Modelarea orientata pe obiecte a unei aplicatii pentru coordonarea unui aparat meteo autonom prin utilizarea diferitelor tipuri de senzori

Student – Dumitru Andreea Emilia

Universitatea Petrol-Gaze din Ploiesti

Programare Orientata pe Obiecte

2020

Cuprins

Capitolul 1 – Introducere

1.1 Descrierea domeniului temei

Meteorologia este disciplina care se ocupa de studiul fenomenelor atmosferice, avand ca obiect principal procesele climatice – precipitatii, temperatura, curenti de aer, descarcari electrice, precum si prognoza acestora. Activitatea in domeniul meteorologiei implica cercetarea conditiilor si fenomenelor meteorologice: prelevarea, analizarea, evaluarea si interpretarea datelor meteorologice pentru prevederea vremii si determinarea conditiilor climatice specifice unor regiuni geografice.

Studiul climatic al unei zone presupune analiza detaliata a elementelor climatice relevante, precum: radiatia solara, temperatura aerului si a solului, presiunea atmosferica, vantul, umezeala aerului, nebulozitatea, precipitatiile atmosferice, evapotranspiratia potentiala, stratul de zapada, ceata, grindina, bruma, etc.

Temperatura aerului este o consecinta a fluxului radiativ solar si a fluxului energetic teluric generat de miezul incandescent al Pamantului. Principalii parametri ce caracterizeaza temperatura sunt: temperatura medie, temperatura extrema, frecventa zilelor cu diferite valori de temperatura, variatiile de lunga durata ale temperaturii aerului, indicii de temperatura etc.

In meteorologie, presiunea atmosferica este definita ca fiind forta prin care aerul apasa pe unitatea de suprafata. Presiunea atmosferica se masoara in milimetri coloana de mercur sau in milibari.

Vantul are doua caracteristici principale: directia si viteza. In afara de acestea, se mai deosebesc taria sau intensitatea vantului si inclinatia sau componenta verticala a acestuia. Viteza vantului este egala cu spatiul parcurs de masa de aer in miscarea sa orizontala, in unitatea de timp si se exprima in m/s sau km/h.

Umezeala aerului se caracterizeaza prin tensiunea vaporilor de apa (partea care revine vaporilor din presiunea totala a atmosferei) si umiditatea absoluta (cantitatea de vapori de apa aflata la un moment dat intr-un metru cub de aer). Nici tensiunea vaporilor de apa si nici umezeala absoluta nu sunt marimi caracteristice extrem de precise ale gradului de umezeala sau de uscaciune al aerului, deoarece la o temperatura ridicata acesta poate fi uscat, iar la o temperatura scazuta poate fi saturat. De aceea, pentru a exprima intr-un mod relevant si evident gradul de umiditate a aerului se folosesc umezeala relativa, dar si cea specifica, care arata starea de umiditate a aerului in procente.

Nebulozitatea sau gradul de acoperire a cerului cu nori este un element climatic care influenteaza desfasurarea celorlalte procese atmosferice. Nebulozitatea are ca parametri valorile medii zilnice, decadale, lunare si anuale, frecventa medie a zilelor cu diferite grade de innorare, frecventa genurilor de nori, variatiile de lunga durata ale nebulozitatii.

Precipitatiile atmosferice prezinta un interes practic deosebit, in domenii precum hidrotehnia, agricultura, turismul, fiind caracterizate de parametri precum cantitatea medie lunara, anotimpuala, semestriala si anuala, cea mai mica si cea mai mare cantitate lunara, anotimpuala si anuala, probabilitatea de producere si gradul de asigurare a diferitelor cantitati de precipitatii, frecventa zilelor cu diferite cantitati de precipitatii, cantitatea maxima intr-un numar relativ de ore, ploile torentiale, indicii pluviometrici etc.

Elaborarea prognozelor meteo reprezinta scopul cel mai important al domeniului meteorologic, intrucat prognoza semnifica o anticipare a caracteristicilor si conditiilor atmosferice pentru un anumit interval de timp in viitor si pe o anumita suprafata, cu ajutorul metodelor stiintifice.

1.2 Descrierea temei si scopul aplicatiei

Un model orientat pe obiecte are la baza obiectul care inglobeaza, atat atribute, cat si comportament, ceea ce face posibila abordarea orientata pe obiecte pentru modelarea de date, dar si pentru modelarea de procese. Flexibilitatea obiectelor, incapsularea, mostenirea, polimorfismul – toate acestea stau la baza dezvoltarii dinamice de obiecte si sisteme de obiecte.

Autonomia presupune abilitatea de administrare in regim de independenta fata de mediu si factori disturbatori.

