OOP - Introduzione

Introduzione alla Programmazione Orientata agli Oggetti

Andrea Colleoni

Lezione 1 - Contenuti

Lezione 1
Glossario e terminologia OOP
Concetti di base

Introduzione

- Object-oriented programming (OOP) is a programming paradigm that uses "objects" and their interactions to design applications and computer programs. It is based on several techniques, including inheritance, modularity, polymorphism, and encapsulation. It was not commonly used in mainstream software application development until the early 1990s. Many modern programming languages now support OOP.

- Procedural Approach
  - Data Structures can be represented as a network of associated structures, referring to one another.
  - Procedures can be represented as a network of routines which call one another, i.e., "call tree"
- Object Oriented Approach
  - Collection of discrete objects that incorporate data structures and behavior.
  - Each data structure has, combined with it, the procedures which apply to that data structure.
  - Contrasts with conventional programming in which data structures and behavior are only loosely connected
  - These entities, called objects, can be associated to one another in one network, rather than two.

Motivazioni

- Programmazione imperativa vs OOP
  - Limitazione dei side-effects
  - Promozione del riuso
  - Promozione dell'incapsulamento
- Ingegneria del SW
  - Gestione del processo di sviluppo
    - Interazione sviluppo committenza
    - Riduzione delle incomprensioni
  - Gestione del cambiamento
  - Capacità previsionale
    - Tempi
    - Costi

Classe

- Classe (o Tipo di dato Astratto o Classe Astratta)
  - In programmazione, un tipo di dato astratto o ADT (Abstract Data Type) è un tipo di dato le cui istanze possono essere manipolate con modalità che dipendono esclusivamente dalla semantica del dato e non dalla sua implementazione.
    Nei linguaggi di programmazione che consentono la programmazione per tipi di dati astratti, un tipo di dati viene definito distinguendo nettamente la sua interfaccia, ovvero le operazioni che vengono fornite per la manipolazione del dato, e la sua implementazione interna, ovvero il modo in cui le informazioni di stato sono conservate e in cui le operazioni manipolano tali informazioni al fine di esibire, all'interfaccia, il comportamento desiderato. La conseguente inaccessibilità dell'implementazione viene spesso identificata con l'espressione incapsulamento (detto anche information hiding: nascondere informazioni). (Wikipedia)

Metodo

- Metodo (o Funzione Astratta o Metodo Astratto)
  - Il termine metodo viene usato principalmente nel contesto della programmazione orientata agli oggetti per indicare un sottoprogramma associato in modo esclusivo ad una classe (metodo statico) oppure ad un oggetto (metodo di istanza). Come in una procedura di un linguaggio di programmazione procedurale, un metodo solitamente consiste in una sequenza di istruzioni scritte per eseguire una determinata azione, eventualmente sulla base di un insieme di parametri, e in grado di restituire al programma chiamante un valore di ritorno (o di output) di un determinato tipo
    Una delle operazioni che possono essere eseguite da un metodo è la lettura/scrittura di dati "privati" memorizzati in un oggetto o in una classe: in questo modo il programmatore può gestire in modo flessibile l'accesso ai dati, prevedendo, ove necessario, opportuni meccanismi di protezione e validazione.

    La differenza fra una routine ed un metodo è che il metodo, essendo associato ad un ben determinato oggetto, può accedere ai dati "privati" di quel oggetto secondo modalità "consistenti" con il comportamento che il programmatore intende dare a quello stesso oggetto. Quindi, ragionando nell'ottica della programmazione orientata agli oggetti, un metodo è qualcosa di più di una semplice sequenza di istruzioni. In effetti potrebbe essere definito come "un modo in cui un oggetto fornisce al programma un certo servizio": la chiamata del metodo, in questa ottica, è la "richiesta" inoltrata all'oggetto, affinché esso esegua un preciso "compito". Un altro modo di esprimere lo stesso concetto è quello di dire che, con l'esecuzione di un metodo, si trasmette un "messaggio" all'oggetto.

