

PROGETTO “IL DIZIONARIO”

IL PROBLEMA

L'obiettivo del progetto è quello di realizzare un dizionario e alcune operazioni in esso contenute.

Una *parola* è una sequenza finita di caratteri appartenenti all'alfabeto di 26 lettere minuscole $\{a, b, c, \dots, z\}$.
 Uno *schema* è una sequenza finita di caratteri appartenenti all'alfabeto $\{a, b, c, \dots, z\}$ e $\{A, B, C, \dots, X\}$ (l'alfabeto delle lettere minuscole e maiuscole).

Un *assegnamento* è una funzione $\sigma: \{A, B, C, \dots, Z\} \rightarrow \{a, b, c, \dots, z\}$ che associa ad ogni lettera maiuscola una lettera minuscola.

Dato uno schema $W = \alpha_1 \dots \alpha_n$ e un assegnamento σ denotiamo con $W(\sigma)$ la parola $w = \beta_1 \dots \beta_n$ tale che $\beta_i = \sigma(\alpha_i)$ se α_i è una lettera maiuscola e $\beta_i = \alpha_i$ se α_i è una lettera minuscola. Una parola w è un'istanza di uno schema W se esiste un assegnamento σ tale che $W(\sigma) = w$.
 Ad esempio la parola “acca” è un'istanza dello schema “aBBa”, “aBCa” e “CdcC”. Mentre lo schema ABBA identifica tutte le parole palindrome di 4 lettere.

Definiamo le seguenti operazioni fondamentali di editing:

- inserimento di un carattere in una qualsiasi posizione di w ; ad esempio *pippo* diventa *pioppo* tramite l'inserimento di *o*;
- cancellazione di un carattere in una qualsiasi posizione di w ; ad esempio *capra* diventa *capa* tramite la cancellazione di *r*;
- sostituzione di un carattere in una qualsiasi posizione di w ; ad esempio *cane* diventa *rane* se la *c* viene sostituita con una *r*;
- scambio di due caratteri adiacenti; ad esempio *trota* diventa *torta* scambiando la *r* con la *o*.

Si richiede di implementare una serie di classi che permettano di gestire un dizionario e le principali operazioni di editing. In particolare, si richiede di implementare:

- una classe **Editing** che implementa le operazioni di editing fondamentali descritte;
- una classe **Assignment** che gestisce gli assegnamenti;
- una classe **Dictionary** che implementa il dizionario, ossia un insieme di parole.

IMPLEMENTAZIONE

La classe **Editing** deve fornire un'implementazione per le operazioni fondamentali di editing. Tali operazioni sono possibili solo se il parametro **position** (vedi di seguito) rappresenta una posizione valida della parola **w** in input. Per i metodi **insert**, **delete** e **replace**, **position** è valida solo se maggiore o uguale a zero e minore della lunghezza di **str**. Per il metodo **swap**, **position** deve essere maggiore o uguale a zero e minore di **str.length() - 1**.

- **String insert(String w, char c, int position).**
 Se **position** è una posizione valida, restituisce la parola ottenuta da **w** inserendo il carattere **c** in posizione **position**. Altrimenti restituisce **null**.
- **String delete(String w, int position).**
 Se **position** è una posizione valida, restituisce la stringa ottenuta da **w** rimuovendo il carattere in posizione **position**. Altrimenti restituisce **null**.
- **String replace(String w, char c, int position).**
 Se **position** è valida, restituisce la stringa ottenuta da **w** rimpiazzando il carattere in posizione

- **position** con il carattere **c**. Altrimenti restituisce **null**.
- **String swap(String w, int position)**.
Se **position** è valida, restituisce la stringa ottenuta da **w** scambiando il simbolo in posizione **position** con quello in posizione **position + 1**. Altrimenti restituisce **null**.

La classe **Assignment** invece deve almeno fornire i seguenti metodi:

- **Assignment()**.
Costruttore che costruisce l'assegnamento vuoto che associa ad ogni lettera maiuscola il carattere fittizio non appartenente all'alfabeto delle lettere minuscole, ad esempio @.
- **char getLower(char upper)**.
Restituisce il carattere minuscolo associare al carattere maiuscolo **upper** oppure @ se il carattere **upper** non è ancora stato assegnato.
- **void set(char upper, char lower)**.
Modifica l'assegnamento corrente in maniera tale da associare al carattere maiuscolo **upper** il carattere minuscolo **lower**. Se **upper** non è un carattere maiuscolo oppure **lower** non è un carattere minuscolo, il metodo lascia l'assegnamento inalterato.
- **void buildAssignment(String schema, String word)**.
Prende un input uno schema e una stringa e costruisce, se esiste, un assegnamento σ tale che la parola **w** è un'istanza dello schema **schema**.
- **boolean isInstance(String schema, String w)**.
Restituisce true se e solo se **w** è un'istanza di **schema** in base all'assegnamento corrente.

La classe **Dictionary** deve fornire i seguenti metodi:

- Un costruttore **Dictionary(String fileName)**
Costruisce il dizionario contenente le parole elencate nel file di testo **fileName**. Le parole contenute in **fileName** sono separate da uno o più spazi (compresi tabulatori e newline). Se il file non esiste crea un dizionario vuoto.
- **void Print()**
Stampa tutte le parole contenute nel dizionario.
- **boolean isIn(String w)**.
Verifica se la parola **w** è presente nel dizionario.
- **void add(String w)**.
Aggiunge, se non presente, la parola **w** nel dizionario.
- **void addInsert(String w, char c, int position)**.
Se **position** è valida, aggiunge nel dizionario la parola ottenuta da **w** inserendo il carattere **c** in posizione **position**.
- **void addDelete(String w, int position)**.
Se **position** è valida, aggiunge nel dizionario la parola ottenuta da **w** rimuovendo il carattere in posizione **position**.
- **void addReplace(String w, char c, int position)**.
Se **position** è valida, aggiunge nel dizionario la parola ottenuta da **w** sostituendo il carattere in posizione **position** con il carattere **c**.
- **void addSwap(String w, int position)**.
Se **position** è valida, aggiunge nel dizionario la parola ottenuta da **w** scambiando il carattere in posizione **position** con quello in posizione **position + 1**.
- **void delete(String w)**.
Rimuove la parola **w** dal dizionario.
- **void Print(String schema)**.

Stampa lo schema **schema** e tutte le parole del dizionario che sono istanze di **schema**.

CONSEGNA DEL PROGETTO

Inviare (per email) al docente una cartella compressa contenente:

- il sorgente delle classi richieste;
- il sorgente di una classe aggiuntiva contenente un **main** che consenta di testare le specifiche richieste;
- la documentazione del progetto generata da **javadoc** (ricordo che questa documentazione ha un qualche significato solo se il codice è stato opportunamente commentato);
- Una breve relazione (al massimo una decina di pagine – escluso il testo del progetto) che descriva sinteticamente le principali scelte algoritmiche effettuate.