

# Esercitazione 9

## Ereditarietà

### Programmazione e Analisi di Dati Mod. A – Programmazione Java

**Esercizio 1.** Scrivere una classe `Dipendente` che estende la classe `Persona` vista a lezione. Ogni dipendente ha un proprio anno di assunzione e un proprio stipendio. Si definiscano costruttori e vari metodi `get` e `set` opportuni. Si ridefinisca inoltre il metodo `visualizza()` opportunamente. Si definisca inoltre un metodo `guadagnaPiuDi` che prende come parametro un altro oggetto `Dipendente` e restituisce `true` se l'oggetto corrente ha uno stipendio maggiore di quello ricevuto come parametro, o `false` altrimenti.

Per testare la classe, scrivere un programma `UsaDipendente` che crea tre oggetti della classe `Dipendente` e li visualizza in ordine di stipendio (usando il nuovo metodo per confrontare gli stipendi).

**Esercizio 2.** Scrivere una classe `Veicolo` che prevede una targa, una marca e un modello. La classe prevede anche una variabile booleana che descrive se il veicolo è guasto. Aggiungere un costruttore opportuno e vari metodi `get` e `set` opportuni. Scrivere la classi `Vettura` e `Motociclo` che estendono la classe `Veicolo`. La classe `Vettura` prevede una stringa che ne descrive la tipologia ("`utilitaria`", "`station wagon`", "`SUV`", ...) mentre la classe `Motociclo` prevede un numero che ne descrive la cilindrata (50, 125, ...).

Per testare le classi, scrivere un programma `UsaVeicoli` che crea un array inizializzato con veicoli delle varie tipologie. Alcuni dei veicoli inseriti nell'array dovranno diventare guasti. Il programma deve stampare la lista delle targhe dei veicoli guasti.

**Esercizio 3.** Scrivere una classe `Officina` che prevede solo un metodo `ripara()` che utilizza veicoli (come definiti nell'esercizio precedente). Tale metodo prende un veicolo come parametro, ne cambia (se necessario) il valore della variabile booleana che descrive lo stato di guasto e restituisce come risultato il prezzo dell'intervento. Il prezzo deve variare a seconda che il veicolo fosse guasto o meno, e a seconda della tipologia di veicolo.

Per testare le classi, scrivere un programma `UsaOfficina` che crea un po' di veicoli e un oggetto di tipo `Officina`, e usa il metodo `ripara()` varie volte su oggetti diversi (guasti o meno) stampando i prezzi ottenuti.

**Esercizio 4.** Scrivere la classe `Quesito` i cui oggetti rappresentano domande di esami orali. Ogni quesito si compone di una domanda, di una risposta corretta e di un punteggio, e mette a disposizione un metodo `ask()` che stampa la domanda, legge la risposta dell'utente e restituisce il punteggio conseguito (0 se la risposta dell'utente è sbagliata).

Scrivere la classe `QuesitoSiNo` che estende la classe `Quesito` in modo da rappresentare domande a cui possa essere risposto solo si o no. Sovrascrivere il metodo `ask()` in modo da garantire che l'utente risponda si o no (prima che venga restituito il punteggio conseguito).

Scrivere la classe `QuesitoNumerico` che estende la classe `Quesito` in modo da rappresentare domande a cui possa essere risposto solo con un valore intero. Sovrascrivere il metodo `ask()` in modo da garantire che l'utente risponda con un valore intero (prima che venga restituito il punteggio conseguito).

Scrivere la classe `QuesitoMultiplo` che estende la classe `QuesitoNumerico` in modo da rappresentare domande che offrono un certo numero di opzioni (prefissato) e alle quali possa essere risposto solo con un valore intero positivo minore o uguale al numero di opzioni disponibili. Sovrascrivere il metodo `ask()` in

modo da garantire che l'utente risponda con un valore consentito (prima che venga restituito il punteggio conseguito).

Scrivere il programma `Test` che riempie un array con quesiti di diversa natura e poi simula un'interrogazione calcolando il punteggio totale ottenuto. A scelta, l'interrogazione può essere fatta estraendo in modo casuale tre quesiti dall'array. Per l'estrazione casuale usare il metodo `nextInt(int n)` della classe `Random` (importare `java.util.Random`).