

EVENTMANAGER

Gestionale di eventi



Lorenzo Piran
(1193526)

INDICE

- Prefazione
 - Progetto e finalità
 - Strumenti utilizzati
 - Minutaggio impiegato
 - Materiale consegnato
- Custom Container e DeepPtr
- Gerarchia
 - Modello
 - Event
 - Commercial Fair
 - Wedding
 - Bachelor Party
 - Marriage
 - Sport
 - Tournament
 - Marathon
 - RatingError
- Gui

PREFAZIONE

Progetto e finalità

Lo scopo del progetto era quello di creare e sviluppare in C++/Qt un gestore di eventi per un'azienda che si occupa di creazione e gestione per conto di terzi di eventi.

Strumenti utilizzati

La codifica di C++ relativa al modello della gerarchia principale è stata realizzata mediante l'ausilio dell' IDE CLion.

In seguito per l'implementazione della parte grafica abbiamo utilizzato il framework qt.

Tutto infine è stato testato con la macchina virtuale con Ubuntu fornita dal professore.

Minutaggio Impiegato

- 7 ore di lezioni seguite sul tutorato/ ricerche relative al framework qt
- 3 ore per l'analisi del problema e della possibile soluzione iniziale
- 2 ore studio autonomo sul framework qt
- 20 ore per la codifica della gerarchia
- 15 ore per la codifica della gui
- 5 ore per i test e i relativi fix da apportare all'applicativo

Le 50 ore sono state di poco sfiorate per problemi dovuti alla risoluzione di alcuni problemi/studio librerie qt

Materiale Consegnato

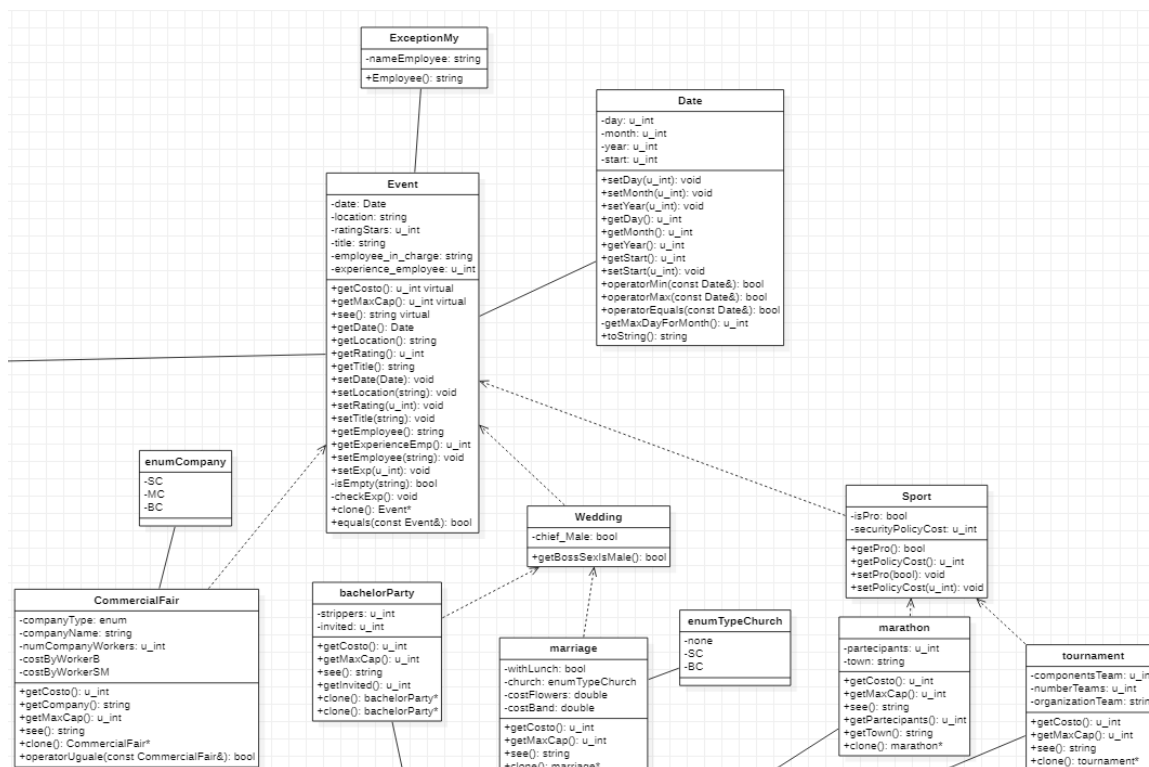
La cartella consegnata contiene:

1. Tutti i file .h e .cpp
2. Una cartella Resources con un'immagine per il contrassegno dell'evento nel calendario
3. Il file .pro, con il quale generare il MakeFile

CUSTOM CONTAINER E DEEPPTR

Come da richieste del progetto è stato implementata una classe DeepPtr<T> per implementare una gestione della memoria “profonda” e polimorfa ai puntatori a T.

GERARCHIA



In UML questa è la stesura della nostra gerarchia, relativa ad un evento e le sue proprietà.

Event

Si tratta della classe base, astratta e polimorfa della gerarchia. Rappresenta un generico evento creato e organizzato dall' azienda posseditrice del nostro applicativo. La classe è composta da attributi e metodi virtuali che le classi derivate andranno ad ereditare.

Attributi:

- Location (dove avviene l'evento)
- Title (univoco per ogni evento)
- Employee(nome impiegato)

- Rating stars (valutazione della difficoltà dell' evento)
- experienceEmployee (valutazione impiegato che deve esse >= del Rating stars)
- date (nostra classe che rappresenta la data dell'evento)

Metodi:

- Getter e setter dei vari campi dati
- getCosto (virtuale pura, che ritorna il costo dell'evento a seconda della tipologia e di altri parametri delle classi derivate)
- getMaxCap(virtuale pura, pensata in un momento delicato come questo in cui il monitoraggio del quantitavi di persone presenti ad un evento è quanto mai fondamentale per garantire la sicurezza)
- see(virtuale pura, che ritorna una specie di sunto relativo a tutte le info più importanti da stampare poi nella label presente a schermo)
- checkExp(fa un controllo su esperienza impiegato e rating evento)
- clone(virtuale pura, clona oggetto che la richiama)
- read(virtuale, per leggere attributi da file)
- write(virtuale, per scrivere su file)
- ridefinizione operatore di uguaglianza

Commercial Fair

Classe derivata da Event, rappresenta un evento commerciale delegato da un azienda, per i propri impiegati/interessi, alla nostra azienda (quella acquirente dell'applicativo).

Con Attributi:

- Type(enum sulla tipologia/grandezza impresa)
- companyName
- numWorkers
- costByWorkerBig(campo dati statico)
- costByWorkerSM(campo dati statico)

Più in specifico il campo dati Type viene rappresentato da un enum contenete i campi:

1. SC
2. MC
3. BC

La classe poi oltre all'implementazione di metodi di get e set, viene poi ridefinita il metodo clone, viene poi ridefinito il metodo getCosto() che ritorna il costo dell'evento a seconda dell'azienda e dei suoi lavoratori, vengono ridefiniti poi i metodi see() che ritorna le informazioni più importanti della fiera e il metodo getMaxCap() sempre per un discorso relativo alla sicurezza e alla capienza massima.

Wedding

Classe a sua volta astratta e derivata da Event, rappresenta una istanza di un evento relativo ad un matrimonio con campi dati:

- isMale(booleano che ci dice se il commitente è di genere maschile o femminile)
- guest(unsigned int che ci dice il numero degli invitati)

come metodi invece abbiamo i vari getter e setter relativi ai campi dati della classe, vengono poi ridefiniti read e write ma non i metodi virtuali puri della classe base Event, ciò rende quindi anche Wedding una classe a sua volta astratta.

Bachelor Party

Classe concreta derivata dalla classe intermedia Wedding, rappresenta l'addio al nubilato organizzato da un uomo o una donna prima del proprio matrimonio, la classe come attributi contiene:

- dancers(numero di ballerine)
- priceForDancer(prezzo da pagare relativo ad ogni ballerina)
- priceForGuest(costo unitario relativo all'invitato)

come metodi invece, essendo la classe concreta vengono implementati tutti i metodi virtuali puri delle classi superiori come getCosto() che ritorna il costo dell'evento a seconda dell'azienda e dei suoi lavoratori, vengono ridefiniti poi i metodi see() che ritorna le informazioni più importanti della fiera e il metodo getMaxCap() sempre per un discorso relativo alla sicurezza e alla capienza massima.

