Универзитет „Св. Кирил и Методиј“ - Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Објектно ориентирано програмирање

Аудиториски вежби 10

Содржина

1. Статички членови и исклучоци

1.1. Задача

Секое плаќање преку картичка има некои подобности. Имено државата сакајќи да го поттикне користењето на картичките, нуди поволни услови за плаќање.

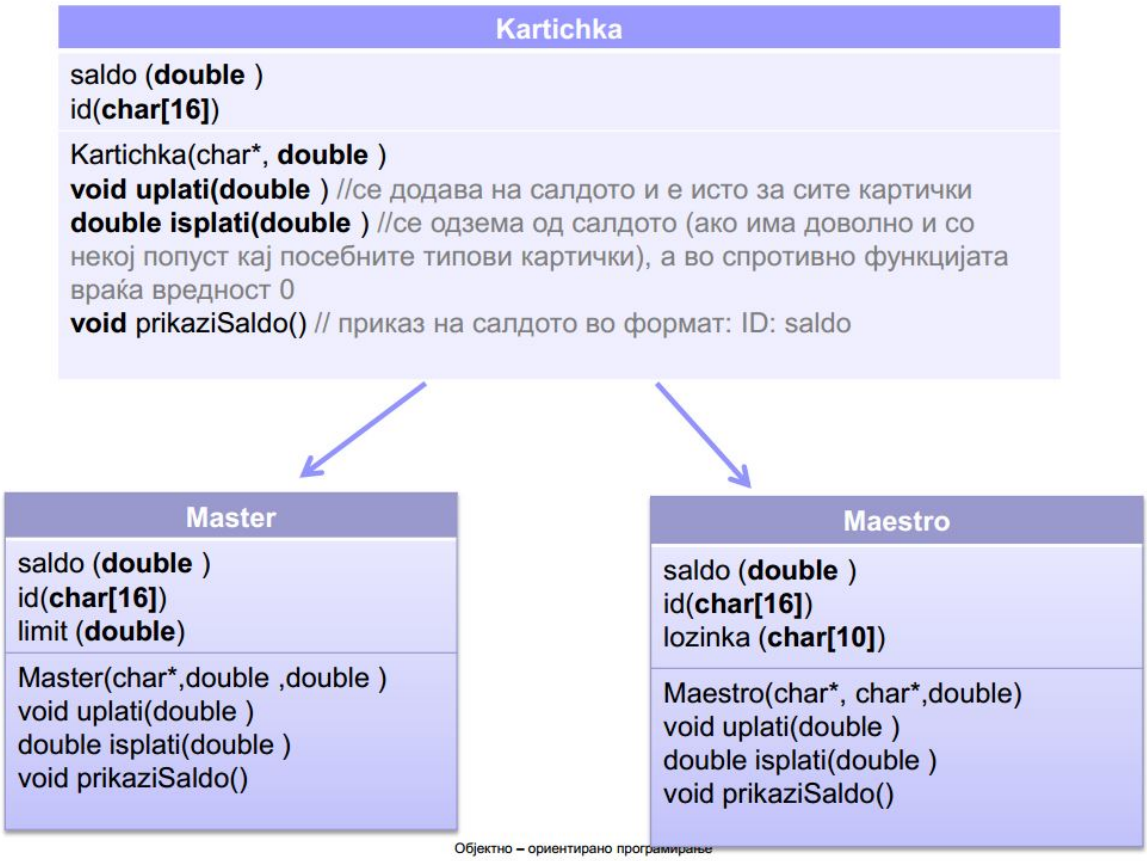
Да се моделира основна класа `Kartichka` како и класи `Master` и `Maestro` кои ја наследуваат. Една картичка е опишана со својот идентификациски број, како и со салдото на сметката која ја претставува.

При плаќање со маестро картичка, секоја сума се плаќа со попуст од 5% за СИТЕ корисници на маестро картичка. Овој процент е фиксен и не смее да се менува!

При плаќање со мастер картичка, ако лимитот на картичката е над 6000 денари тогаш попустот е 10%, наместо стандардните 3% за картички со лимит под 6000 денари.

Попустот од 10% е ист за сите корисници, но тој може да биде променет од страна на Народна Банка.

