsentiti.get(j)))) {

caratteriNonConsentiti.remove(j);

eliminatoUnCarattereDoppio = true;

break etichetta;

}

} while (eliminatoUnCarattereDoppio==true);

for (int i = 0; i<caratteriNonConsentiti.size(); i++)

if (caratteriNonConsentiti.get(i).equals(" "))

elencoCaratteriNonConsentiti += "carattereDiSpaziatura ";

else

elencoCaratteriNonConsentiti += caratteriNonConsentiti.get(i) + " ";

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

}

return elencoCaratteriNonConsentiti;

}

// Metodo privato statico per la verifica di una stringa di testo del campo "testo generico" di una nota o una descrizione. Questo metodo restituisce un valore booleano pari a TRUE se nella stringa vengono trovati solo caratteri consentiti. Questo metodo restituisce un valore booleano pari a FALSE se nella stringa viene trovato almeno un carattere non consentito.

private static boolean verificaCampoTesto(String stringaInput) {

boolean esitoVerifica = true;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!='ª')&&(stringaInput.charAt(i)!='°')&&(stringaInput.charAt(i)!='@')&&(stringaInput.charAt(i)!='\_')&&(stringaInput.charAt(i)!='+')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='\*')&&(stringaInput.charAt(i)!='/')&&(stringaInput.charAt(i)!='^')&&(stringaInput.charAt(i)!='&')&&(stringaInput.charAt(i)!='%')&&(stringaInput.charAt(i)!='$')&&(stringaInput.charAt(i)!='£')&&(stringaInput.charAt(i)!='€')&&(stringaInput.charAt(i)!='!')&&(stringaInput.charAt(i)!='?')&&(stringaInput.charAt(i)!='\"')&&(stringaInput.charAt(i)!='\'')&&(stringaInput.charAt(i)!=',')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!=';')&&(stringaInput.charAt(i)!=':')&&(stringaInput.charAt(i)!='(')&&(stringaInput.charAt(i)!=')')&&(stringaInput.charAt(i)!='[')&&(stringaInput.charAt(i)!=']')&&(stringaInput.charAt(i)!='{')&&(stringaInput.charAt(i)!='}')&&(stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='à')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='è')&&(stringaInput.charAt(i)!='é')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='ì')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='ò')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='ù')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='À')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='È')&&(stringaInput.charAt(i)!='É')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='Ì')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='Ò')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='Ù')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

esitoVerifica = false;

break;

}

return esitoVerifica;

}

// Metodo privato statico per la restituzione degli eventuali caratteri non consentiti presenti in una stringa di testo del campo "testo generico" di una nota o una descrizione. Questo metodo restituisce una stringa vuota se non viene trovato alcun carattere non consentito.

private static String restituisciCaratteriNonConsentitiCampoTesto(String stringaInput) {

String elencoCaratteriNonConsentiti = new String("");

boolean trovatoUnCarattereNonConsentito = false;

stringaInput = eliminaSpaziInEccesso(stringaInput);

for (int i = 0; i<stringaInput.length(); i++)

