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 repository su GitHub relativo al secondo laboratorio:

  https://github.com/TdP-2017/Lab03
- 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 accout GitHub con una copia dei file necessari per l'esecuzione del laboratorio.
- Aprire Eclipse, andare su *File -> Import*. Digitare *Git* e selezionare *Projects from Git -> Next -> Clone URI -> Next*.
- Utilizzare la URL del **proprio** repository che si vuole clonare (**non** quello in TdP-2017!), ad esempio: https://github.com/my-github-username/Lab03
- Fare click su *Next*. Selezionare il branch (*master* è quello di default) fare click su *Next*.
- Selezionare la cartella di destinazione (quella proposta va bene), fare click su *Next*.
- Selezionare *Import existing Eclipse projects*, fare click su *Next* e successivamente su *Finish*.
- Il nuovo progetto Eclipse è stato clonato ed è possibile iniziare a lavorare.
- A fine lavoro ricordarsi di effettuare Git commit e push, utilizzando il menù *Team in Eclipse*.

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 03 – 22 marzo 2017

Obiettivi dell'esercitazione:

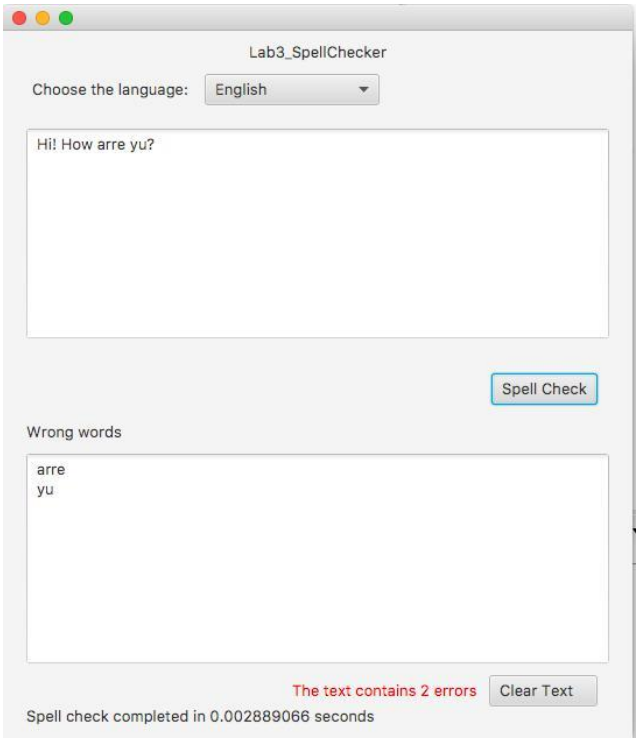
- Utilizzo del pattern MVC
- Creazione di una struttura dati
- Introduzione alla complessità

Esercizio 1

Dopo aver fatto il fork del progetto relativo al laboratorio 3, creare in Java una semplice applicazione dotata di interfaccia grafica 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. L'interfaccia grafica dell'applicazione deve rispecchiare quella riportata in Figura 1.

In particolare, l'utente inserisce un testo che vuole verificare nella casella di testo in alto. Dopo aver selezionato la lingua da utilizzare, fa click sul bottone *Spell Check* per avviare il controllo ortografico. Si suggerisce di filtrare il testo ricevuto in input trasformandolo tutto in minuscolo e eliminando i segni di punteggiatura, utilizzando la funzione *str.replaceAll("[\\p{Punct}]", "")*. Nell'area di testo sottostante, viene restituito l'elenco delle parole errate. Utilizzare il pulsante *Clear Text* per cancellare il testo inserito da entrambe le caselle di testo.

N.B.: sviluppare l'applicazione secondo il pattern **Model View Controller**



Di seguito, una possibile traccia per la soluzione:

1. Creare l'interfaccia grafica con *SceneBuilder*. Associare al bottone *Spell Check* il metodo *doSpellCheck()* ed al bottone *Clear Text* il metodo *doClearText()*, da definirsi nel controllore (*it.polito.tdp.spellchecker.controller.SpellCheckerController*).

2. Creare il package *it.polito.tdp.spellchecker.model* ed una nuova classe (Java Bean) denominata *RichWord*. Ogni istanza di questa classe conterrà una parola del testo in input, e l'indicazione se tale parola è corretta o meno (utilizzare, ad esempio, un booleano).
3. Definire nel package *it.polito.tdp.spellchecker.model* una nuova classe *Dictionary*, il **modello** dell'applicazione, in cui definire i seguenti metodi:

```
public void loadDictionary(String language)
```

Metodo che permette di caricare in memoria il dizionario della lingua desiderata. A questo proposito, utilizzare i file *Italian.txt* e *English.txt* nella cartella *rsc*. I file contentengono una parola per riga. Salvare le parole del dizionario in una struttura dati appropriata. Di seguito viene riportato un esempio della sequenza di operazioni necessarie per leggere le parole dal file:

```
try {
      FileReader fr = new
      FileReader("rsc/English.txt");
      BufferedReader br = new BufferedReader(fr);
      String word;
      while ((word = br.readLine()) != null) {
            // Aggiungere parola alla struttura dati
      }
      br.close();

} catch (IOException e){
      System.out.println("Errore nella lettura del file");
}
```

```
public List<RichWord> spellCheckText(List<String> inputTextList)
```

Metodo che esegue il controllo ortografico su testo di input (rapopresentato da una lista di parole), e restituisce una lista di *RichWord*. Per ogni parola di *inputTextList*, tale metodo controlla se essa è presente nel dizionario. In caso affermativo, la *RichWord* corrispondente, che dovrà essere creata e aggiunta alla lista di ritorno, sarà corretta, altrimenti sarà errata.

4. Definire nel package it.polito.tdp.spellchecker.model una classe TestModel dotata di metodo main per effettuare alcuni test sul funzionamento del model.
5. Utilizzare il modello *Dictionary* per completare i metodi del controllore *SpellCheckerController*

Esercizio 2

Nel metodo *spellCheckText*, per la ricerca delle singole parole all'interno del dizionario, provare ad implementare una ricerca dicotomica (vedere spiegazione al fondo dell'esercizio).

Ci sono differenze di prestazioni con la versione precedente?

Ricerca dicotomica (from Wikipedia):

Sapendo che il vocabolario è ordinato alfabeticamente, l'idea è quella di iniziare la ricerca non 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 invece 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 quanto si arriva al punto in cui tutti gli elementi sono stati scartati: la ricerca termina indicando che il valore non è stato trovato.