Universitatea Tehnică "Gheorghe Asachi" Iaşi

Facultatea de Automatică şi Calculatoare

Calculatoare şi Tehnologia Informaţiei

Proiect POO

Magazin

CUPRINS

ENUNȚUL PROBLEMEI

Să se realizeze o aplicație C++ scrisă în stil POO (codul se va baza pe structurarea aplicației pe clase și lucrul cu obiecte, folosind încapsularea, moștenirea și polimorfismul).

Aplicația trebuie să respecte următoarele cerințe:

1. Să utilizeze cele 3 principii de bază ale POO: încapsularea, moștenirea și polimorfismul;
2. Să implementeze o ierarhie clase pe 3 – 4 nivele (clasă de bază, clase derivate din cea de bază)
3. Să utilizeze biblioteca meniuri consolă de la laborator;
4. Să fie însoțită de o documentație (listată);

Documentația trebuie să conțină următoarele:

1. Enunțul problemei;
2. Analiza aplicației ;
3. Descrierea ierarhiei de clase;
4. Documentația completă a codului – descrierea fiecărei clase, ce reprezintă, și semnificația fiecărui membru (câmpuri și metode);
5. Descrierea interfeței cu utilizatorul (un manual de operare);
6. Listing-ul codului sursă;

ANALIZA APLICAȚIEI

Aplicația are ca scop gestionarea produselor dintr-un magazin. Este utilă deoarece utilizatorul poate să țină cu ușurință evidența produselor din magazin. Utilizatoarul poate oricând să adauge, să șteargă, să caute sau să afișeze produsele din magazin.

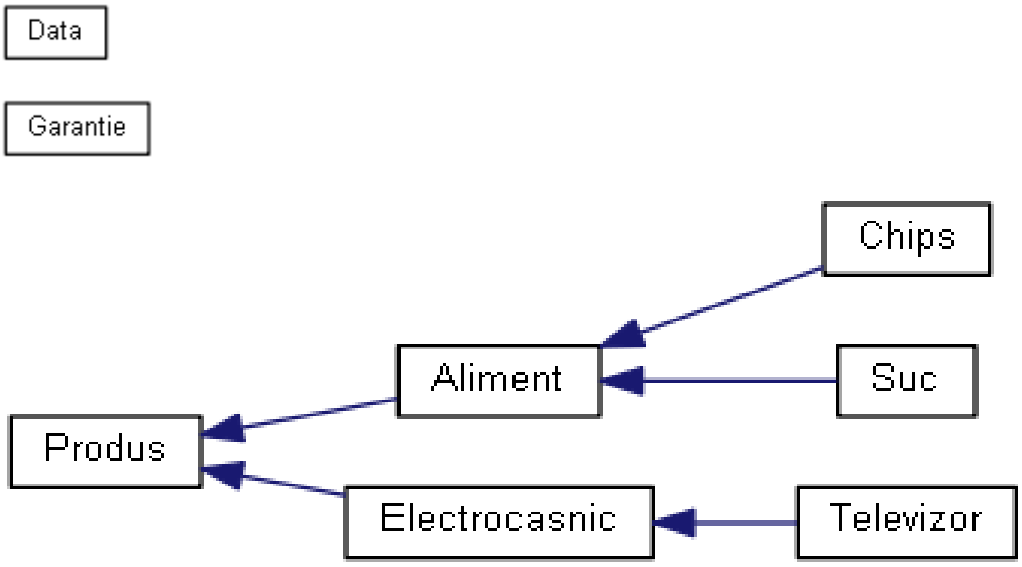
Aplicatia permite gestionarea anumitor tipuri de produse precum: produse alimentare (sucuri, chipsuri) si electrocasnice (televizoare).

Din punct de vedere al programării, aplicația folosește cele trei principii ale programării orientate pe obiecte: încapsularea, moștenirea și polimorfismul; implementează clase de toate tipurile (de bază, derivate, abstracte). Pentru memorarea datelor s-au folosit liste liniare dublu înlănțuite. La lansarea în execuție a aplicației sunt încărcate în program datele din fișierul de lucru Store.txt, dacă acestea există. În timpul execuției se operează pe aceste date, iar la final, înainte de închidere se salvează datele în fișierul de lucru Store.txt.