if ((stringaInput.charAt(i)!='ª')&&(stringaInput.charAt(i)!='°')&&(stringaInput.charAt(i)!='@')&&(stringaInput.charAt(i)!='\_')&&(stringaInput.charAt(i)!='+')&&(stringaInput.charAt(i)!='-')&&(stringaInput.charAt(i)!='\*')&&(stringaInput.charAt(i)!='/')&&(stringaInput.charAt(i)!='^')&&(stringaInput.charAt(i)!='&')&&(stringaInput.charAt(i)!='%')&&(stringaInput.charAt(i)!='$')&&(stringaInput.charAt(i)!='£')&&(stringaInput.charAt(i)!='€')&&(stringaInput.charAt(i)!='!')&&(stringaInput.charAt(i)!='?')&&(stringaInput.charAt(i)!='\"')&&(stringaInput.charAt(i)!='\'')&&(stringaInput.charAt(i)!=',')&&(stringaInput.charAt(i)!='.')&&(stringaInput.charAt(i)!=';')&&(stringaInput.charAt(i)!=':')&&(stringaInput.charAt(i)!='(')&&(stringaInput.charAt(i)!=')')&&(stringaInput.charAt(i)!='[')&&(stringaInput.charAt(i)!=']')&&(stringaInput.charAt(i)!='{')&&(stringaInput.charAt(i)!='}')&&(stringaInput.charAt(i)!=' ')&&(stringaInput.charAt(i)!='0')&&(stringaInput.charAt(i)!='1')&&(stringaInput.charAt(i)!='2')&&(stringaInput.charAt(i)!='3')&&(stringaInput.charAt(i)!='4')&&(stringaInput.charAt(i)!='5')&&(stringaInput.charAt(i)!='6')&&(stringaInput.charAt(i)!='7')&&(stringaInput.charAt(i)!='8')&&(stringaInput.charAt(i)!='9')&&(stringaInput.charAt(i)!='a')&&(stringaInput.charAt(i)!='à')&&(stringaInput.charAt(i)!='b')&&(stringaInput.charAt(i)!='c')&&(stringaInput.charAt(i)!='d')&&(stringaInput.charAt(i)!='e')&&(stringaInput.charAt(i)!='è')&&(stringaInput.charAt(i)!='é')&&(stringaInput.charAt(i)!='f')&&(stringaInput.charAt(i)!='g')&&(stringaInput.charAt(i)!='h')&&(stringaInput.charAt(i)!='i')&&(stringaInput.charAt(i)!='ì')&&(stringaInput.charAt(i)!='j')&&(stringaInput.charAt(i)!='k')&&(stringaInput.charAt(i)!='l')&&(stringaInput.charAt(i)!='m')&&(stringaInput.charAt(i)!='n')&&(stringaInput.charAt(i)!='o')&&(stringaInput.charAt(i)!='ò')&&(stringaInput.charAt(i)!='p')&&(stringaInput.charAt(i)!='q')&&(stringaInput.charAt(i)!='r')&&(stringaInput.charAt(i)!='s')&&(stringaInput.charAt(i)!='t')&&(stringaInput.charAt(i)!='u')&&(stringaInput.charAt(i)!='ù')&&(stringaInput.charAt(i)!='v')&&(stringaInput.charAt(i)!='w')&&(stringaInput.charAt(i)!='x')&&(stringaInput.charAt(i)!='y')&&(stringaInput.charAt(i)!='z')&&(stringaInput.charAt(i)!='A')&&(stringaInput.charAt(i)!='À')&&(stringaInput.charAt(i)!='B')&&(stringaInput.charAt(i)!='C')&&(stringaInput.charAt(i)!='D')&&(stringaInput.charAt(i)!='E')&&(stringaInput.charAt(i)!='È')&&(stringaInput.charAt(i)!='É')&&(stringaInput.charAt(i)!='F')&&(stringaInput.charAt(i)!='G')&&(stringaInput.charAt(i)!='H')&&(stringaInput.charAt(i)!='I')&&(stringaInput.charAt(i)!='Ì')&&(stringaInput.charAt(i)!='J')&&(stringaInput.charAt(i)!='K')&&(stringaInput.charAt(i)!='L')&&(stringaInput.charAt(i)!='M')&&(stringaInput.charAt(i)!='N')&&(stringaInput.charAt(i)!='O')&&(stringaInput.charAt(i)!='Ò')&&(stringaInput.charAt(i)!='P')&&(stringaInput.charAt(i)!='Q')&&(stringaInput.charAt(i)!='R')&&(stringaInput.charAt(i)!='S')&&(stringaInput.charAt(i)!='T')&&(stringaInput.charAt(i)!='U')&&(stringaInput.charAt(i)!='Ù')&&(stringaInput.charAt(i)!='V')&&(stringaInput.charAt(i)!='W')&&(stringaInput.charAt(i)!='X')&&(stringaInput.charAt(i)!='Y')&&(stringaInput.charAt(i)!='Z')) {

elencoCaratteriNonConsentiti += stringaInput.charAt(i) + " ";

trovatoUnCarattereNonConsentito = true;

