TECNICHE DI PROGRAMMAZIONE

10/06/19

Simulazione d’esame: Soluzione laboratorio 12.

Partiamo con lo svolgimento del primo punto:

Immagine che contiene screenshot

Descrizione generata automaticamente

Come prima cosa ci chiediamo sempre quanti sono i vertici. In questo caso, il testo lo dice esplicitamente; ci sono 7 vertici. Essendo 7 un numero molto basso, possiamo usare come strategia la numero 1.

Un’altra cosa su cui ragionare è che cosa sono i vertici. Molte persone avevano creato la classe “Distretto” dato che i vertici erano proprio rappresentati dai distretti. In questo esercizio non serviva crearla in quanto quest’ultimi sono modellati da dei campi INTERI.

Quando nel modello andiamo a definire il grafo, ci basta definirlo “Integer”, non serve l’oggetto “Distretto” con la relativa classe. Non dobbiamo definire gli hashCode e gli equals perché la classe “integer” di Java li ha già.

Il grafo è pesato (lo capiamo dal testo). Inoltre, utilizzando degli interi come vertici, non abbiamo nemmeno bisogno della Mappa. Ci basta quindi creare una lista di distretti.

Sicuramente, come altro attributo della classe Model, abbiamo il DAO “eventsDAO”. Questa classe ha dentro un metodo per ottenere tutte le righe delle tabelle. Ci creiamo quindi il riferimento al DAO.

Nel costruttore del Model andiamo a creare il DAO e il grafo. Non possiamo ancora farci dare i distretti perché il testo chiedere di selezionare prima un determinato anno e quindi alcuni distretti potrebbero non esserci in un determinato anno.

Immagine che contiene screenshot

Descrizione generata automaticamente

Passiamo quindi al metodo void “creaGrafo” (sempre lo stesso). Dobbiamo chiederci se questo metodo riceve qualche parametro; nel nostro caso si (riceve l’anno). Possiamo usare la classe Year o semplicemente “Integer” per modellare l’anno. L’obiettivo di questo metodo è quello di farsi dare i distretti per ogni anno (i vertici del nostro grafo). I distretti, ottenuti dal metodo del DAO “listAllDistricts” (ancora da creare), andranno inseriti nella lista “distretti”.

Andiamo quindi a creare il metodo del DAO. A questo metodo dovremo passare come parametro l’anno selezionato dall’utente e conterrà la query presa da Heidi. La query dovrà permettere di prelevare tutti i distretti di un relativo anno inserito dall’utente (?).

Copio il try-catch dai metodi precedenti e li andiamo a modificare. Impostiamo il parametro dell’anno e ad ogni riga della tabella, aggiungiamo il distretto alla lista “distretti”. Ricordarsi sempre di chiudere la connessione per evitare problemi. Come ultima cosa, ci facciamo restituire i distretti dal metodo.

Il metodo finale sarà quindi il seguente:

Immagine che contiene screenshot

Descrizione generata automaticamente

Possiamo tornare quindi al Modello e aggiungere i vertici trovati dal metodo al grafo. Una volta fatto questo, abbiamo deciso di utilizzare la strategia numero 1 e quindi eseguiamo un doppio ciclo “for” sui vertici. Controlliamo che i due vertici non sono gli stessi mediante l’if (possiamo usare “equals” perché stiamo usando la classe Integere di Java). Se usiamo “!=” per indicare che sono diversi, il programma non funziona perché abbiamo degli OGGETTI Integer e non dei singoli numeri.

Un altro controllo che posso fare è controllare che l’arco tra due vertici non sia “null”. Uno degli errori più comuni è quello di aggiungere un arco e usare il riferimento che ci viene restituito dall’ AddEdge. **Per un grafo non orientato, quando si fa l’ AddEdge, se l’arco è già presente, ci viene restituito “null”.** Logicamente, se poi noi usiamo questo riferimento, ci viene restituito “nullPointerException”. Bisognerebbe quindi sempre controllare che l’arco non sia già presente.

