QONTAINER : Progetto di Programmazione ad Oggetti, a.a. 2018/2019

prof. Francesco Ranzato

1 Specifica

1.1 Scopo

Lo scopo del progetto QONTAINER è lo sviluppo in C++/Qt di un gestore di un contenitore di oggetti di una gerarchia di tipi tramite una interfaccia utente grafica (GUI). Il progetto deve essere sviluppato da un singolo studente in modo indipendente e deve richiedere approssimativamente al più 50 ore di lavoro complessivo.

1.2 Template di classe `Container<T>`

Definire un template di classe `Container<T>` i cui oggetti rappresentano un contenitore di oggetti di tipo `T`. Il template `Container<T>` deve fornire almeno delle funzionalità minime di: (1) inserimento (2) rimozione (3) ricerca. Vi è un unico vincolo da rispettare nello sviluppo del template di classe `Container<T>`: **non** è permesso l'utilizzo dei contenitori STL/Qt (o di altre librerie). Si scelga l'implementazione di `Container<T>` ritenuta più opportuna per usare il contenitore come struttura dati usata dalla GUI del gestore QONTAINER .

1.3 Gerarchia di tipi

Definire una gerarchia G di almeno tre tipi istanziabili ed avente un supertipo base comune `B` (astratto o istanziabile). Ad esempio, il tipo astratto `LibraryItem` può essere un supertipo di `Libro`, `Rivista`, `Quotidiano`, in una gerarchia che modella degli oggetti da catalogare in una biblioteca. I tipi della gerarchia G dovranno essere dotati di opportune interfacce pubbliche. Usare la fantasia ed ispirarsi ai propri interessi nello scegliere la realtà da modellare mediante questa gerarchia di tipi. Naturalmente sono permesse anche gerarchie più complesse che contengono piú di tre tipi.

Si valuti l'opportunità ai fini progettuali di definire un template di classe `DeepPtr<T>` di puntatori polimorfi "smart" al tipo `T`. Quindi `DeepPtr<T>` dovrebbe implementare una gestione automatica della memoria "profonda" e polimorfa di puntatori a `T` richiedendo che `T` sia un tipo che supporti la clonazione e la distruziona polimorfa.

1.4 Requisito Java

La gerarchia di tipi G deve essere implementata anche in Java. A fini illustrativi, dotare la gerarchia Java di un qualche metodo `main` che esemplifichi l'utilizzo dell'interfaccia dei tipi di G.

1.5 GUI in Qt

Sviluppare una GUI in Qt che permetta all'utente di QONTAINER una semplice gestione di un contenitore di oggetti polimorfi della gerarchia G. Ad esempio, la gestione del catalogo di item disponibili in una biblioteca. Si tratta quindi di sviluppare una GUI per la gestione di un contenitore C di puntatori (possibilmente smart) polimorfi al supertipo `A` della gerarchia G, che come requisiti obbligatori permetta:

(1) Inserimenti di oggetti di G in C. Esempio: inserimento di un nuovo libro nel catalogo della biblioteca.

(2) Rimozioni di oggetti da C che soddisfino alcune caratteristiche. Esempio: rimozione di tutti i quotidiani di un certo anno dal catalogo della biblioteca.

(3) Ricerche di oggetti di G in C. Esempio: ricerca di tutti i libri dell'autore "Bjarne Stroustrup". In particolare, le ricerche potrebbero essere basate su alcune caratteristiche pubbliche dei tipi di G, ad esempio, ricerche per: anno di pubblicazione di qualsiasi `LibraryItem`, autore di un `Libro`, editore di una `Rivista`, data di un `Quotidiano`.

(4) Modifiche di oggetti di G in C. Esempio: modifica del costo del prestito per un particolare libro.

(5) Load e save del contenitore C.

(5.1) Caricamento ("load") da file degli oggetti che popolano il contenitore C. Esempio: all'avvio della GUI il caricamento avviene in automatico da un file di default, oppure vi è anche la possibilità di aggiungere da file nuovi oggetti al contenitore C.

(5.2) Salvataggio ("save") su file del contenuto attualmente in memoria del contenitore C. Esempio: all'uscita della GUI avviene il salvataggio automatica del contenitore C in un file di default, oppure vi è anche la possibilità di salvare in qualsiasi momento C (o anche qualche sua parte) in un nuovo file.

(5.3) Per il formato del file di caricamento/salvataggio si possono considerare formati come XML, JSON o altri formati per lo scambio di dati (AXON, YAML, etc.)

(6) La GUI deve visualizzare i diversi tipi di oggetto di G in maniera opportuna. Esempio: la visualizzazione di un libro potrebbe mostrare l'immagine della sua copertina mentre la visualizzazione di un quotidiano potrebbe mostrare solamente del contenuto testuale.

