Universitatea Tehnică “Gheorghe Asachi” Iaşi

Facultatea de Automatică şi Calculatoare

Calculatoare şi Tehnologia Informaţiei

Proiect POO

Fitness Store

Student: Zavalichi Răzvan-Andrei

Grupa: 1207A

Profesor: Cîmpanu Corina

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 de 4-5 clase pe 3 – 4 nivele (clasă de bază, clase derivate din cea de bază)
3. Să conțină clase cu date membre de tip pointer;
4. Să conțină o funcție prietenă cu o clasă;
5. Să utilizeze biblioteca meniuri consolă de la laborator;
6. 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.

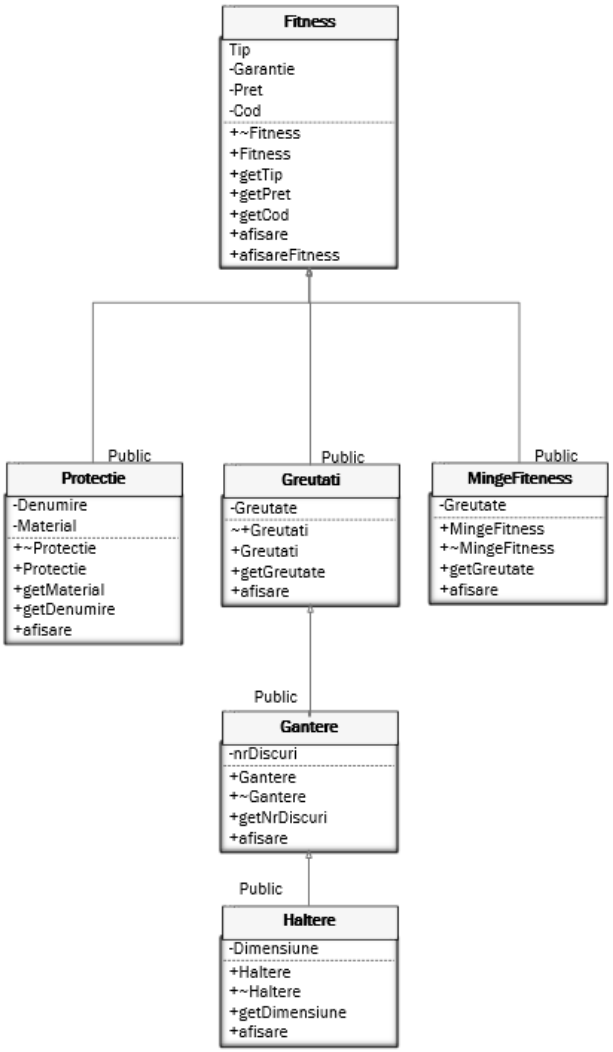
Aplicația permite gestionarea numai anumitor tipuri de produse pentru fitness, și anume: echipamente de protecție, mingi de fitness și greutăți (gantere, haltere). Dar utilizatorul poate oricând să adauge un nou tip de produs, într-o secțiune specială. Aplicația este cu mult mai utilă prin faptul că permite accesul nu numai la fiecare tip de produs în parte, ci și la anumite grupuri. De exemplu se poate lucra cu toate produsele pentru forță sau doar cu greutățile disponibile în magazin.

Din punct de vedere al programării, aplicația folosește cele trei principii ale programării orientate pe obiecte: încapsularea, moștenirea și polimorfismul; implementează foarte multe clase de toate tipurile (de bază, derivate, prietene, abstracte). Pentru memorarea datelor s-au folosit liste liniare simplu înlănțuite. De asemenea majoritatea câmpurilor claselor sunt date de tip pointer. La lansarea în execuție a aplicației sunt încărcate în program datele din fișierul de lucru Magazin.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 Magazin.txt.

