UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

Corso di laura magistrale in Ingegneria Informatica

PROGETTO C++

Fabio Filippo Mandalari

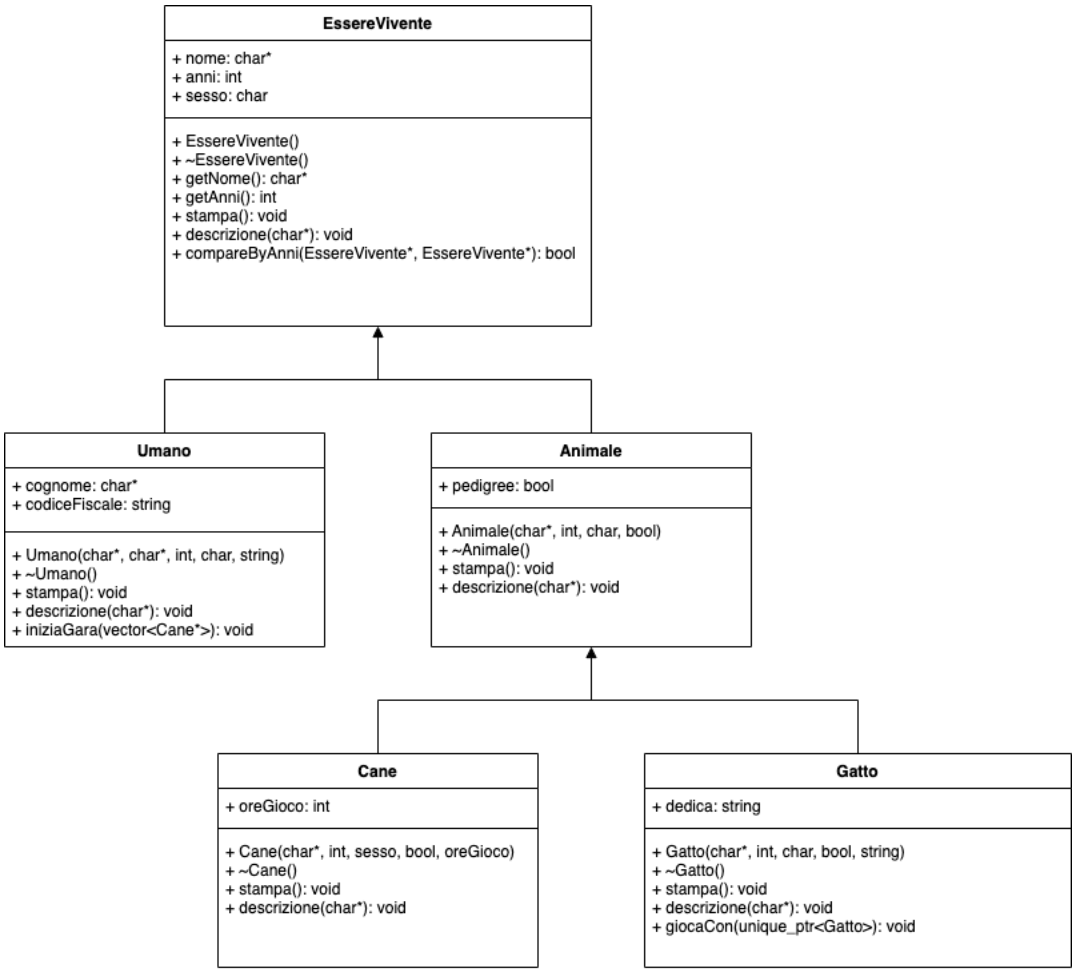
Matricola 1047426

A.A. 2021/2022

IDEA DI PROGETTO

Il punto di partenza per lo sviluppo di questo progetto è stato la costruzione di una gerarchia su tre livelli che prevedesse come padre la classe EssereVivente, come figlie di livello 1 di EssereVivente le due classi Umano e Animale e, da ultimo, come figlie di livello 2 della classe Animale le classi Cane e Gatto.

