Studente.h

mercoledì 28 giugno 2023 21:58

```
#include <iostream>
#include <string>
using namespace std;
#ifndef STUDENTE_H
#define STUDENTE_H
class Studente{
    private:
        int matricola;
        string nome;
        string cognome;
        float media;

    public:
        Studente(int _matricola, string _nome, string _cognome, float
_media) :
            matricola(_matricola), nome(_nome), cognome(_cognome),
media(_media){};

        virtual ~Studente(){};
        int getMatricola() const{ return this->matricola;}
        string getNome() const{return this->nome;}
        string getCognome() const{ return this->cognome;}
        float getMedia() const{ return this->media;}
        friend ostream& operator <<(ostream& os, const Studente &s);
        // virtual void stampa(ostream &os) const;
        virtual void stampa() const;
};
void Studente::stampa() const{
    cout << "Matricola: " << getMatricola() << ",\tnome: " << getNome()
    << ",\t cognome: " << getCognome() << ", \tmedia: " << getMedia() << endl;

}
class BorsaDiStudio{
    private:
        float importo;
        float durata;
    public:
        BorsaDiStudio(float _importo, float _durata) : importo(_importo),
durata(_durata){};
        ~BorsaDiStudio(){};
        float getImporto() const{ return this->importo;}
        float getDurata() const{return this->durata;}
};
class StudenteBorsista : public Studente{
    private:
        BorsaDiStudio* borsa; //indica la borsa di studio assegnata allo
studente
    public:
        StudenteBorsista(int _matricola, string _nome, string _cognome, float
_media, BorsaDiStudio* b) :
            Studente(_matricola, _nome, _cognome, _media),  borsa(b){};
        ~StudenteBorsista(){};
        float get_importo_borsa() const{ return borsa->getImporto(); }
        // friend ostream &operator <<(ostream &os, const StudenteBorsista&
s);
```

```
        // void stampa(ostream &os) const;
        void stampa() const;
};
void StudenteBorsista::stampa() const{
    cout << "Matricola: " << getMatricola() << ",\tnome: " << getNome()
    << ",\t cognome: " << getCognome() << ",\tmedia: " << getMedia() << ",
\tborsa di: "
    << get_importo_borsa() << " euro" << endl;
}


#endif
```

Coda.h

mercoledì 28 giugno 2023 21:59

```
#include <iostream>
#include "Studente.h"
using namespace std;
#ifndef CODA_H
#define CODA_H
class Nodo{
    public:
        Studente* dato;
        Nodo* succ;
    Nodo(Studente* el) : dato(el), succ(nullptr){};
};
class Coda{
    private:
        Nodo* testa;
        Nodo* fine;
    public:
        Coda() : testa(nullptr), fine(nullptr){};
        ~Coda();
        Nodo* getTesta() const{ return this->testa;}
        bool isEmpty() const;
        void enqueue(Studente* el); //inserimento in coda
        Studente* dequeue(); //rimozione
        Studente* peek() const;
        void stampaCoda() const;
};
void Coda::stampaCoda() const{
    Nodo* iter = testa;
    while(iter != nullptr){
        iter->dato->stampa();
        iter = iter->succ;
    }
}

bool Coda::isEmpty() const{
    if(fine == nullptr){
       return true;
    }
    else{
        return false;
    }
}
Coda::~Coda(){
    while(!isEmpty()){
        dequeue();
    }
}
void Coda::enqueue(Studente* el){
    Nodo* nuovo = new Nodo(el);
    nuovo->succ = nullptr;
    if(isEmpty()){
        fine = testa = nuovo;
    }
    fine->succ = nuovo;
    fine = nuovo;
}
```

```
Studente* Coda::dequeue(){
    if(isEmpty()){
        throw runtime_error("coda vuota, impossibile rimuovere elementi");
    }
    Nodo* iter;
    Studente* tmp = testa->dato;
    iter = testa;
    testa = testa->succ;
    if(testa == nullptr){
        fine = nullptr;
    }
    delete iter;
    return tmp;
}
Studente* Coda::peek() const{
    if(isEmpty()){
        throw runtime_error("la coda e' vuota!");
    }
    return testa->dato;
}

#endif
```

