RELAZIONE PROGETTO PROGRAMMAZIONE AD OGGETTI

Gianluca Marraffa
1121080

danishBox

Indice

1 Introduzione

1.1 Scopo del progetto

danishBox è un gioco che simula momenti di vita comuni a chiunque, nella propria infanzia, abbia soggiornato nella casa dei propri nonni per brevi o prolungati periodi di tempo. In particolare l'esecuzione del programma riporta alla mente il fatidico momento della merenda, quando, di fronte ad una scatola di celebri biscotti danesi, si rimaneva delusi nello scoprire l'interno ricolmo di attrezzi da cucito.
In questa simulazione si possono interpretare 3 attori con compiti differenti:
Il ragazzo, che potrà aprire le scatole e mangiare i biscotti (in caso siano presenti) oppure riempirle con i propri giocattoli.
La nonna, che aprendo una scatola di biscotti potrà decidere se mangiarne alcuni e/o cucinarne altri. Questa inoltre potrà gestire le scatole di attrezzi da cucito e decidere di buttare una o più scatole vuote o con oggetti dentro.
Nonna e Ragazzo potranno creare dei nuovi oggetti, ma la prima si limita agli attrezzi da cucito, mentre il secondo avrà accesso solo alla creazione dei giocattoli.
L'amministratore, oltre a i "poteri" dei precedenti attori, potrà creare nuove scatole e modificare (il ruolo)/eliminare gli utenti.
È bene specificare che una scatola puo essere riempita solo con oggetti dello stesso tipo, quindi una scatola di danesi conterrà solo danesi, una di attrezzi da cucito solo attrezzi da cucito e una di giocattoli solo giocattoli.
In caso la scatola sia vuota, sarà possibile inserire qualsiasi tipo di oggetto (rispettando i vincoli nonna-cucito, ragazzo-giocattoli) che determinerà la tipologia della scatola per i prossimi inserimenti.
Un oggetto creato da un utente inoltre, è disponibile anche per tutti gli altri utenti(rispettando i vincoli sopradescritti).

1.2 Il contenitore: QJsonArray

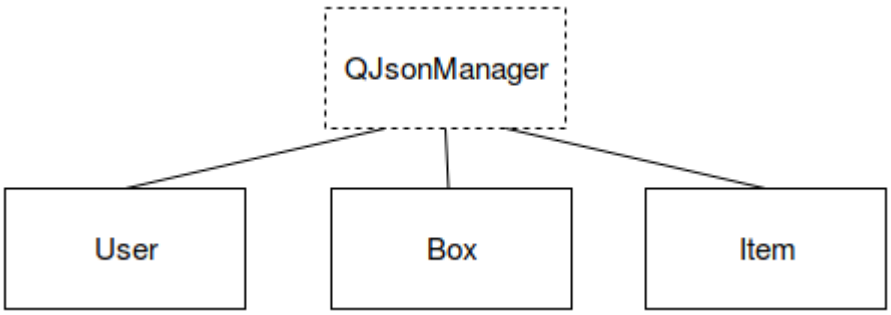
Per la lettura/scrittura da/su disco è stato scelto di utilizzare il contenitore QjsonArray offerto dalla libreria Qt. I file di memorizzazione sono quindi salvati con l'estensione *.json* e hanno la seguente struttura:
[{oggettoJSON}, {oggettoJSON}, …, {oggettoJSON}]
Il formato JSON permette le gestione di diversi tipi di dato senza particolari sforzi, risultando l'ideale per la memorizzazione di modelli diversi.
All'avvio del programma, tramite la classe QFile vengono letti dal disco i 3 file .json (users, items, boxes) contenenti i dati e memorizzati in ram nei rispettivi contenitori QJsonArray, definiti da proprietà statiche delle classi dei modelli.
In diversi punti del programma i contenitori in ram vengono scritti su disco nei rispettivi file.

2 Gerarchie Modello

2.1 Gestione Dati: JsonManager

L'oggetto base che gestisce l'interazione con il contenitore in memoria ed effettua le letture/scritture su/da disco. Tramite l'utilizzo di metodi virtuali e grazie alla versatilità del formato JSON è stato possibile mantenere la logica di interazione in memoria su questa classe.
Questa classe è astratta, non può essere istanziata.
È presente inoltre il distruttore virtuale.

