Information hiding (Incapsulamento) M226A

Definizione d'incapsulamento

L'incapsulamento (information hiding) è la chiave della programmazione orientata agli oggetti.

Tramite esso, una classe riesce ad acquisire caratteristiche di robustezza, indipendenza e riusabilità. Inoltre la sua manutenzione risulterà più semplice al programmatore.

Di seguito sono elencate le caratteristiche dell'incapsulamento:

- Ispirato dalla realtà: tutti gli oggetti del mondo reale sono incapsulati
- Un oggetto deve possedere un **interfaccia pubblica** (che serve per l'interazione dell'oggetto con l'esterno)
- Un oggetto deve possedere **una parte nascosta** (invisibile dall'esterno dell'oggetto).

Esempio

Un semplice esempio è costituito da un oggetto frequentemente adoperato: lo smartphone.

La maggior parte degli utenti sa utilizzare uno smartphone; ignorando il funzionamento interno.

Chiunque può scegliere un contatto dalla rubrica, comporre un numero telefonico, chattare con gli amici e cosi via, ma pochi conoscono in dettaglio la sequenza dei processi scatenati da quei semplici tocchi sul touchscreen.

Per utilizzare uno smartphone, non è necessario essere un programmatore: basta saper usare la sua interfaccia pubblica e non la sua implementatazione interna.

Implementazione

Una qualsiasi classe è costituita da **dati** e **metodi**.

La filosofia dell'incapsulamento si basa sull'accesso controllato ai dati mediante metodi che possono prevenire l'utilizzo da classi non autorizzate ed evitare eventuali errori.

Per applicare l'incapsulamento ad una classe occorre:

- Dichiarare privati gli attributi al fine di renderli inacessibili fuori dalla classe stessa. A tale scopo utilizzando il modificatore ***private.***

- Creare dei metodi pubblici(dichiarati con il modificatore public) che permettono l'accesso agli attributi in maniera controllata. Tali metodi potrebbero realizzare controlli prima di confermare l'accesso ai dati privati, evitando che essi assumano valori non validi.

Metodi get e set

In Java per implementare l'incapsulamento è necessario dichiarare i dati privati e fornire alla classe metodi pubblici di tipo **get** e **set** (detti anchi metodi accessor e mutator) per accedervi rispettivamente in scrittura e lettura.

Questi metodi seguono una convenzione utilizzata anche nella libreria standard.

In presenza di una variabile d'istanza privata, chiamiamo questi metodi con la segeunte sintassi:

getNomeVariabile
setNomeVariabile

Esempio

```
public class Main {
    public static void main(String[] args) {
        Cane pluto = new Cane("Pluto", "Sconosciuta", 10);
        System.out.println(pluto.getPeso());
        pluto.setPeso(15);
        System.out.println(pluto.getPeso());
    }
}
```

