# Home Work 4 – 19/10/2018 Sergio Gabriele Santoro

DETTAGLI IMPLEMENTATIVI:

Il pattern **Data Access Object** (DAO) è un **pattern architetturale** utilizzato per separare i servizi di business dell’application processing layer dalle operazioni di accesso ai dati del data management layer.

Il DAO implementa il meccanismo di accesso richiesto per lavorare con la sorgente dei dati, che potrebbe un RDBMS, un servizio esterno o un repository di file. Il modulo client (l’application processing layer che contiene la business-logic) utilizza l’interfaccia esposta dal DAO. Il Data Access Object nasconde completamente i dettagli dell’interazione con la sorgente dati. L’interfaccia esposta dal DAO al client non cambia quando l’implementazione dell’origine dati sottostante cambia, e questo consente al DAO di adattarsi a diversi schemi di archiviazione senza dover modificare nulla sugli altri layer. In sostanza, il DAO funge da adapter tra il componente della business logic e l’origine dati. Il vantaggio relativo all'uso del DAO è dunque il mantenimento di una rigida separazione tra le componenti di un'applicazione, le quali potrebbero essere il "Modello" e il "Controllo" in un'applicazione basata sul paradigma MVC (Model View Controller).

L’esercitazione comprende lo sviluppo nel package **it.soprasteria.homework4.santoro** di una classe ConnectionManager responsabile della connessione al db.

Nel package **it.soprasteria.homework4.santoro.model** vengono sviluppate delle classi con relativi metodi il cui scopo è quello di rappresentare entità tabellari del nostro db nel nostro caso:

.Candidato.java

.Telefono.java

.Attivita.java

.TitoloStudio.java

. Esperienza.java

.Istruzione.java

Al fine di determinare un maggiore livello di astrazione e manutenibilità. Le classi sono provviste di costruttori e metodi get e set.

Successivamente nel package **it.soprasteria.homework4.santoro.dao** vengono sviluppate le interfacce relative che contengono la definizione dei metodi astratti che permetteranno alle classi che le implementano appartenenti al package **it.soprasteria.homework4.santoro.dao**.i**mpl** di eseguire le operazioni CRUD a tal scopo vengono definite le seguenti interfacce:

.AttivitaDao.java

.CandidatoDao.java

**.**EsperienzaDao.java

.IstruzioneDao.java

.TelefonoDao.java

.TitoloStudio.java

Le classi invece presenti nel package **it.soprasteria.homework4.santoro.dao**.i**mpl** che implementano le interfacce CandidatoDao.java e TelefonoDao.java necessarie per lo svolgimento dell’esercizio sono:

.Default\_Candidato\_Dao.java

.Default\_Telefono\_Dao.java

Queste classi danno un’implementazione ai metodi astratti dichiarati nelle interfacce.

Nella classe Default\_Candidato\_Dao.java sono stati quindi implementati i metodi seguenti:

**public void insert (Candidato candidato) -> permette di inserire un candidato nella tabella Candidato**

**public Candidato SelectByTipo(int codice) ->seleziona il candidato con quel codice dalla tabella Candidato**

**public void update(int codice,String viaResidenza,String capResidenza,String cittaResidenza) ->Permette di aggiornare i campi del candidato con quel particolare codice**

**public LinkedList<String> titoloStudiCandidati() ->Restituisce il titolo di studio di ogni candidato**

**public LinkedList<Candidato> selectAll() -> Restituisce tutti i candidati presenti nella tabella candidati**

Mentre nella classe Default\_Telefono\_Dao.java sono stati quindi implementati i metodi seguenti:

**public void insert (Telefono telefono) -> permette di inserire un telefono nella tabella Telefono**

**public Telefono selectByTipo(int numero)->permette di selezionare la tupla nella tabella con quel particolare numero telefonico**

**public void update(int numero,Telefono telefono)->Permette di cambiare il Candidato associato al numero di telefono nella tabella telefono**

Il client è caratterizzato dalla classe App\_Santoro.java presente nel package **it.soprasteria.homework4.santoro** nella quale è stato realizzato un menu che informerà l’utente delle operazioni che è possibile eseguire:

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MENU HOMEWORK4 SANTORO SERGIO GABRIELE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**UTENTE SELEZIONA FRA UNA DELLE SEGUENTI SCELTE**

**1 : INSERIMENTO DI UN CANDIDATO E DEL SUO NUMERO TELEFONICO**

**2 : RICERCA CANDIDATO TRAMITE CODICE**

**3 : VISUALIZZAZIONE DI TUTTI I TITOLI DI STUDIO DI UN CANDIDATO**

**4 : AGGIORNAMENTO RESIDENZA DI UN CANDIDATO**

**5 : STAMPA CANDIDATI**

**6 : FINE**

E’ possibile eseguire l’inserimento di un candidato selezionando la scelta 1:

Il sistema permetterà di aggiungere un candidato e il suo numero di telefono solo se il nuovo codice non è presente nella tabella e il numero che si sta cercando di inserire non sia già presente in rubrica (non ci possono essere due o più persone con lo stesso numero), mi avvalgo quindi della classe Default\_Candidato\_Dao.java descritta precedentemente. Nel caso in cui tale codice non sia presente nella tabella il sistema chiederà all’utente di inserire tutte le informazioni necessarie all’inserimento (nome, cognome, via residenza, telefono…) Se il codice è presente nella tabella Candidato viene segnalato all’utente l’impossibilità di inserire un nuovo candidato, cosi’ come se si cerca di inserire un numero già presente.

E’ possibile eseguire l’operazione di ricerca selezionando la scelta 2:

Il sistema permetterà di ricercare un particolare candidato nella tabella candidati digitando il tasto 2 infatti il sistema chiederà all’utente di inserire il codice del candidato che vogliamo ricercare nel sistema. Se presente riusciremo a visualizzare a video tutte le informazioni dello stesso altrimenti il sistema segnalerà l’inesistenza dello stesso all’utente tramite una stampa a video.

E’ possibile ricercare tutti i titolo di studio dei candidati digitando il tasto 3:

Il sistema in questo caso mostrerà all’utente l’elenco degli utenti con i rispettivi titoli di studio se nella tabella Candidato non ci sono candidati il sistema informerà l’utente della tabella vuota

E’ possibile aggiornare la residenza di un particolare candidato digitando il tasto 4:

In questo caso il sistema richiederà di inserire il codice del candidato che vogliamo ricercare se il candidato è presente il sistema mi chiederà di inserire i nuovi campi da modificare altrimenti segnalerà all’utente la non presenza del candidato cercato nella tabella;

E’ possibile stampare tutti i candidati presenti nella tabella candidati digitando il tasto 5:

Il sistema stamperà l’elenco dei candidati presenti nella tabella se non ci sono candidati verrà avvertito l’utente.

E’ possibile terminare l’esecuzione del programma digitando il tasto 6.

DETTAGLI BRANCH

Stacco da develop un nuovo branch di nome Sergio\_Santoro\_Homework4\_Curriculum con il comando

git checkout –b Sergio\_Santoro\_Homework4\_Curriculum e mi posiziono su di esso

Eseguo l’operazione git pull in modo tale da aggiornare il mio branch con il develop

Eseguo l’operazione git add . che permette di passare il mio lavoro nell’area Stage

Eseguo l’operazione git commit -m “messaggio” che permette di memorizzare i file in locale e di aggiungere gli stessi nell’area head che punta all’ultimo commit

Per salvare i miei aggiornamenti in remoto eseguo git push origin Sergio\_Santoro\_Homework4\_Curriculum

Accedo a git ed eseguo una new pull request del mio branch su develop