const char\* PORUKA = "\n-------------------------------------------------------------------------------\n"

"1. SVE KLASE TREBAJU POSJEDOVATI ADEKVATAN DESTRUKTOR\n"

"2. NAMJERNO IZOSTAVLJANJE KOMPLETNIH I/ILI POJEDINIH DIJELOVA DESTRUKTORA CE BITI OZNACENO KAO RE\n"

"3. SPASAVAJTE PROJEKAT KAKO BI SE SPRIJECILO GUBLJENJE URADJENOG ZADATKA\n"

"4. NAZIVI FUNKCIJA, TE BROJ I TIP PARAMETARA MORAJU BITI IDENTICNI ONIMA KOJI SU KORISTENI U TESTNOM CODE-U,\n"

"\tOSIM U SLUCAJU DA POSTOJI ADEKVATAN RAZLOG ZA NJIHOVU MODIFIKACIJU. OSTALE\n"

"\tPOMOCNE FUNKCIJE MOZETE IMENOVATI I DODAVATI PO ZELJI.\n"

"5. IZUZETAK BACITE SAMO U FUNKCIJAMA U KOJIMA JE TO NAZNACENO.\n"

"6. FUNKCIJE KOJE NE IMPLEMENTIRATE TREBAJU BITI OBRISANE!\n"

"7. RJESENJA ZADATKA POSTAVITE NA FTP SERVER U ODGOVARAJUCI FOLDER!\n"

"-------------------------------------------------------------------------------\n";

const char\* crt = "\n-------------------------------------------\n";

enum eRazred { PRVI = 1, DRUGI, TRECI, CETVRTI };

char\* GetNizKaraktera(const char\* sadrzaj, bool dealociraj = false) {

if (sadrzaj == nullptr)return nullptr;

int vel = strlen(sadrzaj) + 1;

char\* temp = new char[vel];

strcpy\_s(temp, vel, sadrzaj);

if (dealociraj)

delete[]sadrzaj;

return temp;

}

template<class T1, class T2>

class Kolekcija {

T1\* \_elementi1;

T2\* \_elementi2;

int \_trenutno;

bool \_omoguciDupliranje;

public:

Kolekcija(bool omoguciDupliranje = true) {

\_elementi1 = nullptr;

\_elementi2 = nullptr;

\_omoguciDupliranje = omoguciDupliranje;

}

~Kolekcija() {

delete[]\_elementi1; \_elementi1 = nullptr;

delete[]\_elementi2; \_elementi2 = nullptr;

}

T1& getElement1(int lokacija)const { return \_elementi1[lokacija]; }

T2& getElement2(int lokacija)const { return \_elementi2[lokacija]; }

int getTrenutno() { return \_trenutno; }

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

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

COUT << obj.getElement1(i) << " " << obj.getElement2(i) << endl;

return COUT;

}

};

class Datum {

int\* \_dan, \* \_mjesec, \* \_godina;

public:

Datum(int dan = 1, int mjesec = 1, int godina = 2000) {

\_dan = new int(dan);

\_mjesec = new int(mjesec);

\_godina = new int(godina);

}

~Datum() {

delete \_dan; \_dan = nullptr;

delete \_mjesec; \_mjesec = nullptr;

delete \_godina; \_godina = nullptr;

}

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

COUT << \*obj.\_dan << "." << \*obj.\_mjesec << "." << \*obj.\_godina;

return COUT;

}

};

class Predmet {

char\* \_naziv;

//int se odnosi na ocjenu u opsegu od 1 – 5, a datum na momenat postizanja ocjene

Kolekcija<int, Datum> \_ocjene;

public:

Predmet(const char\* naziv = "", int ocjena = 0, Datum datum = Datum()) {

\_naziv = GetNizKaraktera(naziv);

if (ocjena > 0)

\_ocjene.AddElement(ocjena, datum);

}

~Predmet() {

delete[] \_naziv; \_naziv = nullptr;

}

void AddOcjena(int ocjena, Datum datum) {

\_ocjene.AddElement(ocjena, datum);

}

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

Kolekcija<int, Datum>& GetOcjene() { return \_ocjene; }

};

class Uspjeh {

eRazred \_razred;

//string se odnosi na napomenu o polozenom predmetu

Kolekcija<Predmet, string>\* \_polozeniPredmeti;

public:

Uspjeh(eRazred razred = PRVI) {

\_razred = razred;

}

~Uspjeh() { delete \_polozeniPredmeti; \_polozeniPredmeti = nullptr; }

Kolekcija<Predmet, string>\* GetPredmeti() { return \_polozeniPredmeti; }

eRazred GetERazred() { return \_razred; }

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

COUT << obj.\_razred << " " << \*obj.\_polozeniPredmeti << endl;

return COUT;

}

};

class Kandidat {

char\* \_imePrezime;

string \_emailAdresa;

string \_brojTelefona;

vector<Uspjeh> \_uspjeh;

public:

Kandidat(const char\* imePrezime, string emailAdresa, string brojTelefona) {

\_imePrezime = GetNizKaraktera(imePrezime);

\_emailAdresa = emailAdresa;

\_brojTelefona = brojTelefona;

}

~Kandidat() {

delete[] \_imePrezime; \_imePrezime = nullptr;

}

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

COUT << obj.\_imePrezime << " " << obj.\_emailAdresa << " " << obj.\_brojTelefona << endl;

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

COUT << obj.\_uspjeh[i];

return COUT;

}

vector<Uspjeh>& GetUspjeh() { return \_uspjeh; }

string GetEmail() { return \_emailAdresa; }

string GetBrojTelefona() { return \_brojTelefona; }

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

};

const char\* GetOdgovorNaPrvoPitanje() {

cout << "Pitanje -> Na sta se odnosi pojam reprezentacije tipa?\n";

return "Odgovor -> OVDJE UNESITE VAS ODGOVOR";

}

const char\* GetOdgovorNaDrugoPitanje() {

cout << "Pitanje -> Na koji nacin se moze izbjeci pojavljivanje vise podobjekata bazne klase u slucaju visestrukog nasljedjivanja?\n";

return "Odgovor -> OVDJE UNESITE VAS ODGOVOR";

}

void main() {

cout << PORUKA;

cin.get();

cout << GetOdgovorNaPrvoPitanje() << endl;

cin.get();

cout << GetOdgovorNaDrugoPitanje() << endl;

cin.get();

Datum temp,

datum19062019(19, 6, 2019),

datum20062019(20, 6, 2019),

datum30062019(30, 6, 2019),

datum05072019(5, 7, 2019);

int kolekcijaTestSize = 9;

Kolekcija<int, int> kolekcija1(false);

for (int i = 0; i <= kolekcijaTestSize; i++)

kolekcija1.AddElement(i, i);

try {

//ukoliko nije dozvoljeno dupliranje elemenata (provjeravaju se T1 i T2), metoda AddElement baca izuzetak

kolekcija1.AddElement(3, 3);

}

catch (exception & err) {

cout << err.what() << crt;

}

cout << kolekcija1 << crt;

/\*objekat kolekcija2 ce biti inicijalizovan elementima koji se u objektu kolekcija1 nalaze na lokacijama 1 - 4

ukljucujuci i te lokacije. u konkretnom primjeru to ce biti parovi sa vrijednostima: 1 1 2 2 3 3 4 4\*/

Kolekcija<int, int> kolekcija2 = kolekcija1(1, 4);

cout << kolekcija2 << crt;

try {

//primjeri u kojima opseg nije validan, te bi funkcija trebala baciti izuzetak

Kolekcija<int, int> temp1 = kolekcija1(1, 14);//imamo 10 elemenata

Kolekcija<int, int> temp2 = kolekcija1(-1, 8);//lokacija -1 ne postoji

}

catch (exception & err) {

cout << err.what() << crt;

}

//parametri: nazivPredmeta, prva ocjena, datum

Predmet Matematika("Matematika", 5, datum19062019),

Fizika("Fizika", 5, datum20062019),

Hemija("Hemija", 2, datum30062019),

Engleski("Engleski", 5, datum05072019);

Matematika.AddOcjena(3, datum05072019);

Matematika.AddOcjena(5, datum05072019);

// ispisuje: naziv predmeta, ocjene (zajedno sa datumom polaganja) i prosjecnu ocjenu na predmetu

