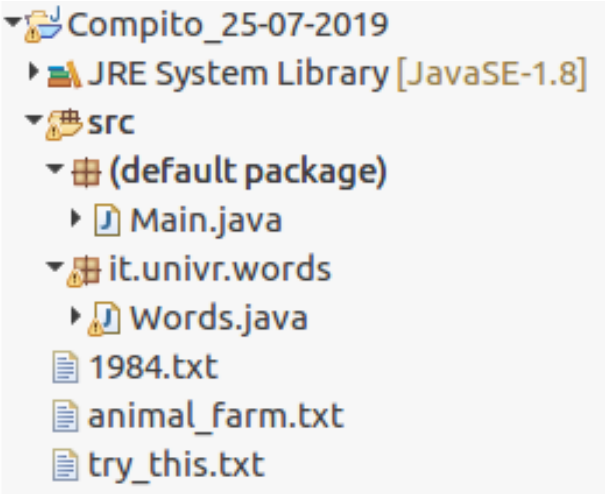
Esame di Programmazione II, 25 luglio 2019

Si crei un progetto Eclipse e, nella directory dei sorgenti, si crei il package ***`it.univr.words`****. Si copi al suo interno la classe* ***`Words.java`****. Si copi dentro il package di default la classe* ***`Main.java`****. Si copino al livello base del progetto i tre file di testo* ***`1984.txt`****,* ***`animal_farm.txt`*** *e* ***`try_this.txt`*** *(quest'ultimo lo dovete generare voi con il comando* ***`touch try_this.txt`****). Il risultato dovrebbe quindi essere come nella figura seguente:*



Se si realizzano nuove classi, le si crei dentro il package ***`it.univr.words`****. Non si modifichi le dichiarazioni dei metodi. Si possono definire altri campi, metodi, costruttori e classi, ma devono essere* ***`private`****. La consegna fornita compila. Anche la soluzione che verrà consegnata dovrà compilare, altrimenti non verrà corretta.*

Esercizio 1 [18 punti, si consegni `Words.java`] Si completi la classe `Words.java`, che implementa una lista di parole lette da un file di testo, modificandola dove indicato con `TODO`. Quando un oggetto `Words` viene creato, esso deve accedere al file di testo indicato per nome, leggerlo parola per parola e aggiungere tali parole a se stesso (`this`). Le modifiche sono le seguenti:

1. si faccia estendere a tale classe una classe della libreria Java standard che rappresenta liste di stringhe;

2. si completino i due costruttori, che estraggono le parole di un file di testo e le aggiungono alla lista `this`. Si noti che i due costruttori si differenziano solo perché il primo estrae tutte le parole, mentre il secondo estrae solo quelle che soddisfano un predicato estrattore. Il tipo per il predicato esiste già nella libreria standard ed è definito come segue:

   ```
   public interface java.util.function.Predicate<T> {
     boolean test(T t); // controlla se t soddisfa il test
   }
   ```

   I due costruttori devono lanciare una `IOException` (della libreria standard) se ci fosse un problema di accesso al file;

3. si ridefinisca il metodo `toString()` in modo da ritornare una stringa del tipo `a list of XX words`, dove `XX` è la lunghezza della lista;

4. si scriva il metodo `mostFrequent()` che restituisce la parola più frequente fra quelle contenute nella lista (in caso di parità, restituisce la prima fra le più frequenti). Si noti che, se la lista fosse vuota, questo metodo dovrà lanciare una `NoSuchElementException`, la cui classe è nella libreria standard.

Suggerimenti: Per leggere un file di testo si crei un oggetto lettore bufferizzato, in questo modo: `new BufferedReader(new FileReader(fileName))`. Chiamando ripetutamente su tale oggetto il metodo `readLine()`, si ottiene una riga (cioè una stringa) alla volta del file di testo e alla fine si ottiene `null` per segnalare la fine del file. Ogni riga può essere divisa in parole, scartando spaziatura e punteggiatura, usando il metodo delle stringhe `split("\\W+")`. Non ci si dimentichi di chiudere il `BufferedReader` alla fine dell'utilizzo.

Esercizio 2 [14 punti, si consegni `Main.java`] Si completi la classe `Main.java` in modo tale che la sua esecuzione:

1. chieda all'utente di inserire da tastiera il nome del file di testo da processare;
2. crei un oggetto `Words` a partire da tale file, stampi a video tale oggetto e quindi stampi a video la sua parola più frequente;
3. crei un oggetto `Words` a partire da tale file, selezionando solo le parole che cominciano con il carattere `J` (maiuscolo), stampi a video tale oggetto e quindi stampi a video la sua parola più frequente;
4. crei un oggetto `Words` a partire da tale file, selezionando solo le parole che siano più lunghe di quattro caratteri, stampi a video tale oggetto e quindi stampi a video la sua parola più frequente;
5. se la creazione di uno di questi tre `Words` fallisse con un'eccezione perché non si riesce ad accedere al file, il `Main` dovrà stampare `There was a problem accessing FILENAME` e terminare;
6. se una delle tre chiamate a `mostFrequent()` fallisse con un'eccezione perché un oggetto `Words` è vuoto e quindi non ha un elemento più frequente, il `Main` dovrà stampare `I have selected zero words` e terminare.

Se tutto è corretto, un'esecuzione del ***`Main`*** *sul file di prova* ***`1984.txt`*** *dovrebbe rassomigliare a quanto segue:*

```
File name: 1984.txt

I have extracted a list of 107799 words
The most frequent word is "the"

I have extracted a list of 153 words that start with J
The most frequent word is "Julia"

I have extracted a list of 39910 words that are longer than four characters
The most frequent word is "Winston"
```

Lo si provi anche con ***`animal_farm.txt`*** *e* ***`try_this.txt`****.*