La GUI può liberamente trarre ispirazione da aggregatori di contenuti disponibili sul web, sia applicazioni per sistemi desktop che applicazioni per sistemi mobili. Si potrà aderire al design pattern **Model-View-Controller** o **Model-View** per la progettazione architetturale della GUI, dove il Model includerà il modello logico di QONTAINER . Qt include un insieme di classi di "view" che usano una architettura "model/view" per gestire la relazione tra i dati logici della GUI ed il modo in cui essi sono presentati all'utente della GUI (si veda `http://qt-project.org/doc/qt-5/model-view-programming.html`). Come noto, la libreria Qt è dotata di una documentazione completa e precisa che sarà la principale guida di riferimento nello sviluppo della GUI, oltre ad offrire l'IDE QtCreator ed il tool QtDesigner.

1.6 Altri Requisiti Obbligatori

Il progetto QONTAINER deve soddisfare i seguenti requisiti obbligatori:

1. **Separazione tra modello logico e GUI**: il modello logico della gerarchia G non ha alcuna relazione con il codice della GUI. La gerarchia G dovrebbe poter essere usata anche da un GUI scritta in un framework diverso da Qt.
2. Deve essere utilizzato del **codice polimorfo** che sfrutti l'organizzazione gerarchica dei tipi di G.

1.7 Tutorato

La libreria Qt offre una moltitudine di classi e metodi per lo sviluppo di GUI dettagliate e user-friendly. I tutor Benedetto Cosentino (studente del III anno della Laurea triennale) ed Alessandro Zangari (studente del I anno della Laurea Magistrale) organizzano degli incontri di tutorato in laboratorio dedicati all'apprendimento delle caratteristiche di base del framework Qt per la progettazione di GUI. Il primo incontro è previsto per lunedì 10 dicembre 2018 ore 16:30 in LabTA. Ulteriori informazioni al gruppo Facebook **TutorJuniorInformaticaUniPD**.

2 Valutazione del Progetto

Un buon progetto dovrà essere sviluppato seguendo i principi fondamentali della programmazione orientata agli oggetti, anche per quanto concerne lo sviluppo dell'interfaccia grafica. La valutazione del progetto prenderà in considerazione i seguenti criteri:

1. **Correttezza**: il progetto deve:
   (a) compilare ed eseguire correttamente nella macchina `ssh.studenti.math.unipd.it`, di cui è stata fornita una immagine da usare come macchina virtuale in VirtualBox. **NB: si tratta di una condizione necessaria per la valutazione del progetto.** La compilazione dovrà essere possibile invocando la sequenza di comandi: `qmake -project` ⇒ `qmake` ⇒ `make`. Se la compilazione del progetto necessita di un project file (.pro) per `qmake` diverso da quello ottenibile tramite l'invocazione di `qmake -project` allora deve anche essere consegnato un file `progetto.pro` che permetta la generazione automatica tramite `qmake` del `Makefile`, e ciò dovrà essere **esplicitamente** documentato nella relazione.
   (b) soddisfare pienamente e correttamente tutti i requisiti obbligatori
2. **Orientazione agli oggetti**: il progetto deve **obbligatoriamente** includere delle chiamate polimorfe. Saranno oggetto di valutazione le seguenti qualità del codice prodotto:
   (a) incapsulamento
   (b) modularità (in particolare, massima separazione tra parte logica e grafica (GUI) del codice)
   (c) estensibilità ed evolvibilità, in particolare mediante polimorfismo
   (d) efficienza e robustezza
3. **Funzionalità**: quante e quali funzionalità il progetto rende disponibili, e la loro qualità.

4. **GUI**: utilizzo corretto della libreria Qt; qualità, usabilità e robustezza della GUI.
5. **Relazione**: chiarezza e qualità della relazione, che verterà sui seguenti aspetti:
   (a) descrizione **obbligatoria** di tutte le gerarchie di tipi usate
   (b) descrizione **obbligatoria** dell'uso di chiamate polimorfe
   (c) descrizione **obbligatoria** del formato del file di caricamento/salvataggio del contenitore
   (d) se l'applicazione lo richiede, manuale utente della GUI
   (e) se necessario, indicazione di eventuali istruzioni di compilazione ed esecuzione
   (f) indicazione delle **ore effettivamente richieste** dalle diverse fasi progettuali: analisi preliminare del problema, progettazione modello e GUI, apprendimento libreria Qt, codifica modello e GUI, debugging, testing. In caso di superamento del previsto monte di 50 ore di lavoro complessivo per persona, giustificazione per le ore in eccesso
   (g) **In caso di progetto riconsegnato:** la relazione dovrà **obbligatoriamente** descrivere in modo dettagliato le modifiche apportate al codice rispetto alla precedente versione consegnata, in particolare elencando in modo puntuale le modifiche al codice che risolvono i punti deboli riscontrati nella precedente valutazione del progetto.

Quindi, il progetto dovrà essere **obbligatoriamente** accompagnato da una relazione scritta di **massimo 8 pagine in formato 10pt**. La relazione deve essere presentata come un file PDF di nome (preciso) `relazione.pdf`. La relazione deve anche specificare il sistema operativo di sviluppo e le versioni precise del compilatore e della libreria Qt.

