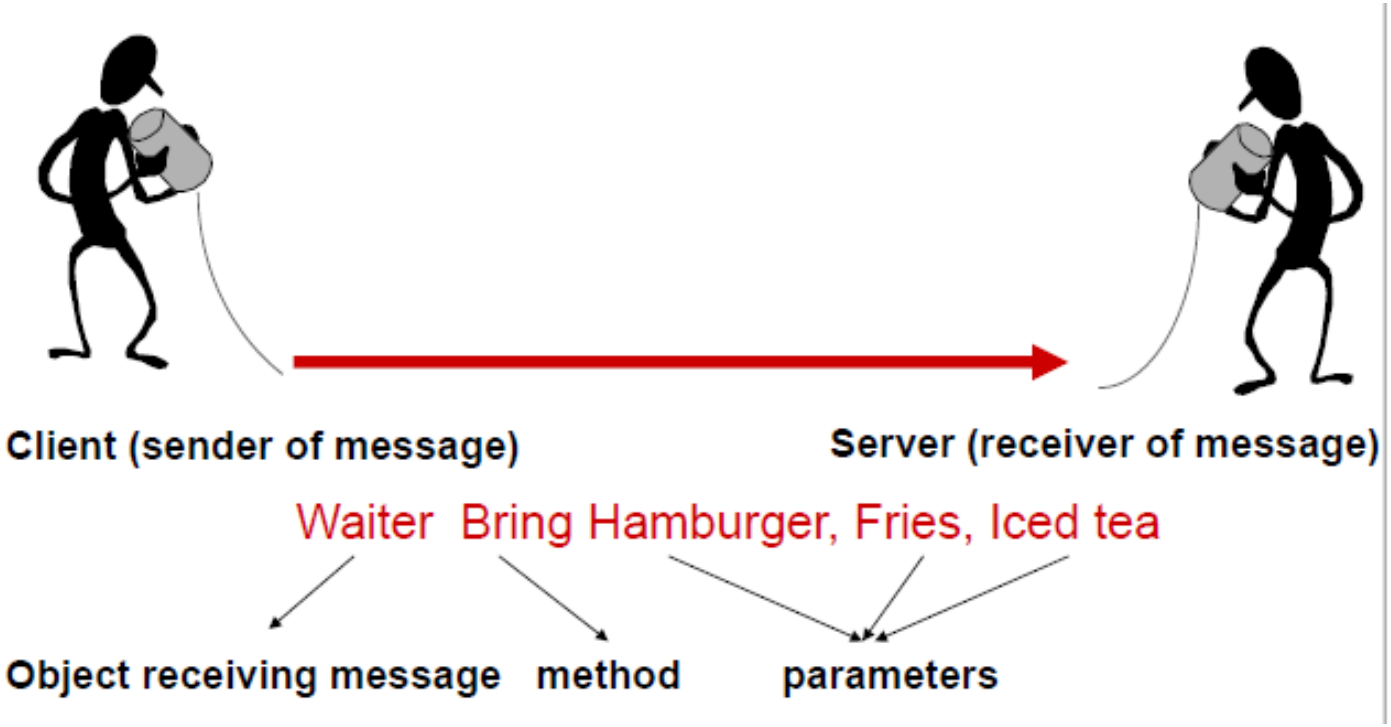
OOP

Lezione 3 Principi fondamentali

Come interagiscono gli oggetti?

Scambiandosi dei messaggi



Principi OOP: concetti basilari

- ASTRAZIONE
- INCAPSULAMENTO
- EREDITARIETÀ
- POLIMORFISMO

Astrazione ➔ Abstraction

- **Definizione:** L'**astrazione** è un procedimento mentale che permette di evidenziare alcune proprietà, ritenute significative, relative ad un determinato fenomeno osservato escludendone altre considerate non rilevanti per la sua comprensione.
  L'uso dell'**astrazione** comporta evidentemente la creazione di **modelli** (nel caso dell'OOP tali modelli sono rappresentati dalle **classi**)

