**NOTA: per ogni esercizio relativo all'ereditarietà realizzare il diagramma delle classi equivalente.**

**Esercizio - 1.**

Creare una classe di nome Person con le variabili di istanza: surname, name, tax code e city 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 bornYear che a partire dal codice fiscale ricavi e restituisca l'anno di nascita di una persona.

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

Costruire una sottoclasse di Person, chiamata Stagista, che contiene 2 variabili di istanza entrambe di tipo intero:

* numberOfPresence, che registra il numero di ore di presenza
* idNumber (numero identificativo).

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

Creare tre oggetti di tipo Stagista memorizzarli in un array e visualizzare lo Stagista più giovane.

**Esercizio - 2.**

Scrivere una classe Employee (dipendente) che estende la classe Person.

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 visualize() opportunamente.

Si definisca inoltre un metodo gainsMore che prende come parametro un altro oggetto Employee 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 TestEmployee che crea tre oggetti della classe Employee e li visualizza in ordine di stipendio (usando il nuovo metodo per confrontare gli stipendi).

**Esercizio - 3.**

Scrivere una classe Vehicle (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 Car e Motocycle che estendono la classe Vehicle. La classe Car prevede una stringa che ne descrive la tipologia ("utilitaria","station wagon", "SUV",....) mentre la classe Motocycle prevede un numero che ne descrive la cilindrata (50, 125, ....).

Per testare le classi, scrivere un programma TestVehicles 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 - 4.**

Scrivere una classe Garage che prevede solo un metodo repair() 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 TestGarage che crea un certo numero di veicoli e un oggetto di tipo Garage, e usa il metodo repair() varie volte su oggetti diversi (guasti o meno) stampando i prezzi ottenuti.

**Esercizio - 5.**

Scrivere la classe Question 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 `e sbagliata).

Scrivere la classe QuestionYesNo che estende la classe Question 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 NumericQuestion che estende la classe Question 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 MultipleQuestion che estende la classe NumericQuestion 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 TestQuestion 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).