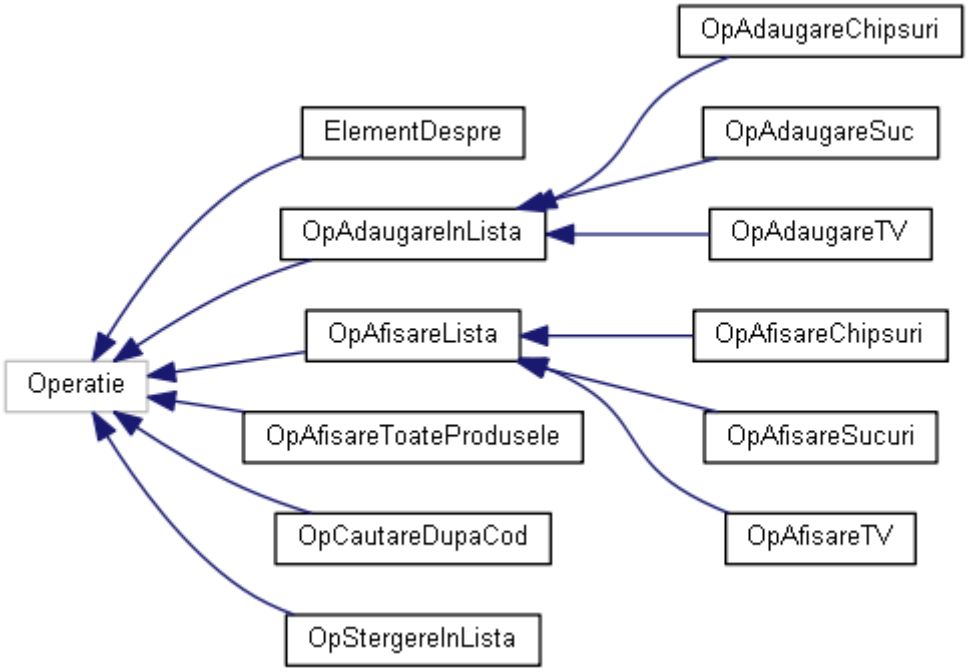
Interfata se utilizeaza usor , aceasta avand un meniu principal si submeniuri.

IERARHIA DE CLASE

În figura de mai jos sunt prezentate clasele de produse:



În figura de mai jos sunt prezentate clasele aferente meniului:



PREZENTAREA CLASELOR

În următoarele pagini vor fi prezentate clasele implementate.

Clasele aferente produselor:

```
class Produs {
	int cod;
	double pret;
protected:
	void afisareProdus();
	void afisareProdusFis();

public:
	Produs():cod(0),pret(0){}
	Produs(int c, double p);
	double getPret() { return pret; }
	int getCod() { return cod; }
	virtual ~Produs() {}
	virtual void afisare();
	virtual void afisareFis();
};
class Data {
	int an;
	int luna;
	int zi;
public:
	Data():zi(0),luna(0),an(0){}
	Data(int a,int b,int c):zi(a),luna(b),an(c){}
	int getAn() { return an; }
	int getLuna() { return luna; }
	int getZi() { return zi; }
	void setAn(int a) { an = a; }
	void setLuna(int b) { luna = b; }
	void setZi(int c) { zi = c; }
	void afisareData();
};
class Aliment :public Produs {
	Data* dataExpirare;
protected:
	void afisareAliment();
	void afisareAlimentFis();
public:
	Aliment():Produs(),dataExpirare(nullptr){}
	Aliment(Produs*, Data*);
	Data* getData();
	~Aliment();
	void afisare();
	void afisareFis();
};
class Garantie {
	int ani;
	int luni;
public:
	Garantie():ani(0),luni(0){}
	Garantie(int a ,int b):ani(a),luni(b){}
```

```
	int getAni() { return ani; }
	int getLuni() { return luni; }
};
class Electrocasnic :public Produs {
	Garantie* garantie;
protected:
	void afisareElectrocasnic();
	void afisareElectrocasnicFis();
public:
	Electrocasnic():Produs(),garantie(nullptr){}
	Electrocasnic(Produs*, Garantie *);
	~Electrocasnic();
	void afisare();
	void afisareFis();
};
class Chips :public Aliment {
	string gust;
	int marimePunga;
protected:
	void afisareChips();
	void afisareChipsFis();
public:
	Chips():Aliment(),gust(string()),marimePunga(0){}
	Chips(Produs*, Data*, string, int);
	~Chips(){}
	string getGust() { return gust; }
	int getMarimePunga() { return marimePunga; }
	void afisare();
	void afisareFis();
};
class Suc :public Aliment
{
	int dimensiuneSticla;
	string marca;
	string gust;
protected:
	void afisareSuc();
	void afisareSucFis();
public:
	Suc():Aliment(),dimensiuneSticla(0),marca(string()),gust(string())
{}
	Suc(Produs*, Data*, int, string,string);
	~Suc(){}
	int getDimensiuneSticla() {  return dimensiuneSticla;}
	string getMarca() { return marca; }
	string getGust() { return gust; }
	void afisare();
	void afisareFis();
};
class Televizor:public Electrocasnic {
	string marca;
	string culoare;
	int diagonala;
protected:
	void afisareTV();
	void afisareTVFis();
public:
```

```
	Televizor():Electrocasnic(),marca(string()),culoare(string()),diag
onala(0){}
	Televizor(Produs*, Garantie*, string, string, int);
	~Televizor(){}
	string getMarca() { return marca; }
	string getCulare() { return culoare; }
	int getDiagonala() { return diagonala; }
	void afisare();
	void afisareFis();
};
```

