```
class A {
private:
  void h() {cout<<" A::h ";}
public:
  virtual void g() {cout <<" A::g ";}
  virtual void f() {cout <<" A::f "; g(); h();}
  void m() {cout <<" A::m "; g(); h();}
  virtual void k() {cout <<" A::k "; g(); h(); m(); }
  A* n() {cout <<" A::n "; return this;}
};

class B: public A {
private:
  void h() {cout <<" B::h ";}
public:
  virtual void g() {cout <<" B::g ";}
  void m() {cout <<" B::m "; g(); h();}
  void k() {cout <<" B::k "; g(); h(); m();}
  B* n() {cout <<" B::n "; return this;}
};

B* b = new B(); A* a = new B();
```

```
// COMPILA?
// ERRORE RUN-TIME?
// COSA STAMPA?

b->f();
b->m();
b->k();
a->f();
a->m();
a->k();
(b->n())->g();
(b->n())->n()->g();
(a->n())->g();
(a->n())->m();
```

```
// COMPILA?
// ERRORE RUN-TIME?
// COSA STAMPA?

b->f();                 // A::f  B::g  A::h
b->m();                 // B::m  B::g  B::h
b->k();                 // B::k  B::g  B::h  B::m  B::g  B::h
a->f();                 // A::f  B::g  A::h
a->m();                 // A::m  B::g  A::h
a->k();                 // B::k  B::g  B::h  B::m  B::g  B::h
(b->n())->g();          // B::n  B::g
(b->n())->n()->g();     // B::n  B::n  B::g
(a->n())->g();          // A::n  B::g
(a->n())->m();          // A::n  A::m  B::g  A::h
```

# Esercizio 11.18

Si consideri il seguente modello di realtà concernente i file audio memorizzati in un riproduttore audio digitale iZod©.

**(A)** Definire la seguente gerarchia di classi.

1. Definire una classe base polimorfa astratta `FileAudio` i cui oggetti rappresentano un file audio memorizzabile in un iZod. Ogni `FileAudio` è caratterizzato dal titolo (una stringa) e dalla propria dimensione in MB. La classe è astratta in quanto prevede i seguenti **metodi virtuali puri**:
   - un metodo di "clonazione": `FileAudio* clone()`.
   - un metodo `bool qualita()` con il seguente contratto: `f->qualita()` ritorna true se il file audio `*f` è considerato di 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:
   - per ogni puntatore `p` a `Mp3`, `p->clone()` ritorna un puntatore ad un oggetto `Mp3` che è una copia di `*p`.
   - per ogni puntatore `p` a `Mp3`, `p->qualita()` ritorna true se il bitrate di `*p` è $\geq$ 192 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 e dall'essere lossless oppure no (cioè con compressione senza perdita oppure con perdita). La classe `WAV` implementa i metodi virtuali puri di `FileAudio` come segue:
   - per ogni puntatore `p` a `WAV`, `p->clone()` ritorna un puntatore ad un oggetto `WAV` che è una copia di `*p`.
   - per ogni puntatore `p` a `WAV`, `p->qualita()` ritorna true se la frequenza di campionamento di `*p` è $\geq$ 96 kHz, altrimenti ritorna false.

**(B)** Definire una classe `iZod` i cui oggetti rappresentano i brani memorizzati in un iZod. La classe `iZod` deve soddisfare le seguenti specifiche:

1. È definita una classe annidata `Brano` i cui oggetti rappresentano un brano memorizzato nell'iZod. Ogni oggetto `Brano` è rappresentato da un puntatore polimorfo ad un `FileAudio`.
   - La classe `Brano` deve essere dotata di un opportuno costruttore `Brano(FileAudio*)` con il seguente comportamento: `Brano(p)` costruisce un oggetto `Brano` il cui puntatore polimorfo punta ad una copia dell'oggetto `*p`.
   - La classe `Brano` ridefinisce costruttore di copia profonda, assegnazione profonda e distruttore profondo.
2. Un oggetto di `iZod` è quindi caratterizzato da un vector di oggetti di tipo `Brano` che contiene tutti i brani memorizzati nell'iZod.
3. La classe `iZod` rende disponibili i seguenti metodi:
   - Un metodo `vector<Mp3> mp3(double, int)` con il seguente comportamento: una invocazione `iz.mp3(dim,br)` ritorna un vector di oggetti `Mp3` contenente tutti e soli i file audio in formato mp3 memorizzati nell'iZod `iz` che: (i) hanno una dimensione $\geq$ `dim` e (ii) hanno un bitrate $\geq$ `br`.
   - Un metodo `vector<FileAudio*> braniQual()` con il seguente comportamento: una invocazione `iz.braniQual()` ritorna il vector dei puntatori ai `FileAudio` memorizzati nell'iZod `iz` che: (i) sono considerati di qualità e (ii) se sono dei file audio `WAV` allora devono essere lossless.
   - Un metodo `void insert(Mp3*)` con il seguente comportamento: una invocazione `iz.insert(p)` inserisce il nuovo oggetto `Brano(p)` nel vector dei brani memorizzati nell'iZod `iz` se il file audio mp3 `*p` non è già memorizzato in `iz`, mentre se il file audio `*p` risulta già memorizzato non provoca alcun effetto.