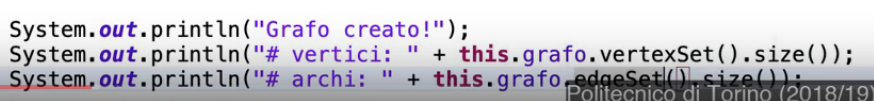
L’arco che dobbiamo aggiungere deve avere come peso la distanza tra i centri geografici dei due distretti connessi. Se noi andiamo nel DB, notiamo che ogni crimine ha una longitudine e una latitudine. Dato un insieme di latitudini e longitudini, per calcolare il centro mi basta calcolare la media. Grazie alla latMedia e alla lonMedia, so dove si trova il centro di un distretto. Userò poi questo valore per trovare la distanza media del centro di un altro distretto.

Per calcolare la latMedia e la lonMedia posso utilizzare due metodo del DAO che attraverso una query mi restituiscano i valori che mi servono. Per trovare la distanza media tra due centri posso usare la classe “LatLngTool” che ha un metodo “distance” a cui bisogna passare i due punti e l’unità di misura (nel nostro caso chilometri). Una volta fatto questo, posso finalmente aggiungere l’arco tra i due vertici e stampare il grafo ottenuto.

Immagine che contiene screenshot

Descrizione generata automaticamente

(ancora nell’if)



Andiamo ora ad implementare i metodi “getLanMedia” e “getLonMedia”.

La query di questi due metodi dovrà calcolare una media “AVG” del campo “geo\_lat” o “geo\_long”. Dovremo inoltre passare l’anno in quanto noi vogliamo sapere la latitudine e la longitudine media di un distretto in un determinato anno. Essendo un punto di domanda nella query, dobbiamo ricordarci di settare il suo valore con il metodo “setInt”.

Un consiglio pratico per questi metodi è quello di chiudere la connessione prima di ogni “return”. xI due metodi saranno quindi i seguenti:

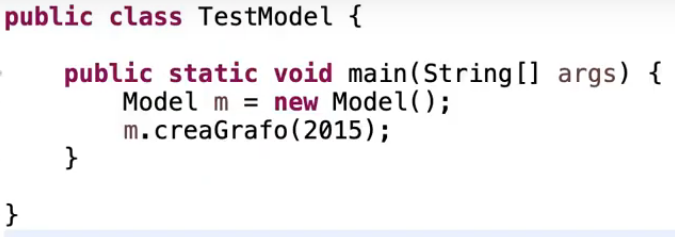
Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Passiamo ora al TestModel in cui creiamo il Modello e testiamo il metodo “creaGrafo”.



Runnando il programma, esce l’eccezione “Unknown column”; abbiamo sbagliato a scrivere il campo (indicato nell’errore) nella query.

Passiamo ora al punto “c” del primo punto. Leggendo il testo, notiamo che dobbiamo modellare due informazioni contemporaneamente: il mio vicino e quanto è distante da me. Queste informazioni non le ho da nessuna parte. Dobbiamo quindi creare una classe che abbia 2 campi (quelli che vogliamo gestire) così mi basta ordinare le istanze di quella classe. Quindi quando l’ultimo punto ci chiede di ordinare i vicini di un vertice secondo un determinato criterio, bisogna sempre fare così.

Creiamo quindi la classe “Vicino” che avrà come variabili il vicino (Integer) e la sua distanza. Aggiungiamo il costruttore e i Getter/Setter.

Nel Model suppongo di avere un metodo public che mi restituisca una lista ordinata di “Vicino” in cui come parametro ricevo il distretto di cui voglio sapere i vicini e la distanza. Come prima cosa, in questo metodo andiamo a prendere la lista di vicini del “distretto” passato come parametro. Per far questo ,uso il metodo “neighborListOf” della classe Graphs.

Questi vicini appena trovati li scorro e mi vado a creare una lista di “Vicino” in cui abbiamo tutti i vicini del distretto scelto con il loro peso. Successivamente, grazie al metodo “sort” della classe “Collections” ordino la lista. Ricordarsi sempre di implementare la classe Comparable perché ci serve il metodo di comparazione dato dal metodo “compareTo”.

Immagine che contiene screenshot

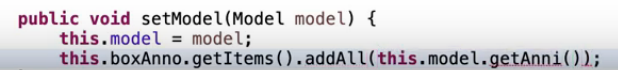
Descrizione generata automaticamente

Possiamo ora collegare tutto al Controller mediante il metodo “doCreaReteCittadina”. In questo metodo come prima cosa recupero l’anno dal box (con controllo su errore). Se tutto va bene, possiamo creare il grafo con il metodo “creaGrafo” del Model.

