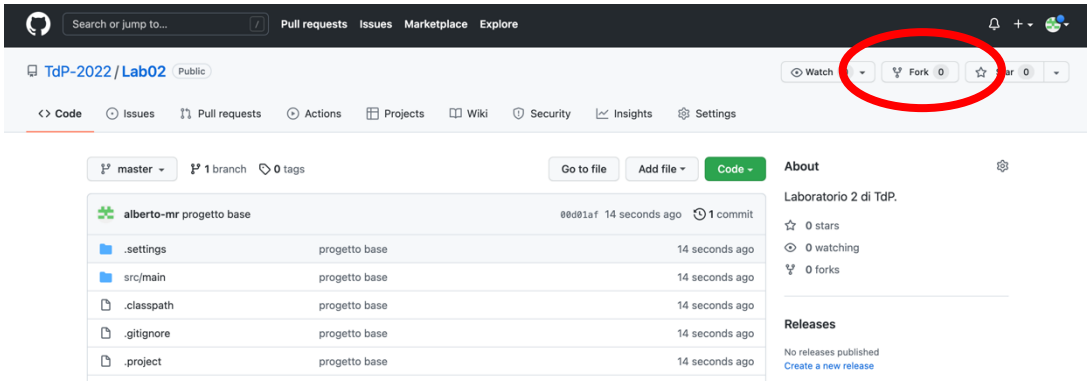
03FYZ TECNICHE DI PROGRAMMAZIONE

Istruzioni per effettuare il fork di un repository GitHub

- Effettuare il login su GitHub utilizzando il proprio username e password.
- Aprire il seguente repository su GitHub:

  https://github.com/TdP-2024/Lab03.git
- Utilizzare il pulsante *Fork* in alto a destra per creare una propria copia del progetto.



L'azione di Fork crea un nuovo repository nel proprio account GitHub con una copia dei file necessari per l'esecuzione del laboratorio.

- Aprire Pycharm, assicurandosi che eventuali precedenti progetti siano chiusi, selezionare *Get From VCS*.
- Utilizzare la URL del **proprio** repository che si vuole clonare (**non** quello in TdP-2024!), ad esempio:

  https://github.com/my-github-username/Lab03
- Selezionare la cartella di destinazione (quella proposta va bene), fare click su *Clone*.
- Il nuovo progetto è stato clonato ed è possibile iniziare a lavorare.
- A fine lavoro ricordarsi di effettuare Git commit e push, utilizzando l'apposito menù.

ATTENZIONE: solo se si effettua Git **commit** e successivamente Git **push** le modifiche locali saranno propagate sui server GitHub e saranno quindi accessibili da altri PC e dagli utenti che ne hanno visibilità.

03FYZ TECNICHE DI PROGRAMMAZIONE

Esercitazione di Laboratorio 19/20 marzo 2024

Obiettivi dell'esercitazione:

- Creazione di una struttura dati
- Introduzione alla complessità

Esercizio 1

Dopo aver fatto il fork del progetto relativo al laboratorio 3, creare in Python una semplice applicazione che funga da correttore ortografico di parole: dato un testo in input, il programma stampa le parole errate, il numero di errori, e il tempo impiegato per effettuare il controllo ortografico.

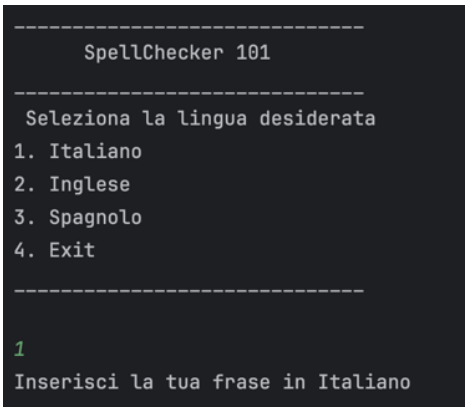
Per interagire con l'applicazione, sarà necessario costruire un menù su più livelli da fornire all'utente tramite riga di comando. Il primo livello permetterà di selezionare la lingua desiderata (e.g. Inglese, Italiano). Il secondo livello richiederà all'utente di inserire una stringa da valutare, ed alla pressione del tasto Invio sarà restituito un sottoinsieme di parole che presentano un errore di ortografia, insieme ad informazioni ausiliarie quali il numero di parole errate ed il tempo di calcolo che è stato richiesto per eseguire il compito. Per questo primo esercizio è richiesto di usare il metodo `contains` di python per effettuare il controllo ortografico.
Si suggerisce di filtrare il testo ricevuto in input trasformandolo tutto in minuscolo ed eliminando i segni di punteggiatura.

