**import** java.io.\*;

**import** java.util.ArrayList;

/\*\*

 \* Questo programma rappresenta un tentativo di implementare un algoritmo per l'ordinamento alfabetico di una serie di stringhe (ad esempio i cognomi di un gruppo di persone).

 \* Questo tipo di ordinamento è già presente in numerosi software, come, a esempio, l'ordinamento numerico o alfabetico (dalla A alla Z) di una serie di celle nella colonna di un foglio di lavoro di EXCEL.

 \* Tenterò di implementare questo algoritmo senza aiuti, senza cercare soluzioni nei vari forum online.

 \* Questo algoritmo mi serve a completamento dell'esercizio EU5\_17 dove ho la necessità di ordinare una serie di oggetti contenuti in un array dinamico, tenendo come parametro di riferimento la variabile di istanza "cognome" (contenuta in ogni singolo oggetto dell'array dinamico).

 \* Questo array dinamico, rappresenta un record logico, e, al mio livello di studio, è utile per simulare un archivio dati, seppur realizzato solo nella RAM del computer (non è presente alcuna operazione di scrittura su file o in un database reale).

 \* **@author** Abramo Gerardo Giordano.

 \* **@version** 1.0 (10 Aprile 2019).

 \*/

**public** **class** **OrdinamentoAlfabeticoStringhe** {

 **public** **static** **void** main(**String** args[]) **throws** **IOException** {

 **String** stringaInput = **new** String();

 **InputStreamReader** lettoreFlussoIngresso = **new** InputStreamReader(**System**.***in***);

 **BufferedReader** tastiera = **new** BufferedReader(lettoreFlussoIngresso);

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

 **final** **int** lunghezzaCampoNome = 50; // Costante utilizzata per ddefinire la lunghezza massima (in caratteri) di un nome.

 **char** sceltaUtente = 'S';

 **boolean** condizioneCicloDoWhile = **false**; // Variabile utilizzata esclusivamnte nei cicli DO... WHILE per la verifica dei dati in ingresso.

 **boolean** condizioneNuovaElaborazioneProgramma = **false**; // Variabile utilizzata per l'esecuzione di una nuova elaborazione del progrmma senza riavviarlo.

 **boolean** condizioneUscitaProgramma = **false**; // Variabile utilizzata per terminare l'esecuzione del programma.

 **boolean** condizioneRiavvioProgramma = **false**; // Variabile utilizzata per il riavvio del programma.

 *stampa*("\n\n\n\n");

 // Punto di inizio per il riavvio del programma.

 **do** {

 condizioneRiavvioProgramma = **false**;

 *stampa*("\n\tProgramma per l'ordinamento alfabetico di una serie di stringhe contenute in un array dinamico (ArrayList).");

 *stampa*("\n\tScrivere \"ESCI\" oppure \"EE\" oppure \"EXIT\" in minuscolo o maiuscolo + INVIO, per uscire dal programma.");

 *stampa*("\n\tScrivere \"RIAVVIA\" oppure \"RR\" oppure \"RESTART\" in minuscolo o maiuscolo + INVIO, per riavviare il programma.");

 *stampa*("\n");

 // Punto di inizio per il riavvio del programma.

 **do** {

 condizioneNuovaElaborazioneProgramma = **false**;

 condizioneUscitaProgramma = **false**;

 **if** (condizioneUscitaProgramma==**false**) {

 costruttoFor: // Crazione di un'etichetta per la gestione di un eventuale salto del blocco di elaborazione mediante il comando BREAK.

 **for** ( ; sceltaUtente=='S'; ) {

 // Inizio del ciclo DO... WHILE per la verifica della stringa inserita dall'utente relativa al nome.

 **do** {

 condizioneCicloDoWhile = **true**;

 *stampa*("\tInserire un nome: ");

 stringaInput = tastiera.readLine();

 **if** (*esci*(stringaInput)==**true**) {

 condizioneCicloDoWhile = **false**;

 condizioneUscitaProgramma = **true**;

 *stampa*("\tTERMINE DELL'ESECUZIONE DEL PROGRAMMA.\n");

 **break** costruttoFor;

 } **else** **if** (*riavvia*(stringaInput)==**true**) {

 condizioneCicloDoWhile = **false**;

 condizioneUscitaProgramma = **true**;

 condizioneRiavvioProgramma = **true**;