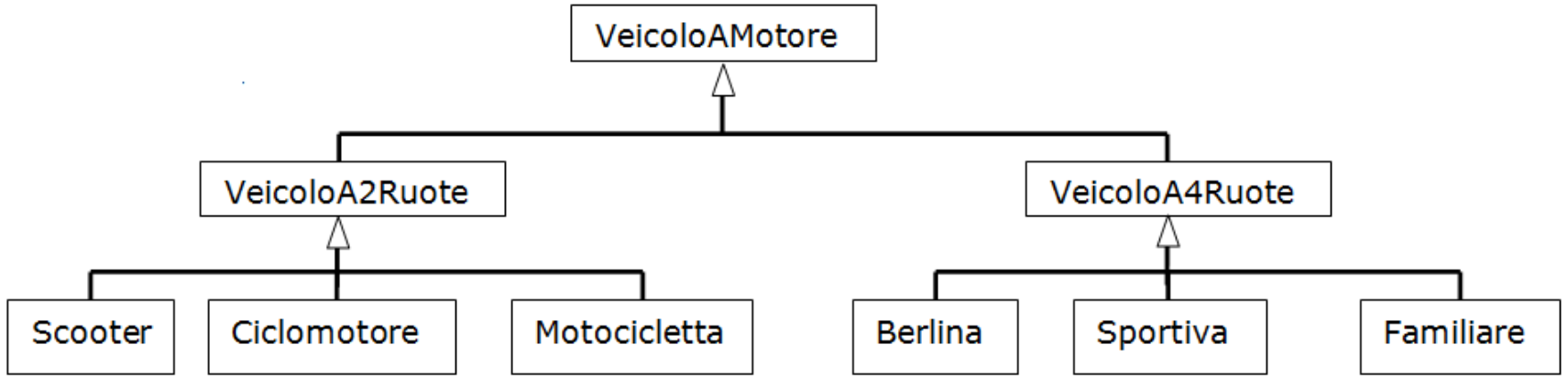
Incapsulamento ➔ Information hiding

- Abbiamo visto che due oggetti dialogano tra loro tramite uno **scambio di messaggi** ed abbiamo supposto finora che tutti i metodi e gli attributi di un oggetto possano essere visibili agli altri oggetti
  In realtà ogni oggetto **mittente non è obbligato a conoscere tutti i metodi dell'oggetto destinatario** poiché sono sufficienti solo quelli che il destinatario ritiene opportuno.
  Con **interfaccia** indichiamo la lista di **proprietà** e **metodi pubblici** che la classe rende noti all'esterno e che sono utilizzabili per interagire con esso.
  In questo modo **l'oggetto chiamante** conosce solo il modo di interagire con **l'oggetto chiamato** anche **se ignora completamente** i dettagli implementativi di quest'ultimo.
  Si crea come una capsula, un contenitore concettuale, che isola l'oggetto dalle cose esterne.

Ereditarietà ➔ Inheritance

- Molto spesso non è indispensabile o utile creare una classe dal nulla, ma è possibile utilizzare una classe già esistente (**ereditarietà semplice**) dalla quale partire per la sua creazione, specificando solo le differenze con quest'ultima.
  L'**ereditarietà** è un meccanismo che, in fase di definizione di classe, permette di specificare solo le differenze rispetto ad una classe (o più) già esistente, detta **classe base**.
  Tutte le altre caratteristiche ed i comportamenti della classe chiamata **derivata** che si sta definendo **saranno gli stessi della classe base.**
  Con questo concetto di ereditarietà si introduce quello di **gerarchia delle classi:**
  - Ad ogni classe possono essere associate una classe che la precede immediatamente sopra nella gerarchia (la sua **classe base**).
  - Ad ogni classe possono essere associate una o più classi che la seguono immediatamente sotto nella gerarchia (le sue **classi derivate**).

Ereditarietà ➔ Inheritance



- La **classe** diventa una combinazione dei metodi e delle proprietà di **tutte le classi base** che la precedono nella **gerarchia delle classi** ai quali vanno aggiunti i metodi e le proprietà specifici di quella classe

Polimorfismo ➔ Polymorphism

- Il **polimorfismo** è la capacità espressa dai metodi ridefiniti di assumere forme (implementazioni) diverse all'interno di **una gerarchia di classi** o all'interno **di una stessa classe**
  In altre parole il **polimorfismo** indica la possibilità dei metodi di possedere diverse **implementazioni**
  Quando un oggetto richiama un metodo di un altro oggetto appartenente ad una certa classe, esso verrà cercato dapprima in quella stessa classe. Se viene trovato (stessa firma della chiamata) sarà eseguito, altrimenti verrà ricercato risalendo nell'albero della gerarchia di classe (tra le sue classi base).
  Esistono due tipi di polimorfismo:
  -OVERLOAD
  -OVERRIDE

Polimorfismo ➔ Polymorphism

- Con il termine **overload** si intende la scrittura di più metodi identificati dallo stesso nome che però hanno, in ingresso, parametri di tipo e numero diverso.
  Una classe può ospitare un metodo con lo **stesso nome** ma del quale vengono fornite diverse implementazioni con firme differenti che il programmatore potrà utilizzare rispettandole all'atto della chiamata

- Con il termine **override** si intende una vera e propria riscrittura di un metodo all'interno di una classe **mantenendo intatta la sua firma** che abbiamo ereditato dalla classe base
  Si dice che la classe derivata espressamente **ridefinisce** (**overriding**) il metodo della classe base

A PIE

https://www.youtube.com/watch?v=m_MQYyJpljg&t=2s&ab_channel=ComputerScience

9min 15sec