Clasele aferente operațiilor efectuate de aplicație:

```
class ElementDespre :public Operatie {
public:
	ElementDespre(char*nume):Operatie(nume){}
	void execOperatie();
};
class OpAdaugareInLista :public Operatie {
protected:
	List *lst;
	Produs* pd;
public:
	OpAdaugareInLista(char* nume, List*&lst);
	virtual void execOperatie() = 0;
};
class OpAfisareLista :public Operatie {
protected:
	List *lst;
public:
	OpAfisareLista(char *nume, List *lst);
	virtual void execOperatie() = 0;
};
int verificareCod(int cod, int v, List *lst);
class OpAdaugareSuc :public OpAdaugareInLista {
public:
	OpAdaugareSuc(char* nume, List *&lst);
	void execOperatie();
};
class OpAfisareSucuri:public OpAfisareLista
{
public:
	OpAfisareSucuri(char*nume, List*lst):OpAfisareLista(nume,lst){}
	void execOperatie();
};
class OpAfisareChipsuri :public OpAfisareLista
{
public:
	OpAfisareChipsuri(char*nume, List*lst) :OpAfisareLista(nume, lst)
{}
	void execOperatie();
```

```
};
class OpAfisareTV :public OpAfisareLista
{
public:
      OpAfisareTV(char*nume, List*lst) :OpAfisareLista(nume, lst) {}
      void execOperatie();
};
class OpStergereInLista :public Operatie
{
      List *lst;
public:
      OpStergereInLista(char*nume, List*&lst);
      void execOperatie();
};
class OpCautareDupaCod :public Operatie
{
protected:
      List*lst;
      Produs*pd;
public:
      OpCautareDupaCod(char*nume, List*lst);
      void execOperatie();
};
class OpAdaugareChipsuri :public OpAdaugareInLista
{
public:
      OpAdaugareChipsuri(char*
nume,List*lst):OpAdaugareInLista(nume,lst){}
      void execOperatie();
};
class OpAdaugareTV :public OpAdaugareInLista
{
public:
      OpAdaugareTV(char*nume,List*&lst):OpAdaugareInLista(nume,lst){}
      void execOperatie();
};
class OpAfisareToateProdusele:public Operatie
{
      List*lsta1;
      List*lsta2;
      List*lsta3;
public:
      OpAfisareToateProdusele(char*nume, List *lsta1, List*lsta2,
List*lsta3);
      void execOperatie();
};
```

MANUALUL DE OPERARE

La lansarea aplicației va aparea un meniu, de unde se pot executa toate operatiile aferente programului. În meniu se poate naviga foarte ușor, utilizând tastele de la ‘0’ la ‘9’. Din meniul principal utilizând corect tastele indicate se poate ajunge în oricare din submeniuri și de asemenea se pot lansa operații de adăugare, afișare, ștergere sau căutare.

Produsele cu care se va lucra sunt: chipsuri, sucuri si televizoare.

Pentru fiecare tip de produs există posibilitatea executării oricăreia din cele patru operații amintite mai sus. Operația de afișare a datelor se poate face pentru toate produsele la un loc.

La introducerea datelor, utilizatorul trebuie să fie atent la comentariile aflate între paranteze, impuse pentru a realiza anumite limitări. Pentru introducerea datelor de tip întreg sau real s-au impus limitări legate de apartenența la un anumit interval de numere. Pentru introducerea datelor de tip caracter este impusă introducerea unui singur cuvânt pentru fiecare caracteristică a unui produs în parte, asadar fără spații între cuvintele introduse.

Noțiunea de cod a fost introdusă pentru a putea identifica mai ușor produsele din magazin. Acesta este reprezentat de un numar intreg si este caracteristic pentru fiecare produs in parte, utilizatorul trebuind sa respecte un set de reguli in momentul introducerii acestuia.

După fiecare operație efectuată asupra unei liste de produse, se afișează lista pentru a putea observa modificările făcute. Ieșirea din aplicație se face utilizând tasta ’ESC’ sau ‘0’ atât pentru a reveni la un meniu anterior cât și pentru a ieși din program.

