Programmazione a Oggetti

Ivano Salvo

Dipartimento di Informatica
Università “La Sapienza”
Stanza 310

Alcune informazioni…

Ricevimento:

Mercoledi' 11:30-12:30

Sito Web:

http://www.dsi.uniroma1.it/~poti

Esame:

progetto + scritto o orale

Laboratorio:

Gabbie, mer/gio/ven mattina

Materiali Didattici

- Lucidi
- Diversi tutorial in rete (inglese)
- Numerose letture consigliate
- Tutto presto disponibile sul sito.

Obiettivi del Corso

- Studio generale del paradigma a oggetti
- Approccio object-oriented alla soluzione dei problemi (linguaggio = modo di pensare)
- Sviluppo analisi critica Linguaggi di Programmazione
- Acquisizione Linguaggi a Oggetti:
  - SmallTalk
  - Java/C++

Linguaggi: perche?

- Cosa ci si aspetta da un Linguaggio di Programmazione?
- **Teoria**: per calcolare tutte le funzioni computabili sono sufficienti:
  - Assegnazione, operazioni aritmetiche, sequenza e while;
  - Anche meno: salto condizionato, successore, trasferimento di memoria

Ma vogliamo questo?

Linguaggi: evoluzione (1)

- Linguaggi Macchina
  - Istruzioni: stringhe di bit
  - Completo controllo del processore
  - Tipi di dato: celle di memoria
  - Controllo: salto (condizionato)
- Linguaggi Assembler
  - Uso di nomi e label simbolici
  - 1:1 con istruzioni macchina
  - Uso di label simboliche: astrazione utile? [**Sugg:** e se devo aggiungere un' istruzione?]

Linguaggi: evoluzione (2)

- Linguaggi Imperativi
  - Fortran (50), Algol (60), C, Pascal (70), …
  - Assegnazione/cicli
  - Tipi di dato: astrazione della cella di memoria
  - Astrazione procedurale/funzionale
- Vantaggi?
  - Orientati al problema (almeno un po‘)
  - Definizione di tipi di dato
  - **Riuso:** procedure e funzioni (opportunamente parametrizzate)

Linguaggi: evoluzione (3)

- Linguaggi Funzionali
  - LISP (58), ML (80), Haskell (90)
  - Funzioni Ricorsive
  - NO stato, ma ambiente di valutazione con associazioni nome-valore
  - Higher Order Function / Lazyness
- Vantaggi?
  - Ricca teoria algebrica
  - No alias, no side-effects (trasparenza referenziale)
  - **Riuso:** polimorfismo, funzioni di ordine superiore…

Linguaggi: evoluzione (4)

- Linguaggi Logici
  - ProLog
  - Significato Dichiarativo vs. Significato Procedurale
  - La definizione della specifica genera un meccanismo computazionale

- Vantaggi?
  - Programmare cosa piuttosto che programmare come (sulla carta, almeno)

Linguaggi: evoluzione (5)

- Linguaggi Object-Oriented
  - —Un po‘ di pazienza, abbiamo molto tempo per parlarne

LdP: cosa ci aspettiamo

- Core language che offra metafore semplici e uniformi (con chiara semantica!)
  - **Leggere**: Dan Ingalls "Design Principles behind SmallTalk" (sul sito)
- Metodi di suddivisione e ricombinazione
  - Procedure/funzioni
  - moduli
- Meccanismi di Astrazione
- Librerie ben integrabili e strumenti di Sviluppo
  - Debugger, IDE, integrazione col sistema …

Astrazione

- Identificare proprieta‘ importanti di cosa si vuole descrivere.
- Concentrarsi sulle questioni rilevanti e ignorare le altre.
- Cosa e‘ rilevante dipende dallo scopo del progetto.
- Essenziale per gestire la complessita‘
- Astrazione
  - Sul controllo
  - Sui dati