Modelarea orientata pe obiecte in coordonarea unui aparat meteo autonom presupune crearea de diferite obiecte apartinand diferitelor clase inglobate in modelarea aparatului. Aparatul meteo functioneaza prin utilizarea diferitelor tipuri de senzori care masoara principalele caracteristici ale vremii, adica principalii factori climatici. Acesta este proiectat sa lucreze intr-un regim autonom, adica independent de mediu sau de utilizator, facand masuratorile factorilor climatici in mod automatizat. Pe langa calculul acestor caracteristici, aparatul dispune si de functii de alerta care se activeaza atunci cand valorile inregistrate de senzorii aparatului depasesc valorile medii standard general admise, implementate in software-ul programului.

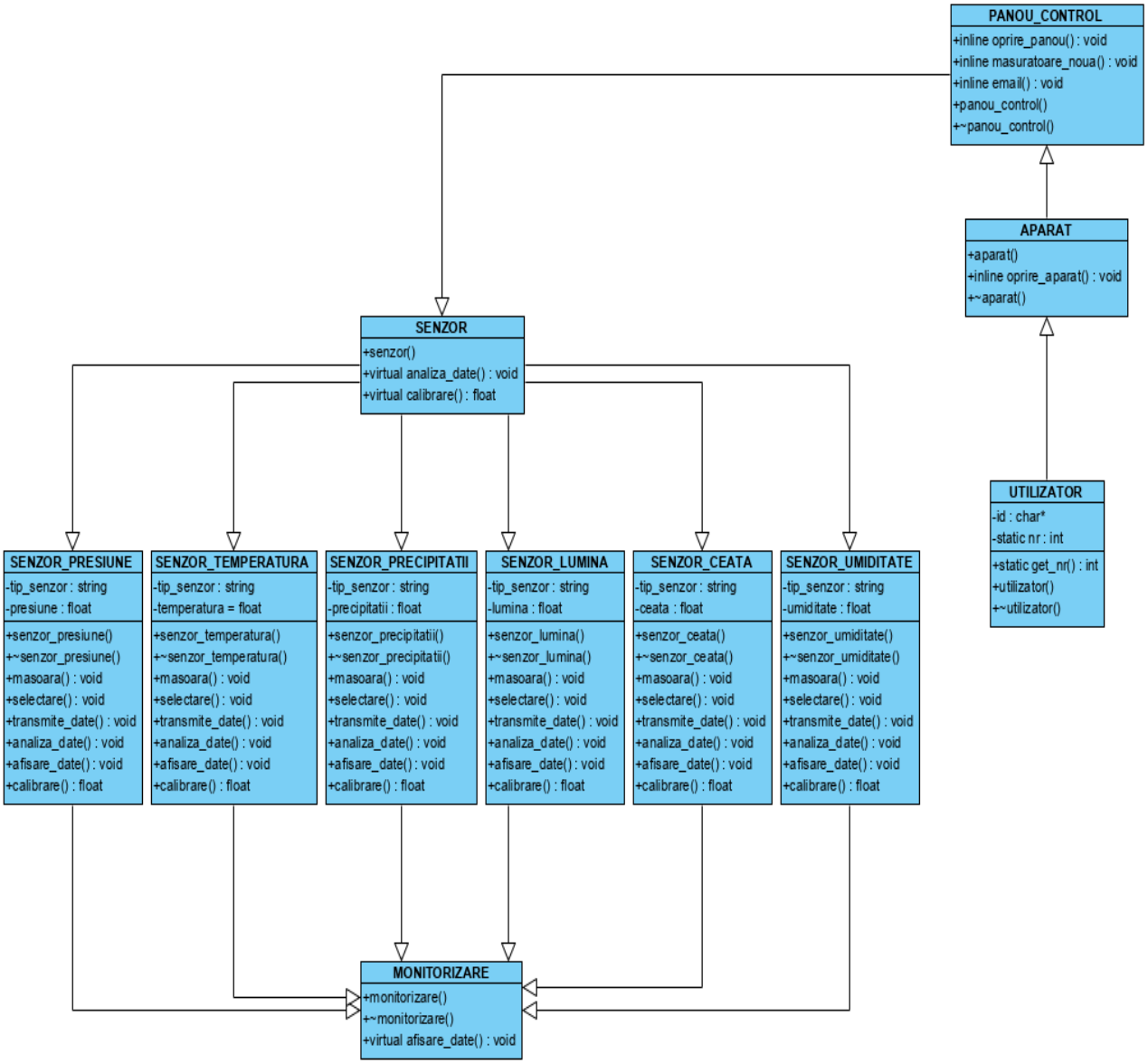
Aparatul meteo are ca si scop masurarea cat mai exacta a valorilor presiunii, temperaturii, precipitatiilor, gradului luminos, gradului de ceata si a umezelii, in vederea afisarii acestora printr-o interfata la care are acces utilizatorul. Aparatul dispune de functiile de alerta si de analiza a datelor, rezultatele fiind furnizate utilizatorului si prin intermediul unor email-uri informative care cuprind datele si alertele inregistrate in ziua curenta.