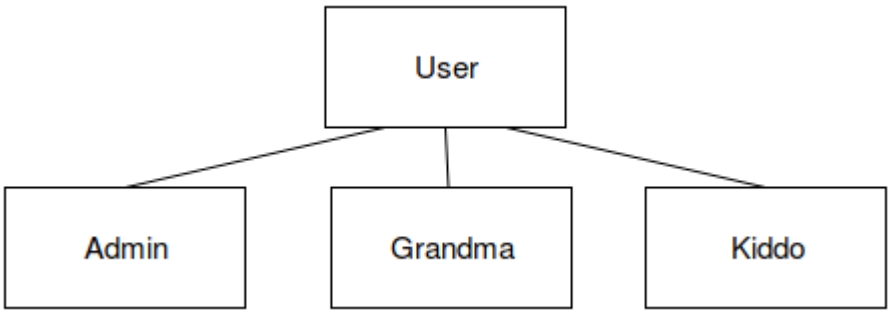
2.2 Modelli Base: User, Box, Item



Le 3 classi che ereditano direttamente da JsonManager, definiscono la propria conversione in JSON, il proprio contenitore e gestiscono la base della logica applicativa.
Queste classi implementano i metodi virtuali puri del loro padre, rendendone possibile l'istanziazione.
User rappresenta l'utente (name, surname, username, password) e, oltre alle definizione dei metodi di accesso e registrazione, contiene tutte le informazione base (tramite metodi e attributi) relative all'utente e definisce i permessi di accesso ai contenitori utenti e oggetti.

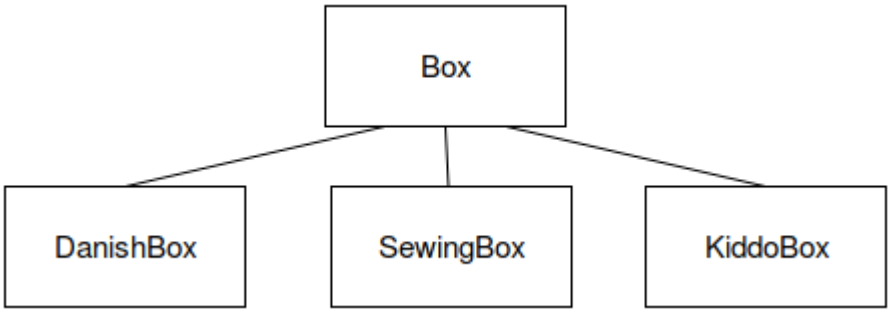
Box rappresenta le scatole (code, capacity) che vengono manipolate dall'utente e al suo interno è definito il contenitore logico degli oggetti (Item) e i metodi che servono per interagirci (push, pop). In questa classe sono definite le informazioni base delle scatole e dei metodi per facilitare l'utilizzo degli oggetti. Questa classe gestisce inoltre copia e distruzione profonda dei relativi oggetti istanziati.
Item rappresenta gli oggetti (code) che verranno inseriti nelle scatole.

2.3 Utenti: Admin, Kiddo, Grandma



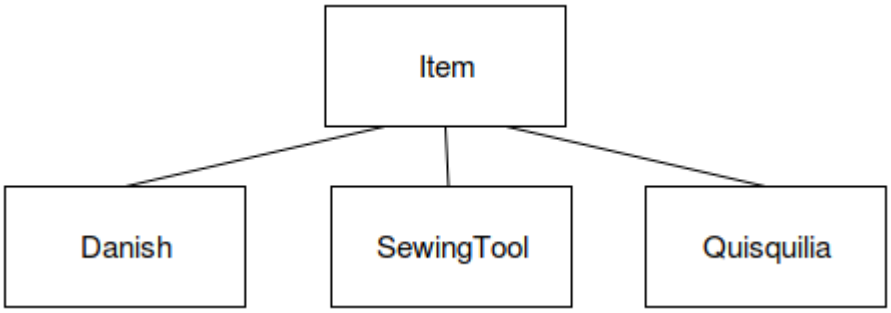
La classe utente si dirama in 3 composizioni che definiscono i 3 attori del programma:
Admin, l'amministratore
Kiddo, il ragazzo
Grandma, la nonna
Oltre a specificare le informazioni relative al tipo di utente, personalizzano l'accesso ai contenitori.