 *stampa*("\tRIAVVIO DEL PROGRAMMA IN CORSO ...\n");

 **break** costruttoFor;

 } **else** **if** (*eliminaSpazi*(stringaInput).length()==0)

 *stampa*("\tATTENZIONE: non hai inserito alcun nome.\n");

 **else** **if** (*eliminaSpaziInEccesso*(stringaInput).length()>lunghezzaCampoNome)

 *stampa*("\tATTENZIONE: la lunghezza massima consentita per un nome è di " + lunghezzaCampoNome + " caratteri.\n");

 **else** **if** (*verificaCampoNome*(stringaInput)==**false**)

 *stampa*("\tATTENZIONE: sono stati trovati uno o più caratteri non consentiti: " + *restituisciCaratteriNonConsentitiCampoNome*(stringaInput) + "\n");

 **else** {

 condizioneCicloDoWhile = **false**;

 stringaInput = *eliminaSpaziInEccesso*(stringaInput);

 nomi.add(stringaInput);

 }

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

 // Inizio del ciclo DO... WHILE per la verifica dell'opzione scelta dall'utente relativa alla richiesta del programma di inserire un altro nome.

 **do** {

 condizioneCicloDoWhile = **true**;

 *stampa*("\tVuoi inserire un altro nome (S) o (N)? ");

 stringaInput = tastiera.readLine();

 **if** (*esci*(stringaInput)==**true**) {

 condizioneCicloDoWhile = **false**;

 condizioneUscitaProgramma = **true**;

 *stampa*("\tTERMINE DELL'ESECUZIONE DEL PROGRAMMA.\n");

 **break** costruttoFor;

 } **else** **if** (*riavvia*(stringaInput)==**true**) {

 condizioneCicloDoWhile = **false**;

 condizioneUscitaProgramma = **true**;

 condizioneRiavvioProgramma = **true**;

 *stampa*("\tRIAVVIO DEL PROGRAMMA IN CORSO ...\n");

 } **else** **if** (*eliminaSpazi*(stringaInput).length()==0)

 *stampa*("\tATTENZIONE: non hai scelto alcuna opzione.\n");

 **else** {

 stringaInput = *eliminaSpazi*(stringaInput);

 **if** (stringaInput.length()>1)

 *stampa*("\tATTENZIONE: hai scelto un'opzione non valida.\n");

 **else** {

 stringaInput = stringaInput.toUpperCase();

 sceltaUtente = stringaInput.charAt(0);

 **if** ((sceltaUtente!='S')&&(sceltaUtente!='N'))

 *stampa*("\tATTENZIONE: hai scelto un'opzione non in elenco.\n");

 **else** **if** (sceltaUtente!='S') {

 condizioneCicloDoWhile = **false**;

 **break**;

 } **else**

 condizioneCicloDoWhile = **false**;

 }

 }

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

 }

 }

 **if** (condizioneUscitaProgramma==**false**) {

 { // Inizio del blocco di elaborazione per l'ordinamento alfabetico dei nomi inseriti dall'utente.

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

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

 **int** lunghezzaMassimaNome = 0;

 **int** variazioneLunghezzaNome = 0;

 **boolean** effettuatoScambio = **false**;

 /\*

 Come sappiamo, una stringa non è altro che un insieme di caratteri, quindi può essere considerato come un array che contiene soltanto dati di tipo carattere (char).

 Da questo punto di vista, l'array dinamico "nomi" rappresenta un array di array, ovvero una matrice. Se i nomi inseriti dall'utente hanno lunghezza diversa, la matrice non è rettangolare.

 Per implementare un algoritmo che ordini in senso alfabetico (dalla A alla Z) tutti i nomi presenti nell'array dinamico di oggetti stringa "nomi", dobbiamo per prima cosa rendere la matrice rettangolare.

 Per fare questo aggiungiamo una serie di caratteri di spaziatura alla fine di ogni nome contenuto nell'array dinamico "nomi", in modo da ottenere soltanto nomi con la stessa lunghezza.

 \*/

 { // Blocco di elaborazione A.

 // 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 "lunghezzaMassimaNome".

