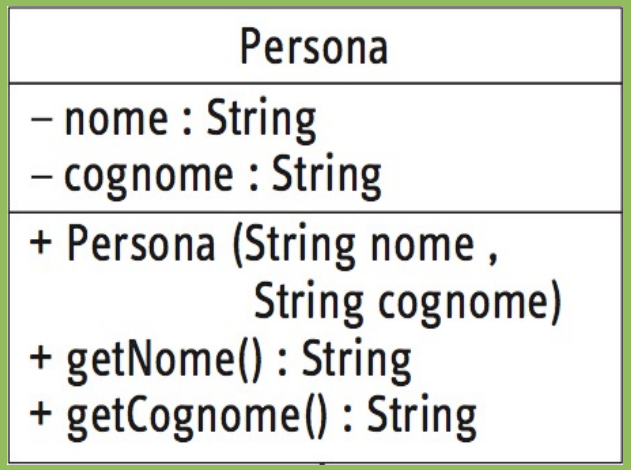
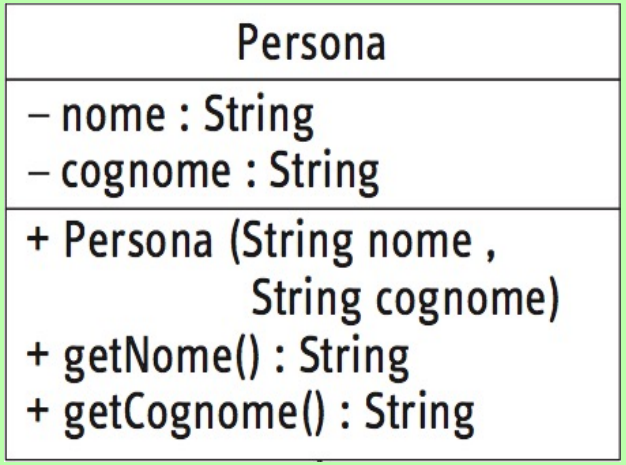
Esercizio

- Creare una classe di nome Persona con le variabili di istanza: nome, cognome di tipo stringa.
- La classe deve contenere un costruttore parametrico; i metodi getNome e getCognome.



