#include<iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. SVE KLASE TREBAJU POSJEDOVATI ADEKVATAN DESTRUKTOR

2. NAMJERNO IZOSTAVLJANJE KOMPLETNIH I/ILI POJEDINIH DIJELOVA DESTRUKTORA KOJI UZROKUJU RUNTIME ERROR ĆE BITI OZNACENO KAO "RE"

3. SPAŠAVAJTE PROJEKAT KAKO BI SE SPRIJEČILO GUBLJENJE URAĐENOG ZADATKA

4. NAZIVI FUNKCIJA, TE BROJ I TIP PARAMETARA MORAJU BITI IDENTIČNI ONIMA KOJI SU KORIŠTENI U TESTNOM CODE-U, OSIM U SLUČAJU DA POSTOJI ADEKVATAN RAZLOG ZA NJIHOVU MODIFIKACIJU. OSTALE, POMOĆNE FUNKCIJE MOŽETE IMENOVATI I DODAVATI PO ŽELJI.

5. IZUZETAK BACITE U FUNKCIJAMA U KOJIMA JE TO NAZNAČENO.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const char\* nedozvoljena\_operacija = "Nedozvoljena operacija";

const char\* not\_set = "NOT\_SET";

enum Sortiranje { ASC, DESC };

enum Predmet { MATEMATIKA, FIZIKA, HEMIJA, GEOGRAFIJA, NOT\_SET };

enum Razred { I = 1, II, III, IV };

char \* AlocirajNizKaraktera(const char \* sadrzaj) {

if (sadrzaj == nullptr)

return nullptr;

int vel = strlen(sadrzaj) + 1;

char \* temp = new char[vel];

strcpy\_s(temp, vel, sadrzaj);

return temp;

}

template <class T1, class T2, int max>

class Kolekcija {

T1 \* \_elementi1[max] = { nullptr };

T2 \* \_elementi2[max] = { nullptr };

int \_trenutnoElemenata;

bool \_dozvoliDupliranje;

Sortiranje \_sortiranje;

public:

Kolekcija(Sortiranje sortiranje = ASC, bool dozvoliDupliranje = true) {

\_trenutnoElemenata = 0;

\_sortiranje = sortiranje;

\_dozvoliDupliranje = dozvoliDupliranje;

}

~Kolekcija() {

for (size\_t i = 0; i < \_trenutnoElemenata; i++) {

delete \_elementi1[i]; \_elementi1[i] = nullptr;

delete \_elementi2[i]; \_elementi2[i] = nullptr;

}

}

int GetTrenutno() const { return \_trenutnoElemenata; }

int GetMax() const { return max; }

T1& GetElement1(int lokacija) const {

if (lokacija < 0 || lokacija >= \_trenutnoElemenata)

throw exception(nedozvoljena\_operacija);

return \*\_elementi1[lokacija];

}

T2& GetElement2(int lokacija) const {

if (lokacija < 0 || lokacija >= \_trenutnoElemenata)

throw exception(nedozvoljena\_operacija);

return \*\_elementi2[lokacija];

}

void SetElement2(int lokacija, T2 ele2) {

\_elementi2[lokacija] = ele2;

}

friend ostream &operator<<(ostream &COUT, const Kolekcija &obj) {

for (size\_t i = 0; i < obj.GetTrenutno(); i++)

COUT << obj.GetElement1(i) << " " << obj.GetElement2(i) << endl;

return COUT;

}

};

class Aktivnost {

shared\_ptr<Razred> \_razred;

string \_opis;

int \_ocjena;//1-5

public:

Aktivnost(Razred razred = I, int ocjena = 0, string opis = not\_set) {

\_ocjena = ocjena;

\_opis = opis;

\_razred = make\_shared<Razred>(razred);

}

int GetOcjenu() const { return \_ocjena; }

string GetOpis() const { return \_opis; }

Razred GetRazred() const { return \*\_razred; }

friend ostream &operator<<(ostream &COUT, const Aktivnost &obj) {

COUT << \*obj.\_razred << " " << obj.\_ocjena << " " << obj.\_opis << endl;

return COUT;

}

};

class Polaznik {

protected:

char \* \_imePrezime;

string \_brojTelefona;

public:

Polaznik(string imePrezime, string brojTelefona) : \_imePrezime(AlocirajNizKaraktera(imePrezime.c\_str())) {

\_brojTelefona = brojTelefona;

}

~Polaznik() { delete[] \_imePrezime; }

char \* GetImePrezime() { return \_imePrezime; }

string GetTelefon() { return \_imePrezime; }

virtual void PredstaviSe() = 0;

};

class Ucenik {

Kolekcija<Predmet, Aktivnost, 16> \* \_aktivnosti;

public:

~Ucenik() { delete \_aktivnosti; \_aktivnosti = nullptr; }

Kolekcija<Predmet, Aktivnost, 16> & GetAktivnosti() { return \*\_aktivnosti; };

friend ostream& operator<<(ostream& COUT, Ucenik & n)

{

COUT << "PODACI O UCENIKU I NJEGOVIM AKTIVNOSTIMA....IMPLEMENTIRATI" << endl;

return COUT;

}

};

class Skola {

char\* \_naziv;

vector<Ucenik> \_ucenici;

public:

Skola(const char \* naziv = nullptr) {

\_naziv = AlocirajNizKaraktera(naziv);

}

~Skola() {

delete[] \_naziv; \_naziv = nullptr;