Vengono poi implementati i due gettere e setter relativi alle ballerine, viene ridefinito l'operatore di uguaglianza, il metodo clone e i metodi read e write, per la lettura e scrittura su file dei dati.

Marriage

Classe concreta derivata dalla classe intermedia Wedding, rappresenta l'evento relativo al matrimonio e quindi al giorno delle nozze organizzato da un uomo o una donna, la classe come attributi contiene:

- lunch(booleano che indica se nell'evento è prevista anche l'organizzazione e quindi il pagamento di un pranzo di matrimonio)
- church(enum)
- numFlower(unsigned int che dice il numero di fiori da comprare e utilizzare nel matrimonio)
- numWaiters(unsigned int che dice il numero di camerieri)
- flowerPrice(campo dati statico double riguardante il prezzo di un fiore)

- bandPrice(campo dati statico double riguardante il prezzo della band)
- waitersPrice(campo dati statico double riguardante il prezzo del servizio di un cameriere)
- priceForGuest(campo dati statico double riguardante il prezzo unitario per invitato)

come metodi invece, essendo la classe concreta vengono implementati tutti i metodi virtuali puri delle classi superiori come getCosto() che ritorna il costo dell'evento a seconda dell'azienda e dei suoi lavoratori, vengono ridefiniti poi i metodi see() che ritorna le informazioni più importanti della fiera e il metodo getMaxCap() sempre per un discorso relativo alla sicurezza e alla capienza massima.

Vengono poi implementati i gettere e setter relativi, viene ridefinito l'operatore di uguaglianza, il metodo clone e i metodi read e write, per la lettura e scrittura su file dei dati.

Sport

Classe a sua volta astratta e derivata da Event, rappresenta una istanza di un evento relativo ad un evento sportivo, con committente quindi una società sportiva o un'organizzazione per l'organizzazione di una competizione, con campi dati:

- isPro(booleano che ci dice se l'organizzazione dell'evento è fatto per professionisti o meno per un discorso di assicurazioni diverse che contengono quindi prezzi differenti che cambiano l'ammontare finale)
- securityPolicyCost(campo dati statico double che dice il costo unitario per sportivo riguardante l'assicurazione)

come metodi invece abbiamo i vari getter e setter relativi ai campi dati della classe, vengono poi ridefiniti read e write ma non i metodi virtuali puri della classe base Event, ciò rende quindi anche Sport una classe a sua volta astratta.

Tournament

Classe concreta derivata dalla classe intermedia Sport, rappresenta la creazione e gestione di un torneo sportivo, la classe come attributi contiene:

- componentsNumber(numero di componenti di tutte le squadre)
- teamNumber(numero di squadre)
- organizationTeam(nome della squadra organizzativa del torneo)
- tSport(enum)

L'enumerazione relativa al campo dati tSport contiene:

1. basket
2. Football

Ed in un'ottica di estensione dell'applicativo aumenteranno.

Come metodi invece, essendo la classe concreta vengono implementati tutti i metodi virtuali puri delle classi superiori come `getCosto()` che ritorna il costo dell'evento a seconda dell'azienda e dei suoi lavoratori, vengono ridefiniti poi i metodi `see()` che ritorna le informazioni più importanti della fiera e il metodo `getMaxCap()` sempre per un discorso relativo alla sicurezza e alla capienza massima.

Vengono poi implementati i getter e setter relativi, viene ridefinito l'operatore di uguaglianza, il metodo `clone` e i metodi `read` e `write`, per la lettura e scrittura su file dei dati.

Marathon

Classe concreta derivata dalla classe intermedia `Sport`, rappresenta la creazione e gestione di una maratona in una determinata città, la classe come attributi contiene:

- `participants`(numero di partecipanti alla maratona)
- `town`(città in cui si svolge la maratona)
- `length`(lunghezza della maratona a seconda dell'evento, può essere per esempio la metà dei classici 42.5 km per un evento maratona particolare)
- `subPrice`(Prezzo unitario per partecipante)

Come metodi invece, essendo la classe concreta vengono implementati tutti i metodi virtuali puri delle classi superiori come `getCosto()` che ritorna il costo dell'evento a seconda dell'azienda e dei suoi lavoratori, vengono ridefiniti poi i metodi `see()` che ritorna le informazioni più importanti della fiera e il metodo `getMaxCap()` sempre per un discorso relativo alla sicurezza e alla capienza massima.