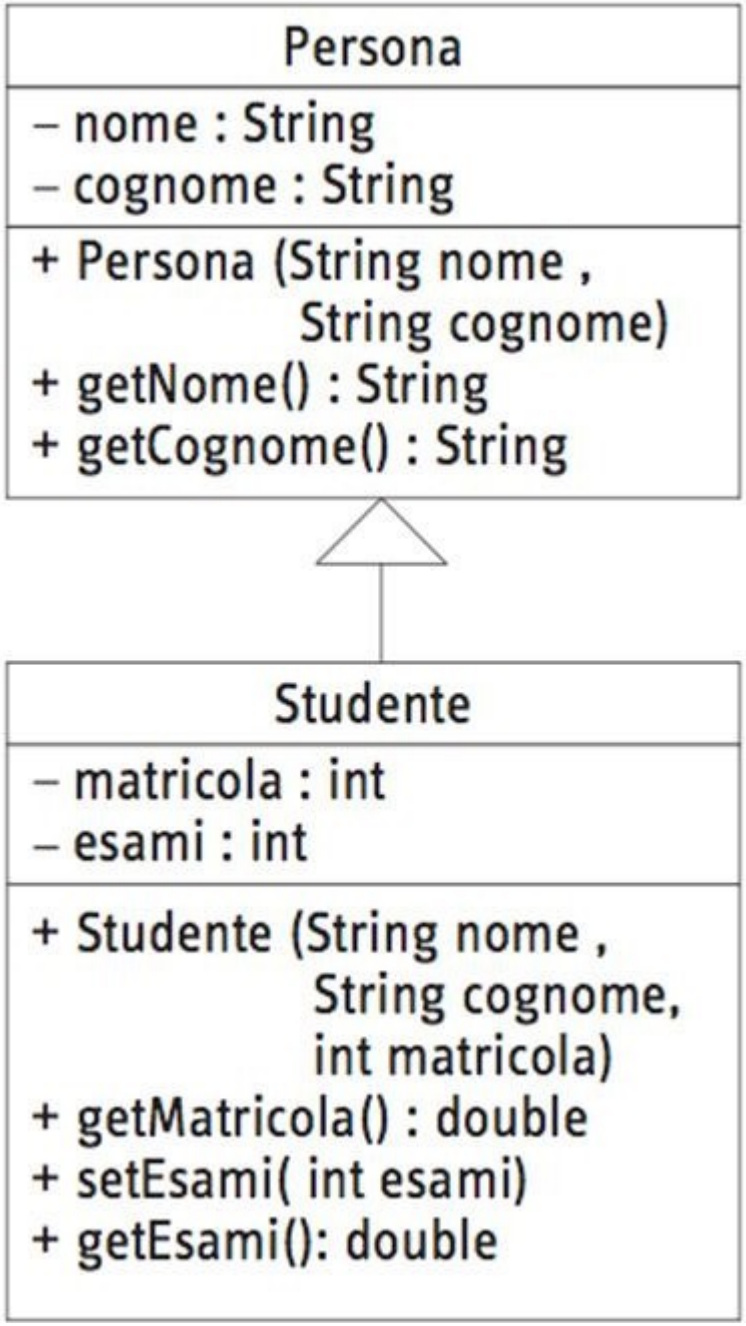
Esempio class Persona

```
class Persona {
  private String nome;
  private String cognome;

  public Persona(String nome,String cognome){
    this.nome=nome;
    this.cognome=cognome;
  }

  public String getNome(){
    return nome;
  }

  public String getCognome(){
    return cognome;
  }
}
```



Esercizio

- Costruire una sottoclasse di Persona, chiamata Studente, che contiene le variabili di istanza: *matricola* ed *esami*, che registra il numero di esami sostenuti, e devono essere entrambe di tipo intero.
- La sottoclasse deve contenere un costruttore parametrico ed i metodi set e get.


Persona
– nome : String
– cognome : String
+ Persona (String nome ,
String cognome)
+ getNome() : String
+ getCognome() : String
Studente
– matricola : int
– esami : int
+ Studente (String nome ,
String cognome,
int matricola)
+ getMatricola() : double
+ setEsami(int esami)
+ getEsami(): double

```
class Studente extends Persona {
  private int esami;
  private int matricola;
  public Studente(String nome,
                  String cognome,  int matricola) {
    super (nome,cognome);
    this.matricola=matricola;

  }
  public int getEsami(){
    return esami;
  }
  public void setEsami( int esami ){
    this.esami=esami;
  }
  public int getMatricola(){  return
    matricola;
  }
}
```

costruttore della superclasse

Studente
– matricola : int – esami : int
+ Studente (String nome , String cognome, int matricola) + getMatricola() : double + setEsami(int esami) + getEsami(): double

Esercizio

- Creare una classe di nome Persona con le variabili di istanza: cognome,nome, codice fiscale e città di tipo stringa.
- La classe deve contenere un costruttore di default che inizializzi le variabili di istanza con NULL; un costruttore parametrico; i metodi set e get ed un metodo chiamato AnnoNascita che a partire dal codice fiscale ricavi e restituisca l'anno di nascita di una persona.

Esercizio

- Creare un'applicazione Java che istanzi un oggetto della classe Persona e ne visualizzi in seguito nome, cognome ed anno di nascita.

Esercizio

- Costruire una sottoclasse di Persona, chiamata Impiegato, che contiene 1 variabile di istanza di tipo double:
  - *stipendio*, che registra la paga lorda annua.

- La sottoclasse deve contenere un costruttore parametrico ed i metodi set e get.

