Programmazione a Oggetti
Lezione 2

Concetti Base della programmazione Object-Oriented

ADT: limiti

- Poca Estensibilita'

  Supponete di aver definito un ADT code: definire l'ADT code con priorita' implica ridefinire TUTTE le operazioni

Programmazione Object Oriented 1

- Individuare oggetti, entita' con un proprio comportamento, rilevanti per la soluzione del problema
- Selezionare le caratteristiche interessanti da esportare (interfaccia)
- Controllo: scambio messaggi tra oggetti

Programmazione Object Oriented 2

- Considerare le relazione tra oggetti
- Specializzare il comportamento di oggetti ottenendone di piu' specifici (raffinare l'analisi)
- Generalizzare raggruppando oggetti con alcune caratteristiche comuni
  - Vantaggio: ottimizzando o correggendo una funzione gli effetti vengono percepiti da tutti gli oggetti che la ereditano (localizzazione)

Linguaggi Object Oriented

- Incapsulazione

  Simile agli ADT, ma grazie a:

- Dynamic Lookup
- Sottotipaggio
- Ereditarieta'

  genera sistemi estensibili

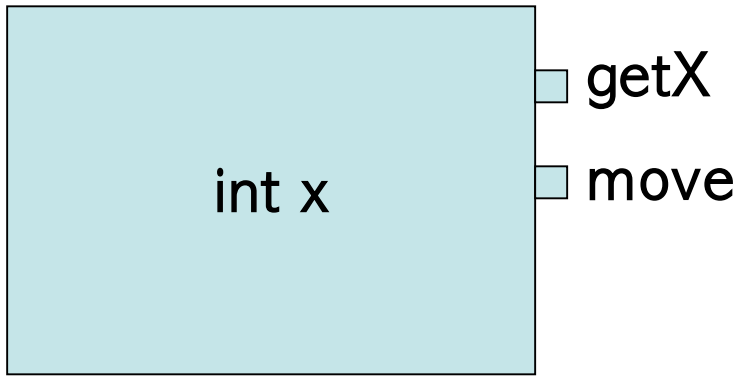
Oggetti

- Un oggetto consiste di:
  - Stato nascosto (variabili di istanza)
  - Operazioni pubbliche (metodi)
  - Eventuali metodi ausiliari privati
- Meccanismo computazionale
  - Scambio messaggi tra oggetti
  - Analogia tra invocazione di metodi e chiamate di funzione

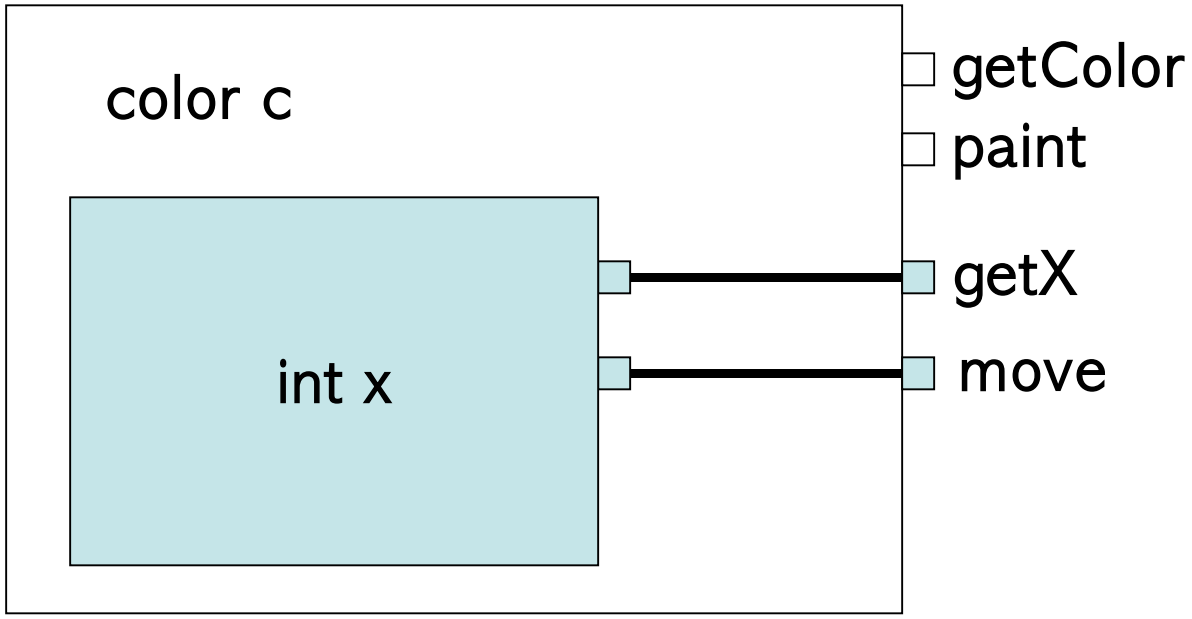
Classi

- Definiscono dei template da cui creare oggetti:
  - Struttura e comportamento degli oggetti
  - codice di inizializzazione (costruttori)
  - relazioni con altre classi (superclassi/sottoclassi)
- Linguaggi object-based
  - Oggetti vengono creati estendendone altri
- Multi-Methods

Esempio: Punto



Esempio: Punto Colorato



Non viene ridefinito il codice per getX e move

Ereditarietà e sottotipaggio

- Interfaccia
  - Insieme di messaggi a cui risponde un oggetto
- Sottotipaggio (o sostituvita')
  - Relazione di inclusione tra interfacce

- Implementazione
  - Rappresentazione interna di un oggetto
- Ereditarieta'
  - Relazione tra implementazioni

Sostitutività (Subtyping)

- Ogni funzione sui punti puo' essere applicata ai punti colorati
- L'interfaccia dei punti colorati contiene l'interfaccia dei punti
- Il tipo PuntoColorato e' sottotipo del tipo Punto
- Interessa il Cliente delle classi

Sostitutività

- Puo' essere *indipendente* dall'ereditarieta'
  - esempio: un algoritmo di ordinamento chiede che i dati da ordinare rispondano ad un metodo **`compare()`**
- Permette il *riuso* del codice, inducendo una forma di *polimorfismo*

Dynamic Lookup

- Il codice eseguito per effetto di un messaggio *dipende dal ricevente*
  - esempio: il metodo **`compare()`** esegue codici diversi su tipi diversi
- Un metodo puo' essere ridefinito (*overridden* o *specializzato*) all' interno della stessa gerarchia

Dynamic Lookup/Overloading

- Operatori come "+" eseguono codici diversi su tipi diversi (int, real, string), sono overloaded

- L'overloading viene risolto staticamente (*compile time*) mentre il dynamic lookup dinamicamente (*run time*)

Object Oriented Programming

- Metodologia di Programmazione
  - Organizzare concetti in oggetti e classi
  - Costruire sistemi etensibili
- Costrutti di Linguaggi OO
  - Incapsulare dati e funzioni negli oggetti
  - Subtyping permette di costruire dati estensibili
  - Ereditarieta' permette riuso del codice

Metodologia Object Oriented

[Booch]

- 4 passi:
  1. Identificare oggetti rilevanti a un certo livello di astrazione
  2. Identificare la semantica (comportamento) degli oggetti
  3. Identificare le relazioni tra gli oggetti (eventualmente classificando e raggruppando)
  4. Implementare gli oggetti
- Processo iterativo
  - Gli oggetti vengono implementati ripetendo 1-4
- Non necessariamente top-down