}

if (trovatoUnCarattereNonConsentito==true)

elencoCaratteriNonConsentiti = elencoCaratteriNonConsentiti.substring(0, elencoCaratteriNonConsentiti.length()-2);

return elencoCaratteriNonConsentiti;

}

// Metodo privato statico per l'ordinamento di una serie di stringhe (contenute in un array dinamico) dalla A alla Z.

private static void ordinaStringheAZ(ArrayList<String> nomi) {

String stringa01 = new String();

String stringa02 = new String();

int lunghezzaMaxNome = 0;

int variazioneLunghezzaNome = 0;

boolean effettuatoScambio = false;

// Questo ciclo FOR cerca all'interno dell'array dinamico "nomi" il nome con lunghezza maggiore (in caratteri) e assegna il valore numerico della lunghezza del nome alla variabile "lunghezzaMaxNome".

for (int i = 0; i<nomi.size(); i++)

if (nomi.get(i).length()>lunghezzaMaxNome)

lunghezzaMaxNome = nomi.get(i).length();

// Questo ciclo FOR aggiunge una serie di caratteri di spaziatura alla fine di ogni nome contenuto nell'array dinamico "nomi" in nodo da ottenere soltanto nomi della stessa lunghezza.

for (int i = 0; i<nomi.size(); i++)

if (nomi.get(i).length()<lunghezzaMaxNome) {

stringa01 = nomi.get(i);

variazioneLunghezzaNome = lunghezzaMaxNome-stringa01.length();

for (int j = 0; j<(variazioneLunghezzaNome); j++)

stringa01 += " ";

nomi.remove(i);

nomi.add(i, stringa01);

}

{ // Questo blocco di elaborazione ordina le stringhe all'interno dell'array dinamico "nomi" dalla A alla Z.

do {

effettuatoScambio = false;

for (int i = 0; i<nomi.size()-1; i++)

for (int j = 0; j<lunghezzaMaxNome; j++)

if ((nomi.get(i).toLowerCase().substring(0, j).equalsIgnoreCase(nomi.get(i+1).toLowerCase().substring(0, j)))&&(((int) nomi.get(i).toLowerCase().charAt(j))>((int) nomi.get(i+1).toLowerCase().charAt(j)))) {

stringa01 = nomi.get(i);

stringa02 = nomi.get(i+1);

nomi.remove(i);

nomi.add(i, stringa02);

nomi.remove(i+1);

nomi.add(i+1, stringa01);

effettuatoScambio = true;

}

} while (effettuatoScambio==true);

}

}

// Metodo privato statico per l'ordinamento di una serie di stringhe (contenute in un array dinamico) dalla Z alla A.

private static void ordinaStringheZA(ArrayList<String> nomi) {

String stringa01 = new String();

String stringa02 = new String();

int lunghezzaMaxNome = 0;

int variazioneLunghezzaNome = 0;

boolean effettuatoScambio = false;

// Questo ciclo FOR cerca all'interno dell'array dinamico "nomi" il nome con lunghezza maggiore (in caratteri) e assegna il valore numerico della lunghezza del nome alla variabile "lunghezzaMaxNome".

for (int i = 0; i<nomi.size(); i++)

if (nomi.get(i).length()>lunghezzaMaxNome)

lunghezzaMaxNome = nomi.get(i).length();

// Questo ciclo FOR aggiunge una serie di caratteri di spaziatura alla fine di ogni nome contenuto nell'array dinamico "nomi" in nodo da ottenere soltanto nomi della stessa lunghezza.

for (int i = 0; i<nomi.size(); i++)

if (nomi.get(i).length()<lunghezzaMaxNome) {

stringa01 = nomi.get(i);

variazioneLunghezzaNome = lunghezzaMaxNome-stringa01.length();

for (int j = 0; j<(variazioneLunghezzaNome); j++)

stringa01 += " ";

nomi.remove(i);

nomi.add(i, stringa01);

}

{ // Questo blocco di elaborazione ordina le stringhe all'interno dell'array dinamico "nomi" dalla Z alla A.