Во продолжение е даден шематски приказ на изгледот на класите!



Решение `oop_av101.cpp`

```
#include<iostream>
#include<cstring>
using namespace std;

class Kartichka
{
private:
    char id[16];
    double saldo;

public:
    Kartichka(char* id = "", double  saldo = 0)
    {
        this->saldo = saldo;
        strcpy(this->id, id);
    }
    void prikaziSaldo()
    {
        cout << id << ": " << saldo << endl;
    }
    void uplati(double  suma)
    {
        this->saldo += suma;
    }
    virtual double isplati(double suma)
    {
        if (this->saldo > suma){
            this->saldo -= suma;
            return suma;
        }
        else return 0;
    }

protected:
    virtual double isplati(double suma, double limit)
    {
        if (this->saldo + limit > suma){
```

```
            this->saldo -= suma;
            return suma;
        }
        else return 0;
    }
};

class Maestro : public Kartichka{
private:
    char lozinka[10];
    const static double popust; //static clen na klasa

public:
    Maestro(char* lozinka = "", char* id = "", double  saldo = 0) :Kartichka(id, saldo) {
        strcpy(this->lozinka, lozinka);
    }

    static double  getPopust() { //static funkcija koja raboti so static clen
        return popust;
    }

    double isplati(double  cena)
    {
        double  suma = cena*(1 - popust); // non-static funkcii moze da gi koristat
                                          // static podatocnite elementi
        return Kartichka::isplati(suma);
    }
};
const double Maestro::popust = 0.05; // inicijaliziranje na static clen

class Master : public Kartichka{
private:
    double  limit;
    const static double popust1;  // fiksen popust
    static double popust2;       // popust koj sto moze da se promeni

public:
    Master(double  limit = 0, char* id = "", double  saldo = 0) :Kartichka(id, saldo) {
        this->limit = limit;
    }
    static double  getPopust1(){
        return popust1;
    }
    static double  getPopust2() {
        return popust2;
    }
    static void setPopust2(double  popust2) {
        Master::popust2 = popust2;
    }
    /* ne smeeme da go napravime ova:
    static void setPopust1 (double  popust1) {
        Master::popust1 = popust1;
    }
    //poradi toa sto stanuva zbor za konstanta
    */
    double isplati(double  cena)
    {
        if (this->limit<6000){
            double  suma = cena*(1 - popust1);
            return Kartichka::isplati(suma, limit);
        }
        else
        {
            double  suma = cena*(1 - popust2);
            return Kartichka::isplati(suma, limit);
        }
    }
};
const double Master::popust1 = 0.03; //pri inicijalizacija ne se pisuva klucniot zbor
static
double Master::popust2 = 0.1;
```

1.2. Задача

Да се креира класа `Kasa` која што треба да ја претставува касата на една продавница во која муштериите можат да плаќаат во готово или со картичка. За секоја каса (Kasa) се водат две суми за дневното работење. Едната е вредноста на средствата добиени од готовина, а другата е сумата на средствата од картички. Исто така, секој објект од класата се креира во различен ден, па затоа за секој објект се чува и денот, месецот и годината кога касата е отворена.

Во класата `Kasa` има функција `kasaPrimi()` со која ќе се овозможи примање на парични средства на сметката на продавницата. За плаќањето во готовина да се креира една функција со потпис `kasaPrimi(double )` со која на сумата во касата се додава цената на сметката.

За плаќањето со картичка да се направи функција со потпис `kasaPrimi (double , Kartichka )` која како аргумент покрај вредноста на сметката има и референца кон самата картичка (мастер или маестро). Имено со повик на оваа функција потребно е да се ажурираат податоците и во касата и во картичката со која се плаќа некоја сметка.

Во класата да се дефинира и функција `prikaziKasa()` со која ќе се прикажат информациите – заедно со вкупната дневна добивка кои се значајни за една каса.