3 Esame Orale e Registrazione Voto

La partecipazione all'esame orale è possibile solo dopo:

1. avere superato con successo (cioè, con voto $\geq$ 18/30) l'esame scritto
2. avere consegnato il progetto entro la scadenza stabilita, che verrà sempre comunicata in Moodle.
3. essersi iscritti alla lista Uniweb dell'esame orale

Il giorno dell'esame orale (nel luogo ed all'orario stabiliti) verrà comunicato l'esito della valutazione del progetto (non vi saranno altre modalità di comunicazione della valutazione del progetto) assieme ad un sintetico **feedback** sui punti deboli riscontrati nella valutazione del progetto. Tre esiti saranno possibili:

(A) Valutazione positiva del progetto con registrazione del voto complessivo proposto **con esenzione dell'esame orale**. Nel caso in cui il voto proposto non sia ritenuto soddisfacente dallo studente, sarà possibile rifiutare il voto oppure richiedere l'esame orale, che potrà portare a variazioni in positivo o negativo del voto proposto.

(B) Valutazione del progetto da completarsi con un **esame orale obbligatorio**. Al termine dell'esame orale, o verrà proposto un voto complessivo sufficiente oppure si dovrà riconsegnare il progetto per un successivo esame orale.

(C) Valutazione negativa del progetto che comporta quindi la **riconsegna del progetto** per un successivo esame orale (il voto dell'esame scritto rimane valido).

Lo studente che decida di rifiutare il voto finale proposto, con o senza orale, dovrà riconsegnare il progetto per un successivo orale (tranne al quinto orale), cercando quindi di porre rimedio ai punti deboli segnalati nel feedback di valutazione e descrivendo obbligatoriamente le modifiche apportare al progetto nella relazione aggiornata. Il voto sufficiente dell'esame scritto rimane comunque valido. All'eventuale esame orale lo studente dovrà saper motivare **ogni** scelta progettuale e dovrà dimostrare la **piena conoscenza** di ogni parte del progetto.

4 Regole

4.1 Compilatore e libreria Qt

Il progetto deve compilare ed eseguire correttamente sulla macchina **Linux** `ssh.studenti.math.unipd.it` (di cui è stata fornita una immagine da utilizzarsi come macchina virtuale per VirtualBox) con il compilatore GNU `g++` 5.x (correntemente 5.4.0), la libreria Qt in versione 5.x (correntemente 5.5.1) e con compilatore e macchina virtuale Java (correntemente 1.8.0). È naturalmente possibile sviluppare il progetto su altri sistemi operativi come MacOS/Windows. In tal caso, prima di consegnare il progetto, ricordarsi di effettuare una prova di compilazione, esecuzione e funzionamento sulla macchina Linux `ssh.studenti.math.unipd.it`.

4.2 Cosa consegnare

Tutti i file sorgente `.h` e `.cpp` per il codice C++ e `.java` per il codice Java, assieme al file `relazione.pdf` contenente la relazione **individuale**, e ad eventuali file che memorizzano dati necessari per il corretto funzionamento del programma (ad esempio, dei file di input/output necessari al programma). Se la compilazione del progetto necessita di un project file (.pro) per `qmake` diverso da quello ottenibile tramite l'invocazione di `qmake -project` allora ciò dovrà essere dichiarato esplicitamente nella relazione e dovrà anche essere consegnato un file `progetto.pro` che permetta la generazione automatica tramite `qmake` del `Makefile`.

Cosa non consegnare: codice oggetto, eseguibile, file di back-up generati automaticamente da editor o IDE e tutto quanto non necessario per la corretta compilazione ed esecuzione del programma.

4.3 Come consegnare

Dalle macchine del laboratorio invocando il comando

```
consegna progetto-pao-2019
```

dalla directory contenente **tutti e soli** i file da consegnare. **Attenzione:** La dimensione massima complessiva di tutti i file che verranno consegnati è 50MB (se la dimensione è maggiore il comando di consegna non funzionerà correttamente). **Non saranno accettate altre modalità di consegna** (ad esempio via email). Naturalmente è possibile consegnare remotamente il progetto tramite il server

```
ssh.studenti.math.unipd.it
```

e opportuni comandi/programmi come `ssh`, `sftp`, `scp`, etc.

4.4 Scadenze di consegna

Il progetto dovrà essere consegnato rispettando **tassativamente** le scadenze **ufficiali** (data e ora) previste che verranno rese note tramite le liste Uniweb di iscrizione agli esami scritti ed orali e tramite il Moodle del corso. Approssimativamente la scadenza sarà circa 8-12 giorni prima dell'esame orale.

Per i progetti ritenuti insufficienti, lo studente dovrà consegnare una nuova versione del progetto per un successivo appello orale.

Prima sessione regolare di esami orali: Le date degli esami orali della sessione regolare con relative scadenze tassative di consegna del progetto sono le seguenti:

Primo orale: lunedì 11 febbraio 2019, scadenza di consegna: domenica 3 febbraio 2019 ore 23:59

Secondo orale: venerdì 22 febbraio 2018, scadenza di consegna: mercoledì 13 febbraio 2019 ore 23:59