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

 **if** (nomi.get(i).length()>lunghezzaMassimaNome)

 lunghezzaMassimaNome = 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()<lunghezzaMassimaNome) {

 stringaInput = nomi.get(i);

 variazioneLunghezzaNome = lunghezzaMassimaNome-stringaInput.length();

 **for** (**int** j = 0; j<(variazioneLunghezzaNome); j++)

 stringaInput += " ";

 nomi.remove(i);

 nomi.add(i, stringaInput);

 }

 }

/\* { // Blocco di elaborazione per la verifica dello stato dell'array dinamico "nomi" e degli oggetti stringa contenuti al suo interno.

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

 stampa("\n\t" + nomi.get(i));

 stampa("\n\n");

 } \*/

 /\*

 A questo punto dell'elaborazione, abbiamo una matrice rettangolare.

 Per ordinare i nomi in senso alfabetico, dalla A alla Z, dobbiamo partire dalla considerazione che a ogni lettera, ovvero a ogni variabile di tipo char, possiamo associare il corrispondente valore numerico ottenibile dalle tabelle dei cosici ASCII di tutte le lettere dell'alfabeto.

 Utilizzando l'operatore di cast "(int)" possiamo rivavare il valore numerico di ogni lettera.

 Quindi, l'algoritmo per l'ordinamento di una serie di nomi in un array, può essere ricondotto all'algoritmo per l'ordinamento di una serie di numeri interi in un array, ovvero all'algoritmo "BUBBLE SORT".

 \*/

 { // Blocco di elaborazione B.

 // Incominciamo a effettuare un primo ordinamento partendo dalla 1ª lettera di ogni stringa contenuta nell'array dinamico "nomi".

 // Per comodità trasformiamo tutti i nomi in minuscolo nella fase di confronto delle singole lettere.

 **do** {

 effettuatoScambio = **false**;

 **for** (**int** i = 0; i<nomi.size()-1; i++)

 **if** (((**int**) nomi.get(i).toLowerCase().charAt(0))>((**int**) nomi.get(i+1).toLowerCase().charAt(0))) {

 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**);

 }

/\* { // Blocco di elaborazione per la verifica dello stato dell'array dinamico "nomi" e degli oggetti stringa contenuti al suo interno.

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

 stampa("\n\t" + nomi.get(i));

 stampa("\n\n");

 } \*/

 /\*

 A questo punto dell'elaborazione, abbiamo ordinato i nome in senso alfabetico solo in riferimento alla 1ª lettera di ogni nome contenuto nell'array dinamico di oggetti stringa "nomi".

 Per ordinare i nome in riferimento anche alle lettere successive, dobbiamo effettuare un confronto anche sulla prima parte della stringa, ovvero:

 se ordiniamo i nomi in riferimento alla 2ª lettera, dobbiamo assicurarci che questi inizino con la stessa lettera (ordiniamo, in riferimento alla 2ª lettera, tutti i nomi che iniziano con 'a', tutti i nomi che iniziano con 'b', tutti i nomi che iniziano con 'c', ecc ...);

 se ordiniamo i nomi in riferimento alla 3ª lettera, dobbiamo assicurarci che questi abbiano le prime due lettere uguali (ordiniamo, in riferimento alla 3ª lettera, tutti i nomi che iniziano con 'aa', tutti i nomi che iniziano con 'ab', ...., tutti i nomi che iniziano con 'ba', tutti i nomi cha iniziano con 'bb', fino ai nomi che iniziano con 'zz').

 Quindi, nella fase di confronto fra il valore numerico di ogni lettera (ricordando sempre che le lettere, in fase di confronto, si trovano nella stessa posizione all'interno dei vari nomi), dobbiamo assicurarci che anche la prima parte della stringa resti uguale.

 È necessario, quindi, effettuare un'elaborazione più complessa utilizzando le sottostringe.

 \*/

 { // Blocco di elaborazione C.

 **do** {

 effettuatoScambio = **false**;

 **for** (**int** i = 0; i<nomi.size()-1; i++)

 **for** (**int** j = 1; j<lunghezzaMassimaNome; j++) // Poniamo j = 1, perchè l'ordinamento dei nomi in riferimento alla 1ª lettera (posizione 0 di ogni stringa) è gia stato effettuato.

 **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**);

 }

/\* { // Blocco di elaborazione per la verifica dello stato dell'array dinamico "nomi" e degli oggetti stringa contenuti al suo interno.

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

 stampa("\n\t" + nomi.get(i));

 stampa("\n\n");

 } \*/

 /\*

 I due blocci di elaborazione B e C, possono essere uniti in un unico blocco di elaborazione (riportato di seguito), con l'utilizzo delle sottostringhe e l'indice j del ciclo FOR annidato che parte da 0.

