Developer: ***Luca Canessa***
Badge Number: ***516639***
Class: ***B***
Date: ***26-11-2016***

Graph Creator

Progettino 1 Programmazione 2

LISTA FILE

Elemento.java	Implementazione dei nodi di grafi
Graph.java	Interfaccia per implementare MyGraph.java
MyGraph.java	Implementazione di grafi
Test.java	File per testare i casi limite e non di un grafo
README.pdf	Documentazione del progetto

DESCRIZIONE

Il software ha la funzione di creare grafi relazionali e fare su di esso operazioni manipolando nodi e archi.
Il progetto è stato suddiviso in 4 file:

Elemento.java
in questo file viene dichiarata la struttura dati principale del software cioè il nodo generico del grafo con le variabili di istanza e i vari metodi che mi permettono di interagire con il nodo. Per comodità di programmazione mi è stato utile inserire tra le variabili la lista degli elementi dei nodi che sono collegati direttamente al nodo in questione e le variabili per la gestione dell'algoritmo (BFS) per visitare l'albero che successivamente verrà creato.

- public void changeE(E el);
  - aggiorna l'elemento di tipo E
- public int getNedge();
  - restituisce il numero di archi che ha l'oggetto, che corrisponde alla lunghezza della lista dove sono salvati gli elementi collegati
- public E getE();
  - restituisce l'elemento di tipo E presente nell'oggetto
- public List<E> getEdge();
  - copio la lista di archi in una lista temporanea per preservare quella originale

- public void addE(E x);
  - aggiunge alla lista degli archi l'elemento che si vuole collegare all'oggetto
- public void removeE(E x);
  - rimuove dalla lista degli archi l'elemento che si vuole scollegare dall'oggetto
- public String toString();
  - restituisce una stringa con le informazioni principali dell'oggetto
- public boolean equals(Elemento<E> el);
  - eguaglia la variabile dato con el

Graph.java

è l'interfaccia della classe concreta in cui setto e rendo pubblici i metodi che definiscono la struttura dei grafi in modo tale da usarli in altre classi.

- public int min_edge(E start, E end);
  - calcola la distanza minima tra due nodi del grafo utilizzando l'algoritmo di ricerca in ampiezza denominato BFS
- public int diameter();
  - utilizza la funzione precedentemente dichiarata (min_edge) per calcolare il cammino minimo tra tutti i nodi del grafo per poi restituire il massimo. Questo tipo di algoritmo è esponenziale.

MyGraph.java

è la classe concreta in cui creo il grafo e adotto alcune scelte implementative per un miglior utilizzo delle strutture dati. In questo file è presente un frammento di codice che permette di visualizzare il grafo su una pagina web in modo da facilitare la scrittura di metodi e verificarne la validità. Un esempio di tale utilizzo è sicuramente la misurazione della distanza minima tra due nodi e del diametro del grafo.

Test.java

è il file di test del codice. Tale file mi permette di verificare l'effettiva funzionalità del codice attraverso un grafo abbastanza semplice con alcune particolarità. Inoltre, commentati, sono stati inseriti metodi che sollevano eccezioni, cioè sono stati scritti metodi che vengono detti casi limite, casi in cui il software non sarebbe valido in base alle scelte implementative che ho adottato.

Adottando l'approccio di programmazione difensiva col fine di ottenere un software più sicuro, le variabili di tipo primitivo e non, sono state restituite attraverso metodi appositi e/o copiate in variabili temporanee, in modo tale che il cliente anche se fosse fornito dell'oggetto primario è impossibilitato a modificarle.
Inoltre per essere sicuri che le scelte implementative fossero mantenute, sono state inserite delle clausole che sollevano eccezioni e bloccano l'esecuzione del codice così che se venissero modificati o inseriti dati in modo errato, l'invariante di rappresentazione non si modifica e il software rimane valido in ogni situazione.