Classe (esempio)

```
class Bicycle {

    int speed = 0;
    int gear = 1;

    void changeGear(int newValue) {
        gear = newValue;
    }

    void speedUp(int increment) {
        speed = speed + increment;
    }

    void applyBrakes(int decrement) {
        speed = speed - decrement;
    }

    void printStates() {
        System.out.println("Speed:"+speed+" gear:"+gear);
    }
}
```

Campo

- Campo (o Attributo)
  - Un campo è una porzione di un oggetto che contiene un dato o un riferimento ad un dato.
    Il dato può essere di un qualunque tipo, anche di un tipo di dato astratto (classe).

- Campi e Metodi vengono anche detti membri di una classe

Oggetto

- Oggetto (o istanza di una classe)
  - Un oggetto è una istanza di una classe. Un oggetto occupa memoria, la sua classe definisce come sono organizzati i dati in questa memoria.
    Ogni oggetto possiede tutti gli attributi definiti nella classe, ed essi hanno un valore, che può mutare durante l'esecuzione del programma come quello di qualsiasi variabile. Il paradigma OOP suggerisce un principio noto come information hiding che indica che si debba accedere agli attributi dell'istanza solo tramite metodi invocati su quello stesso oggetto.
  - Costruttori, Distruttori e Garbage Collection
- Puntatore o riferimento
  - New
  - This, me, self
  - Cast

Concetti OOP...

- Ereditarietà
  - Inheritance provides a powerful and natural mechanism for organizing and structuring your software.
- Polimorfismo
  - In computer science, polymorphism is the use of a general interface to manipulate things of various specialized types. The concept of polymorphism applies to both data types and functions. A function that can evaluate to or be applied to values of different types is known as a polymorphic function. A data type that can appear to be of a generalized type is known as a polymorphic data type as is the generalized type from which such specializations are made.
- Interfaccia
  - An interface is a contract between a class and the outside world. When a class implements an interface, it promises to provide the behavior published by that interface.

Ereditarietà (esempio)

```
class MountainBike extends Bicycle {

    // new fields and methods defining a mountain
    bike would go here

}
```

Interfaccia (esempio)

```
interface Bicycle {

        void changeCadence(int newValue);

        void changeGear(int newValue);

        void speedUp(int increment);

        void applyBrakes(int decrement);
}
```

...Concetti OOP...

- Binding e scoping
  - **Binding**: In programming languages, name binding is the association of values with identifiers. An identifier bound to a value is said to reference that value. Since computers themselves have no notion of identifiers, there is no binding at the machine language level — name binding is an abstraction provided by programming languages. Binding is intimately connected with scoping, as scope determines when binding occurs.
  - **Scoping**: In computer programming, scope is an enclosing context where values and expressions are associated. Various programming languages have various types of scopes. The type of scope determines what kind of entities it can contain and how it affects them -- or semantics. Scopes can:
    - contain declarations or definitions of identifiers;
    - contain statements and/or expressions which define an executable algorithm or part thereof;
    - nest or be nested
- Package o Namespace
  - A package is a namespace for organizing classes and interfaces in a logical manner. Placing your code into packages makes large software projects easier to manage.

Binding (esempi)

List è un'interfaccia che consente di operare su liste di valori. Ha varie implementazioni (ad es. ArrayList, LinkedList etc.).

```
public void foo(List<String> l) {
    l.add("bar");
}
```

```
LinkedList<String> l;
l = new LinkedList<String>();
l.add("foo");
l = null;
```

...Concetti OOP...

- Coesione e accoppiamento: principi di Parnas
  - High cohesion
  - Loose coupling
- Information hiding (o incapsulamento)

Possibili problematiche OOP

- Problema della classe base fragile
- Diamond problem
- Circle-ellipse problem e il principio di sostituzione di Liskov