do {

effettuatoScambio = false;

for (int i = 0; i<nomi.size()-1; i++)

for (int j = 0; j<lunghezzaMaxNome; j++)

if ((nomi.get(i).toLowerCase().substring(0, j).equalsIgnoreCase(nomi.get(i+1).toLowerCase().substring(0, j)))&&(((int) nomi.get(i).toLowerCase().charAt(j))<((int) nomi.get(i+1).toLowerCase().charAt(j)))) {

stringa01 = nomi.get(i);

stringa02 = nomi.get(i+1);

nomi.remove(i);

nomi.add(i, stringa02);

nomi.remove(i+1);

nomi.add(i+1, stringa01);

effettuatoScambio = true;

}

} while (effettuatoScambio==true);

}

}

// Metodo private statico per l'ordinamento di una serie di stringhe (contenute in un array) dalla A alla Z.

private static void ordinaStringheAZ(String nomi[]) {

String stringa01 = new String();

String stringa02 = new String();

int lunghezzaMaxNome = 0;

int variazioneLunghezzaNome = 0;

boolean effettuatoScambio = false;

// Questo ciclo FOR cerca all'interno dell'array "nomi" il nome con lunghezza maggiore (in caratteri) e assegna il valore numerico della lunghezza del nome alla variabile "lunghezzaMaxNome".

for (int i = 0; i<nomi.length; i++)

if (nomi[i].length()>lunghezzaMaxNome)

lunghezzaMaxNome = nomi[i].length();

// Questo ciclo FOR aggiunge una serie di caratteri di spaziatura alla fine di ogni nome contenuto nell'array "nomi" in nodo da ottenere soltanto nomi della stessa lunghezza.

for (int i = 0; i<nomi.length; i++)

if (nomi[i].length()<lunghezzaMaxNome) {

stringa01 = nomi[i];

variazioneLunghezzaNome = lunghezzaMaxNome-stringa01.length();

for (int j = 0; j<(variazioneLunghezzaNome); j++)

stringa01 += " ";

nomi[i] = stringa01;

}

{ // Questo blocco di elaborazione ordina le stringhe all'interno dell'array "nomi" dalla A alla Z.

do {

effettuatoScambio = false;

for (int i = 0; i<nomi.length-1; i++)

for (int j = 0; j<lunghezzaMaxNome; j++)

if ((nomi[i].toLowerCase().substring(0, j).equalsIgnoreCase(nomi[i+1].toLowerCase().substring(0, j)))&&(((int) nomi[i].toLowerCase().charAt(j))>((int) nomi[i+1].toLowerCase().charAt(j)))) {

stringa01 = nomi[i];

stringa02 = nomi[i+1];

nomi[i] = stringa02;

nomi[i+1] = stringa01;

effettuatoScambio = true;

}

} while (effettuatoScambio==true);

}

}

// Metodo privato statico per l'ordinamento di una serie di stringhe (contenute in un array) dalla Z alla A.

private static void ordinaStringheZA(String nomi[]) {

String stringa01 = new String();

String stringa02 = new String();

int lunghezzaMaxNome = 0;

int variazioneLunghezzaNome = 0;

boolean effettuatoScambio = false;

// Questo ciclo FOR cerca all'interno dell'array "nomi" il nome con lunghezza maggiore (in caratteri) e assegna il valore numerico della lunghezza del nome alla variabile "lunghezzaMaxNome".

for (int i = 0; i<nomi.length; i++)

if (nomi[i].length()>lunghezzaMaxNome)

lunghezzaMaxNome = nomi[i].length();

// Questo ciclo FOR aggiunge una serie di caratteri di spaziatura alla fine di ogni nome contenuto nell'array "nomi" in nodo da ottenere soltanto nomi della stessa lunghezza.

for (int i = 0; i<nomi.length; i++)

if (nomi[i].length()<lunghezzaMaxNome) {

stringa01 = nomi[i];

variazioneLunghezzaNome = lunghezzaMaxNome-stringa01.length();

for (int j = 0; j<(variazioneLunghezzaNome); j++)

stringa01 += " ";

nomi[i] = stringa01;