Esercizio

- Costruire una sottoclasse di Persona, chiamata Stagista, che contiene 2 variabili di istanza entrambe di tipo intero:
  - *numeroPresenze*, che registra il numero di ore di presenza, e
  - *numeroIdentificativo*, che identifica l'identificativo dello stagista
- La sottoclasse deve contenere un costruttore parametrico ed i metodi set e get.

Esercizio

- Creare 10 oggetti scelti in modo casuale tra Stagista e Impiegato
- Memorizzarli in un array
- Stampare Nome, Cognome di ogni persona
- Individuare la persona più giovane e più vecchia
- Stampare se è un impiegato o uno stagista

Esercizio Advanced

- Ordinare l'array per cognome, nome
- Chiedere all'utente una persona da cercare (nome, cognome) cercarla nell'array
- Stampare tutti i dati della persona se trovata altrimenti segnalare all'utente che la persona non esiste

Relazione tra ereditarietà ed incapsulamento

L'incapsulamento rappresenta una tecnica per rendere robusto il programma mentre l'ereditarietà è considerata come un ottimo strumento di sviluppo e semplificazione;

Per utilizzare entrambi i paradigmi andranno però fatte alcune considerazioni.

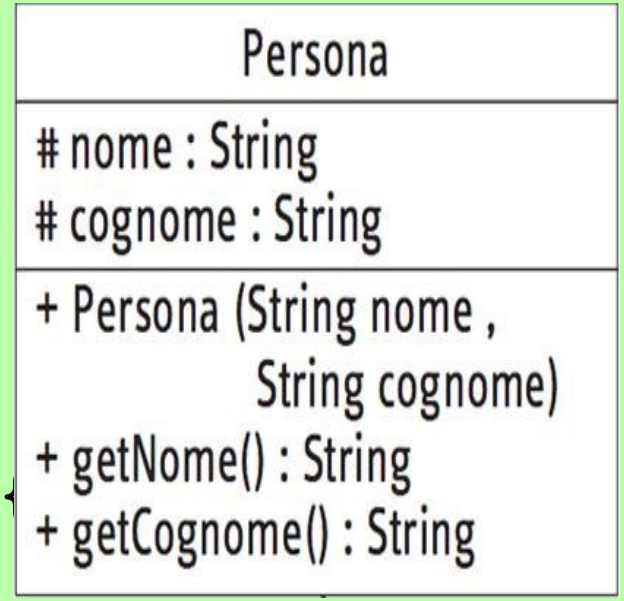
Ereditando una classe nella quale viene applicato l'incapsulamento, i suoi campi privati sarebbero accessibili solo attraverso l'utilizzo dei metodi set e get!

Per ovviare a questa limitazione esiste la possibilità di utilizzare il modificatore di accesso ***protected;***

una variabile dichiarata protected in una certa classe ***sarà accessibile da tutte le classi dello stesso package e da classi di package diversi che estenderanno la classe più generale***.

Esempio class Persona

```
public class  Persona {
  protected String cognome:
  protected String nome;

  public Persona(String nome, String cognome){
    this.nome=nome;
    this.cognome=cognome;
  }
  public String getNome(){
    return nome;
  }
  public String getCognome(){
    return cognome;
  }
}
```



Il modificatore di accesso `protected`:

- rende accessibile un campo *a tutte le sottoclassi*, presenti *e future*
- costituisce perciò un *permesso di accesso* valido per *ogni possibile sottoclasse che possa essere definita*.

Modificatori di accesso

- **private:** disponibile solo dall'ambito della stessa classe
- **protected**: disponibile nell'ambito delle sottoclassi
- **public**: disponibile per qualsiasi altra classe

- default… disponibile per qualsiasi altra classe appartenente allo stesso package

La parola chiave *super*

- nella forma `super(...)`, invoca un costruttore della classe base
- nella forma `super.val`, consente di accedere al campo val della classe base *(sempre che esso non sia private)*
- nella forma `super.metodo()`, consente di invocare il metodo metodo() della classe base *(sempre che esso non sia private)*