Esercitazione - Classi (309)

Fondamenti 2 - Corso di Laurea in Informatica

Esercizio 1: Progettare e implementare una classe Concessionaria tenendo conto delle seguenti specifiche:

- La concessionaria può contenere due tipologie di veicoli: Automobili e Moto. Utilizzare l'ereditarietà, creando la classe Veicolo, per generalizzare le classi Automobile e Moto.
- Ogni veicolo contiene le informazioni riguardanti la targa, il prezzo, la casa produttrice ed il nome del veicolo. La targa del veicolo deve essere univoca per ogni veicolo.
- Deve essere presente un metodo che restituisca il prezzo scontato, calcolato come segue:
  - Per le automobili con un prezzo inferiore ai 10000 euro, lo sconto è del 5%
  - Per le automobili con un prezzo inferiore ai 20000 euro, lo sconto è del 10%
  - Per le moto con un prezzo inferiore ai 7000 euro, lo sconto è del 3%
  - Per le moto con un prezzo inferiore ai 15000 euro, lo sconto è del 7.5%
  - Negli altri casi il prezzo non è scontato
- Deve essere possibile aggiungere e rimuovere un veicolo dal concessionario.
- Deve essere possibile stampare, utilizzando l'operatore << di Veicolo, tutti i veicoli presenti.

Progettare e implementare una classe GestoreVeicoli che si occupi di gestire i veicoli da aggiungere alla concessionaria.
Inoltre, implementare il seguente menù:
Premi 1 per aggiungere un'automobile al gestore veicoli
Premi 2 per aggiungere una moto al gestore veicoli
Premi 3 per aggiungere un veicolo del gestore veicoli alla concessionaria (usando la targa)
Premi 4 per rimuovere un veicolo dalla concessionaria (usando la targa)
Premi 5 per stampare il prezzo scontato di un veicolo (usando la targa)

Premi 6 per stampare tutti i veicoli presenti nella concessionaria
Premi 9 per uscire

Esercizio 2: Realizzare una classe CodaEreditaria che gestisca una coda di studenti. Utilizzando la classe Studente (con campi privati matricola, isee, media), la classe deve ereditare in modo privato list<Studente> e implementare i seguenti metodi:

- void aggiungi(Studente s); aggiunge uno studente solo se uno studente con la stessa matricola non è inserito nella coda.
- Studente prossimo() const; restituisce il prossimo studente in coda
- void rimuovi(); rimuove il prossimo studente dalla coda
- unsigned int size() const; restituisce la dimensione della coda

La coda ha il seguente ordine di priorità:

- Prima gli studenti con la media più alta del 28 in ordine di arrivo
- Poi tutti gli studenti con una media compresa tra 26 e 28 ordinati per matricola
- Poi tutti gli altri studenti ordinati per isee

Esercizio 3: Utilizzando le classi Computer.h, GestoreComputers.h e GestoreComputers.cpp allegate alla traccia, implementare i 4 metodi in GestoreComputers.cpp.