2.4 Scatole: DanishBox, KiddoBox, SewingBox



Come per gli utenti, anche Box si espande in 3 sotto classi:
DanishBox, la scatola di danesi
KiddoBox, la scatola di giocattoli
SewingBox, la scatola di attrezzi da cucito
Oltre a specificare le informazioni relative al tipo di scatola, controllano i vincoli di inserimento degli Item al loro interno.

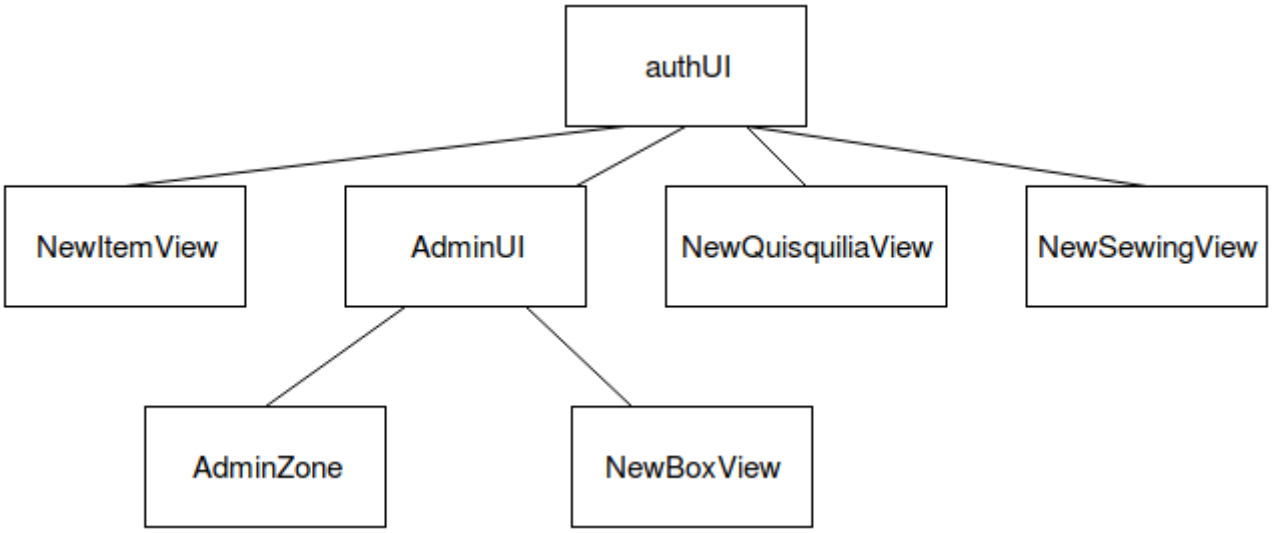
2.5 Oggetti: Danish, Quisquilia, SewingTool



L'ultima gerarchia di modelli discende da Item:
Danish, il biscotto danese
Quisquilia, il giocattolo
SewingTool, l'attrezzo da cucito
Queste sotto classi aggiungono attributi specifici per il tipo di oggetto e reimplementano la conversione in JSON.

Questi gli attributi aggiunti da ogni sotto tipo:
Danish→dimension, good
Quisquilia→name, affective
SewingTool→name, utility

3 Gerarchie GUI



3.1 Autorizzazione Base: authUI

Assieme alle varie viste, esiste una gerarchia che garantisce l'accesso autenticato alla vista e fornisce una procedura base per la popolazione delle finestre.
Questa classe prende il nome di *authUI* è presenta un messaggio di errore in caso l'utente non sia autenticato.

3.2 Finestre Admin: AdminUI, AdminZone, NewBoxView

La classe *AdminUI* discende da *authUI* reimplementa il metodo di autorizzazione e permette l'accesso alla vista solo ad utenti amministratori.
Da questa classe discendono *AdminZone* e *NewBoxView* che gestiscono rispettivamente la gestione degli utenti e la creazione di nuove scatole.

