UNIVERSITÀ DEGLI STUDI DI PADOVA

Relazione Progetto Programmazione ad Oggetti

Anno Accademico 2015-2016

Titolo: QtDelivery

Studente: Scarparo Davide 1049135

Indirizzo email referente: davide.scarparo@studenti.unipd.it

Indice

1 Introduzione

Il progetto ***QtDelivery*** è stato progettato su un sistema Windows 10, con l'ausilio dell'IDE QtCreator in versione 3.6.1.
Le librerie di QtCreator utilizzate sono in versione 5.3 e la versione del compilatore g++ è MinGW 4.6.1; per la realizzazione dell'interfaccia grafica si è deciso di non far uso dello strumento QtDesigner fornito dall'ambiente Qt.
Il progetto è stato inoltre testato sulle macchine del laboratorio del dipartimento in ambiente Linux Ubuntu 12.04 con compilatore g++ in versione 4.7 e librerie Qt versione 5.3.2.
La cartella consegnata contiene già un project file *QtDelivery.pro* che è necessario per compilare correttamente il progetto, quindi occorre lanciare semplicemente i comandi *qmake* e *make*.

2 Progetto

2.1 Scopo

Lo scopo del progetto QtDelivery è lo sviluppo e la realizzazione di un'applicazione in C++/Qt per la gestione di un magazzino appartenente ad un'azienda che fornisce servizi di spedizioni nazionali ed internazionali.
Si è scelto di rappresentare le seguenti diverse tipologie di pacchi: Regular National Parcel, Fast National Parcel ed International Parcel.
L'applicazione fa uso di un database esterno per la lettura e la memorizzazione dei dati; è stato scelto il formato XML, poichè Qt offre librerie apposite alla gestione di file in tale formato.

2.2 Funzionalità

Le funzionalità offerte dall'applicazione sono:

- tenere traccia dei quantitativi dei pacchi (nazionali ed internazionali);
- aggiungere un nuovo pacco;
- visualizzare la lista dei pacchi presenti nel magazzino;
- spedire il primo pacco;
- spedire l'ultimo pacco;
- spedire l'i-esimo pacco;
- spedire tutti i pacchi di una certa tipologia;
- svuotare l'intero magazzino;
- ricercare il pacco con costo di spedizione maggiore;
- ricercare tutti i pacchi di una certa categoria (small, medium, large).

3 Classi

3.1 Incapsulamento

Il progetto QtDelivery fa uso dei seguenti tipi definiti da utente, incapsulati in apposite classi:

- **Address** (rappresenta un indirizzo civico, caratterizzato da via, numero civico e CAP);
- **Measures** (rappresenta le tre dimensioni di un generico oggetto, quali altezza, larghezza e profondità);
- **Entity** (rappresenta un individuo caratterizzato da nome, cognome e indirizzo);
- **ShippingLabel** (rappresenta un'etichetta di spedizione che viene apposta su di un qualsiasi pacco, ed è caratterizzata dalle informazioni relative a mittente e destinatario);

Alcune delle classi della gerarchia che si andrà a descrivere nel successivo paragrafo, sono definite modularmente per favorire il riutilizzo e l'estensibilità del codice.

3.2 Gerarchia

La gerarchia sviluppata per il progetto QtDelivery è rappresentata in figura 1.

