Fondamenti di Informatica T-1

Ereditarietà

Tutor:
Angelo Feraudo
angelo.feraudo2@unibo.it

a.a. 2017/2018

Ereditarietà: definizione

Meccanismo per definire una nuova classe (classe derivata o classe figlio) come specializzazione di un'altra (classe base o classe padre)

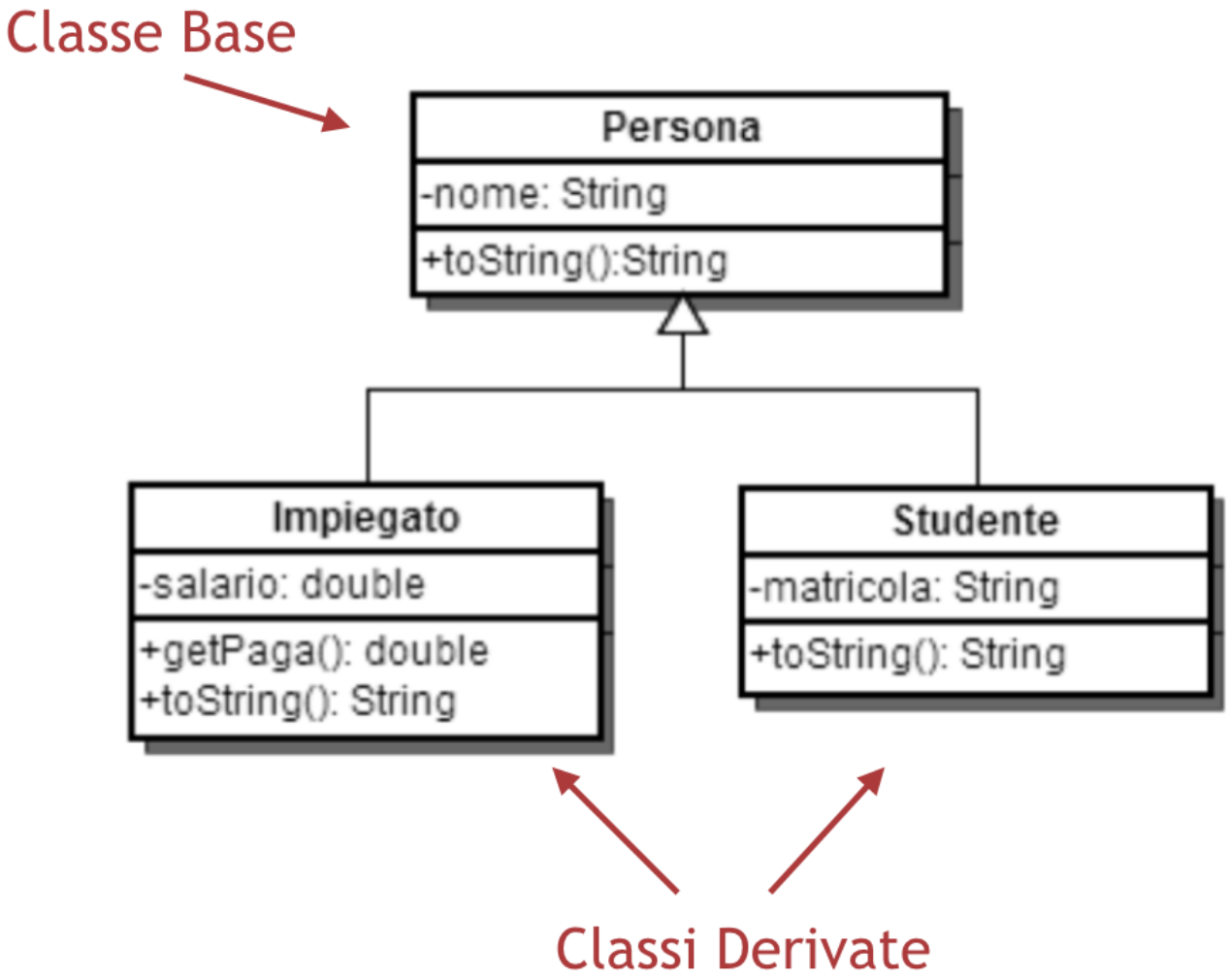
- La classe base modella un concetto generico
- La classe derivata modella un concetto più specifico

Ereditarietà: classe derivata

La classe derivata:

- Dispone di tutte le funzionalità (attributi e metodi) della classe base
- Può aggiungere funzionalità proprie
- Può ridefinire il funzionamento di metodi esistenti (polimorfismo)

Ereditarietà: esempio(1)



Ereditarietà: esempio(2)

Classe Persona

```
public class Persona {

    protected String nome;


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

    public String getNome() {
        return nome;
    }

    public String toString(){
        String s = "";
        s += "Nome: " + nome + "\n";
        return s;
    }
```

Ereditarietà: esempio(3)

Classe Impiegato

```
public class Impiegato extends Persona{

    private double salario;

    public Impiegato(String nome, double salario) {
        super(nome);
        this.salario = salario;
    }


    public double getPaga() {
        return salario;
    }

    @Override
    public String toString(){
        String s = super.toString();
        s += "Salario: " + salario + "€\n";
        return s;
    }
```

Ereditarietà: esempio(4)

Classe Studente

```
public class Studente extends Persona{

    private String matricola;

    public Studente(String nome, String matricola) {
        super(nome);
        this.matricola = matricola;
    }

    @Override
    public String toString(){
        String s = super.toString();
        s += "Matricola: " + matricola + "\n";
        return s;
    }



}
```

Ereditarietà: parole chiave

Parola chiave "extends" :
Specifica da quale classe eredita.
Nell'esempio, Studente eredita da Persona

Parola chiave "super" :
Consente di invocare un metodo, un costruttore o un attributo della classe base purché non privati

Annotazione "@Override" :
Permette al compilatore di verificare la ridefinizione di un metodo della superclasse. Il nuovo metodo deve avere stesso nome, parametri e tipo di ritorno (magari anche stessa semantica)

```
public class Studente extends Persona{
    private String matricola;
    public Studente(String nome, String matricola) {
        super(nome);
        this.matricola = matricola;
    }
    @Override
    public String toString() {
        return super.toString() +
        ", numero di matricola: " + this.matricola;
    }
}
```

ESERCIZIO 1

Scrivere un programma per la gestione di racconti brevi.

- La classe **Testo** è composta da un autore, un titolo ed un contenuto.
- Scrivere inoltre una classe figlio chiamata **TestoCensurato** che presenta un ulteriore attributo, chiamato "parola proibita".
- Oltre ad i metodi getter e setter, ridefinire per entrambe le classi il metodo **toString()**, facendo in modo che la classe TestoCensurato restituisca il proprio contenuto andando a sostituire ogni occorrenza della parola proibita con una successione di tre asterischi.
- Scrivere inoltre una classe contenente un main di prova per testare il corretto funzionamento delle classi.

ESERCIZIO 2 (1)

Scrivere un programma che consenta di gestire il pagamento degli stipendi dei dipendenti di un'azienda.
Di ogni **dipendente** si memorizzano il nome, l'indirizzo, il telefono e lo stipendio.
Alcuni dipendenti sono **dipendenti giornalieri**, con stipendio calcolato in base al numero di giorni di lavoro in un mese (numero di giorni * base di retribuzione giornaliera).

ESERCIZIO 2 (2)

L'azienda deve gestire, oltre ai dati dei dipendenti, le **buste paga** che sono caratterizzate da:

- un riferimento al dipendente
- una data nel formato `gg/mm/aaaa`
- un id univoco che deve essere generato automaticamente dal sistema nel seguente formato:
  `<data>-<progressivoNumerico>`
  dove `<data>` corrisponde alla data della busta paga e `<progressivoNumerico>` è un usuale numero progressivo
- un importo corrispondente allo stipendio da attribuire al dipendente

ESERCIZIO 2 (3)

- L'**azienda** gestisce dipendenti e buste paga in vettori di massimo 200 dipendenti e 2500 buste paga, non necessariamente pieni
- Si crei un metodo **inserisciBustaPaga()** che permette, se possibile, l'inserimento di una busta paga nell'array di buste paga
- Si crei un metodo **giornoDiPaga()** che, relativamente ad una certa data, crei le buste paga dei dipendenti, le memorizzi e le mostri a video.
- Si implementino metodi per ottenere le liste di dipendenti e delle buste paga.
- Nel **main** si istanzi un'azienda, si inseriscano dei dipendenti, si stampino le liste dei dipendenti e si esegua almeno un aggiornamento delle buste paga.