In notazione UML la gerarchia si presenta così:



CLASSE ESSEREVIVENTE

La classe EssereVivente si preoccupa di fornire una base su cui sviluppare l'intero progetto:

- Il metodo `getNome()` serve per poter estrarre il nome dell'oggetto che lo invoca. Tale metodo viene invocato in tre diverse situazioni per stampare i nomi di alcuni oggetti Umano, di alcuni oggetti Cane e di alcuni oggetti Gatto.
- Il metodo `getAnni()` serve per poter estrarre il dato relativo all'età dell'oggetto che lo invoca. Tale metodo viene invocato solo una volta nel contesto di confronti che vengono effettuati tra oggetti della classe Umano.
- Il metodo `stampa()` serve poter stampare tutte le informazioni che compongono ciascun oggetto. Questo metodo viene ridefinito in tutte le sottoclassi in modo che ognuna di queste possa aggiungere le informazioni specificate nella classe stessa che non sono presenti nella superclasse.
- Il metodo `descrizione()` stampa a schermo una scritta relativa alla tipologia della classe di appartenenza del particolare oggetto che lo invoca. Di questo metodo se ne fa l'override in tutte le sottoclassi della gerarchia.
- Il metodo `compareByAnni()` serve per poter effettuare dei confronti basati sull'età al fine di riordinare un vettore di oggetti appartenenti ad una qualunque classe della gerarchia.

CLASSE UMANO

Nella classe Umano oltre a fornire sia la ridefinizione del metodo `stampa()` sia l'override del metodo `descrizione()` vi è la definizione del metodo `iniziaGara()`. L'idea alla base di questo metodo è la seguente: un giudice (oggetto della classe Umano) può indire una gara di bellezza per cani una volta che ottiene un vettore contenente oggetti della classe Cane (`vector<Cane*>`). La logica di funzionamento viene illustrata nel contesto della spiegazione della classe Cane.

Oltre a quanto implementato nel metodo `iniziaGara()` ho pensato di creare all'interno del `main()` un vettore che possa contenere oggetti della classe Umano. Tale vettore viene passato come parametro ad un altro oggetto che, tramite un'implementazione basata sull'approccio del design pattern Facade, si preoccupa sia di riordinare i suoi elementi in base all'età specificata sia di stampare l'array riordinato.

Va sottolineato che il vettore atto a contenere gli oggetti della classe Umano, in realtà, è pensato per contenere oggetti di qualunque classe della gerarchia EssereVivente dato che la segnatura del vettore è la seguente: `vector<EssereVivente*>`.

CLASSE ANIMALE

La classe Animale consta solamente della ridefinizione del metodo `stampa()` e dell'override del metodo `descrizione()`.

CLASSE CANE

Nella classe Cane, al pari della classe Animale, sono presenti solamente la ridefinizione del metodo `stampa()` e l'override del metodo `descrizione()`. Ciò che è davvero interessante per gli oggetti della classe Cane è che questi, all'interno del main, vengono aggiunti ad un vettore che vuole rappresentare una lista di cani partecipanti ad una fantomatica gara di bellezza.

La logica con cui ho pensato di implementare la gara prevede che gli oggetti di `vector<Cane*>` vengano aggiunti ad una `map` che vuole simulare una sorta di classifica tra i cani. L'uso della map si rende vantaggioso perché permette di ottenere un ordinamento dei suoi elementi sulla base del valore della chiave contestuale alla loro aggiunta nella map stessa. Nella fattispecie, il valore della chiave è un intero che viene elaborato randomicamente e rappresenta un punteggio compreso tra 1 e 100 che viene assegnato ad ogni cane dalla giuria.

Ottenuta la classifica, il problema da risolvere è diventato la stampa dei cani sul podio. A tal fine ho creato un vettore di appoggio contenente gli stessi elementi della map. L'uso del vector di appoggio ha reso più facile l'accesso ai campi stringhe per permettere una più agevole stampa. Per mezzo di una iterazione sono riuscito ad ottenere le stampe volute (1^ posto, 2^ posto, 3^ posto).

CLASSE GATTO

Nella classe Gatto, oltre a fornire sia la ridefinizione del metodo `stampa()` sia l'override del metodo `descrizione()` vi è la definizione del metodo `giocaCon()`. La logica del metodo è semplice:

- A seguito della creazione di due oggetti della classe Gatto, uno di questi chiama il metodo passando l'altro come parametro;
- Il metodo ha il compito di estrarre un numero casuale compreso tra 0 e 255 effettuandone contestualmente un cast per mezzo dell'operatore `char`:
  - se il numero estratto è compreso tra 0 e 127 il cast non produce alcun effetto e il numero estratto positivo continua ad essere positivo;
  - se il numero estratto è compreso tra 128 e 255 il cast ha l'effetto di calcolarne il complemento a 2, quindi il numero estratto positivo diventa considerato come negativo.
- Se il numero estratto è positivo il metodo stampa una stringa che afferma che i due gatti stanno giocando insieme e si stanno divertendo, mentre se è negativo ne stampa una che afferma che i due gatti non si stanno divertendo.