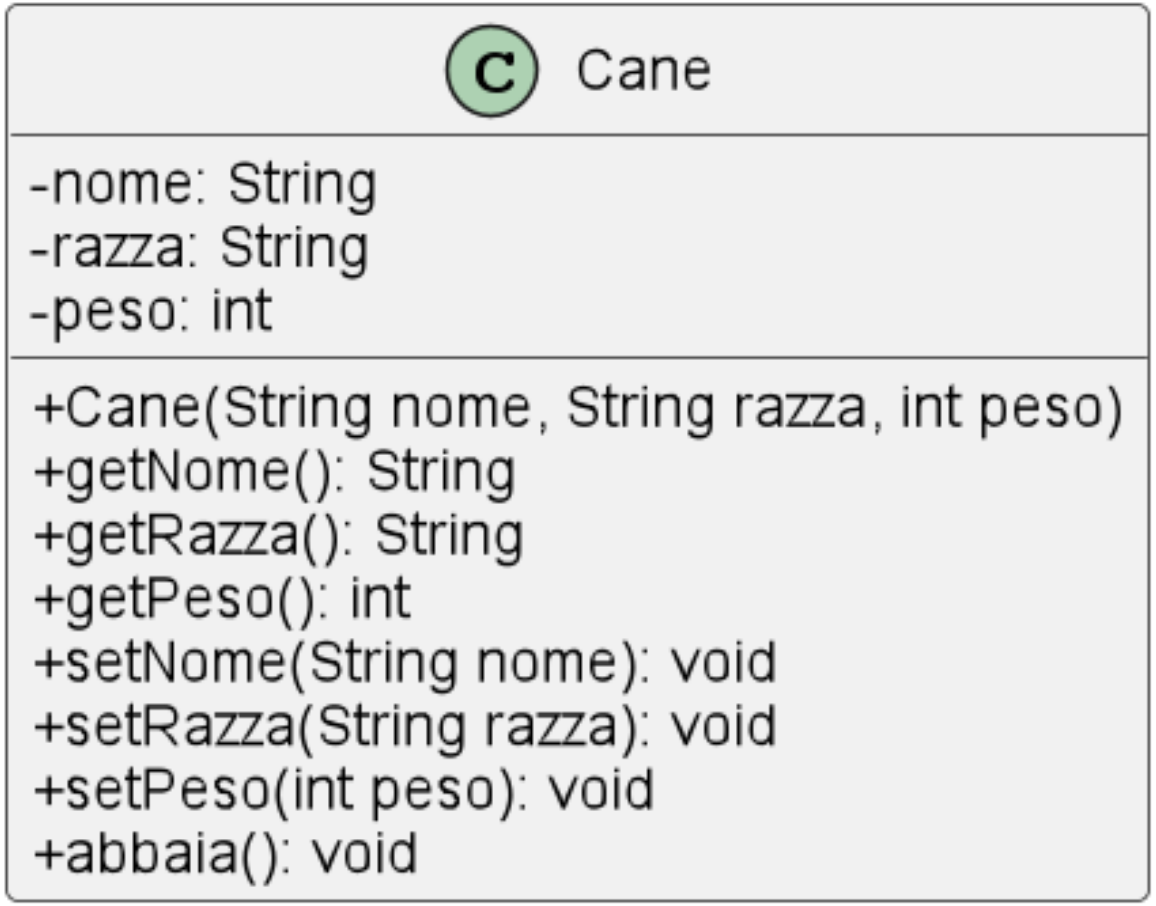
```
public class Cane{
    private String nome;
    private String razza;
    private int peso;

    public Cane(String nome, String razza, int peso){
        this.nome = nome;
        this.razza = razza;
        this.peso = peso;
    }
    public String getNome(){
      return nome;
    }
    public String getRazza(){
      return razza;
    }
    public int getPeso(){
      return peso;
    }
    public void setNome(String nome){
      this.nome = nome;
    }
    public void setRazza(String razza){
      this.razza = razza;
    }
    public void setPeso(int peso){
      if(peso >0){
        this.peso = peso;
      }
    }
}
```

Evitare immissioni errate

Grazie all'utilizzo del metodo setter possiamo evitare errori di immissione.
Rendendo più sicuro l'accesso ai dati dall'esterno.

```
public void setPeso(int peso){
    if(peso >0){
        this.peso = peso;
    }
```

UML (classe Cane)



Getter per attributi booleani

Generalmente il nome del metodo getter di un attributo di tipo boolean è formulato applicando il prefisso più efficace tra **is** o **has**.

Ad esempio **isMale** è meglio di **getMale** mentre **hasFleas** è meglio di **getFleas**

Esempio

```
public class Cane{
    private String nome;
    private String razza;
    private int peso;
    private boolean bassotto;
    private boolean pulci;

    …….

    public void setBassotto(boolean bassotto{…}
    public boolean isBassotto(){...}
    public boolean hasPulci(){...}

    ……..

}
```

Riassunto

Un oggetto è costituito da un insieme di metodi e attributi incapsulati in esso.

Un oggetto rende nota all'esterno una lista di metodi utilizzabili per interagire con se stesso, identificati da nome, numero e tipo dei parametri e, soprattutto, dalla parola chiave public.

Tale lista rappresenta l'interfaccia pubblica dell'oggetto.

L'implementazione dei metodi non è resa disponibile all'esterno: si realizza così **l'information hiding**, cioè non si consente di vedere l'implementazione dei metodi, se ne consente solo l'utilizzo.

Tutto ciò che è dichiarato **private** fa parte dell'informazione nascosta.