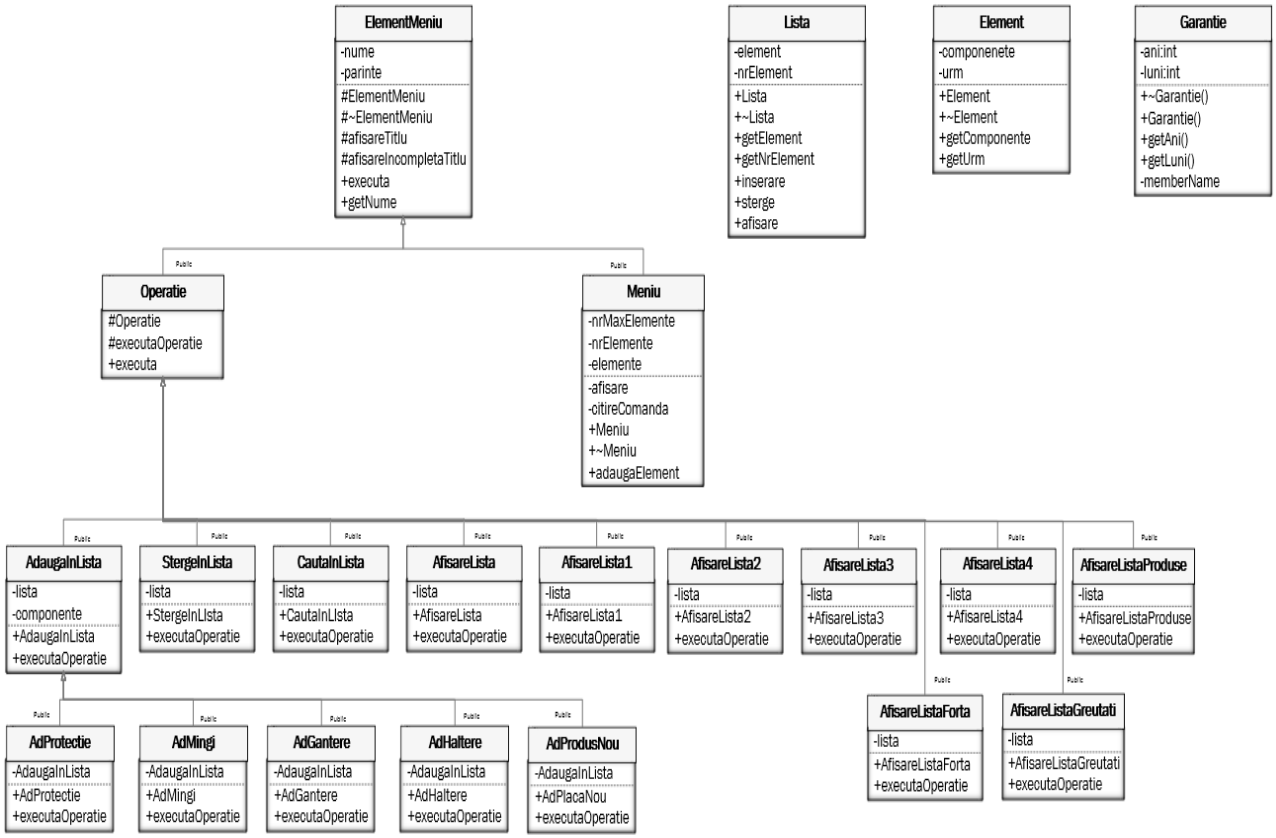
Interfața este una prietenoasă, ușor de utilizat, având un meniu principal și submeniuri.

IERARHIA DE CLASE

În figura de mai jos sunt prezentate clasele de produse:



În figura de mai jos sunt prezentate clasele aferente meniului:



Ierarhia de clase poate fi văzută mai clar în imaginea anexată documentației.

