Programmazione ad Oggetti

a.a. 2016/2017

Client Overview

Giovanni Motterle

Matricola: 1097626

Introduzione

L'obbiettivo di questo progetto è lo sviluppo in C++/Qt di un sistema di memorizzazione dei clienti di un hotel, che permetta la fruizione di varie funzionalità a seconda della tipologia dell'utente autenticato.

Quest'ultimo può essere un amministratore, un utente premium oppure un utente base.

Il primo può sia leggere che modificare il contenitore, in particolare può aggiungere e rimuovere gli utenti e i clienti; inoltre può modificarne i dati e salvare le modifiche.

L'utente premium, come quello base può accedere al contenitore in sola lettura, in particolare l'utente premium può vedere una panoramica di tutti i clienti ed eseguire una selezione di questi secondo specifici parametri.

L'unica funzionalità a disposizione dell'utente base è quella della visualizzazione generica dei clienti senza possibilità di ricerca.

L'hotel in questione è caratterizzato da tre tipologie di clientela:

- il cliente di ristorante, non pernotta nella struttura ma usufruisce solo del ristorante per pranzi o cene.
- Il cliente della Spa, neanche lui pernotta nella struttura ma usufruisce dei servizi offerti dalla Spa dell'hotel come la palestra, la piscina ecc.
- il cliente d'hotel, risiede almeno una notte in hotel ed inoltre può usufruire dei servizi della Spa e del ristorante.

I dati sono salvati su due file xml: clienti.xml e utenti.xml.

All'avvio del programma vengono caricati automaticamente, e in caso di situazioni eccezionali (es. mancanza dei suddetti file) l'utente riceve alcune informazioni tramite una finestra di dialogo.

Il salvataggio permanente su file delle modifiche deve avvenire sempre tramite la funzione Salva → Salva, abilitata solo per l'utente amministratore dato che è l'unico con i permessi di scrittura. Nel caso in cui si tenti di uscire in un momento in cui esistono modifiche non salvate, si riceve un messaggio d'avvertimento.

Questo progetto è stato sviluppato su Ubuntu 16.04 LTE con:

- compilatore g++ versione 5.4.0

- libreria Qt versione 5.5.1

Gerarchie di tipi

1) Parte logica

- Abstract_Client è la classe base astratta da cui derivano tutte le classi che rappresentano i clienti. Gestisce le caratteristiche comuni a tutti, come il nome, cognome, codice fiscale e indirizzo.

  E' astratta in quanto dichiara un metodo virtuale puro , costo_totale( ), che calcola il totale pagato da ogni cliente.

- Restaurant_Client deriva virtualmente e pubblicamente da Abstract_Client e rappresenta il cliente di ristornate. Si distingue per la presenza di un campo dato privato che incapsula l'insieme di tutti gli scontrini (Lista_Scontrini).

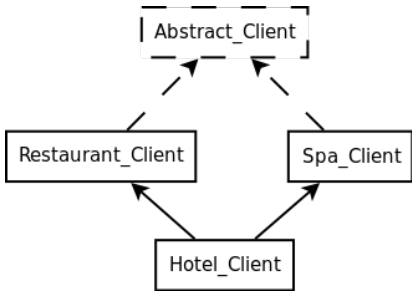
  Definisce un overriding di costo_totale( ).

- Spa_Client deriva virtualmente e pubblicamente da Abstract_Client e rappresenta il cliente della spa. Si distingue per la presenza di un campo dato privato che incapsula l'insieme di tutti i servizi usufruiti (Contenitore_Servizi).

  Definisce un overriding di costo_totale( ).

- Hotel_Client deriva pubblicamente da Spa_Client e Restaurant_Client e rappresenta il cliente d'hotel che risiede nell'hotel per almeno una notte.

  E' infatti caratterizzato, oltre3 che dall'insieme di scontrini e da quello dei servizi, da una data di check-in e una di check-out.

  Definisce un overriding di costo_totale( ).

  Queste classi usufruiscono delle funzionalità di alcune classi "secondarie" come:

  - Indirizzo
  - Data
  - Lista_Scontrini

- Contenitore_Servizi

- Utente rappresenta un utente del programma, è caratterizzato da uno username, una password e una tipologia.
- Clienti è il contenitore principale del progetto, si occupa di collezionare i clienti attraverso una lista di nodi, ognuno con un puntatore ad un Abstract_Client allocato sullo heap. Permette l'aggiunta in coda alla lista e la rimozione di un nodo in tempo costante, oltre che l'iterazione del suo contenuto mediante la classe Iteratore definita internamente.
- Utenti è il contenitore di tutti gli utenti riconosciuti dal programma, offre funzionalità come aggiunta, rimozione e verifica di esistenza di un utente oltre che modifica dei campi dati e loro lettura

2) Parte grafica

- MainWindow è la finestra principale; oltre che genitore delle tre gui per le tre tipologie di utente, possiede anche un puntatore al contenitore dei clienti ed uno a quello degli utenti.

