**Relazione primo progetto intermedio Programmazione 2 anno accademico 2017-2018**

**-Analisi Problema**

La consegna del Progetto richiede lo sviluppo di due classi volte all’implementazione dell’interfaccia DataCounter<E,V> assegnata. Le due classi sono Hashtable<E,V> e Treemap<E,V>, da me rinominate HashDataCounter<E,V> e TreeDataCounter<E,V>. Nella prima parte del progetto, per lo sviluppo di queste classi è stata scelta la tecnica defensive programming che prevede l’uso di requires, throws, modifies ed effects per descrivere al meglio il comportamento e la funzione di ogni metodo sviluppato. Nella seconda parte del progetto, viene richiesto lo sviluppo di un’applicazione che utilizzi DataCounter<E,V> e che sia in grado di 1) contare tutti i caratteri presenti in un file di testo, comprendendo ed escludendo gli spazi, 2) contare tutte le parole presenti in questo file di testo. L’applicazione infine deve 3) stampare una tabella in cui nella prima colonna è presente un indice, nella seconda delle stringhe, ossia le parole presenti nel file di testo (di cui tutti i caratteri maiuscoli sono stati resi minuscoli) e nella terza colonna, le occorrenze di ogni parola. Inoltre, le parole all’interno della tabella sono ordinate prima per occorrenza e poi per le varie occorrenze; le parole vengono ordinate in ordine lessicografico.

**-Implementazione**

La prima parte del progetto è stata implementata agevolmente, considerando entrambe le strutture dati con simili funzioni d’astrazione e invarianti di rappresentazione. Ovvero entrambe vengono interpretate come funzioni parziali. Le classi degli iteratori vengono invece interpretate come insieme di coppie modificabili: <chiave,indice>, dove chiave rappresenta gli elementi del Vector<E> restituito, indice rappresenta l’indice della chiave contenuta in Vector<E>. Riguardo la seconda parte del progetto, questa è stata ostica da implementare. Per poter verificare la correttezza delle strutture dati HashDataCounter e TreeDataCounter viene richiesta una scelta inserendo un intero (0 HashDataCounter e 1 TreeDataCounter). Per acquisire l’intero file di testo ho scelto di utilizzare la funzione FileReader, già implementata nella libreria java.io. Il file di testo da inserire deve essere nominato “prova.txt”. Nella cartella questo è già presente e contiene lo stesso esempio usato nella consegna del progetto. Per eliminare da ogni stringa punteggiatura ho scelto di usare il metodo, già presente in java.util.\*, replaceAll. In tal modo, sono riuscito ad isolare caratteri e singole parole. Per trasformare i caratteri da maiuscoli a minuscoli ho usato il metodo toLowerCase(). Per svolgere, infine, l’ordinamento e quindi portare a compimento la seconda richiesta della tabella, ho modificato l’iteratore di HashDataCounter e di TreeDataCounter. Nella modifica degli iteratori, ho costruito un vector che contenesse tutte le chiavi della struttura dati. Successivamente ho eseguito un ciclo per trovare occorrenza massima, e con un ulteriore ciclo, ho selezionato tutte le chiavi con l’occorrenza attuale che ho poi inserito in un nuovo Vector definito “nuovo”. Dopo aver ordinato tale Vector, con Collections.sort(nuovo) (ciò è stato possibile perché ho scelto di far estendere ad E Comparable<E>), ho inserito tutte le chiavi di nuovo nel vettore che restituirà l’iteratore ovvero “ultima”. Infine, viene diminuita l’occorrenza e il ciclo ricomincia.

**-Maggiori difficoltà incontrate**

I punti più ardui da svolgere nel progetto sono stati: la scelta delle funzioni d’astrazione e dell’invariante di rappresentazione e sviluppare il metodo costruttore degli iteratori. Per quanto riguarda la scelta delle funzioni d’astrazione, si poteva optare per un insieme di coppie (chiavi-valore), ma ho scelto l’utilizzo della funzione parziale per mettere in evidenza la corrispondenza dizionario.get(data) = valore o albero.get(data) = valore; così facendo, il tutto viene espresso in modo più chiaro e diretto. Riguardo lo sviluppo dei costruttori degli iteratori, si è rivelato complesso riuscire a permettere all’iteratore di ordinare le varie stringhe, soprattutto nel punto di dover paragonare due V. Dunque ho scelto di fare estendere a V Integer così da poter confrontare più facilmente le occorrenze delle stringhe per svolgere ciò che ho indicato nel paragrafo precedente (Implementazione).

**-Indicazioni per eseguire il programma**

Per eseguire il programma basta compilare la classe contenente il metodo main ovvero Application.java ed eseguire quest’ultima.