03FYZ TECNICHE DI PROGRAMMAZIONE

Istruzioni per clonare un repository GitHub su Eclipse tramite EGit

- Iscriversi all'assignment di GitHub Classroom utilizzando il seguente link:

  https://classroom.github.com/assignment-invitations/263e2397ace00f525de9a181b90ad9c3

  In automatico viene creato un nuovo repository personale 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 repository che si vuole clonare, ad esempio:

  https://github.com/orgs/TdP-2016-Lab/Lab1-123456

- 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 01 – 9 Marzo 2016

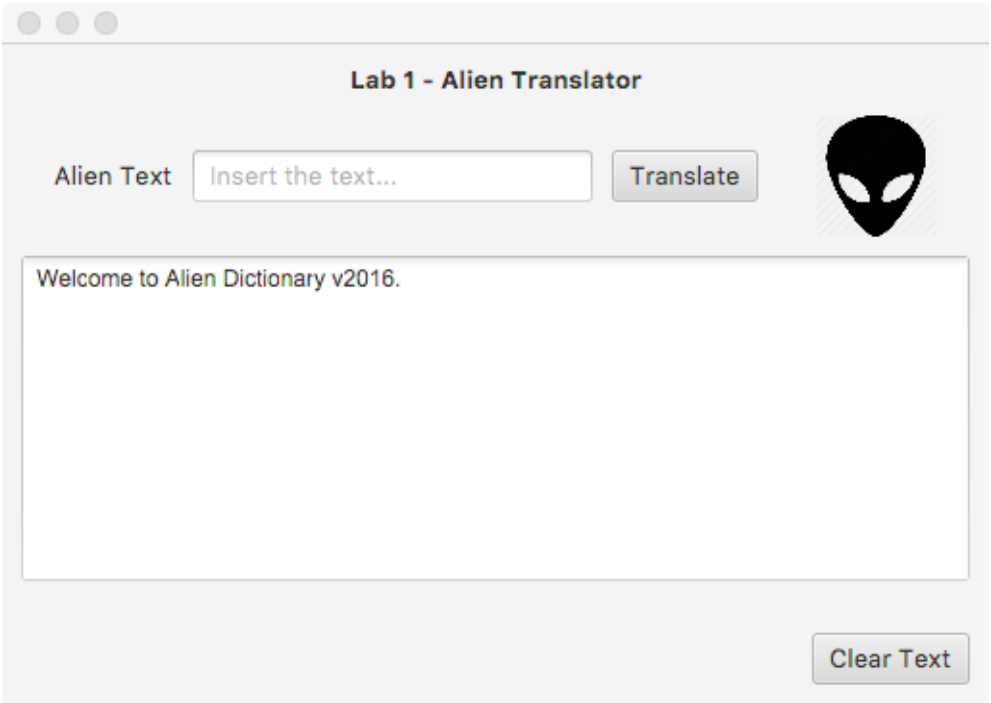
Obiettivi dell'esercitazione:

- Introduzione all'utilizzo degli strumenti di sviluppo: Java, Eclipse, JavaFX, SceneBuilder
- Sperimentare la realizzazione di interfacce grafiche
- Sperimentare l'utilizzo delle liste.

Dopo anni di studi, alcuni scienziati sono riusciti a decifrare un linguaggio alieno trasmesso da un remoto pianeta. Per poter scoprire i messaggi che gli alieni hanno trasmesso alla terra nell'ultimo decennio, gli scienziati hanno chiesto di ideare un traduttore che possa aiutarli.

Scopo dell'esercitazione: Realizzare in linguaggio Java una semplice applicazione dotata di interfaccia grafica che funga da traduttore di parole aliene: deve essere possibile sia l'aggiunta di nuove parole che la ricerca di quelle esistenti.

Funzionamento previsto: L'utente inserisce una nuova parola utilizzando l'apposita casella di testo, secondo il seguente pattern: <parola aliena> <traduzione> (separate da uno spazio). Cliccando sul bottone *Translate* la parola viene aggiunta al dizionario. In alternativa, l'utente può cercare la traduzione di una parola esistente inserendo: <parola aliena> e facendo click nuovamente sul bottone *Translate*.
Utilizzare l'area di testo sottostante per visualizzare la traduzione o eventuali messaggi di errore. Gli unici caratteri ammessi sono [a-zA-Z] (ossia solo le lettere alfabetiche, siano essere maiuscole o minuscole), ma la ricerca deve essere *case insensitive* (si suggerisce di convertire tutto il testo ricevuto in minuscolo prima di elaborarlo).



Esercizio 1.1 Utilizzando la libreria *javafx* ed il tool *SceneBuilder*, creare un'interfaccia grafica simile a quella sopra mostrata. Associare al bottone *Translate* un metodo *doTranslate()* ed al bottone *Clear* un metodo *doReset()*.

Esercizio 1.2 Nel package *it.polito.tdp.alien.* creare una classe *Word* con le seguenti variabili:

```
private String alienWord;
private String translation;
```

Definire il metodo:

```
public String compare(String alienWord)
```

Il metodo viene utilizzato per controllare se la parola *alienWord* è già presente nel dizionario (vedi punto 1.3).

Esercizio 1.3 Nel package *it.polito.tdp.alien* creare una classe *AlienDictionary* con i seguenti metodi:

```
public void addWord(String alienWord, String translation)
```

Il metodo viene chiamato dal Controller per l'aggiunta, nel dizionario, di una nuova *alienWord* con corrispondente *translation*. Se *alienWord* è già presente, la traduzione deve essere aggiornata.

```
public String translateWord(String alienWord)
```

Il metodo viene chiamato dal Controller per la traduzione della parola *alienWord* passata come parametro. Il metodo restituisce la parola tradotta, altrimenti *null* se *alienWord* non è presente nel dizionario.

La classe implementa il dizionario come una lista di oggetti di tipo *Word*.

Esercizio 1.4 Implementare, nel controller, il metodo *doTranslate()* e tutta la rimanente logica dell'applicazione necessaria al suo funzionamento. Effettuare alcuni test.

Esercizio 1.5 Definire nel package *it.polito.tdp.alien* una nuova classe *WordEnhanced.* La classe è simile all'esistente *Word*, ma il metodo *addWord()* consente di associare più traduzioni ad una singola parola aliena. Definire una lista per tenere traccia delle possibili traduzioni:
`private List<String> translations`
Effettuare alcuni test usando la nuova classe *WordEnhanced* al posto della vecchia *Word*.

Esercizio 1.6 Implementare la ricerca di una parola con wildcard: quando nella parola aliena compare il simbolo "?", il carattere corrispondente può essere qualsiasi. È ammesso un solo "?" per ogni parola da cercare.
Ad esempio, se la traduzione della parola ALIENO corrisponde ad ANDREA, cercando ALI?NO si deve ottenere ANDREA. Attenzione a cosa succede se vi solo più parole aliene corrispondenti (ad esempio ALINNO).