}

{ // Questo blocco di elaborazione ordina le stringhe all'interno dell'array "nomi" dalla Z alla A.

do {

effettuatoScambio = false;

for (int i = 0; i<nomi.length-1; i++)

for (int j = 0; j<lunghezzaMaxNome; j++)

if ((nomi[i].toLowerCase().substring(0, j).equalsIgnoreCase(nomi[i+1].toLowerCase().substring(0, j)))&&(((int) nomi[i].toLowerCase().charAt(j))<((int) nomi[i+1].toLowerCase().charAt(j)))) {

stringa01 = nomi[i];

stringa02 = nomi[i+1];

nomi[i] = stringa02;

nomi[i+1] = stringa01;

effettuatoScambio = true;

}

} while (effettuatoScambio==true);

}

}

\*/

}

/\*

// Metodo privato statico per l'ordinamento degli oggeti all'interno di un array dinamico (ArrayList), dalla A alla Z, tenendo come riferimento la stringa contenuta nella variabile di istanza "campoOrdinamento", presente in ogni oggetto dell'array dinamico.

// ATENZIONE, IMPORTANTE: ogni oggetto contenuto nell'arrayDinamico, deve avere la variabile di istanza di tipo stringa "campoOrdinamento".

private static void ordinaOggettiArrayDinamicoA\_Z(ArrayList<> arrayDinamicoOggetti) {

String stringa01 = new String();

String stringa02 = new String();

int lunghezzaMassimaStringa = 0;

int variazioneLunghezzaStringa = 0;

boolean effettuatoScambio = false;

{ // Blocco di elaborazione A.

// Questo ciclo FOR cerca la variabile di istanza "campoOrdinamento" di lunghezza maggiore (in caratteri) in ogni singolo oggetto dell'array dinamico "arrayDinamicoOggetti" e ne assegna il valore numerico alla variabile "lunghezzaMassimaStringa".

for (int i = 0; i<arrayDinamicoOggetti.size(); i++) {

if (arrayDinamicoOggetti.get(i).campoOrdinamento().length()>lunghezzaMassimaStringa)

lunghezzaMassimaStringa = arrayDinamicoOggetti.get(i).campoOrdinamento().length();

}

// Questo ciclo FOR aggiunge dei caratteri di spaziatura al termine della stringa contenuta nella variabile di istanza "campoOrdinamento", icapsulata in ogni oggetto dell'array dinamico "arrayDinamicoOggetti", fino a ottenere variabili di istanza (in ogni singlo oggetto) con lo stesso numero di caratteri (matrice rettangolare).

for (int i = 0; i<arrayDinamicoOggetti.size(); i++)

if (arrayDinamicoOggetti.get(i).campoOrdinamento().length()<lunghezzaMassimaStringa) {

variazioneLunghezzaStringa = lunghezzaMassimaStringa - arrayDinamicoOggetti.get(i).campoOrdinamento().length();

stringa01 = arrayDinamicoOggetti.get(i).campoOrdinamento();

for (int j = 0; j<variazioneLunghezzaStringa; j++)

stringa01 += " ";

arrayDinamicoOggetti.get(i).campoOrdinamento(stringa01);

}

}

{ // Blocco di elaborazione B.

// Questo ciclo DO... WHILE viene eseguito fino a quando viene effettuato un cambio nella posizione degli oggetti all'interno dell'array dinamico "arrayDinamicoOggetti".

do {

effettuatoScambio = false;

// Questo ciclo FOR ordina ogni oggetto dell'array dinamico "arrayDinamicoOggetti" in senso alfabetico dalla A alla Z, in riferimento alla variabile di istanza "campoOrdinamento".