3.3 Autorizzazioni Specifiche: NewItemView, NewQuisquiliaView, NewSewingView

Da authUI discendono queste 3 viste che permettono accessi ad utenti specifici:
NewItemView è l'interfaccia che contiene le due viste successive.
NewQuisquiliaView permette l'accesso ad utenti *Admin* e *Kiddo* e permette la creazione di nuovi giocattoli.
NewSewingView permette l'accesso ad utenti *Admin* e *Grandma* e permette la creazione di nuovi oggetti da cucito.

4 Codice Polimorfo

4.1 JsonManager

virtual QString filename() const = 0;→Indica il percorso del file dove vengono salvati i dati, virtuale puro viene implementato da Box, User e Item per specificare i relativi percorsi.
virtual QJsonArray* getStorage() const = 0;→Ritorna il puntatore alll'oggetto QjsonArray che gestisce i dati in ram, ognuna delle classe figlie ha il suo specifico contenitore (statico per la classe).
virtual QJsonObject toJson() const;→Converte l'oggetto in QJsonObject per essere inserito nel contenitore, viene implementato da Box, User, Item, Danish, Quisquilia e SewingTool.
virtual void insert() const; →Utilizza il metodo toJson() per inserire un nuovo elemento nel contenitore
virtual QJsonObject remove(); →Rimuove l'oggetto dal contenitore in memoria e lo ritorna.
virtual bool checkUniq() const; →Controlla l'unicità di specifici attributi per evitare duplicati, viene reimplementato da Item e Box.
virtual bool create() const;→Utilizza la combinazione di checkUniq e insert per creare un nuovo oggetto da inserire nel contenitore.

4.2 Box
virtual bool pushItem(Item*); →Dato un item lo inserisce nel contenitore interno di box, ogni scatola (DanishBox, SewingBox, KiddoBox) implementa il controllo del vincolo dell'oggetto inserito.
virtual Item* popItem();→Rimuove l'Item dal contenitore interno e lo restituisce.
virtual Type type() const; →Ritorna il tipo di scatola per facilitarne l'utilizzo nelle viste, viene implementato da ogni tipo di scatola.(DanishBox, SewingBox, KiddoBox)
virtual QString itemHeader() const; →Ritorna una stringa che definisce le informazioni generali degli oggetti inseriti al suo interno, anche questo è un supporto per le viste e viene implementato da ogni tipo di scatola (DanishBox, SewingBox, KiddoBox).

4.3 Item
virtual QString infoLine() const;→Fornisce il valore degli attributi di Item separati da una virgola, ogni sottoclasse (Danish, Quisquilia, SewingTool) aggiunge i suoi attributi specifici.
virtual QString iconPath() const;→Fornisce il percorso per l'icona del tipo di oggetto, utile nelle viste. Ogni sottoclasse (Danish, Quisquilia, SewingTool) ha la sua personalizzata.
virtual Type getType() const;→ Ritorna il tipo di oggetto, utile per le viste. Ogni classe (Danish, Quisquilia, SewingTool) implementa il suo.
.
4.4 User
virtual QString iconPath() const; →Fornisce il percorso per l'icona del tipo di oggetto, utile nelle viste. Ogni sottoclasse (Admin, Grandma, Kiddo) ha la sua personalizzata.
virtual Type role() const;→Ritorna il ruolo dell'utente, utile nelle viste. Ogni sottoclasse (Admin, Grandma, Kiddo) implementa il suo.
virtual QVector<User*> availableUsers() const; → Ritorna un vettore con gli utenti che può visualizzare l'utente. Di default il vettore contiene una copia di se stesso, Admin invece lo implementa per ottenere tutti gli utenti salvati nel sistema.
virtual QVector<Item*> availableItems() const; → Ritorna un vettore con tutti gli Item disponibili all'utente. Di default torna un array vuoto. Admin può visualizzare tutti gli Item, Grandma solo gli attrezzi da cucito, Kiddo solo i giocattoli.
virtual User* clone() const;→Effettua una copia profonda dell'utente.

4.5 authUI
virtual void init();→ Combina i successivi metodi polimorfi per generare la vista.
virtual bool authorized() const; →Determina le politiche di autorizzazione per la vista. Viene implementato da Admin, NewItemView, NewQuisquiliaView e NewSewingView.
virtual void notAuthorizedUI();→Popola la vista in caso di fallimento dell'autorizzazione. authUI ne implenta una standard che viene rivisitata da NewQuisquiliaView e NewSewingView;
virtual void authorizedUI();→Popola la vista in caso di successo dell'autorizzazione. authUI fornisce una finestra vuota che viene rivisitata da AdminZone, NewBoxView, NewItemView, NewQuisquiliaView e NewSewingView.