  Si occupa di invocare i metodi di caricamento dei dati da file xml e di tenere traccia di eventuali modifiche al contenuto dei contenitori non ancora salvate su file.
- AutenticazioneDialog deriva da QDialog e permette, attraverso un metodo di verifica della classe Utenti, l'autenticazione.
- TabWidgetAmministratore deriva da QTabWidget, è l'interfaccia utilizzabile da un amministratore e consiste di tre tab:
  - il primo è una panoramica dei clienti con possibilità di eliminazione o apertura dei dettagli.

    La visualizzazione dei dettagli e loro modifica è implementata grazie a dettagliCliente, che deriva da QDailog e in base al tipo dinamico del cliente puntato mostra la lista degli scontrini e/o quella dei servizi usufruiti oltre a dei pulsanti per ulteriori aggiunte (aggiungiScontrino e aggiungiServizio) e il prezzo totale pagato.
  - Il secondo mostra tutti gli utenti riconosciuti con possibilità di modifica e rimozione (ad eccezione dell'ultimo amministratore che non è possibile rimuovere).

        - Il terzo prevede di effettuare una selezione dei clienti in base alla loro tipologia, prezzo pagato, e area geografica di provenienza.
    - TabWidgetPremium deriva da QTabWidget, è l'interfaccia utilizzabile da un utente premium (es. receptionist) e consiste di due tab:
        - l primo è una panoramica dei clienti con la sola possibilità di lettura delle informazioni generali
        - il secondo è lo stesso tab di ricerca dell' utente amministratore.
    - TabWidgetBase deriva da QTabWidget, è l'interfaccia utilizzabile da un utente base (es. HouseKeeper) e consiste di un solo tab, ovvero la stessa panoramica dei clienti dell'utente premium.

      Come già detto, l'amministratore può aggiungere sia clienti che nuovi utenti, quindi sono state definite ulteriori due classi:

    - aggiungiNuovoCliente deriva da QDialog e permette, in base alla tipologia di cliente selezionato, di inserire le informazioni richieste.
    - aggiungiNuovoUtente deriva da QDialog e permette di selezionare il tipo di utente e inserire username e password.

Polimorfismo

Il codice polimorfo riguarda il calcolo del totale pagato da ogni cliente.

I servizi sono implementati con una gerarchia di altezza 1 e ampiezza 5. La classe Servizio_astratto dichiara un metodo virtuale puro (getCosto( )) che è poi implementato differentemente nelle classi derivate. Le chiamate polimorfe a questo metodo sono usate nel calcolo totale dei servizi usufruiti da parte della classe Contenitore_Servizi.

Ogni classe della gerarchia di clienti definisce un overriding del metodo virtuale puro costo_totale( ) in Abstract_Client.

L'overriding in Restaurant_Client ritorna il totale calcolato da Lista_Scontrini (unico campo dato proprio)

Quello in Spa_Client ritorna il costo totale dei servizi calcolato da Contenitore_Servizi.

L'ultimo overriding (quello in Hotel_Client) richiama le definizioni delle sue superclassi dirette e aggiunge un costo fisso moltiplicato per il numero di giorni di residenza del cliente. Il valore di ritorno di questo metodo è usato in dettagliCliente.

Credenziali d'accesso

E' possibile autenticarsi con ognuno dei tipi di utenti, in particolare:

Amministratore: username → Admin password → Admin

Premium: username → Premium password → Premium

Base: username → Base password → Base

Fasi progettuali

Una prima parte non trascurabile e difficilmente quantificabile di tempo è dovuta al processo di apprendimento dei primi rudimenti del framework Qt e alla decisione del "topic" principale del progetto.

- Progettazione e codifica del modello 30h

  in particolare la decisione del formato di memorizzazione e la codifica del "modulo" di caricamento e salvataggio dei dati non è stato banale.

- Progettazione e codifica GUI 20h

  non essendo possibile usare i file .ui generati da Qt Desiner, è stato necessario un periodo iniziale per comprendere i meccanismi propri di Qt.

- Debugging 4h
- Testing 2h
- Relazione 3h

.