Università della Calabria

Corso di Laurea in Ingegneria Elettronica

Programmazione Orientata agli Oggetti

A.A. 2006/2007

Classi e oggetti – Prima parte

Paolo Trunfio *

* DEIS, Università della Calabria – http://si.deis.unical.it/~trunfio

Oggetti

I linguaggi orientati agli oggetti (*object oriented*) sono basati sul concetto di ***oggetto***.

Un *oggetto* è una entità software che incapsula un insieme di ***attributi*** (o *dati*) e comprende un insieme di ***operazioni*** (o *metodi* o *funzioni*) che manipolano i dati.

L'insieme dei valori assunti dai suoi attributi costituisce lo ***stato*** dell'oggetto.

L'insieme delle sue operazioni definiscono il ***comportamento*** dell'oggetto.

Oggetti

Gli oggetti sono generalmente ***entità passive***: in un dato istante nell'esecuzione di un programma un solo oggetto è ***attivo*** (ovvero *in esecuzione*).

Un oggetto diventa attivo quando riceve l'invocazione di un metodo da parte di un altro oggetto. Mentre l'oggetto ricevente è attivo, il richiedente è passivo (non in esecuzione) in attesa dei risultati del metodo invocato.

Una volta che i risultati sono stati inviati al richiedente, l'oggetto ricevente torna in uno stato passivo ed il richiedente prosegue la sua esecuzione.

Classi

Ogni oggetto è istanza di una determinata ***classe***.

Una *classe* definisce una tipologia di oggetti che possono essere descritti dallo stesso insieme di attributi e dallo stesso insieme di metodi.

Due istanze della stessa classe condividono le stesse ***proprietà strutturali*** (attributi) e le medesime ***proprietà comportamentali*** (metodi), ma sono comunque oggetti distinti.

Classi

Il concetto di classe fornisce al programmatore uno strumento per creare *nuovi tipi* che possono essere usati facilmente come i *tipi predefiniti* (come *int*, *float*, *double*, etc.).

Un tipo è la rappresentazione concreta di un *concetto*. Ad es., il tipo predefinito *double*, insieme alle sue operazioni +, -, * e /, fornisce una rappresentazione del concetto di numero reale.

Una classe è quindi un nuovo tipo definito dal programmatore. Si progetta un nuovo tipo per fornire la definizione di un concetto che non è definito direttamente tra i tipi predefiniti.

Ad esempio, può essere utile definire le classi *Resistore*, *Condensatore*, o *Diodo* in un programma per l'analisi o la progettazione di circuiti elettronici.

Proprietà fondamentali della POO

Classificazione. Ogni oggetto è istanza di una classe, che rappresenta un insieme di oggetti aventi le stesse proprietà strutturali e comportamentali.

Incapsulamento. Meccanismo che rende possibile il principio dell'*information hiding*, ovvero la capacità degli oggetti di nascondere al mondo esterno la propria organizzazione in termini di struttura e logica interna.

Ereditarietà. Meccanismo attraverso il quale una classe incorpora struttura e comportamento definiti da una classe più generale. La classe da cui si eredita è detta *superclasse*, mentre la classe discendente è detta *sottoclasse*.

Polimorfismo. Capacità di supportare operazioni con la medesima firma e comportamenti diversi, situate in classi diverse ma derivanti da una stessa superclasse.

Esempio: una classe e due oggetti

```
#include<iostream>
using namespace std;

class Data {
  public:
    int giorno;
    int mese;
    int anno;
    void init(int g, int m, int a) {
      giorno = g;
      mese = m;
      anno = a;
    }
};
```

attributi

metodi

```
int main () {
  Data a;
  a.init(1,1,1980);
  Data b;
  b.init(15,8,2006);
  cout << a.mese << "\n"; // 1
  cout << a.anno << "\n"; // 1980
  cout << b.giorno << "\n"; // 15
  b.giorno = a.giorno + 7;
  cout << b.giorno << "\n"; // 8
  system("pause");
}
```


Campi pubblici e campi privati

```
#include<iostream>
using namespace std;

class Data {
  private:
    int giorno;
    int mese;
    int anno;
  public:
    void init(int g, int m, int a) {
      giorno = g;
      mese = m;
      anno = a;
    }
    void stampa() {
      cout << giorno << "-" << mese
           << "-" << anno << endl;
    }
};
```

```
int main () {
  Data a;
  a.init(1,1,1980);
  // cout << a.mese; // errore
  // a.mese = 3; // errore
  a.stampa(); // 1-1-1980
  Data b;
  b.init(15,8,2006);
  // b.giorno = a.giorno + 7; // errore
  b.stampa(); // 15-8-2006
  system("pause");
}
```

Metodi definiti fuori dal corpo della classe

```
#include<iostream>
using namespace std;

class Data {
  private:
    int giorno;
    int mese;
    int anno;
  public:
    void init(int g, int m, int a);
    void stampa();
};

void Data::init(int g, int m, int a) {
   giorno = g;
   mese = m;
   anno = a;
}
```

```
void Data::stampa() {
   cout << giorno << "-" << mese
        << "-" << anno << endl;
}

int main () {
  Data a;
  a.init(3,5,2005);
  a.stampa(); // 3-5-2005
  system("pause");
}
```

Nota: Quando le funzioni sono definite all'interno del corpo della classe (come nei lucidi precedenti) sono implicitamente dichiarate come funzioni "inline".

Costruttori

```
#include<iostream>
using namespace std;

class Data {
  private:
    int giorno;
    int mese;
    int anno;
  public:
    Data(int g, int m, int a);
    void stampa();
};

Data::Data(int g, int m, int a) {
   giorno = g;
   mese = m;
   anno = a;
}
```

```
void Data::stampa() {
   cout << giorno << "-" << mese
        << "-" << anno << endl;
 }

int main () {
  Data a(3,3,2003);
  a.stampa(); // 3-3-2003
  system("pause");
}
```

Nota: I costruttori hanno lo stesso nome della classe e non specificano alcun valore di ritorno.

Metodi di accesso

```
#include<iostream>
using namespace std;

class Data {
  private:
    int giorno;
    int mese;
    int anno;
  public:
    Data(int g, int m, int a);
    int getGiorno();
    int getMese();
    int getAnno();
    void stampa();
};

Data::Data(int g, int m, int a) {
   giorno = g; mese = m; anno = a;
}
```

```
int Data::getGiorno() {
  return giorno;
}
int Data::getMese() {
  return mese;
}
int Data::getAnno() {
  return anno;
}
void Data::stampa() {
  cout << giorno << "-" << mese
       << "-" << anno << endl;
}
int main () {
  Data a(3,3,2003);
  // cout << a.anno; // errore
  cout << a.getAnno(); // 2003
  system("pause");
}
```


Metodi di accesso definiti nel corpo della classe

```
#include<iostream>
using namespace std;

class Data {
  private:
    int giorno;
    int mese;
    int anno;
  public:
    Data(int g, int m, int a);
    int getGiorno() { return giorno; }
    int getMese() { return mese; }
    int getAnno() { return anno; }
    void stampa();
};

Data::Data(int g, int m, int a) {
   giorno = g; mese = m; anno = a;
}
```

```
void Data::stampa() {
  cout << giorno << "-" << mese
       << "-" << anno << endl;
}

int main () {
  Data a(3,3,2003);
  // cout << a.anno; // errore
  cout << a.getAnno(); // 2003
  system("pause");
}
```

Nota: I metodi di accesso sono spesso definiti all'interno del corpo della classe