PREZENTAREA CLASELOR

În următoarele pagini vor fi prezentate clasele noi implementate

Clasele aferente produselor:

```
class Garantie{ //Garantia in ani si luni a unui produs
private:
	int ani;
	int luni;
public:
	Garantie() {}
	Garantie(int ani,int luni);
	~Garantie();
	int getAni(); //returneaza anii
	int getLuni(); //returneaza lunile
};

class Fitness{ //reprezinta informatia legata de un produs
private:
	char *cod; //cod unic repartizat fiecărui produs
	char *tip; //tipul produsului
	double pret; //pretul
	Garantie *garantie; //garantia produsului
protected:
	void afisareFitness(); //afiseaza in consola
	void afisareFinessF(); //afiseaza in fisier
public:
	Fitness(){}
	Fitness(char *tip, double pret, char *cod, Garantie *garantie);
	virtual ~Fitness();
	Garantie *getGarantie(); //returneaza pointer la garantie
	char *getCod(); //returneaza codul
	char *getTip(); //returneaza pointer la tip
	double getPret(); //returneaza pretul
	virtual void afisare(); //afiseaza informatia despre produs
	virtual void afisareF(); //scrie in fisier
};

class Protectie: public Fitness{ //reprezinta informatia legata de o
echipamentele de protectie
private:
	char *denumire; //denumirea produsului
	char *material; //tipul de material al produsului
public:
	Protectie(){}
	Protectie(Fitness *fit, char *denumire, char *material);
	~Protectie();
	char *getDenumire(); //returneaza pointer la denumire
	char *getMaterial(); //returneaza pointer la material
	void afisare(); //afiseaza informatia despre produs
	void afisareF(); //scrie in fisier
};

class MingeFitness: public Fitness{ //reprezinta informatia legata de
mingile de fitness
private:
```

```
	double greutate; //greutatea in kg a produsului
public:
	MingeFitness(){}
	MingeFitness(Fitness *fit,double greutate);
	~MingeFitness();
	double getGreutate(); //returneaza greutatea
	void afisare(); //afiseaza informatia despre produs
	void afisareF(); //scrie in fisier
};

class Greutati:public Greutati{ //reprezinta informatia legata de
greutati
private:
	double greutate; //masa produsului
protected:
	void afisareGreutati(); //afiseaza informatia despre greutati
	void afisareGreutatiF(); //scrie in fisier
public:
	Greutati(){}
	Greutati(Fitness *fit,double pret);
	~Greutati();
	char *getGreutate(); //returneaza pointer la greutate
	void afisare(); //afiseaza informatia despre produs
	void afisareF(); //scrie in fisier
};

class Gantere:public Greutati{ //reprezinta informatia legata de
gantere
private:
	int nrDiscuri; //numarul de discuri
protected:
	void afisareGantere(); //afiseaza informatia despre produs
	void afisareGantereF(); //scrie in fisier
public:
	Gantere(){}
	Gantere(Fitness *fit,double greutate,int nrDiscuri);
	~Gantere();
	double getNrDiscuri(); //returneaza numarul de discuri
	void afisare(); //afiseaza informatia despre produs
	void afisareF(); //scrie in fisier
};


class Haltere:public Gantere{ //reprezinta informatia legata de haltere
private:
	double dimensiune; //dimensiunea barei
public:
	Haltere(){}
	Haltere(Fitness *fit,double greutate, int nrDiscuri, double
dimensiune);
	~Haltere();
	double getDimensiune(); //returneaza pointer la dimensiune
	void afisare(); //afiseaza informatia despre produs
	void afisareF(); //scrie in fisier
};
```

Clasele aferente lucrului cu liste liniare simplu înlănțuite:

```
class Element
{
private:
	Fitness *produs; //produsul reprezinta informatia
	Element *urm; //legatura catre urmatorul element
public:
	Element() {}
	Element(Fitness *produs, Element *urm);
	Fitness *getProdus(); //returneaza pointerul la produs
	Element *getUrmator(); //returneaza pointerul la urmatorul element
	~Element();
	friend class Lista;
};

class Lista
{
private:
	Element *el; //capul listei
	int nrEl; //nr de elemente
public:
	Lista();
	~Lista();
	Element *getElement(); //returneaza pointerul la elementul curent
	int getNrEl(); //returneaza numarul de elemente din lista
	void operator += (Fitness *&p); //Adauga un element in lista
	void sterge(); //sterge un element din lista
	void afisare(); //afiseaza lista de produse
	int afisare2(int nr); //afiseaza lista de produse in alt mod
	void afisareF(); //scrie o lista in fisier
};
```