main.cpp

mercoledì 28 giugno 2023 21:59

```
#include <iostream>
#include <string>
#include "Studente.h"
#include "Coda.h"
using namespace std;

int main(){
    BorsaDiStudio *b = new BorsaDiStudio(880, 2);
    BorsaDiStudio *b2 = new BorsaDiStudio(500, 2);
    BorsaDiStudio *b3 = new BorsaDiStudio(600, 2);
    Studente *s1 = new Studente (1001, "Mario", "Rossi", 25.5);
    Studente *s2 = new StudenteBorsista(1002, "Anna", "Verdi", 28.0,  b);
    Studente *s3 = new Studente (1003, "Luca", "Bianchi", 26.0);
    Studente *s4 = new Studente (1004, "Mario", "Rossi", 27.5);
    Studente *s5 = new StudenteBorsista (1005, "Marco", "Gialli", 24.0, b2);
    Studente *s6 = new StudenteBorsista (1006, "Laura", "Marroni", 29.0, b3);

    s1->stampa();
    s2->stampa();
    s3->stampa();
    s4->stampa();
    s5->stampa();
    s6->stampa();

    Coda tail;
    tail.enqueue(s1);
    tail.enqueue(s2);
    tail.enqueue(s3);
    tail.enqueue(s4);
    tail.enqueue(s5);
    tail.enqueue(s6);
    cout << "\nSTUDENTI TOTALI:" << endl;
    tail.stampaCoda();


    Coda tail2;

    Studente* tmp;

    while(!tail.isEmpty()){
        tmp = tail.dequeue();
        if(tmp->getMedia() >= 25){
            tail2.enqueue(tmp);
        }
        else{
            cout << "\nlo studente seguente e' stato rimosso: " << endl;
            tmp->stampa();
        }
    }
    cout << "\n\nSTUDENTI CON MEDIA >= 25: " << endl;
    tail2.stampaCoda();
    float totaleBorseDiStudio = 0;
        for (Nodo* iter = tail2.getTesta(); iter != nullptr; iter = iter->
succ) {
            StudenteBorsista* sb = dynamic_cast<StudenteBorsista*>(iter->
```

```
dato);
            if (sb != nullptr) {
                totaleBorseDiStudio += sb->get_importo_borsa();
        }
            }
    cout << "Totale degli importi delle borse di studio: "
<< totaleBorseDiStudio << " euro" << endl;

    delete s1;
    delete s2;
    delete s3;
    delete s4;
    delete s5;
    delete s6;
    return 0;
}
```

Testo

mercoledì 28 giugno 2023 22:00

PROVA D'ESAME 2

Si consideri una classe Studente che rappresenta uno studente universitario. La classe ha i seguenti attributi privati:

- *matricola, nome, cognome, media*

La classe ha i seguenti metodi pubblici:

- *costruttore, get degli attributi, stampa*

Si consideri una classe BorsaDiStudio che rappresenta una borsa di studio assegnata a uno studente. La classe ha i seguenti attributi privati:

- *importo, durata*

Si consideri una classe StudenteBorsista che deriva dalla classe Studente e ha un attributo privato aggiuntivo:

- *borsa: un puntatore a BorsaDiStudio che indica la borsa di studio assegnata allo studente*

La classe ha i seguenti metodi pubblici:

- *get_importo_borsa: restituisce l'importo della borsa di studio assegnata allo studente*
- *stampa: stampa a video le informazioni dello studente nel formato "matricola: nome cognome - media [borsa di importo euro]", se è titolare di borsa, altrimenti stampa le informazioni dello studente senza la borsa*

Si scriva un programma che crea una coda di studenti di dimensione 10 e inserisce nella coda i seguenti studenti:

- *matricola 1001, nome "Mario", cognome "Rossi", media 25.5*
- *matricola 1002, nome "Anna", cognome "Verdi", media 28.0, borsa di 880 euro*
- *matricola 1003, nome "Luca", cognome "Bianchi", media 26.0*
- *matricola 1004, nome "Sara", cognome "Neri", media 27.5*
- *matricola 1005, nome "Marco", cognome "Gialli", media 24.0, borsa di 500 euro*
- *matricola 1006, nome "Laura", cognome "Marroni", media 29.0, borsa di 600 euro*

Il programma poi controlla se ci sono studenti nella coda che hanno una media inferiore a 25. In tal caso, il programma deve rimuovere dalla coda gli studenti insufficienti e stampare un messaggio di avviso con il nome e la matricola degli studenti rimossi.

Il programma poi stampa il contenuto della coda e calcola il totale degli importi delle borse di studio assegnate agli studenti nella coda usando i metodi get_importo_borsa degli studenti borsisti.

Output atteso:

Contenuto della coda:

1001: Mario Rossi - 25.5

1002: Anna Verdi - 28.0 [borsa di 880 euro]

1003: Luca Bianchi - 26.0

1004: Sara Neri - 27.5

1006: Laura Marroni - 29.0 [borsa di 600 euro]

Totale degli importi delle borse di studio: 1480 euro