 \*/

 { // Blocco di elaborazione D.

 **do** {

 effettuatoScambio = **false**;

 **for** (**int** i = 0; i<nomi.size()-1; i++)

 **for** (**int** j = 0; j<lunghezzaMassimaNome; 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**);

 }

 /\*

 A questo punto dell'elaborazione non rimane altro che eliminare gli spazi in eccesso nei vari nomi contenuti nell'array dinamico "nomi", ottenendo così nuovamente una matrice non rettangolare.

 \*/

 { // Blocco di elaborazione E.

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

 stringaInput = nomi.get(i);

 stringaInput = *eliminaSpaziInEccesso*(stringaInput);

 nomi.remove(i);

 nomi.add(i, stringaInput);

 }

 }

/\* { // Blocco di elaborazione per la verifica dello stato dell'array dinamico "nomi" e degli oggetti stringa contenuti al suo interno.

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

 stampa("\n\t" + nomi.get(i));

 stampa("\n\n");

 } \*/

 /\*

 L'algoritmo è stato testato e funziona perfettamente.

 Questo algoritmo verrà inserito come nuovo metodo statico all'interno della mia classe "AGG" con passaggio di un array dinamico di oggetti stringa (e anche di un array normale di oggetti stringa) per riferimento come argomento d'ingresso del metodo, utilizzando il meccanismo dell'overloading.

 \*/

 }

 }

 **if** (condizioneUscitaProgramma==**false**) {

 { // Inizio del blocco di elaborazione per la stampa a video dei nomi ordinati in senso alfabetico.

 *stampa*("\n\tSTAMPA, IN ORDINE ALFABETICO, DI TUTTI I NOMI INSERITI DALL'UTENTE:");

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

 *stampa*("\n\t" + nomi.get(i));

 *stampa*("\n\n");

 }

 }

 **if** (condizioneUscitaProgramma==**false**) {

 // Inizio del ciclo DO... WHILE per la verifica della scelta effettuata dall'utente relativa alla richiesta del programma di eseguire una nuova elaborazione.

 **do** {

 condizioneCicloDoWhile = **true**;

 *stampa*("\tVuoi eseguire nuovamente il programma (S) o (N)? ");

 stringaInput = tastiera.readLine();

 **if** (*esci*(stringaInput)==**true**) {

 condizioneCicloDoWhile = **false**;

 condizioneUscitaProgramma = **true**;

 *stampa*("\tTERMINE DELL'ESECUZIONE DEL PROGRAMMA .\n");

 } **else** **if** (*riavvia*(stringaInput)==**true**) {

 condizioneCicloDoWhile = **false**;

 condizioneUscitaProgramma = **true**;

 condizioneRiavvioProgramma = **true**;

 *stampa*("\tRIAVVIO DEL PROGRAMMA IN CORSO ...\n");

 } **else** **if** (*eliminaSpazi*(stringaInput).length()==0)

 *stampa*("\tATTENZIONE: non hai scelto alcuna opzione.\n");

 **else** {

 stringaInput = *eliminaSpazi*(stringaInput);

 **if** (stringaInput.length()>1)

 *stampa*("\tATTENZIONE: hai scelto un'opzione non valida.\n");

 **else** {

 stringaInput = stringaInput.toUpperCase();

 sceltaUtente = stringaInput.charAt(0);

 **if** ((sceltaUtente!='S')&&(sceltaUtente!='N'))

 *stampa*("\tATTENZIONE: hai scelto un'opzione non in elenco.\n");

 **else** **if** (sceltaUtente=='S') {

 condizioneCicloDoWhile = **false**;

 condizioneNuovaElaborazioneProgramma = **true**;

 *stampa*("\n");

 } **else**

 condizioneCicloDoWhile = **false**;

 }

 }

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

 }

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

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

 *stampa*("\n\n\n\n");

 }

 // 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 "condizioneUscita" 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 "condizioneUscita" 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 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 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)!='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**;

 **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("");

 stringaInput = *eliminaSpaziInEccesso*(stringaInput);

 **boolean** trovatoCarattereNonValido = **false**;

 **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)!='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) + " ";

 trovatoCarattereNonValido = **true**;

 }

 **if** (trovatoCarattereNonValido==**true**)

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

 **return** elencoCaratteriNonConsentiti;

 }

}