Clasele aferente operațiilor efectuate de aplicație:

```
class ElementDespre:public Operatie
{
public:
	ElementDespre(char *nume);
	void execOperatie();
};

class OpAdaugareInLista:public Operatie //Adauga un produs in lista
{
protected:
	Lista *lista;
	Fitness *fit; //pointer de tip Fitness
public:
	OpAdaugareInLista(char *nume, Lista *&lista);
	virtual void execOperatie()=0;
```

```
};

class OpAdaugareProdusNou:public OpAdaugareInLista //Adauga un produs
nou in lista
{
public:
      OpAdaugareProdusNou(char *nume,Lista *&lista);
      void execOperatie();
};

class OpAdaugareProtectie:public OpAdaugareInLista  //Adauga un produs
de protectie in lista
{
public:
      OpAdaugareProtectie(char *nume, Lista *&lista);
      void execOperatie();
};

class OpAdaugareMingeFitness:public OpAdaugareInLista  //Adauga o minge
fitness in lista
{
public:
      OpAdaugareMingeFitness(char *nume, Lista *&lista);
      void execOperatie();
};

class OpAdaugareGantere : public OpAdaugareInLista //Adauga un
produs(gantera) in lista
{
public:
      OpAdaugareGantere(char *nume, Lista *&lista);
      void execOperatie();
};

class OpAdaugareHaltere : public OpAdaugareInLista //Adauga un
produs(haltere) in lista
{
public:
      OpAdaugareHaltere(char *nume, Lista *&lista);
      void execOperatie();
};

class OpAfisareLista:public Operatie //Afiseaza continutul unei liste
de Accesorii
{
private:
      Lista *lista;
public:
      OpAfisareLista(char *nume, Lista *lista);
      void execOperatie();
};

class OpAfisareLista1:public Operatie //Afiseaza continutul unei liste
de produse pentru protectie
{
private:
      Lista *lista;
public:
```

```
      OpAfisareLista1(char *nume, Lista *lista);
      void execOperatie();
};

class OpAfisareLista2:public Operatie //Afiseaza continutul unei liste
cu mingi fitness
{
private:
      Lista *lista;
public:
      OpAfisareLista2(char *nume, Lista *lista);
      void execOperatie();
};

class OpAfisareLista3:public Operatie //Afiseaza continutul unei liste
de Greutati(gantere)
{
private:
      Lista *lista;
public:
      OpAfisareLista3(char *nume, Lista *lista);
      void execOperatie();
};

class OpAfisareLista4:public Operatie //Afiseaza continutul unei liste
de greutati(haltere)
{
private:
      Lista *lista;
public:
      OpAfisareLista4(char *nume, Lista *lista);
      void execOperatie();
};

class OpAfisareListeProduse:public Operatie //Afiseaza toate listele cu
toate produsele
{
private:
      Lista *lista1,*lista2,*lista3,*lista4,*lista5,*lista6;
public:
      OpAfisareListeProduse(char *nume, Lista *lista1, Lista *lista2,
Lista *lista3, Lista *lista4, Lista *lista5, Lista *lista6);
      void execOperatie();
};

class OpAfisareListeForta:public Operatie //Afiseaza toate listele cu
toate produsele pentru forta
{
private:
      Lista *lista1,*lista2,*lista3,*lista4;
public:
      OpAfisareListeForta(char *nume, Lista *lista1, Lista *lista2,
Lista *lista3, Lista *lista4);
      void execOperatie();
};

class OpAfisareListeGreutati:public Operatie //Afiseaza toate listele
cu toate greutatile(gantere/haltere)
```

```
{
private:
	Lista *lista1,*lista2;
public:
	OpAfisareListeGreutati(char *nume, Lista *lista1, Lista *lista2);
	void execOperatie();
};



class OpCautareDupaCod:public Operatie //Afiseaza un produs dupa cod
{
private:
	Lista *lista;
public:
	OpCautareDupaCod(char *nume, Lista *lista);
	void execOperatie();
};


class OpStergereInLista:public Operatie  //Sterge un produs din lista
{
private:
	Lista *lista;
public:
	OpStergereInLista(char *nume, Lista *&lista);
	void execOperatie();
};
```

MANUALUL DE OPERARE

La lansarea aplicației va apare 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 se împart în echipamente pentru protectie, echipamente pentru forta și produse noi, după care echipamentele pentru forta se împart în mingi de fitness si greutati, iar greutatile se împart în gantere și haltere. Produsele noi se referă la alte tipuri de produse fitness. Această organizare este vizibilă și în meniul aplicației.

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, pentru toate echipamentele de forta sau toate greutatile. Deci operația de afișare se poate face și pe grupări de produse.

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, deci fără spații între cuvintele introduse. O excepție de la introducerea normală a unui șir de caractere o face introducerea codului pentru fiecare produs, cod unic și care respectă anumite reguli. Noțiunea de cod a fost introdusă pentru a putea identifica mai ușor produsele din magazin.

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

LISTING COD SURSĂ

Funcția de citire din fișier:

```
void citireListe(Lista *lista1,Lista*lista2,Lista *lista3,Lista
*lista4,Lista *lista5,Lista *lista6)
{
	int nr,i=1; //nr reprezinta numarul de elemente pentru fiecare
lista ce urmeaza a fi citite
	f>>nr;
	f.get();
	Fitness *a=new Fitness(); //a este pointer catre clasa Fitness
	assert(a);
	while(i<=nr)
	{
	a=citesteProtectieF(); //construim obiectul de tip Protectie
	(*lista1)+=a; //adaugam un nou produs in lista de echipamente
pentru protectie
	i++;
	f.get();
	}
	i=1;
	f>>nr;
	f.get();
	while(i<=nr)
	{
	a=citesteMingeFitnessF(); //construim obiectul de tip MingeFitness
	(*lista2)+=a; //adaugam un nou produs in lista de mingi
	f.get();
	i++;
	}
	i=1;
	f>>nr;
	f.get();
	while(i<=nr)
	{
	a=citesteGreutatiF(); //construim obiectul de tip Greutati
	(*lista3)+=a; //adaugam un nou produs in lista de greutati
	i++;
	f.get();
	}
	i=1;
	f>>nr;
	f.get();
	while(i<=nr)
	{
	a=citesteGantereF(); //construim obiectul de tip Gantere
	(*lista4)+=a; //adaugam un nou produs in lista de gantere
	i++;
	f.get();
	}
	i=1;
	f>>nr;
	f.get();
	while(i<=nr)
	{
	a=citesteHaltereF(); //construim obiectul de tip Haltere
```

```
	(*lista5)+=a; //adaugam nou produs in lista de haltere
	i++;
	f.get();
	}
	i=1;
	f>>nr;
	f.get();
	while(i<=nr)
	{
	a=citesteFitnessF(); //construim obiectul de tip Fitness
	(*lista6)+=a; //adaugam nou pordus in lista de accesorii
	i++;
	f.get();
	}
	inchideF();
}
```