5 Manuale Utente

5.1 Login
All'apertura del software si apre la finestra di login, all'interno sono già configurate le seguenti credenziali {username, password}:
{admin, admin}
{kiddo, kiddo}
{grandma, grandma}
In alternativa, cliccando su "Non sei iscritto" si aprirà la finestra per effettuare la registrazione.

5.2 Registrazione
La procedura di registrazione prevede i seguenti campi: nome, cognome, username, password, conferma password, tipo utente.
È possibile registrarsi come ragazzo o nonna.
Per effettuare la procedura correttamente è necessario compilare tutti i campi e inserire un username non presente in archivio, in caso di fallimento gli errori saranno segnalati sotto il pulsante di registrazione.
In caso di successo si accederà alla finestra di menu.
Sia per la finestra di login che di registrazione è possibile premere il pulsante invio al posto che cliccare sul bottone.

5.3 Menu

Una volta eseguita con successo la procedura di login/registrazione si aprirà la finestra di menu che avrà a disposizione i seguenti pulsanti:

Per tutti

- Apri scatole
- Crea oggetti
- Logout (Ritorna alla finestra di login)

Per Admin e Nonna

- Butta scatole

Solo per Admin

- Crea Scatole
- Gestione utenti

5.4 Apri scatole

Il gioco vero e proprio, in questa finestra è presente una lista di scatole con il rispettivo codice e colore. Cliccando su una si aprirà la finestra relativa alla scatola dove sarà possibile inserire/rimuovere oggetti o mangiare/cucinare biscotti, a seconda della scatola aperta.
L'apertura delle scatole rispetta i vincoli, se una nonna apre una scatola di giocattoli apparirà un messaggio di errore, così come in caso il ragazzo apra una scatola di oggetti da cucito.
In caso dell'apertura di una scatola vuota sarà possibile inserire un oggetto o cucinare un biscotto, con l'inserimento del primo elemento però, per i successivi bisognerà rispettare il vincolo della scatola.
Nel titolo della finestra e specificato il tipo di scatola e l'indicazione *elementi inseriti/capacità*, una volta che la scatola viene riempita il bottone di inserimento oggetti viene disabilitato.
In archivio è stata già memorizzata una scatola per tipo.

5.5 Crea Scatole

In questa vista l'amministratore potrà creare una nuova scatola vuota da inserire in archivio, i campi richiesti sono codice (univoco), capacità (da 1 a quanto specificato dalla variabile statica MAX_CAPACITY di Box) e colore.

5.6 Butta Scatole

Questa vista contiene la lista di scatole presenti nell'archivio. La nonna e l'amministratore potranno selezionare una scatola alla volta ed eliminarla dalla memoria e dal disco.
In caso la scatola contenga degli Item, questi verranno eliminati.

5.7 Crea Oggetti

In questa finestra è possibile creare nuovi oggetti, la vista cambia in base all'utente che accede:
Il ragazzo potrà creare solo giocattoli, la nonna solo attrezzi da cucito e l'admin entrambi.
Il giocattolo richiede i campi codice (univoco), nome, ha valore affettivo? (si, no).
L'attrezzo da cucito richiede i campi codice(univoco), nome, utilità (da 1 a 100).

5.8 Gestione Utenti

In questa finestra l'admin può rimuovere un utente o cambiarne il ruolo.
Le modifiche non vengono salvate fino al click del pulsante salva, che indicherà il corretto funzionamento della procedura o mostrerà un messaggio di errore in caso contrario.

6 Indicazione Ore

Progettazione: 5h
Infarinatura Qt: 5h
Modello scrittura/lettura: 5h
Logica applicativa: 15h
Codifica viste: 20h
Refactoring: 3h
Stesura relazione: 6h
Totale: 59h

7 Specifiche

Il progetto è stato sviluppato su sistema operativo Ubuntu-Gnome 16.10 e compilato con il compilatore gcc 6.2.0. La versione di Qt utilizzata è la 5.7.1.