

APPELLO 14-06-2016

Quesito 1

Si consideri il seguente modello di realtà concernente una app di riproduzione di file audio denominata MusicPlay.

(A) Definire la seguente gerarchia di classi

1. Definire una classe base polimorfa astratta *FileAudio* i cui oggetti rappresentano un file audio memorizzabile da MusicPlay. Ogni *FileAudio* è caratterizzato dalla propria dimensione in MB. La classe è astratta in quanto prevede i seguenti **metodi virtuali puri**:
 - un metodo di clonazione polimorfa;
 - un metodo *bool quality()* con il seguente contratto puro: *f->quality()* ritorna true se il file audio **f* è considerato di alta qualità, altrimenti ritorna false.
2. Definire una classe concreta *Mp3* derivata da *FileAudio* i cui oggetti rappresentano un file audio in formato mp3. Ogni oggetto *Mp3* è caratterizzato dal proprio bitrate espresso in Kbit/s. La classe *Mp3* implementa i metodi virtuali puri di *FileAudio* come segue:
 - l'usuale implementazione del metodo di clonazione polimorfa;
 - *m->quality()* ritorna true se il bitrate di **m* è ≥ 256 kbit/s, altrimenti ritorna false.
3. Definire una classe concreta *WAV* derivata da *FileAudio* i cui oggetti rappresentano un file audio in formato WAV. Ogni oggetto *WAV* è caratterizzato dalla propria frequenza di campionamento espressa in kHz. La classe *WAV* implementa i metodi virtuali puri di *FileAudio* come segue:
 - l'usuale implementazione del metodo di clonazione polimorfa;
 - *w->quality()* ritorna true se la frequenza di campionamento di **w* è ≥ 192 kHz, altrimenti ritorna false.

(B) Definire una classe *MusicPlay* i cui oggetti rappresentano i brani memorizzati in una installazione dell'app MusicPlay. Un oggetto *MusicPlay* deve essere rappresentato mediante una lista (della libreria STL) di puntatori di tipo *const FileAudio**. La classe *MusicPlay* rende disponibili i seguenti metodi:

1. Un metodo *vector<WAV*> qualityWAV(double)* con il seguente comportamento: una invocazione *mp.qualityWAV(dim)* ritorna un vector contenente tutti e soli i puntatori ai file audio in formato WAV memorizzati da *mp* che hanno una dimensione $\leq dim$ e sono di alta qualità.
2. Un metodo *list<FileAudio*> nonConstCopy()* con il seguente comportamento: una invocazione *mp.nonConstCopy()* ritorna una copia profonda della lista di puntatori (di tipo *const FileAudio**) memorizzata da *mp* in cui il tipo dei puntatori memorizzati in tale copia è non costante.
3. Un metodo *vector<Mp3> removeMp3(unsigned int)* con il seguente comportamento: una invocazione *mp.removeMp3(br)*: (i) rimuove dalla lista di puntatori memorizzata da *mp* tutti e soli i puntatori a file audio in formato mp3 che abbiano un bitrate minore di *br*; (ii) ritorna un vector contenente tutti e soli gli oggetti *Mp3* rimossi nel punto (i).

Quesito 2

Definire una unica gerarchia di classi che includa:

1. una classe base polimorfa A alla radice della gerarchia;
2. una classe derivata astratta B;
3. una sottoclasse C di B che sia concreta;
4. una classe D che non permetta la costruzione pubblica dei suoi oggetti, ma solamente la costruzione di oggetti D che siano sottooggetti;
5. una classe E definita mediante derivazione multipla con base virtuale, che abbia D come supertipo, e con l'assegnazione ridefinita pubblicamente con comportamento identico a quello dell'assegnazione standard di E.

Quesito 3

Siano A, B, C e D quattro diverse classi polimorfe. Si considerino le seguenti definizioni:

```
template <class T, class Y>
X* fun(X* p) { return dynamic_cast<Y*>(p); }

main() {
    C c;
    fun <A,B> ( &c );
    if ( fun <A,B> (new C( ) ) == 0 )
        cout<<"Programazione";
    if ( dynamic_cast <C*> ( new B( ) ) == 0 )
        cout<<"ad oggetti";
    A* p = fun <D,B> (new D( ) );
}
```

Si supponga che:

1. il *main()* compili correttamente ed esegua senza provocare errori a run time;
2. l'esecuzione del *main()* provochi in output su *cout* la stampa "Programmazione ad oggetti".

In tali ipotesi, per ognuna delle relazioni di sottotipo $T1 \leq T2$ nelle seguenti tabelle segnare:

1. VERO per indicare che T1 **sicuramente** è sottotipo di T2;
2. FALSO per indicare che T1 **sicuramente non** è sottotipo di T2;
3. POSSIBILE altrimenti.

A<=B
A<=C
A<=D
B<=A
B<=C
B<=D
C<=A
C<=B
C<=D
D<=A
D<=B
D<=C