GIOCO DEL FORZA4

Due oggetti scelti in modo randomico possono partecipare ad una gara di Forza4. L'estrazione dei partecipanti viene effettuata all'interno del `main()`. I due giocatori scelti vengono assegnati alle due fazioni tipiche del gioco: il giocatore Rosso e il giocatore Verde.

A turno, ogni giocatore deve decidere in quale colonna della griglia far cadere un gettone dello stesso colore di quello della fazione. Obiettivo del gioco è quello di posizionare consecutivamente quattro gettoni dello stesso colore. Il giocatore che riesce a perseguire l'obiettivo vince. Le possibilità di vincita sono tre:

1. Quattro gettoni in orizzontale;
2. Quattro gettoni in verticale;
3. Quattro gettoni posizionati in modo obliquo.

Per realizzare il gioco è stato necessario creare:

- La classe Gettone;
- Le sottoclassi GettoreRosso e GettoneVerde di Gettone;
- La classe Griglia per la rappresentazione della griglia di gioco;
- La classe Forza4 per la gestione delle dinamiche di gioco.

ESEMPI DI OUTPUT

Output del metodo `descrizione()` eseguito da oggetti delle classi Umano, Cane e Gatto:

```
Fabio è un oggetto della classe Umano
Maya è un oggetto della classe Cane
Pasqualino è un oggetto della classe Gatto
```

Output del metodo `stampa()` eseguito da oggetti delle classi Umano, Cane e Gatto:

```
Nome: Marcelo
Età: 5 anni
Sesso: M
Cognome: Rossi
Codice fiscale: Codice2Fiscale3

Nome: Merlin
Età: 12 anni
Sesso: M
Pedigree: True
oreGioco: 4

Nome: Ares
Età: 10 anni
Sesso: M
Pedigree: True
Dedica: Distruggi tutti i divani
```

Output prodotto dall'approccio Facade per il riordinamento e la successiva stampa di un vettore:

```
Nome: Marcelo -> età: 5
Nome: Antonia -> età: 20
Nome: Fabio -> età: 25
```

Output tipico prodotto dall'esecuzione del metodo `iniziaGara()`:

```
Classifica generale:
1^ candidato: Kira      Punteggio: 63
2^ candidato: Elvis     Punteggio: 65
3^ candidato: Merlin    Punteggio: 68
4^ candidato: Loki      Punteggio: 73
5^ candidato: Maya      Punteggio: 90

Podio:
1^ classificato: Maya   Punteggio: 90
2^ classificato: Loki   Punteggio: 73
3^ classificato: Merlin Punteggio: 68
```

Output tipici prodotti dal metodo giocaCon():

```
Pasqualino e Ares si stanno divertendo!
```

```
Pasqualino e Ares non si stanno divertendo!
```

Punto di partenza per giocare una partita di Forza4:

```
Estrazione dei partecipanti per giocare a Forza 4

Giocatori estratti: Marcelo e Antonia

Marcelo sarà il giocatore Rosso
Antonia sarà il giocatore Verde

        ____________________
Riga: 8|                    |
Riga: 7|                    |
Riga: 6|                    |
Riga: 5|                    |
Riga: 4|                    |
Riga: 3|                    |
Riga: 2|                    |
Riga: 1|                    |
Colonne: 1 2 3 4 5 6 7 8 9 10
```

Combinazioni per vincere una partita di Forza4:

```
        ____________________
Riga: 8|                    |
Riga: 7|                    |
Riga: 6|                    |
Riga: 5|                    |
Riga: 4|              R     |
Riga: 3|              R     |
Riga: 2|              R     |
Riga: 1| V V V        R     |
Colonne: 1 2 3 4 5 6 7 8 9 10

Ha vinto il giocatore Rosso!!!
```

```
        ____________________
Riga: 8|                    |
Riga: 7|                    |
Riga: 6|                    |
Riga: 5|                    |
Riga: 4|                    |
Riga: 3|                    |
Riga: 2| R R                |
Riga: 1| R R      V V V V   |
Colonne: 1 2 3 4 5 6 7 8 9 10

Ha vinto il giocatore Verde!!!
```

```
        ____________________
Riga: 8|                    |
Riga: 7|                    |
Riga: 6|                    |
Riga: 5|                    |
Riga: 4|          R         |
Riga: 3|        R V         |
Riga: 2|      R R R         |
Riga: 1|   V R R V V     V V |
Colonne: 1 2 3 4 5 6 7 8 9 10

Ha vinto il giocatore Rosso!!!
```