Решение `oop_av102.cpp`

```
#include<iostream>
#include<cstring>
using namespace std;

class Kartichka
{
private:
    char id[16];
    double saldo;

public:
    Kartichka(char* id = "", double  saldo = 0)
    {
        this->saldo = saldo;
        strcpy(this->id, id);
    }
    void prikaziSaldo()
    {
        cout << id << ": " << saldo << endl;
    }
    void uplati(double  suma)
    {
        this->saldo += suma;
    }
```

```
    virtual double isplati(double suma)
    {
        if (this->saldo > suma){
            this->saldo -= suma;
            return suma;
        }
        else return 0;
    }

protected:
    virtual double isplati(double suma, double limit)
    {
        if (this->saldo + limit > suma){
            this->saldo -= suma;
            return suma;
        }
        else return 0;
    }
};

class Maestro : public Kartichka{
private:
    char lozinka[10];
    const static double popust; //static clen na klasa

public:
    Maestro(char* lozinka = "", char* id = "", double  saldo = 0) :Kartichka(id, saldo) {
        strcpy(this->lozinka, lozinka);
    }

    static double  getPopust() { //static funkcija koja raboti so static clen
        return popust;
    }

    double isplati(double  cena)
    {
        double  suma = cena*(1 - popust); // non-static funkcii moze da gi koristat
                                          // static podatocnite elementi
        return Kartichka::isplati(suma);
    }
};
const double Maestro::popust = 0.05; // inicijaliziranje na static clen

class Master : public Kartichka{
private:
    double  limit;
    const static double popust1;  // fiksen popust
    static double popust2;       // popust koj sto moze da se promeni

public:
    Master(double  limit = 0, char* id = "", double  saldo = 0) :Kartichka(id, saldo) {
        this->limit = limit;
    }
    static double  getPopust1(){
        return popust1;
    }
    static double  getPopust2() {
        return popust2;
    }
    static void setPopust2(double  popust2) {
        Master::popust2 = popust2;
    }
    /* ne smeeme da go napravime ova:
    static void setPopust1 (double  popust1) {
        Master::popust1 = popust1;
    }
    //poradi toa sto stanuva zbor za konstanta
    */
    double isplati(double  cena)
    {
        if (this->limit<6000){
            double  suma = cena*(1 - popust1);
            return Kartichka::isplati(suma, limit);
        }
        else
        {
```

```
            double  suma = cena*(1 - popust2);
            return Kartichka::isplati(suma, limit);
        }
    }
};
const double Master::popust1 = 0.03; //pri inicijalizacija ne se pisuva klucniot zbor
static
double Master::popust2 = 0.1;

class Kasa {
private:
    double  sumaVoKasa;
    double  sumaOdKartichka;
    int den, mesec, godina;

public:

    Kasa(double  sumaVoKasa, int den, int mesec, int godina) {
        this->sumaVoKasa = sumaVoKasa;
        this->sumaOdKartichka = 0;
        this->den = den;
        this->mesec = mesec;
        this->godina = godina;
    }
    void kasaPrimi(double smetka){
        this->sumaVoKasa += smetka;
    }
    void kasaPrimi(double smetka, Kartichka &k){
        this->sumaOdKartichka += k.isplati(smetka);
    }
    void prikaziKasa(){

        cout << "Den: \t" << den << endl;
        cout << "Mesec: \t" << mesec << endl;
        cout << "Godina: " << godina << endl;
        cout << "Prihod-vkupno: " << this->vratiPrihod() << endl;
        cout << endl;
    }

    double  vratiPrihod(){
        return this->sumaOdKartichka + this->sumaVoKasa;
    }
};

int main(){
    Kasa deneshna(10000, 22, 4, 2014);
    Kartichka *k;
    deneshna.prikaziKasa();

    cout << "Primam vo gotovo!" << endl;
    deneshna.kasaPrimi(5000);
    deneshna.prikaziKasa();

    k = new Master(10000.00, "1234567890123456", 54000.00);
    cout << "Primam so kartichka!" << endl;
    deneshna.kasaPrimi(10000.00, *k);
    deneshna.prikaziKasa();

    k = new Maestro("lozinka", "1234567890123456", 54000.00);
    cout << "Primam so kartichka!" << endl;
    deneshna.kasaPrimi(10000, *k);
    deneshna.prikaziKasa();

    Master::setPopust2(0.07);
    k = new Master(10000, "4567891234567890", 3000);
    cout << "Primam so kartichka!" << endl;
    deneshna.kasaPrimi(10000, *k);
    deneshna.prikaziKasa();
    return 0;
}
```