// ukoliko predmet nema niti jednu ocjenu prosjecna treba biti 0

cout << Matematika << endl;

if (ValidirajEmail("text.text@edu.fit.ba"))

cout << "Email validan" << crt;

if (ValidirajEmail("texttext@edu.fit.ba"))

cout << "Email validan" << crt;

if (ValidirajEmail("texttext@fit.ba"))

cout << "Email validan" << crt;

if (ValidirajEmail("texttext@fit.com"))

cout << "Email validan" << crt;

if (ValidirajEmail("texttext@edu.fit.org"))

cout << "Email validan" << crt;

/\*

email adresa mora biti u formatu text.text (sa ili bez tacke),

dok domena moze biti fit.ba ili edu.fit.ba

nakon institucije (fit ili edu.fit), osim drzavne (.ba), dozvoljene su domene .com i .org.

za provjeru validnosti email adrese koristiti globalnu funkciju ValidirajEmail, a unutar nje regex metode.

validacija email adrese ce se vrsiti unutar konstruktora klase Kandidat, a u slucaju da nije validna

postaviti je na defaultnu adresu: notSet@edu.fit.ba

\*/

Kandidat jasmin("Jasmin Azemovic", "jasmin@fit.ba", "033 281 172");

Kandidat adel("Adel Handzic", "adel@edu.fit.ba", "033 281 170");

Kandidat emailNotValid("Ime Prezime", "korisnik@klix.ba", "033 281 170");

/\*

uspjeh (tokom srednjoskolskog obrazovanja) se dodaje za svaki predmet na nivou razreda.

tom prilikom onemoguciti:

- dodavanje istih (moraju biti identicne vrijednosti svih clanova) predmeta na nivou jednog razreda,

- dodavanje predmeta kod kojih je prosjecna ocjena manja od 2.5

- dodavanje vise od 5 predmeta na nivou jednog razreda

razredi (predmeti ili uspjeh) ne moraju biti dodavani sortiranim redoslijedom (npr. prvo se moze dodati uspjeh za II razred, pa onda za I razred i sl.).

Funkcija vraca true ili false u zavisnosti od (ne)uspjesnost izvrsenja

\*/

if (jasmin.AddPredmet(DRUGI, Fizika, "Napomena 1"))

cout << "Predmet uspjesno dodan!" << crt;

if (jasmin.AddPredmet(DRUGI, Hemija, "Napomena 2"))

cout << "Predmet uspjesno dodan!" << crt;

if (jasmin.AddPredmet(PRVI, Engleski, "Napomena 3"))

cout << "Predmet uspjesno dodan!" << crt;

if (jasmin.AddPredmet(PRVI, Matematika, "Napomena 4"))

cout << "Predmet uspjesno dodan!" << crt;

//ne treba dodati Matematiku jer je vec dodana u prvom razredu

if (jasmin.AddPredmet(PRVI, Matematika, "Napomena 5"))

cout << "Predmet uspjesno dodan!" << crt;

/\*nakon evidentiranja uspjeha na bilo kojem predmetu kandidatu se salje email sa porukom:

FROM:info@fit.ba

TO: emailKorisnika

Postovani ime i prezime, evidentirali ste uspjeh za X razred. Dosadasnji uspjeh (prosjek)

na nivou X razreda iznosi Y, a ukupni uspjeh u toku skolovanja iznosi Z.

Pozdrav.

FIT Team.

ukoliko je prosjek na nivou tog razreda veci od 4.5 kandidatu se salje SMS sa porukom: "Svaka cast za uspjeh 4.X u X razredu".

slanje poruka i emailova implemenitrati koristeci zasebne thread-ove.

\*/

cout << jasmin << crt;

//vraca kolekciju predmeta koji sadrze najmanje jednu ocjenu evidentiranu u periodu izmedju proslijedjenih datuma

//float se odnosi na prosjecan broj dana izmedju ostvarenih ocjena na predmetu

Kolekcija<Predmet, float> jasminUspjeh = jasmin(Datum(18, 06, 2019), Datum(21, 06, 2019));

cout << jasminUspjeh << crt;

Uspjeh\* uspjeh\_Irazred = jasmin[PRVI];//vraca uspjeh kandidata ostvaren u prvom razredu

if (uspjeh\_Irazred != nullptr)

cout << \*uspjeh\_Irazred << crt;

cin.get();

system("pause>0");

}