PAROLE E FILE

Spesso è interessante analizzare uno o più file di testo in una determinata lingua per comprendere quali siano le parole più frequenti (o anche le lettere più frequenti). Vogliamo realizzare la classe FileAndParole, che effettua alcune di queste operazioni ed è composta dai seguenti metodi:

- stampaParole(Path file): questo metodo prende in input un oggetto della classe Path del package java.nio.file rappresentante un file di testo e stampa riga per riga tutte le parole contenute nel testo. Si supponga che ogni parola sia separata da spazio e non vi siano segni di punteggiatura.

- occorrenzeParole(Path file): questo metodo prende in input un oggetto della classe Path del package java.nio.file rappresentante un file di testo e ritorna una HashMap dove le chiavi sono le parole e i valori sono il numero di occorrenze delle parole stesse nel testo. Si supponga che ogni parola sia separata da spazio, siano tutte minuscole e non vi siano segni di punteggiatura.

- occorrenzeParoleTesto(Path file): questo metodo prende in input un oggetto della classe Path del package java.nio.file rappresentante un file di testo e ritorna una HashMap dove le chiavi sono le parole e i valori sono il numero di occorrenze delle parole stesse nel testo. In questo caso il testo può comprendere punteggiatura e lettere maiuscole. SUGGERIMENTO: per eliminare la punteggiatura da una riga di testo utilizzare il metodo riga.split("\W+").

- paroleInRima(Path file): questo metodo prende in input un oggetto della classe Path del package java.nio.file rappresentante un file di testo in italiano e il suo obiettivo è quello di catalogare le parole in rima. Due parole in italiano di intendono in rima se:
  Terminano con una sequenza vocale-consonante-vocale identica.
  Terminano con una sequenza vocale-consonante-consonante-vocale identica.
  Il metodo deve ritornare una HashMap dove le chiavi sono i dittonghi finali (le rime) e i valori sono ArrayList contenenti tutte le parole che terminano con quel dittongo (rima).

Ad esempio se il testo è il seguente:
```
"Il soldato è tornato passando per un ponte vicino a una fonte"
```
Il metodo deve tornare:
```
{
  "ato": ["soldato", "tornato"],
  "ando": ["passando"],
  "onte": ["ponte", "fonte"],
  "ino": ["vicino"]
}
```
NB: possiamo ignorare completamente le parole con lunghezza minore di 4 lettere.