Vengono poi implementati i getter e setter relativi, viene ridefinito l'operatore di uguaglianza, il metodo `clone` e i metodi `read` e `write`, per la lettura e scrittura su file dei dati.

RatingError

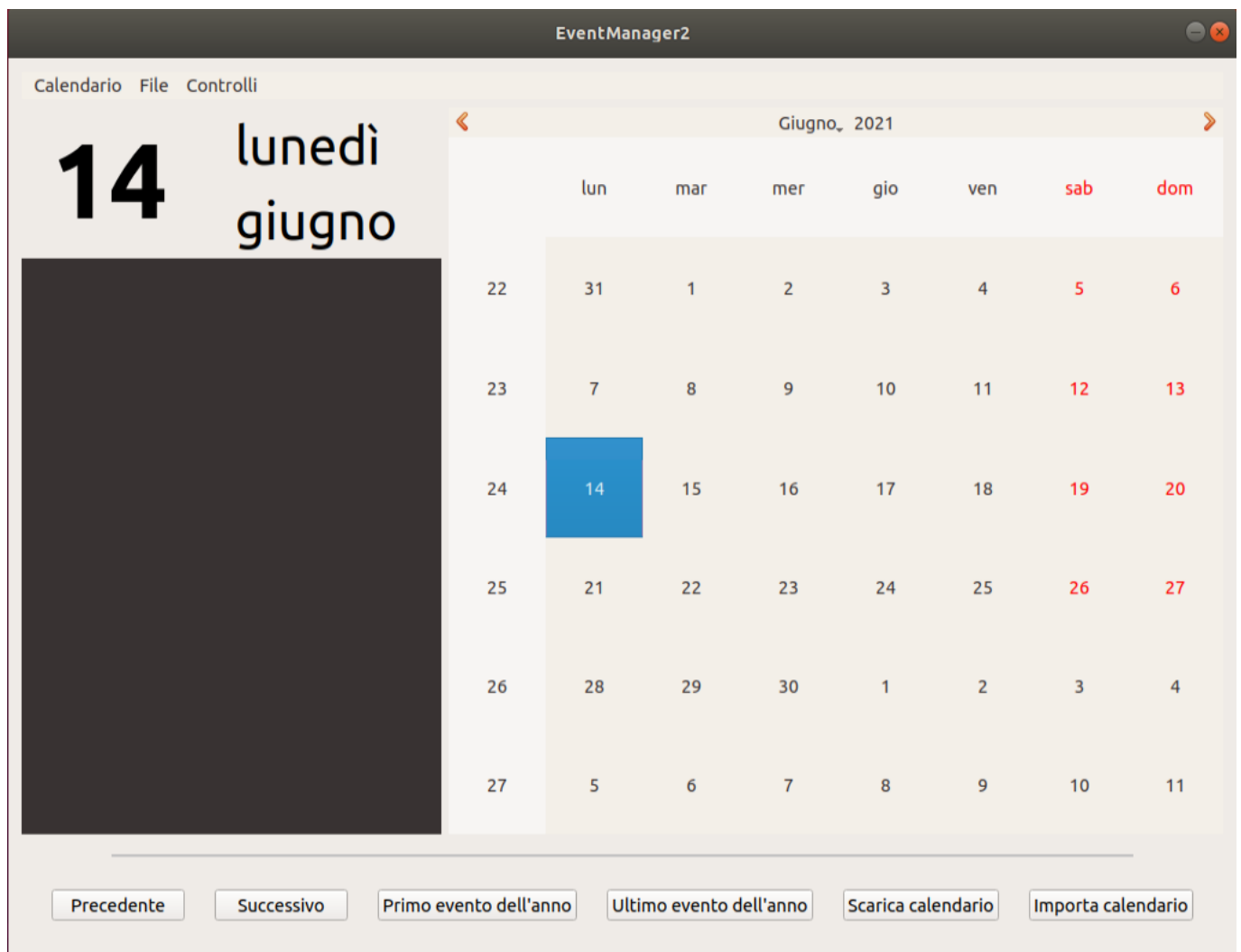
Classe di eccezione derivata, la classe come attributi contiene:

- `nameEmployee`(nome impiegato non sufficientemente esperto)

Come metodi invece è presente una funzione `Employee()` che ritorna una stringa che verrà poi visualizzata dall'utente tramite il dialog di errore.