}

char\* GetNaziv()const { return \_naziv; }

vector<Ucenik> & GetUcenici() { return \_ucenici; };

friend ostream& operator<<(ostream& COUT, Skola& obj) {

COUT << "Skola: " << obj.\_naziv << endl;

COUT << "Ucenici: " << endl;

for (size\_t i = 0; i < obj.\_ucenici.size(); i++)

COUT << obj.\_ucenici[i] << endl;

return COUT;

}

};

int main() {

Kolekcija<int, int, 10> kolekcija1(DESC, false);

try {

kolekcija1.AddElement(1, 2);

//dupliranje elemenata nije dozvoljeno

kolekcija1.AddElement(1, 2);

}

catch (exception & ex) {

cout << ex.what();

}

/\*nakon svakog dodavanja, elemente sortirati prema T1 i vrijednosti atributa \_sortiranje\*/

for (size\_t i = 1; i < kolekcija1.GetMax() - 1; i++)

kolekcija1.AddElement(rand(), rand());

cout << kolekcija1 << endl;

try {

//prekoracen maksimalan broj elemenata

kolekcija1.AddElement(rand(), rand());

}

catch (exception & ex) {

cout << ex.what();

}

cout << kolekcija1 << endl;

Kolekcija<int, int, 10> kolekcija2(kolekcija1);

cout << kolekcija2 << endl;

Skola gimnazijaMostar("GIMNAZIJA MOSTAR");

//dodaje novog ucenika u skolu

gimnazijaMostar("Jasmin Azemovic", "+387(61)111-222");

gimnazijaMostar("Adel Handzic", "+387(61)333-444");

/\*

koristeci regex, osigurati sljedeci format za broj telefona: +387(6X)XXX-XXX ili +387 6X XXX-XXX

onemoguciti pojavljivanje samo jedne zagrade, a ukoliko format nije adekvatna koristiti vrijednost not\_set

\*/

gimnazijaMostar("Telefon NotValidFormat", "387 61)333-444");

try

{

/\*onemoguciti dodavanje ucenika sa istim imenom i prezimenom ili brojem telefona\*/

gimnazijaMostar("Adel Handzic", "+387(61)333-444");

}

catch (exception & ex)

{

cout << ex.what() << endl;

}

if (gimnazijaMostar.DodajAktivnost("Jasmin Azemovic", MATEMATIKA, Aktivnost(I, 4, "Priprema za takmicenje iz Matematije koje se odrzava u Konjicu 07.02.2019")))

cout << "Aktivnost uspjesno dodana" << endl;

/\*na nivou svakog razreda se mogu evidentirati maksimalno 4 aktivnosti, a takodjer, na nivou razreda se ne smiju ponavljati aktivnosti iz istog predmeta\*/

if (!gimnazijaMostar.DodajAktivnost("Jasmin Azemovic", MATEMATIKA, Aktivnost(I, 4, "Aktivnosti iz matematike")))

cout << "Aktivnost nije uspjesno dodana" << endl;

if (gimnazijaMostar.DodajAktivnost("Jasmin Azemovic", HEMIJA, Aktivnost(I, 5, "Priprema otopina za vjezbe iz predmeta Hemija")))

cout << "Aktivnost uspjesno dodana" << endl;

if (gimnazijaMostar.DodajAktivnost("Jasmin Azemovic", FIZIKA, Aktivnost(I, 4, "Analiza stepena apsorpcije materijala ")))

cout << "Aktivnost uspjesno dodana" << endl;

/\*u slucaju da je ucenik uspjesno (ocjenom vecom od 1) realizovao aktivnosti na nivou odredjenog razreda, te posjeduje validan broj telefona,

u okviru zasebnog thread-a se salje SMS sa sadrzajem "Uspjesno ste okoncali aktivnosti u okviru X razreda sa prosjecnom ocjenom X.X"\*/

if (gimnazijaMostar.DodajAktivnost("Jasmin Azemovic", GEOGRAFIJA, Aktivnost(I, 4, "Izrada 5 reljefa Mostara")))

cout << "Aktivnost uspjesno dodana" << endl;

if (gimnazijaMostar.DodajAktivnost("Adel Handzic", MATEMATIKA, Aktivnost(I, 5, "Izrada skripte na temu integralni racun")))

cout << "Aktivnost uspjesno dodana" << endl;

//ispisuje sve ucenike i njihove aktivnosti

cout << gimnazijaMostar << endl;

pair<Polaznik \*, float> par = gimnazijaMostar.GetNajboljegUcenika();

cout << "Najbolji ucenik je " << par.first->GetImePrezime() << " sa prosjekom " << par.second << endl;

/\*U fajl (npr. Gimnazija.txt) upisati podatke (podatke upisati kao obicni tekst) o skoli i svim ucenicima.

Nakon upisa, potrebno je ispisati sadrzaj fajla. Parametar tipa bool oznacava da li ce ranije dodani sadrzaj fajla prethodno biti pobrisan\*/

if (gimnazijaMostar.SpasiUFajl("Gimnazija.txt"))

cout << "Podaci o ucenicima uspjesno pohranjeni u fajl" << endl;

if (gimnazijaMostar.SpasiUFajl("Gimnazija.txt", true))

cout << "Podaci o ucenicima uspjesno pohranjeni u fajl" << endl;

cin.get();

system("pause>0");

return 0;

}