Astrazione e modularità

- Componente
  - Unita‘ di programma: funzione, struttura dati, modulo
- Interfaccia
  - Tipi e operazioni definiti in un componente che sono visibili *fuori* del componente stesso.
- Specifica
  - Funzionamento “inteso” del componente, espresso mediante proprieta‘ osservabili attraverso l’interfaccia.
- Implementazione
  - Strutture dati e funzioni definiti *dentro al componente, non necessariamente visibili da fuori.*

Esempio: Componente "funzione"

- Componente
  - Funzione che calcola la radice quadrata
- Interfaccia
  - float sqroot (float x)
- Specifica
  - Se x>1, allora sqroot (x)*sqroot(x) = x
- Implementazione

```
Float sqroot (float x) {
  float y = x/2; float step = x/4; int i;
  for (i=0; i<20; i++) {if ((y*y)<x) y=y+step;
                        else y=y-step; step=step/2;
  return y;
}
```

Esempio: Tipo di dato

- Componente
  - Coda a priorit_ : struttura dati che restituisce elementi in ordine decrescente di priorit_
- Interfaccia
  - Tipo: `PrioQueue`
  - Operazioni:
```
empty: PrioQueue
insert: ElemenType * PrioQueue -> PrioQueue
deletemax: PrioQueue -> ElemenType * PrioQueue
```
- Specifica
  - `insert`: Aggiunge all'insieme di elementi memorizzati
  - `Deletemax`: Restituisce l'elemento a amssima priorit_ e la coda degli elementi rimanenti

Astrazione: Tipi di Dato

- Cella di memoria
  - Nessuna protezione contro accessi incoerenti
- Variabili (con tipo)
- Costruttori di Tipo
  - Il programmatore puo' costruire nuovi tipi
- Tipi Astratti di Dato
  - Separazione tra interfaccia e implementazione
- Classi/Oggetti
  - Classificazioni e specificazione dei tipi di dato

Astrazione: Information Hiding

- Tipo di dato = valori e operazioni
  - Valori: `{-maxint..+maxint}`
  - Operazioni: `{+, -, *, div, mod}`
  - Non sono possibili altre operazioni
    (p.e., non e' ammesso lo shift)

- Information hiding
  - La raprresentazione dei dati e delle operazioni _
    inaccessibile all'utente.

Tipi di dato astratti

- Separazione interfaccia/implementazione
- Separazione garantita dal controllo di tipo
  - L'utente (il programma che utilizza il tipo di dato astratto) ha accesso alle sole operazioni dell'interfaccia.
  - L'implementazione _ incapsulata in opportuni costrutti
- Indipendenza dalla rappresentazione
  - Due implementazioni corrette di un tipo di dato astratto non sono distinguibili dai clienti di quel tipo.
    - Le modifiche alle implementazioni non richiedono modifiche al cliente.
    - Il cliente deve essere scritto senza conoscenza dei dettagli di implementazione del tipo di dato.

Astrazione: controllo

- go to
  - Nessuna limitazione sulla struttura del controllo
- Cicli (programmazione strutturata)
  - Linguaggi a blocchi: ogni blocco ha una sola entrata e una sola uscita
  - Programmi leggibili e debuggabili
- Funzioni e procedure
  - Favoriscono il riuso (parametri)
  - Metodologia top-down (mi riferisco a un esecutore piu' astratto)
  - Piu' facile valutare la correttezza (stato locale, …)

Verso l'OOP

- Integrazione di astrazione procedurale e sui dati (incapsulazione)
  - Oggetti composti di uno stato e da operazioni
- Distinzione netta tra significato dichiarativo (interfaccia) e implementazione (black-boxes)
  - Per interagire con un'automobile non devo conoscere la struttura del motore
- Riuso favorito da un'organizzazione gerarchica degli oggetti (ereditarieta')
  - Fiat 500 e Maggiolino sono entrambe automobili e avranno caratteristiche comuni
- Polimorfismo
  - Molte procedure possono essere definite sulle autombili in generale e applicabili indipendemente dal modello.