GUI

MainWindow

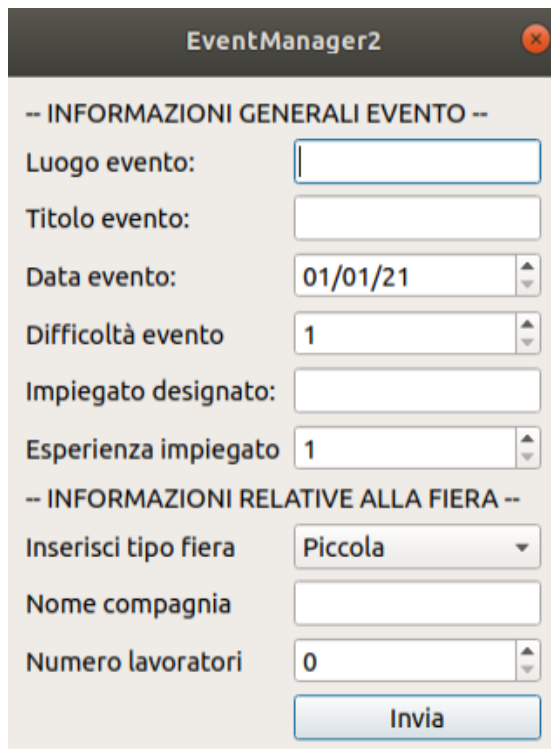


La gui del nostro applicativo si divide in 4 parti fondamentali, la prima in alto a sinistra con tre controlli Calendario che si suddivide in Esporta eventi (che li salverà nella cartella desiderata dall'utente), Importa Eventi (per importare dentro il nostro calendario gli eventi salvati in precedenza o mandati da un amico) e Termina Programma, dopodichè è presente File che si suddivide in Inserisci nuovo utente, Elimina un evento, elimina tutti gli eventi e Ricerca evento, e l'ultimo denominato Controlli contiene i controlli presenti poi anche nella parte inferiore per navigare tra le slides.

La seconda parte è quella centrale di sinistra con la label superiore che segna il giorno e quella inferiore che mostra i dati relativi all'evento.

La terza parte quella di destra è effettivamente il nostro calendario, e l'ultima parte quella inferiore ha i bottoni relativi alla navigazione tra gli eventi del calendario.

Inserimento Evento



The screenshot shows a dialog box titled "EventManager2" with a close button (X) in the top right corner. The dialog is divided into two sections: "INFORMAZIONI GENERALI EVENTO" and "INFORMAZIONI RELATIVE ALLA FIERA".

INFORMAZIONI GENERALI EVENTO

- Luogo evento:
- Titolo evento:
- Data evento:
- Difficoltà evento:
- Impiegato designato:
- Esperienza impiegato:

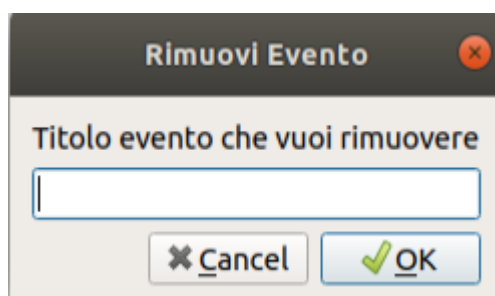
INFORMAZIONI RELATIVE ALLA FIERA

- Inserisci tipo fiera:
- Nome compagnia:
- Numero lavoratori:

At the bottom of the dialog is a button labeled "Invia".

Questo è un esempio di dialog da compilare per inserire manualmente un evento all'interno del calendario.

Eliminazione Evento



The screenshot shows a dialog box titled "Rimuovi Evento" with a close button (X) in the top right corner. The dialog contains a text input field labeled "Titolo evento che vuoi rimuovere".

Below the input field are two buttons: "Cancel" (with a red X icon) and "OK" (with a green checkmark icon).

Questo dialog invece serve per eliminare un evento tramite titolo(univoco) all'interno del calendario.