**IL TEST DEL SOFTWARE**

# Analisi

Si vuole realizzare un’applicazione software denominata Progetto Presenze si basa sul funzionamento di una comune timbratrice, avendo inoltre un archivio consultabile di tutti gli orari dei giorni passati (quindi di orari lavorativi già conclusi e non in corso) fino ad un massimo di 30 giorni oltre i quali il tutto sarà salvato su file.

* Eseguire la timbratura (inizio lavoro)
* Eseguire una timbratura di uscita (fine lavoro)
* Visualizzare tutti i dipendenti timbrati (in tutti e 2 gli ordini cronologici)
* Visualizzare tutte le timbrature (anche quelle passate quindi dei dipendenti non presenti oggi)



* Visualizzare i dati relativi ad un dipendente (tramite il numero di matricola) in un dato giorno.



* Eliminazione di tutti i dati di 1 dipendente tramite numero matricola (con annesso salvataggio su apposito file)

Scelte progettuali:

Le classi VisualizzaRiordinato, SaveRestore, Menu e Controlli sono classi con metodi statici che permettono in ordine di:

* Visualizzare i dipendenti presenti in ordine cronologico.
* Salvare e restaurare i dati, rendendo il programma dotato di un semplicissimo database.
* Gestire le varie voci del menu in modo da non intasare il Main e avere sempre a disposizione tutto ciò di cui si ha bisogno.
* Eseguire controlli, che principalmente sono 2: il primo per verificare che l’input sia effettivamente numerico per evitare eccezioni di mismatching non desiderate e il secondo è un controllo che permette di assicurarsi che l’utente non timbri un’entrata mentre è già presente (così da evitare la pluritimbratura).

Test unitari

# Test della classe Presenza (indica la singola presenza)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Nome** | | **Descrizione** | | **Comando** |
|  |  |  |  | **Test Costruttori** | |
|  | Test del | | Creare una timbratura. | | Presenza P0=new  Presenza(int, int, oraIngresso) |
|  | costruttore | | Richiede unicamente l’inserimento del numero di amtricola, il resto sarà gestito dal programma | |  |
|  | Test del costruttore  vuoto | | Creare una timbratura vuota. | | Presenza P0=new Presenza() |
|  |  |  |  | **Test Getter e Setter** | |
|  | Getter | | GetcodiceidentificativoPresenza    GetNumeroMatricola      GetOraIngresso        GetOraUscita | | P0.get… |
|  | Setter: | | Vengono testati nei Getter. | | Si crea una timbratura vuota si settano i dati coi Setter e li si ottengono coi Getter, così si testano entrambi in un test unico. |

# Test della classe Presenze (Lista)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nome** |  | | **Descrizione** | | | **Comando** |
|
|  |  | |  |  | **Test costruttore** | |
| Costruttore |  | | | Crea una lista vuota pronta per essere popolata. | | Presenze() |
|  |  | |  |  | Test getter e altri metodi | |
| GetPresenza |  | | | Ottiene 1 oggetto dalla lista (estraendo dalla lista l’informazione codificata nell’attributo info); richiede un intero che indica la posizione in cui cercare. | | Lista.GetPresenza(int) |
| GetElementi |  | | | Ottiene il numero di elementi presenti nella lista, sfruttando l’attributo Elementi che nella nostra lista è stato appositamente creato. | | Lista.GetElemnti() |
| InserisciInTesta | |  | | Necessita di un oggetto (elemento) da porre nella lista, in testa. Viene anche invocato da  InserisciInPosizione | | Lista.InserisciInTesta(Object); |
| InserisciInCoda | |  | | Necessita di un oggetto (elemento) da porre nella lista, in coda. Viene anche invocato da  InserisciInPosizione | | Lista.InserisciInCoda(Object); |
| InserisciInPosizione | |  | | Permette di inserire un oggetto in qualsiasi posizione della lista. (eccezioni: se la posizione risulta essere l’ultima viene invocato inserisci in coda); necessita oltre all’oggetto della posizione come intero | | Lista.InserisciInPosizione(Object, int); |
| EliminaInTesta | |  | | Elimina il primo oggetto della lista. Viene anche invocato da  EliminaPosizione | | Lista.EliminaInTesta(); |
| InserisciInCoda | |  | | Elimina l’ultimo oggetto della lista. Viene anche invocato da  EliminaInPosizione | | Lista.EliminaInCoda(); |
| EliminaInPosizione | |  | | Permette di eliminare un oggetto in qualsiasi posizione della lista. (eccezioni: se la posizione risulta essere l’ultima viene invocato inserisci in coda); necessita della posizione come intero | | Lista.EliminaInPosizione(int); |

13

# TEST DI INTEGRAZIONE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | Azione da svolgere | Eccezioni gestite | Eccezioni non gestite | Verifica finale |
| 0 | Aggiungere una presenza | Dipendente già timbrato; Inserimento di tipologia sbagliata (String al posto di Int). | Nessuna | Tutto completo. |
| 1 | Timbrare un’uscita | Dipendente non timbrato in entrata; Inserimento di tipologia sbagliata. | Nessuna | Tutto completo. |
| 2 | Visualizzazione presenti (in ordine cronologico e cronologico inverso) | Nessuna eccezione da gestire | / | Tutto completo. |
| 3 | Visualizza tutti i dati nella lista  (timbrature vecchie fino a 30 giorni) | Nessuna eccezione da gestire. | / | Tutto completo. |
| 4 | Eliminazione manuale di tutti i dati di 1 dipendente tramite numero matricola | Numero matricola non presente nella  lista;  Inserimento di tipologia sbagliata. | Nessuna | Tutto completo. |
| 5 | Ricerca in base a data e numero di matricola | Numero matricola non presente nella  lista;  Inserimento di tipologia sbagliata; Data inserita non valida in quanto l’anno è postumo all’attuale anno, il mese o il giorno sfora i canoni. | Nessuna | Tutto completo. |
| 6 | Salva su file | Tiene conto dei possibili errori relativi all’input e output su file. | Nessuna | Tutto completo. |
| 7 | Carico dal file | Tiene conto dei possibili errori relativi all’input e output su file. | Nessuna | Tutto completo. |
| 8 | Elimino e salvo su file le timbrature più vecchie di 30 giorni | Tiene conto dei possibili errori relativi all’input e output su file. | Nessuna | Tutto completo. |

DIAGRAMMA DELLE CLASSI

