Esercizi di Programmazione ad Oggetti, a.a. 10/11

Esercizio 1

Definire una superclasse `Auto` i cui oggetti rappresentano generiche automobili e due sue sottoclassi `Benzina` e `Diesel`, i cui oggetti rappresentano automobili alimentate, rispettivamente, a benzina e a diesel (ovviamente non esistono automobili non alimentate e si assume che ogni auto è o benzina o diesel). Ci interesserà l'aspetto fiscale delle automobili, cioè il calcolo del bollo auto. Queste classi devono soddisfare le seguenti specifiche:

- Ogni automobile è caratterizzata da un numero di cavalli fiscali. La tassa per cavallo fiscale è unica per tutte le automobili (sia benzina che diesel) ed è fissata in 5 euro. La classe `Auto` fornisce un metodo `tassa()` che ritorna la tassa di bollo fiscale per l'automobile di invocazione.
- La classe `Diesel` è dotata (almeno) di un costruttore ad un parametro intero `x` che permette di creare un'auto diesel di `x` cavalli fiscali. Il calcolo del bollo fiscale per un'auto diesel viene fatto nel seguente modo: si moltiplica il numero di cavalli fiscali per la tassa per cavallo fiscale e si somma una addizionale fiscale unica per ogni automobile diesel fissata in 100 euro.
- Un'auto benzina può soddisfare o meno la normativa europea Euro4. La classe `Benzina` è dotata di (almeno) un costruttore ad un parametro intero `x` e ad un parametro booleano `b` che permette di creare un'auto benzina di `x` cavalli fiscali che soddisfa Euro4 se `b` vale `true` altrimenti che non soddisfa Euro4. Il calcolo del bollo fiscale per un'auto benzina viene fatto nel seguente modo: si moltiplica il numero di cavalli fiscali per la tassa per cavallo fiscale; se l'auto soddisfa Euro4 allora si detrae un bonus fiscale unico per ogni automobile benzina fissato in 50 euro, altrimenti non vi è alcuna detrazione.

Si definisca inoltre una classe `ACI` i cui oggetti rappresentano delle filiali ACI addette all'incasso dei bolli auto. Ogni oggetto ACI è caratterizzato da un vector di puntatori ad auto, cioè un oggetto `vector<Auto*>`, che rappresenta la lista delle automobili gestite da una filiale ACI. La classe `ACI` fornisce un metodo `aggiungiAuto(const Auto& a)` che aggiunge l'auto `a` alla lista gestita dalla filiale di invocazione. Inoltre, la classe `ACI` fornisce un metodo `incassaBolli()` che ritorna la somma totale dei bolli che devono pagare tutte le auto gestite dalla filiale di invocazione.

Definire infine un esempio di `main()` in cui viene costruita una filiale ACI a cui vengono aggiunte quattro automobili in modo tale che l'incasso dei bolli ammonti a 1600 euro.

Esercizio 2

```
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:
  virtual 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();
```

La compilazione delle precedenti definizioni non provoca errori (con gli opportuni `include` e `using`). Si supponga che ognuno dei seguenti frammenti sia il codice di un `main()` che può accedere alle precedenti definizioni. Tali `main()` compilano o provocano un errore run-time? In caso compilino ed eseguano senza errori, quali stampe provocano in output?

```
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();
```