Funcția de ștergere din listă:

```
void Lista::sterge()
{
	Element *aux, *cap=el; //initializam cap cu primul element din
lista
	int i, nr;
	if (el==0)
		cout<<"Lista produselor este vida!"<<endl;
	else
	{
		do
		{
			cout<<"Dati numarul de ordine al produsului in lista
(numar pozitiv)";
			cin>>nr; //citire pozitia de la care se va sterge
		} while (nr<1);
		if (nr==1) //daca este chiar primul element
		{
			aux=el; //pastram primul element in aux
			el=el->urm; //al doilea element va deveni primul
			cap=el; //cap este initializat cu primul element
			delete aux; //stergem primul element
			nrEl--; //decrementam numarul de elemente din lista
		}
		else //daca nu este primul element
		{
			for (i=2;(i<nr)&&(el->urm!=0);i++) //parcurgem lista
pana ajungem la elemntul ce precede pe cel ce trebuie eliminat
				el=el->urm;
			if (el->urm==0) //daca am ajuns la capatul listei
				cout<<"Numar de ordine incorect"<<endl;
			else
			{
				aux=el->urm; //aux va primi elementul ce trebuie
eliminat
				el->urm=aux->urm; //refacem legatura in lista
				delete aux; //stergem elementul
				nrEl--; //decrementam numarul de elemente din
lista
```

```
                }
            }
            el=cap; //el va fi iar primul element din lista
      }
}
```

Funcția de verificare a unicității codului:

```
int verificareCod(char *c,char *ch,Lista *lista)
{
      int ok,nr;
      char *aux;
      aux=new char[strlen(c)+1];
      assert(aux);
      strcpy(aux,c);
      strcpy(aux,aux+3);
      if(strncmp(c,ch,3)==0) ok=0; //daca coincide cu prefixul pentru
fiecare tip de produs
            else return 1;
      if(strlen(c)==7)  ok=0; //daca s-au introdus 7 caractere
                  else return 1;
      if(isdigit(aux[0])) nr=atoi(aux); //luam numarul
      if(nr<1000 || nr>9999) return 1; //si vedem daca e in intervalul
1000-9999
      Element *cap=lista->getElement(); //initializa cap cu primul
element din lista
      while(cap!=NULL)
            {
                  if(strcmp(cap->getFitness()->getCod(),c)==0)
                        {
                              ok=1; //produsul avand codul dat este gasit
                              cap=NULL;
                        }
                  else
                        cap=cap->getUrmator(); //trecem la urmatorul
element din lista
            }
      if(ok==1) cout<<"Exista deja un produs in aceasta lista cu acest
cod."<<endl<<"Va rog introduceti alt cod."<<endl;

      return ok;
}
```

Funcția de căutare după cod:

```
void OpCautareDupaCod::execOperatie()
{
      int ok=0; //variabila de control pentru a vedea daca s-a afisat
vreun produs
      char c[10];
      cout<<"Introduceti un cod: ";
      cin>>c;
```

```
	Element *cap=this->lista->getElement(); //initializa cap cu primul
element din lista
	if (cap==0)
		cout<<"Lista este vida !"<<endl;
	else
	{
		while(cap!=NULL)
		{
			if(strcmp(cap->getFitness()->getCod(),c)==0)
				{
					cap->getFitness()->afisare(); //afisam
produsul care are codul dat
					ok=1; //produsul avand codul dat este gasit
					cap=NULL;
				}
			else
				cap=cap->getUrmator(); //trecem la urmatorul
element din lista
		}
	if (ok==0) cout<<"Produsul cu codul dat nu exista in lista"<<endl;
//nu am gasit produs cu codul dat
	}
	pauza();
}
```