for (int i = 0; i<arrayDinamicoOggetti.size()-1; i++) {

stringa01 = arrayDinamicoOggetti.get(i).campoOrdinamento().toLowerCase();

stringa02 = arrayDinamicoOggetti.get(i+1).campoOrdinamento().toLowerCase();

for (int j = 0; j<lunghezzaMassimaStringa; j++) {

if ((stringa01.substring(0, j).equalsIgnoreCase(stringa02.substring(0, j)))&&(((int)stringa01.charAt(j))>((int)stringa02.charAt(j)))) {

effettuatoScambio = true;

arrayDinamicoOggetti.add(i, arrayDinamicoOggetti.get(i+1));

arrayDinamicoOggetti.remove(i+2);

}

}

}

} while (effettuatoScambio==true);

}

}

// Metodo privato statico per l'ordinamento degli oggeti all'interno di un array dinamico (ArrayList), dalla Z alla A, tenendo come riferimento la stringa contenuta nella variabile di istanza "campoOrdinamento", presente in ogni oggetto dell'array dinamico.

// ATENZIONE, IMPORTANTE: ogni oggetto contenuto nell'arrayDinamico, deve avere la variabile di istanza di tipo stringa "campoOrdinamento".

private static void ordinaOggettiArrayDinamicoZ\_A(ArrayList<> arrayDinamicoOggetti) {

String stringa01 = new String();

String stringa02 = new String();

int lunghezzaMassimaStringa = 0;

int variazioneLunghezzaStringa = 0;

boolean effettuatoScambio = false;

{ // Blocco di elaborazione A.

// Questo ciclo FOR cerca la variabile di istanza "campoOrdinamento" di lunghezza maggiore (in caratteri) in ogni singolo oggetto dell'array dinamico "arrayDinamicoOggetti" e ne assegna il valore numerico alla variabile "lunghezzaMassimaStringa".

for (int i = 0; i<arrayDinamicoOggetti.size(); i++) {

if (arrayDinamicoOggetti.get(i).campoOrdinamento().length()>lunghezzaMassimaStringa)

lunghezzaMassimaStringa = arrayDinamicoOggetti.get(i).campoOrdinamento().length();

}

// Questo ciclo FOR aggiunge dei caratteri di spaziatura al termine della stringa contenuta nella variabile di istanza "campoOrdinamento", icapsulata in ogni oggetto dell'array dinamico "arrayDinamicoOggetti", fino a ottenere variabili di istanza (in ogni singlo oggetto) con lo stesso numero di caratteri (matrice rettangolare).

for (int i = 0; i<arrayDinamicoOggetti.size(); i++)

if (arrayDinamicoOggetti.get(i).campoOrdinamento().length()<lunghezzaMassimaStringa) {

variazioneLunghezzaStringa = lunghezzaMassimaStringa - arrayDinamicoOggetti.get(i).campoOrdinamento().length();

stringa01 = arrayDinamicoOggetti.get(i).campoOrdinamento();

for (int j = 0; j<variazioneLunghezzaStringa; j++)

stringa01 += " ";

arrayDinamicoOggetti.get(i).campoOrdinamento(stringa01);

}

}

{ // Blocco di elaborazione B.

// Questo ciclo DO... WHILE viene eseguito fino a quando viene effettuato un cambio nella posizione degli oggetti all'interno dell'array dinamico "arrayDinamicoOggetti".

do {

effettuatoScambio = false;

// Questo ciclo FOR ordina ogni oggetto dell'array dinamico "arrayDinamicoOggetti" in senso alfabetico dalla A alla Z, in riferimento alla variabile di istanza "campoOrdinamento".

for (int i = 0; i<arrayDinamicoOggetti.size()-1; i++) {

stringa01 = arrayDinamicoOggetti.get(i).campoOrdinamento().toLowerCase();

stringa02 = arrayDinamicoOggetti.get(i+1).campoOrdinamento().toLowerCase();

for (int j = 0; j<lunghezzaMassimaStringa; j++) {

if ((stringa01.substring(0, j).equalsIgnoreCase(stringa02.substring(0, j)))&&(((int)stringa01.charAt(j))<((int)stringa02.charAt(j)))) {

effettuatoScambio = true;

arrayDinamicoOggetti.add(i, arrayDinamicoOggetti.get(i+1));

arrayDinamicoOggetti.remove(i+2);

}

}

}

} while (effettuatoScambio==true);

}

}

\*/