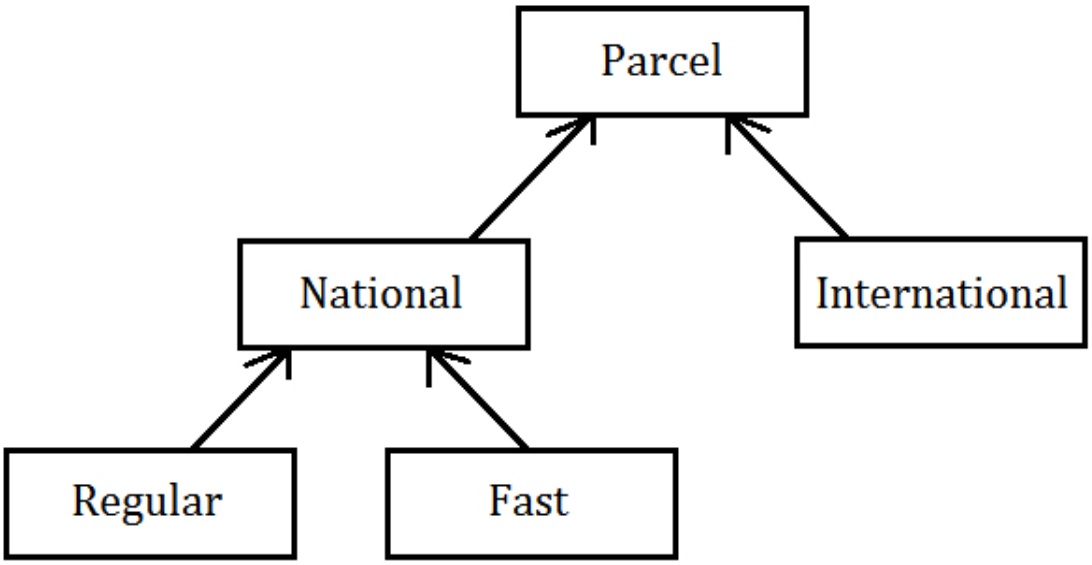

Figura 1 – Gerarchia di classi

La classe base **Parcel** rappresenta un generico pacco, avente un peso, delle dimensioni, un'etichetta di spedizione e due campi statici, quali la tariffa per kg e il coefficiente del peso volumetrico.
È presente un distruttore virtuale affinchè tutti i distuttori nelle classi derivate siano anch'essi virtuali; inoltre, sono stati definiti altri due metodi virtuali, il cui comportamento verrà ridefinito nelle classi derivate.
Il primo, *shippingCost()*, si occupa di calcolare il costo finale di spedizione per il pacco di invocazione; il secondo, *typology()*, fornisce il nome del tipo di pacco (Base, Regular, Fast, International).
Per il calcolo del costo di spedizione di un pacco, si fa uso della seguente regola: si

prende il maggiore tra peso e peso volumetrico, e lo si moltiplica per la tariffa per kg (il peso volumetrico è calcolato come volume/coefficiente peso volumetrico).
La classe **NationalParcel** deriva direttamente dalla base Parcel e rappresenta un pacco di spedizione nazionale; è caratterizzato da una tassa nazionale e calcola il costo di spedizione come *shippingCost()* della base, sommato alla tassa nazionale.
La classe **RegularNP** deriva direttamente da NationalParcel e rappresenta un pacco di spedizione nazionale di tipo regular.
Questi pacchi sono condizionati da una tassa fissa regular, che serve nel calcolo del costo di spedizione del pacco stesso: *shippingCost()* ridefinito in NationalParcel moltiplicato per la tassa regular.
La classe **FastNP** deriva direttamente da NationalParcel e rappresenta un pacco di spedizione nazionale di tipo fast.
Questi pacchi sono condizionati da una tassa fissa fast, che serve nel calcolo del costo di spedizione del pacco stesso: *shippingCost()* ridefinito in NationalParcel moltiplicato per la tassa fast.
Per capire la differenza tra il concetto di pacco regular e il concetto di pacco fast, basta pensare, ad esempio, ad un differente tempo lavorativo nella consegna del pacco: il progetto, però, non si occupa della gestione delle dati e delle consegne, ma vuole dare semplicemente una divisione a livello logico nella gerarchia.
La classe **InternationalParcel** deriva direttamente dalla base Parcel e rappresenta un pacco di spedizione internazionale; è caratterizzato da una tassa per l'estero e calcola il costo di spedizione come *shippingCost()* della base, sommato alla tassa internazionale.

3.3 Contenitore

Per la definizione di un contenitore per oggetti polimorfi, si è scelto di implementare una lista doppiamente linkata, con gestione della memoria in modalità profonda.
Per fare ciò, si è fatto uso dei template di classe e template di funzione; inoltre, come da richiesta, è stata implementata una classe Iterator per la gestione degli iteratori.
La classe **Depot** è la classe contenitore per il progetto QtDelivery, e consente le classiche operazioni in tempo costante: aggiunta in testa, aggiunta in coda, rimozione in testa e rimozione in coda.
Ogni nodo **Link** della lista Depot è caratterizzato da un campo info e da due puntatori: uno che punta al nodo precedente e uno a quello successivo.
Per la classe **Iterator** sono stati ridefiniti gli operatori di uguaglianza, disuguaglianza, incremento e decremento postfisso, incremento e decremento prefisso, dereferenziazione, costruttore di copia e assegnazione.
La classe **Stock**, invece, è la classe che utilizza il contenitore: ha un campo privato di tipo Depot<Parcel*>, ovvero un contenitore di puntatori ad oggetti polimorfi.

4 GUI

Per la parte relativa all'interfaccia grafica e all'interazione con l'utente, sono state progettate le seguenti classi: MainWindow, MainWidget, Input, Table, MyForm, MessageToUser.
La classe **MainWindow** rappresenta la finestra principale dell'applicazione, dove è presente un classico menu orizzontale che si sviluppa in verticale con sottovoci a tendina.

Tutti i contenuti vengono visualizzati all'interno di questa finestra, attraverso il MainWidget, il cui contenuto è dinamico e varia a seconda delle azioni eseguite dall'utente dell'applicazione.
La classe **MainWidget**, per l'appunto, contiene il widget principale attivo al momento; fondamentalmente, è un box di layout verticale, ciò significa che ogni oggetto (item) aggiunto, viene inserito dall'alto verso il basso.
Il MainWidget può contenere semplici oggetti, quali QPushButton, QLabel, QLCDNumber, ma anche oggetti più complessi, come una Table.
La classe **Table** rappresenta in forma tabellare l'intero contenuto del magazzino dell'azienda, fornendo per ogni pacco le principali informazioni.
La classe **MyForm**, invece, rappresenta il form per l'inserimento dei dati relativi ad un nuovo pacco che si vuole aggiungere e registrare nel magazzino; questa classe a sua volta, fa uso della classe **Input**, la quale rappresenta i vari campi del form dove inserire i dati.
Per l'inserimento di valori di tipo double, si è scelto di utilizzare un QDoubleSpinBox, mentre per i valori di tipo int un QSpinBox; infine, per le stringhe si utilizzano delle QLineEdit.
MyForm ha il compito di validare i dati inseriti, e in caso di errore, segnalarlo all'utente con una descrizione annessa; solo una volta validati i dati inseriti dall'utente, si procede con l'operazione di aggiunta al contenitore e scrittura sul database XML.
Affinchè un input sia valido, occorre che:

- il peso e le dimensioni del pacco non siano 0;
- una stringa contenga solo caratteri alfabetici (vale per nome, cognome e indirizzo);
- nel caso dell'indirizzo, la stringa deve iniziare con "via" o "piazza";
- il CAP contenga 5 caratteri numerici.

Infine, la classe **MessageToUser**, ha il compito di gestire tutti i messaggi che l'applicazione fornisce all'utente, sia messaggi negativi (errore), sia mesaggi positivi (operazione completata).

5 Uso del polimorfismo

Il polimorfismo viene utilizzato, ad esempio, nel metodo *showParcelRemoved(Parcel*)*, per ottenere le informazioni sul pacco puntato dal puntatore parametro.
Grazie al **late binding**, le chiamate *pacco->shippingCost()*, *pacco->typology()*, ecc..., sono relative al tipo dinamico del puntatore, ovvero al tipo effettivo dell'oggetto a cui punta il puntatore.
Si ricorda, infatti, che il contenitore contiene puntatori polimorfi, ovvero che hanno come tipo statico quello della classe base Parcel*, mentre il tipo dinamico è individuato a run time, con il quale si seleziona il metodo corretto.
In rari casi, viene utilizzato il **dynamic_cast** come operatore di RTTI, come si può vedere in alcuni metodi della classe Stock; per il resto, si usano prevalentemente chiamate polimorfe, al fine di rendere il codice il più estensibile possibile.

6 Note

Qui di seguito, verranno messe in evidenza alcune note fondamentali riguardanti il progetto, quali scelte e vincoli ragionevoli:

- durante i test del progetto in laboratorio, sono state riscontrate delle leggere differenze di visualizzazione grafica di alcuni contenuti della finestra dell'applicazione (barra menu, spinbox, caratteri speciali come il simbolo dell'euro e lettere accentate);
- affinchè il progetto venga eseguito correttamente, deve essere sempre presente all'interno della cartella del progetto il file database *stock.xml*;
- la struttura del file *stock.xml* è molto semplice per facilitare la lettura/scrittura con le librerie Qt;
- tutte le immagini utilizzate sono contenute nell cartella images, perciò il progetto fa uso di un file di risorse QtDelivery.qrc, necessario al fine di una corretta compilazione e di una corretta visualizzazione delle immagini;
- si è scelto di non fornire funzionalità per modificare i pacchi, poichè l'utilizzo di questa applicazione è rivolta a personale addetto di un'azienda specializzata in consegne, che deve avere molta cura nel registrare i dati di nuovi pacchi (una volta che un pacco è imballato, non ha senso modificare informazioni come peso, dimensioni o destinatario);
- la sezione *Informazioni* della voce *Help* del menu principale, fornisce le caratteristiche e tutte le più importanti info riguardanti l'applicazione (versione, autore, contatti, funzionalità, ecc...);
- si presuppone la seguente categorizzazione per le dimensioni dei pacchi: small (inferiore a 9000 cm^3), medium (tra 9000 e 15000 cm^3) e large (superiore a 15000 cm^3);
- le dimensioni di un generico pacco (altezza, larghezza e profondità) sono da intendersi come da figura 2.

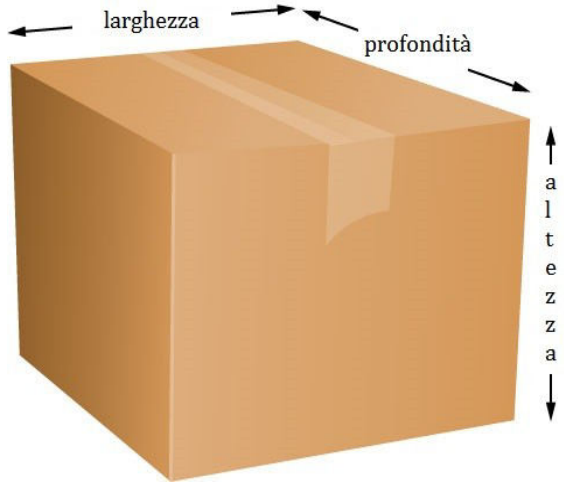

Figura 2 – Dimensioni pacco

- il programmatore che fornisce l'applicazione, può cambiare le tasse (nazionale e verso l'estero), le tariffe (per kg, regular, fast) e il coefficiente di peso volumetrico, in base alla richiesta del cliente.