**ИКОНОМИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА**

**ФАКУЛТЕТ „ИНФОРМАТИКА“**

**КАТЕДРА „ИНФОРМАТИКА“**

**![A black and white logo

Description automatically generated]()**

**Курсова работа по Обектно ориентирано програмиране на тема**

**„Продажба на мебели“**

|  |  |
| --- | --- |
| **Изготвили:** | **Проверил:** |
| **Елис Рафет, 2 курс, 35 група, ФН: 126416** | **Проф. д.н. Павел Петров** |
| **Беррин Мустафа, 2 курс, 35 група, ФН:** **126835** | **Гл. ас. д-р Стойчо Стоев** |

**ВАРНА, 2024**

**Съдържание**

[**Описание** 3](#_Toc185003305)

[**Програмен код** 4](#_Toc185003306)

# **Описание**

Курсовият проект представлява информационна система за автоматизиране на част от дейността на фирма за мебели. Проектът симулира процеса на поръчки за мебели. Потребителите могат да правят поръчки за различни видове мебели (маси, дивани и столове), да виждат приетите вече поръчки и да избират команда за обработка им.

Основни функции на програмата:

* Добавяне на поръчки: Клиентите могат да избират различни мебели (маси, дивани, столове), да задават техните характеристики като размер, материал или други специфики и начална цена съобразена с техния бюджет.
* Извеждане на списък с информация за приетите поръчки: При избиране на тази опция се извежда списък с поръчки. За всяка поръчка се извежда вида мебел, неговото име, характеристика и крайна цена, която се изчислява спрямо избраните характеристики от клиента.
* Обработка на поръчки: При избиране на тази опция потребителят може да избере коя поръчка да се обработи и съответно да се изтрие от списъка с поръчки.

Използвани обектно-ориентирани концепции:

* Класове и наследяване:

Родителски (базов) клас „Мебели“– основният клас, който описва общите характеристики на всички мебели (име и цена). Включва методи за извеждане на информация (print\_info()) и за изчисляване на цена (calc\_cena()), които се променят за всеки тип мебел.

Класове-наследници – те са 3: Клас „Маса“ – има 3 размера маси: малка, средна и голяма. Цената на малката е същата като на началната цена, средната е с 20% по-скъпа и голямата с 50%. Цената също така зависи и от материала: ако е стъклена цената се увеличава с 80лв., ако е дървена с 50 лв. и желязна със 70 лв. Вторият клас е „Диван“ – има 3 размера и опция дали да бъде разтегателен или не. Ако е разтегателен цената се увеличава с 500 лв. Третият клас „Стол“ – има 3 вида столове: дървени на цена равна на началната, железни +50 лв. и въртящи се +100 лв.

Клас за управление – управлява списъка от поръчки чрез използване на динамична структура от данни (вектор). Включва 3 метода: за добавяне на поръчка, за извеждане на поръчките и за обработка и изтриване на поръчката след обработка.

* Използване на полиморфизъм и виртуални методи:

Методи като calc\_cena() и print\_info() се предефинират във всеки наследник, което осигурява специфична логика за различните видове мебели. Чрез виртуалните методи, програмата динамично определя коя имплементация на методите да използва, в зависимост от типа на обекта.

* Меню:

Реализирано е интерактивно меню, което предоставя лесен достъп до основните функции.

# **Програмен код**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Mebeli

{

public:

Mebeli() {};

~Mebeli() {};

virtual void print\_info() = 0;

virtual int calc\_cena() = 0;

protected:

string name;

int price;

};

class Masa :public Mebeli

{

public:

Masa(string ime, string razmer, string mat, int cena) {

name = ime; size = razmer; material = mat; price = cena;

};

~Masa() {};

virtual int calc\_cena() {

if (size == "1") {

if (material == "1")

return price + 80;

else if (material == "2")

return price + 50;

else return price + 70;

}

else if (size == "2"){

if (material == "1")

return price \*1.2 + 80;

else if (material == "2")

return price \* 1.2 + 50;

else return price \* 1.2 + 70;

}

else {

if (material == "1")

return price \* 1.5 + 80;

else if (material == "2")

return price \* 1.5 + 50;

else return price \* 1.5 + 70;

}

}

virtual void print\_info() {

cout << "Masa - " << name << " Razmer - " << size << ", Material - " << material << ", Cena - " << calc\_cena() << "lv." << endl;

}

private:

string size; string material;

};

class Divan :public Mebeli

{

public:

Divan(string ime, string razmer, string razteg, int cena) {

name = ime; size = razmer; raztegatelen = razteg; price = cena;

};

~Divan() {};

virtual int calc\_cena() {

if (size == "1") {

if (raztegatelen == "1") return price + 500;

else return price;

}

else if (size == "2") {

if (raztegatelen == "1") return price + 500;

else return price;

}

else {

if (raztegatelen == "1") return price + 500;

else return price;

}

}

virtual void print\_info(){

cout << "Diwan - " << name << ", Razmer - " << size << ", Raztegatelen - " << raztegatelen << ", Cena - " << calc\_cena() << "lv." << endl;

}

private:

string size;

string raztegatelen;

};

class Stol :public Mebeli

{

public:

Stol(string ime, string vid, int cena) {

name = ime; type = vid; price = cena;

};

~Stol() {};

virtual int calc\_cena() {

if (type == "1") return price;

else if (type == "2") return price + 50;

else return price + 100;

}

virtual void print\_info() {

cout << "Stol - " << name << ", Wid stol - " << type << ", Cena - " << calc\_cena() << "lv." << endl;

}

private:

string type;

};

class Proizvodstwo

{

public:

Proizvodstwo() {};

~Proizvodstwo() {};

void Add(Mebeli\* obj) {

spisyk.push\_back(obj);

cout << "Poruchkata e prieta." << endl << endl;

};

void List() {

if (spisyk.empty()) {

cout << "Nqma porychki za mebeli.\n";

}

else {

cout << "Prorychkite za mebeli sa: \n";

for (int x = 0; x < spisyk.size(); x++)

spisyk[x]->print\_info();

}

};

int Obrabotka() {

if (spisyk.empty()) {

cout << "Nqma porychki za mebeli.\n";

return 0;

}

else {

cout << "Wywedete nomer na porychka: "; cin >> e;

if ((e > 0) && (e - 1< spisyk.size())) {

int price = spisyk[e-1]->calc\_cena();

spisyk.erase(spisyk.begin() + (e -1));

return price;

}

else {

cout << "Greshen nomer na porychka!!!!!" << endl << endl;

return 0;

}

}

};

private:

vector <Mebeli\*> spisyk;

int e;

};

int main()

{

Proizvodstwo m;

string opciq;

while (true) {

cout << "---------------------Menu---------------------" << endl;

cout << "1.Porychka na masa\n2.Porychka na divan\n3.Poruchka na stol\n4.Izvejdane na spisyka s poruchki\n5.Obrabotka na porychka\n0.Izhod" << endl << endl;

cout << "Izberete opciq: "; getline(cin, opciq);

if (opciq == "0") break;

if (opciq == "1") {

string name, size, material; int price;

cout << "Izberete model na masa: Victoria, Sophia, Mara.\nModel: "; getline(cin, name);

cout << "Izberete edin ot slednite razmeri:\n1.Malka\n2.Sredna\n3.Golqma\nRazmer: "; getline(cin, size);

cout << "Izberete edin slednite materiali: \n1.Styklo\n2.Dyrwo\n3.Jelqzo\nMaterial: "; getline(cin, material);

cout << "Kakwa da e nachalnata cena? "; cin >> price;

m.Add(new Masa(name, size, material, price));

}

if (opciq == "2") {

string name, size, raztegatelen; int price;

cout << "Izberete model na diwan: Victoria, Sophia, Mara.\nModel: "; getline(cin, name);

cout << "Izberete edin ot slednite razmeri:\n1.Malyk\n2.Sreden\n3.Golqm\nRazmer: "; getline(cin, size);

cout << "Da byde li raztegatelen? 1.Da 2.Ne\nOtgovor: "; getline(cin, raztegatelen);

cout << "Kakwa da e nachalnata cena? "; cin >> price;

m.Add(new Divan(name, size, raztegatelen, price));

}

if (opciq == "3") {

string name, vid; int price;

cout << "Izberete model na stol: Victoria, Sophia, Mara.\nModel: "; getline(cin, name);

cout << "Izberete edna ot slednite opcii: \n1.Dyrwen\n2.Jelezen\n3.Wyrtqsht se\nOpciq: "; getline(cin, vid);

cout << "Kakwa da e nachalnata cena? "; cin >> price;

m.Add(new Stol(name, vid, price));

}

if (opciq == "4") m.List();

if (opciq == "5") cout << "Obrabotena porucka - " << m.Obrabotka() << endl;

}

}

A screenshot of a computer

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

A screenshot of a computer program

Description automatically generated