Излез од програмата е:

```
Den:    22
Mesec:  4
Godina: 2014
Prihod-vkupno: 10000

Primam vo gotovo!
Den:    22
Mesec:  4
Godina: 2014
Prihod-vkupno: 15000

Primam so kartichka!
Den:    22
Mesec:  4
Godina: 2014
Prihod-vkupno: 24000

Primam so kartichka!
Den:    22
Mesec:  4
Godina: 2014
Prihod-vkupno: 33500

Primam so kartichka!
Den:    22
Mesec:  4
Godina: 2014
Prihod-vkupno: 42800
```

1.3. Задача

Дел од производите во една продавница по новата политика на продавницата мора да имаат одреден попуст. За таа цел во системот на продавницата потребно е да се моделира апстрактната класа `Discount`. Оваа класа како податок ги има курсевите на евра и долари во денари и методите кои мора секоја класа што ќе наследи од неа да ги имплементира:

- `float discount_price();`
- `float price();`
- `void print_rule();`

За секој производ треба да се чуваат информации за името и цената на производот. Со ова треба да се моделира класата `Product`. Во неа покрај конструкторите треба да се имплементираат сите потребни методи.

Производите се поделени на неколку типови: `FoodProduct`, `Drinks` и `Cosmetics`.

Според новата политика на продавницата храната нема попуст. Пијалоците и тоа алкохолните поскапи од 20 евра имаат попуст 5%, а неалкохолните од

брендот Coca-Cola имаат попуст од 10%. Сите козметички производи поскапи од 5 евра имаат попуст 12%, а оние поскапи од 20 долари имаат попуст 14%.

Да се пресмета вкупната цена на сите производи заедно со попустот.

Исто така да се креира функција `changePrice(float )` во класата `Product` која што ќе нуди можност за промена на постоечката цена на производот. Ако се направи обид да се внесе негативна вредност за цената да се фрли исклучок (објект од класата `NegativeValueException`). Фатете го исклучокот во главната функција каде што ќе ги излистате сите производи од тип `Cosmetics` и ќе им ја промените цената.

