1. #include<iostream>

#include "Source.h"

using namespace std;

//U programu implementirati nasljeðivanje izmeðu odgovarajuæih klasa

//Klasu deklarisati kao apstraktnu

class Osoba//proglasiti je apstraktnom

{

protected:

char\* \_ime;

char\* \_prezime;

char\* \_adresa;

char \_telefon[15];

public:

//Potrebne konstruktor i destruktor funkcije

//Kreirati funkciju Info za ispis svih vrijednosti atributa klase

};

static int pom = 1;

class Klijent : public Osoba

{

protected:

int \_id;

double \_mjesecnaPotrosnja;

char\* \_email;

public:

char \* getIme() { return this->\_ime; }

char \* getPrezime() { return this->\_prezime; }

//Potrebne konstruktor i destruktor funkcije, konstruktor kopije

//Kreirati funkciju za izmjenu mail adrese.

void IzmijeniMail(char \*novi) {

}

void setMail(char \* Mail) {

}

char \* GetMail() { }

double getPotrosnja() {}

};

class Zaposlenik : public Osoba {

protected:

char \_radnoMjesto[30];

int \_godineStaza;

public:

//Potrebne konstruktor i destruktor funkcije, konstruktor kopije

char \* GetRM() {

//vraca radno mjesto zaposlenika

}

};

class VIPKlijent :public Klijent

{

double \_popust;

char\* \_parkingOznaka;

public:

//Potrebne konstruktor i destruktor funkcije, kao i konstruktor kopije

void SmanjiPotrosnju() {

//Funkciju koja treba da izmijeni vrijednost mjeseène potrošnje na naèin da se

//uraèuna popust (prosjeèna potrošnja treba da se umanji)

}

};

int pretragaPotrosnjeKlijenata(Osoba\* niz[], int max) {

//Funkcija treba da pronaðe sve one klijente koji su ostvarili mjeseènu potrošnju veću

//od zadate. Ispisati podatke i vratiti ukupan broj pronaðenih klijenata.

}

void pretragaVIPKlijenata(Osoba\* niz[], int max) {

//Pronaći sve VIP klijente, te im zaraèunati popust na kupljene artikle i ispisati

//informacije o onima koji su ostvarili potrošnju veæu od 50 KM mjeseèno.

}

void generisiMailAdrese(Osoba\* niz[], int max) {

//Svim klijentima postaviti mail adresu u formatu: ime.prezime@firma.ba.

}

void pretragaKlijenataPoImenu(Osoba\* niz[], int max) {

//Omogućiti pretragu klijenata po imenu ili prezimenu (ispisati informacije

//o svima koji sadrže uneseni tekst u dijelu imena ili prezimena.

}

void pretragaZaposlenika(Osoba\* niz[], int max) {

//Omogućiti pretragu zaposlenika na osnovu radnog mjesta koje definiše korisnik.

}

void main()

{

//Testirati sve dostupne funkcionalnosti uz obaveznu provjeru globalnih funkcija

system("pause>0");

}

2.

#include<iostream>

using namespace std;

class Artikal {

char \*sifra;

char \*naziv;

double cijena;

int kolicinaNaSkladistu;

public:

//potrebne konstruktor i destruktor funkcije (imati na umu i konstruktor kopije)

//UmanjiKolicinuNaSkladistu na način da se stanje na skladištu artikla promijeni za proslijeđenu vrijednost

//funkcija treba da bude tipa bool, tj vraća informaciju o tome da li je uspješno promjenjeno stanje na skladištu

//Operator UvecajKolicinuNaSkladistu na način da se stanje na skladištu artikla promijeni za proslijeđenu vrijednost

//UvecajCijenu na način da se cijena artikla uveća za 1

//Operator UmanjiCijenu na način da se cijena artikla umanji za 1

//AreEqual koji poredi artikle na osnovu sifre

//Info

//Unos

int GetKolicinaNaSkladistu() { }

double GetCijena() { }

char \*GetSifra() { }

};

class StavkaRacuna {

Artikal artikal;

int \*kolicina;

public:

//potrebne konstruktor i destruktor funkcije (imati na umu i konstruktor kopije)

//UvecajStanjeNaSkladistu koji uvećava stanje na skladištu artikla

//UmanjiStanjeNaSkladistu koji umanjuje stanje na skladištu artikla

//AreEqual koji poredi stavke po sifri artikla

// UvecajKolicinuStavke koji uvećava količinu stavke racuna

//Info

double GetCijena() {//vraća cijenu stavke racuna (kolicina \* cijena artikla)

}

int GetKolicina() { //vraća količinu stavke računa

}

};

int pom = 1000;

class Racun {

int trenutno;

int brojRacuna;

StavkaRacuna stavke[100];

int brojStavki;

double iznosBezPDV;

double iznosSaPDV;

bool naplaceno;

public:

//potrebne konstruktor i destruktor funkcije (imati na umu i konstruktor kopije)

//DodajStavku - dodaje nove artikle u stavke racuna (ulazni parametri su artikal i kolicina), kolicina nije obavezni parametar

//Voditi računa o tome da li je račun već naplaćen

//Ukoliko nema dovoljno artikala na stanju, ispisati odgovarajuću poruku

//Ukoliko artikal već postoji, potrebno je samo uvećati količinu za odgovarajuću stavku

//Prilikom svakog dodavanja artikla, azurirati vrijednosti iznosBezPDV i iznosSaPDV

//UkloniArtikal uklanja artikal iz stavki na osnovu njegove sifre

//azurirati vrijednosti iznosBezPDV i iznosSaPDV

double PrintRacun() {

//Ispisuje sve stavke računa zajedno sa iznosom za plaćanje

//Omogućiti da se unese iznos koji je korisnik uplatio, te se

//vraća iznos kusura

}

};

void main() {

//main je moguće mijenjati i prilagođavati

//ali je ovo napisano sasvim dovoljno za testiranje funkcionalnosti zadatka

Racun racun;

Artikal artikli[3];

for (int i = 0; i < 3; i++)

{

artikli[i].Unos();

racun.DodajStavku(artikli[i], 2);

}

//pokusaj dodavanja istog artikla na racun

racun.DodajStavku(artikli[0], 2);

char sifra[20];

cout << "Unesite sifru artikla kojeg želite ukloniti: ";

cin.getline(sifra, 20);

racun.UkloniArtikal(sifra);

racun.DodajStavku(artikli[1], 1);

cout << "Kusur: " << racun.PrintRacun() << " KM" << endl;

system("pause>0");

}