Università degli Studi di Trento 19 giugno 2019

Linguaggi di Programmazione — Programmazione 2

Prova al calcolatore *Prof. Gian Pietro Picco*

Obiettivi e funzionalità richieste

Si vuole gestire l'archivio dati (semplificato) degli abbonati che frequentano un impianto sportivo.

Il programma deve gestire l'anagrafica degli abbonati come segue.

- per ogni persona viene registrata la coppia cognome e nome, il codice fiscale e l'anno di nascita;
- per gli studenti viene altresì registrata l'università di provenienza e il tipo di corso di laurea (triennale oppure magistrale);
- per gli atleti professionisti viene altresì registrata la disciplina sportiva praticata e la rilevanza dell'atleta (nazionale o internazionale).

Queste informazioni vengono utilizzate per determinare la scontistica applicata alla tariffa base mensile dell'abbonamento, che normalmente ammonta a 1000 EUR:

- alle persone nate prima del 1968 viene applicato uno sconto "senior" pari al 35%;
- agli studenti viene applicato uno sconto *i)* del 40% se iscritti a un corso di laurea triennale oppure *ii)* del 20% se iscritti a un corso di laurea magistrale;
- agli atleti professionisti viene praticato uno sconto *i)* del 50% se di rilevanza internazionale oppure *ii)* del 30% se di rilevanza nazionale;
- nel caso in cui un abbonato ricada in più di una categoria viene applicata la tariffa più bassa: si presti attenzione ai dati di esempio descritti nel seguito.

~~I dati anagrafici degli abbonati vengono visualizzati insieme alla tariffa corrispondente mediante l'interfaccia grafica in figura, realizzata in una finestra di 500 × 250 pixel.~~

~~L'interfaccia offre 6 diverse modalità di visualizzazione:~~

- ~~i 3 bottoni in alto a sinistra consentono di~~ visualizzare, rispettivamente, tutti gli abbonati nell'archivio, i soli studenti, i soli atleti; e relative tariffe
- ~~i 2 bottoni in alto a destra consentono di~~ ordinare gli abbonati ~~attualmente selezionati~~ (vedi sopra) secondo il loro nome oppure il loro anno di nascita, in ambedue i casi con ordinamento ascendente;
- ~~la selezione di una modalità fa sì che il bottone corrispondente sia disabilitato, in modo da rendere evidente all'utente quale combinazione è attualmente attiva. Il bottone viene abilitato nuovamente quando un altro bottone dello stesso gruppo viene premuto (e a sua volta disabilitato).~~

~~Il bottone *Exit* in basso a destra chiude la finestra e causa l'uscita dal programma.~~

Requisiti utilizzare i generics ovunque possibile!

- **Si inizializzi l'archivio obbligatoriamente con i dati riportati in figura, pena significativa riduzione punti in fase di correzione.** Ciò può essere fatto ad esempio nel costruttore dell'applicazione principale.
- Si usi, quando evidente/utile, una gerarchia di ereditarietà, sfruttando ove possibile il polimorfismo.
- Si usino costanti ove ragionevole, e si badi alla pulizia del codice (es., linee guida Java) evitando duplicazioni e codice inutilmente complesso.
- ~~La disposizione degli elementi grafici deve rispecchiare fedelmente quanto mostrato; tuttavia, ciò è considerato meno importante rispetto alla corretta funzionalità dell'interfaccia e del resto dell'applicazione.~~
- ~~È richiesto il *class diagram* UML. È sufficiente la vista di alto livello (no attributi/metodi) ma devono essere mostrate le associazioni più importanti, oltre a quella di ereditarietà, analogamente a quanto fatto a lezione. Il diagramma UML deve essere consegnato su un foglio protocollo indicando nome, cognome, numero di matricola.~~

~~Suggerimenti~~

- ~~Per realizzare il layout dell'interfaccia grafica: una possibilità è usare una combinazione di `BorderPane`, `AnchorPane` e `HBox`.~~
- ~~Per realizzare la barra colorata in alto: si può settare il background del *pane* corrispondente con~~

```
top.setBackground(new Background(
        new BackgroundFill(Color.CADETBLUE, CornerRadii.EMPTY, Insets.EMPTY)));
```