Решение `oop_av103.cpp`

```
#include <iostream>
#include <cstring>
using namespace std;

//klasa za iskluchokot
class NegativeValueException{
private:
    char text[50];
public:
    NegativeValueException(char *text)
    {
        strcpy(this->text, text);
    }
    void print() { cout << text; }
};

class Discount {
public:
    static float euro;
    static float dollar;
    virtual float discount_price() = 0;
    virtual float price() = 0;
    virtual void print_rule() = 0;
};

float Discount::euro = 61.7;
float Discount::dollar = 44.5;

class Product {
protected:
    char name[100];
    float price;
public:
    Product(const char *name = "", const float price = 0) {
        strcpy(this->name, name);
        this->price = price;
    }
    float getPrice() {
        return price;
    }
    void print() {
        cout << "Product{ name=" << name << ", price=" << price << "}" << endl;
    }
    void changePrice(float price){
        if (price < 0) throw NegativeValueException("Vnesena e negativna vrednost za
cena!\n");
        this->price = price;
    }
};
```

```
class Cosmetics : public Product, public Discount {
private:
    int weight;
public:
    Cosmetics(const char *name = "", const float price = 0,
        const int weight = 0) :Product(name, price){
        this->weight = weight;
    }
    float discount_price() {
        if (getPrice() / Discount::dollar > 20)
            return 0.86 * getPrice();
        if (getPrice() / Discount::euro > 5)
            return 0.88 * getPrice();
        return getPrice();
    }
    float price() {
        return getPrice();
    }
    void print_rule(){
        cout << "Site kozmeticki proizvodi poskapi od 5 evra imaat popust od 12%, dodeka
pak onie koi se poskapi od 20 dolari imaat popust 14%" << endl;
    }
};

class FoodProduct : public Product, public Discount{
private:
    float callories;
public:
    FoodProduct(const char *name = "", const float price = 0,
        const float callories = 0) : Product(name, price) {
        this->callories = callories;
    }

    float discount_price() {
        return getPrice();
    }

    float price() {
        return getPrice();
    }

    void print_rule(){
        cout << "Nema popust za proizvodite od tip na hrana" << endl;
    }
};

class Drinks : public Product, public Discount {
private:
    char brand[100];
    bool alcoholic;
public:
    Drinks(const char *name = "", const float price = 0,
        const char *brand = "", const bool alcoholic = false) : Product(name, price) {
        strcpy(this->brand, brand);
        this->alcoholic = alcoholic;
    }
    float discount_price() {
        if (this->alcoholic && (getPrice() / Discount::euro > 20))
            return 0.95 * getPrice();
        if (!this->alcoholic && (strcmp(this->brand, "Coca-Cola") == 0))
            return 0.90 * getPrice();
        return getPrice();
    }
    float price() { return getPrice(); }
    void print_rule(){
        cout << "Site alkoholni pijaloci poskapi od 20 evra imaat popust od 5 % , dodeka
pak nealkoholnite od brendot Coca - Cola imaat popust od 10 % "<<endl;
    }
};

float total_discount(Discount **d, int n) {
    float discount = 0;
    for (int i = 0; i < n; ++i) {
        discount += d[i]->discount_price();
        cout << "Prvicna cena: " << d[i]->price() << endl;
        cout << "So popust: " << d[i]->discount_price() << endl;
```

```
        d[i]->print_rule();
    }
    return discount;
}

int main() {
    int n = 0;
    float newPrice;
    Discount **d = new Discount*[10];
        d[n++] = new FoodProduct("leb", 30);
        d[n++] = new Drinks("viski", 1350, "Jack Daniel's", true);
        d[n++] = new FoodProduct("sirenje", 390, 105);
        d[n++] = new Drinks("votka", 850, "Finlandia", true);
        d[n++] = new Cosmetics("krema", 720, 100);
        d[n++] = new Drinks("sok", 50, "Coca-Cola", false);
        d[n++] = new Cosmetics("parfem", 3500, 50);

    cout << "Vkupnata cena na site proizvodi e: " << total_discount(d, n) << endl;

    //se menuva cenata na site Kozmeticki proizvodi
    cout << "Promena na cenata na kozmetickite proizvodi " << endl;
    for (int i = 0; i < n; ++i) {
        Cosmetics* c = dynamic_cast<Cosmetics *>(d[i]);
        if (c != 0){
            c->print();
            cin >> newPrice;
            try{
                c->changePrice(newPrice);
            }
            catch (NegativeValueException i){
                i.print();
            }
        }
    }

    for (int i = 0; i < n; ++i) {
        delete d[i];
    }
    delete[] d;

    return 0;
}
```

Излез од програмата е:

```
Prvicna cena: 30
So popust: 30
Nema popust za proizvodite od tip na hrana
Prvicna cena: 1350
So popust: 1282.5
Site alkoholni pijaloci poskapi od 20 evra imaat popust od 5 % , dodeka pak neal
koholnite od brendot Coca - Cola imaat popust od 10 %
Prvicna cena: 390
So popust: 390
Nema popust za proizvodite od tip na hrana
Prvicna cena: 850
So popust: 850
Site alkoholni pijaloci poskapi od 20 evra imaat popust od 5 % , dodeka pak neal
koholnite od brendot Coca - Cola imaat popust od 10 %
Prvicna cena: 720
So popust: 633.6
Site kozmeticki proizvodi poskapi od 5 evra imaat popust od 12%, dodeka pak onie
 koi se poskapi od 20 dolari imaat popust 14%
Prvicna cena: 50
So popust: 45
Site alkoholni pijaloci poskapi od 20 evra imaat popust od 5 % , dodeka pak neal
koholnite od brendot Coca - Cola imaat popust od 10 %
Prvicna cena: 3500
So popust: 3010
Site kozmeticki proizvodi poskapi od 5 evra imaat popust od 12%, dodeka pak onie
 koi se poskapi od 20 dolari imaat popust 14%
Vkupnata cena na site proizvodi e: 6241.1
Promena na cenata na kozmetickite proizvodi
Product{ name=krema, price=720}
750
Product{ name=parfem, price=3500}
-3600
Vnesena e negativna vrednost za cena!
```

2. Изворен код од примери и задачи

https://github.com/finki-mk/OOP/

Source code ZIP