Nota1: per eliminare tutti i segni di punteggiatura una strategia potrebbe essere attraverso un apposito metodo:

```
def replaceChars(text):
    chars = "\\`*_{}[]()>#+-.!$%^;,=_~"
    for c in chars:
        text = text.replace(c, "")
    return text
```

Nota2: sviluppare l'applicazione dividendo opportunamente i metodi di gestione dei dati dai metodi di logica.

Esempi:




```
Il dinosaauro in pigiama decise di andare a fare la spesa, dimenticandosi però che non avva portasfoglio né pantaloni, suscitando sguradi perplexi tra i pasanti.
------------------------------
Using contains
dinosaauro
avva
portasfoglio
sguradi
perplexi
pasanti
Time elapsed 0.1059122085571289
------------------------------
Using Linear search
dinosaauro
avva
portasfoglio
sguradi
perplexi
pasanti
Time elapsed 0.18631696701049805
------------------------------
Using Dichotomic search
dinosaauro
avva
portasfoglio
sguradi
perplexi
pasanti
Time elapsed 0.000286102294921875
```

```
The library was clossed today, so I couldn't borrow the books I wanted, which was quaite disapointing and slightly inconvinient for my studies.
------------------------------
Using contains
clossed
quaite
disapointing
inconvinient
Time elapsed 0.03913402557373047
------------------------------
Using Linear search
clossed
quaite
disapointing
inconvinient
Time elapsed 0.04754519462585449
------------------------------
Using Dichotomic search
clossed
quaite
disapointing
inconvinient
Time elapsed 7.033348083496094e-05
```

Di seguito, una possibile traccia per la soluzione:

1. Definire una classe `dictionary`, che può essere parzialmente ereditata dal Lab 2 con opportune modifiche. Questa classe avrà il compito di gestire il dizionario di una singola lingua, dovrà leggere il file e raccogliere le informazioni in una lista locale, oltre che a dare modo alle altre classi di consultare il dizionario.

2. Definire una classe `RichWord`. Ogni istanza di questa classe conterrà una parola del testo in input, ge l'indicazione se tale parola è corretta o meno (utilizzare un `boolean`). Questa classe può essere utile per filtrare le parole corrette durante l'esecuzione del programma.

3. Definire una classe `MultiDictionary` che gestirà l'accesso ai vari dizionari a lingua singola, e implementerà gli algoritmi di ricerca. Il metodo `searchWord` sarà dedicato a cercare una specifica parola nel dizionario della lingua selezionata e restituisce una lista di `RichWord`, con indicazione se la parola è stata trovata nel dizionario (e quindi è corretta) oppure no.

4. La classe `SpellChecker` farà da interfaccia fra l'utente e la classe `MultiDictionary.`

Esercizio 2

Modificare l'algoritmo di ricerca del metodo `searchWord` implementando una ricerca lineare ed una dicotomica (vedere spiegazione nella pagina seguente). Si consiglia di creare nel model due nuovi metodi che sostituiscono `searchWord`: `searchWordLinear` e `searchWordDichotomic`, dove il primo utilizza una ricerca lineare, mentre il secondo quella dicotomica.

Confrontare le differenze di prestazioni tra le due implementazioni utilizzando sia un `ArrayList` ed una `LinkedList`. Riempire la tabella nella pagina seguente con i tempi di esecuzione per ciascun caso.
Quale implementazione utilizza il metodo `contains` di python?

Ricerca lineare (source Wikipedia):

Iterare su tutti gli elementi del vocabolario a partire dal primo. La ricerca termina quando viene trovato l'elemento cercato o si raggiunge l'ultimo, nel caso in cui l'elemento cercato non sia presente nella lista.

Ricerca dicotomica (source Wikipedia):

Sapendo che il vocabolario è ordinato alfabeticamente, l'idea è quella di non iniziare la ricerca dal primo elemento, ma da quello centrale, cioè a metà del dizionario. Si confronta questo elemento con quello cercato:

- Se corrisponde, la ricerca termina indicando che l'elemento è stato trovato
- se è superiore, la ricerca viene ripetuta sugli elementi precedenti (ovvero sulla prima metà del dizionario), scartando quelli successivi
- se è inferiore, la ricerca viene ripetuta sugli elementi successivi (ovvero sulla seconda metà del dizionario), scartando quelli precedenti.

Il procedimento viene ripetuto iterativamente fino a quando o si trova l'elemento cercato, o tutti gli elementi vengono scartati. In quest'ultimo caso la ricerca termina indicando che il valore non è stato trovato.

TABELLA CONFRONTO PRESTAZIONI

	ArrayList	LinkedList
list.contains()		
Linear search		
Dichotomic search		