Sectiunea Despre Program este descrisa de urmatoarea functie:

```
void ElementDespre::execOperatie()
{
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 70);
	cout << "\tProgramul simuleaza gestionarea produselor dintr-un
magazin . \n"
	 << "\tMagazinul are drept produse: Alimente(Chipsuri si
sucuri) si Electrocasnice(Televizoare)." << endl;
	cout << "\tExista posibilitatea de a cauta produse dupa cod, de
sterge anumite produse dupa indexul din magazin si de afisare a
produselor pe categorii" << endl;
	cout << "\tCodurile produselor sunt formate din 5 cifre (de la 1 la
9) si se prezinta in felul urmator: " << endl;
	cout << "\tToate produsele au pe prima pozitie cifra 1." << endl;
	cout << "\tAlimentele au pe a doua pozitie cifra 1. Pe a treia
pozitie a alimentelor se regaseste pentru:" << endl;
	cout << "\t\tSucuri - 1;" << endl;
	cout << "\t\tChipsuri - 2;" << endl;
	cout << "\tElectrocasnicele au pe a doua pozitie cifra 2. Pe a doua
pozitie a acestora se regaseste pentru:" << endl;
	cout << "\t\tTelevizoare - 1;" << endl;
	cout << "\tIn magazin se pot adauga, sterge sau extrage diferite
produse folosind tastele 0-9" << endl;
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 67);
	pauza();
}
```

LISTING COD SURSĂ

Funcția de citire din fișier:

```
void citireListe(List *lista1, List *lista2, List *lista3)
{
	int nr, i = 1;
	f >> nr;
	f.get();
	Produs *a = new Produs();
	assert(a);
	while (i <= nr)
	{
		a = citesteSucuriFis();
		(*lista1) += a;
		i++;
		f.get();
	}
	i = 1;
	f >> nr;
	f.get();
	while (i <= nr)
	{
		a = citesteChipsuriFis();
		(*lista2) += a;
		f.get();
		i++;
	}
	i = 1;
	f >> nr;
	f.get();
	while (i <= nr)
	{
		a = citesteTVFis();
		(*lista3) += a;
		i++;
		f.get();
	}
	inchideFis();
}
```

Funcția de ștergere din listă:

```
void List::sterge()
{
	Nod *a;
	Nod*el = cap;
	int cod;
	if (cap == 0)
	{
		SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 46);
		cout << "Lista este vida!" << endl;
		SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 67);
	}
	else
	{
		cout << "Dati codul produsului pe care doriti sa il stergeti:
";
		cin >> cod;
		if (cap->getProdus()->getCod() == cod)
		{
			a = cap;
			cap = cap->next;
			cap->prev = 0;
			el = cap;
			delete a;
			nrEl--;
		}
		else
		{
			while (cap->next != 0 && cap->next->getProdus()-
>getCod()!= cod)
				cap = cap->next;
			if (cap->next == nullptr)
			{

	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 46);
				cout << "Produsul cu codul " << cod << " nu se
afla in magazin." << endl;

	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 67);
			}
			else
			{
				a = cap->next;
				cap ->next = a->next;
				if(a->next!=nullptr)
					a->next->prev = cap;
				delete a;
				nrEl--;
			}
			cap = el;
		}
	}
}
```

Funcția de verificare a unicității codului:

```
int verificareCod(int cod, int v, List *lst)
{
	int ok = 0;
	if (v > 0 && v < 10)
	{
		if (cod / 10000 != v)
			return 1;
	}
	else if (v > 10 && v < 99)
	{
		if (cod / 1000 != v)
			return 1;
	}
	else if (v > 100 && v < 9999)
	{
		if (cod / 100 != v)
			return 1;
	}
	Nod *cap = lst->getCap();
	while (cap != nullptr)
	{
		if (cap->getProdus()->getCod() == cod)
		{
			ok = 1;
			cap = nullptr;
		}
		else
		{
			cap = cap->getNext();
		}
	}
	if (ok == 1)
	{
		SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
46);
		cout << "Acest cod a mai fost folosit pentru un alt produs!"
<< endl;
		cout << "Reintroduceti codul: ";
		SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),
67);
	}
	return ok;
}
/*  verificareCod - Returneaza 1 in cazul in care codul nu respecta
normele amintite in sectiunea Despre Program
	In cazul in care codul e deja folosit afiseaza mesaj de eroare si
returneaza 1
	Cazul in care returneaza 0 este singurul acceptat. Doar atunci
codul introdus este corect si disponibil.
*/
```

Funcția de căutare după cod:

```
void OpCautareDupaCod::execOperatie()
{
	int ok = 0;
	int cod;
	cout << "Introduceti codul: ";
	cin >> cod;
	Nod*cap = this->lst->getCap();
	if (cap == nullptr)
	{
		cout << "Lista este vida!" << endl;
	}
	else
	{
		while (cap != nullptr)
		{
			if (cap->getProdus()->getCod() == cod)
			{
				ok = 1;
				cap->getProdus()->afisare();
				cap = nullptr;
			}
			else
				cap = cap->getNext();
		}
		if (ok == 0)
			cout << "Produsul avand codul dat  nu exista!"<<endl;
	}
	cout << endl;
	pauza();
}
```