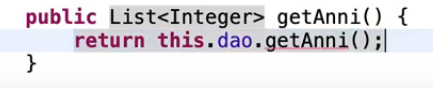
Immagine che contiene screenshot

Descrizione generata automaticamente

Impostiamo il valore della ComboBox “boxAnno” con valori “Integer” e andiamo a riempirla nel metodo setModel.



Dobbiamo ancora definire il metodo “getAnni” che ci permette, attraverso una query, di estrarre tutti gli anni dalla tabella del Database. Questo metodo del Model deve solo richiamare il metodo “getAnni” del DAO (anch’esso da creare ancora).

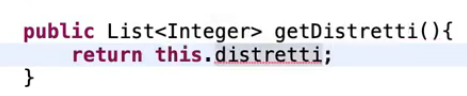


Andiamo quindi nella classe DAO e creiamo questo metodo (molto semplice).

Immagine che contiene screenshot

Descrizione generata automaticamente

Posso tornare ora nel Controller, nel metodo “doCreaReteCittadina” e posso utilizzare il metodo “getVicini” (creato prima) su tutti i vertici del grafo. Mi serve un metodo “getDistretti” del Model che mi restituisce tutti i distretti.



Quindi, dal Controller, per ogni distretto, vado a recuperare la lista dei vicini. Una volta che ho questa lista, mi basta stamparla.

Immagine che contiene screenshot

Descrizione generata automaticamente

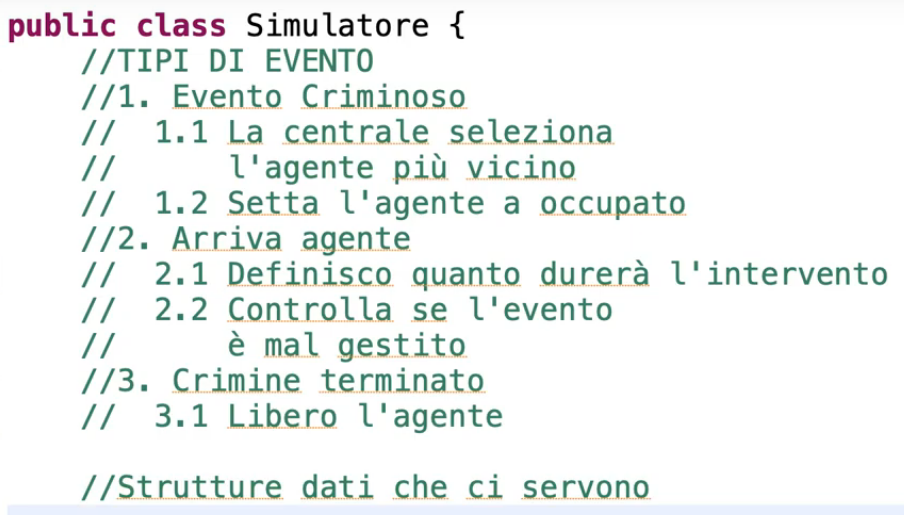
Possiamo passare al secondo punto.

Immagine che contiene screenshot

Descrizione generata automaticamente

Questo punto richiede una simulazione quindi andiamo subito a creare la classe Simulatore e la classe Evento. Non confondere la classe “Evento” appena creata con la classe “Event” già presente nel programma.

Partiamo dal Simulatore e cerchiamo di gestire i seguenti punti:



Per quanto riguarda le strutture dati, ci viene richiesto dal testo il numero di eventi mal gestiti. I parametri da gestire sono invece il numero di agenti e la data. Per gestire quest’ultima, creiamo 3 variabili Integer (giorno, mese, anno).

Creiamo successivamente una Mappa che collega Agenti e Distretti. Questo perché, oltre a sapere quanti sono gli agenti liberi, noi vogliamo anche sapere in che distretto questi agenti sono liberi.

Come fa la centrale a scegliere l’agente più vicino?

Dovremo calcolare il cammino minimo tra tutti i vertici che hanno degli agenti liberi e il vertice in cui è avvenuto il crimine. Vedremo come fare la prossima lezione.

FINE.