Utilizatorul are la dispozitie functia de calibrare a aparatului pentru a reporni masuratorile incepand cu valorile 0, dar si functia de selectare a unui anumit senzor pentru realizarea unei masuratori a parametrului care este in interesul utilizatorului sa-l cunoasca.

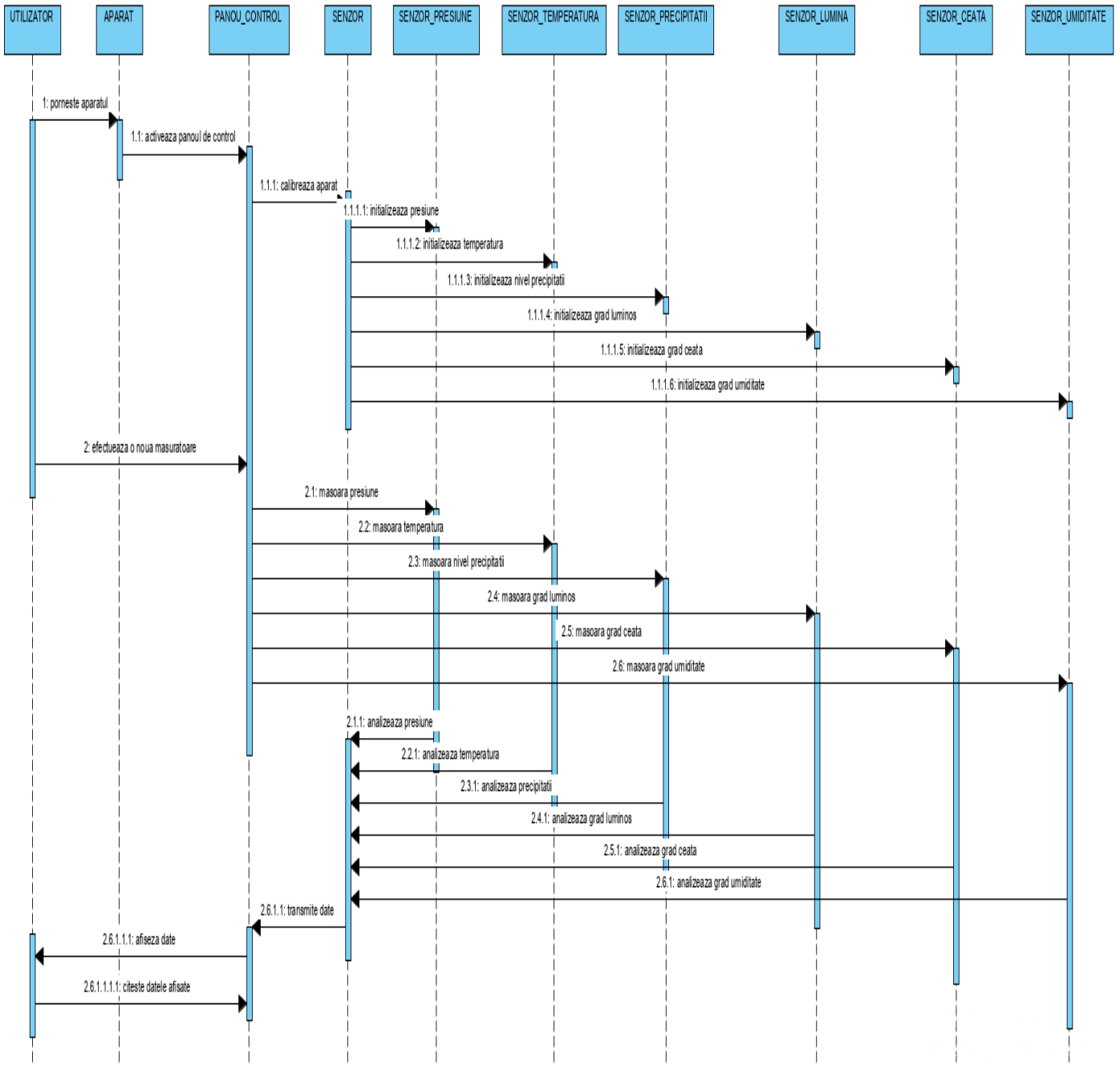
Scopul aplicatiei dezvoltate pentru un aparat meteo autonom este de a pune la dispozitia utilizatorului un meniu de functii cu capacitatea de a exploata constructia si abilitatile aparatului achizitionat de catre utilizator. Aplicatia pune in evidenta o multitudine de functii care definesc comportamentul autonom al dispozitivului meteo, precum functiile de selectare individuala a senzorilor, functia de trimitere a unui email cu datele inregistrate, functia de afisarea datelor intr-o interfata la care utilizatorul sa aibe acces sau functia de analiza a datelor care aduce la cunostiinta utilizatorului numarul de parametrii care au depasit valorile normale. Totodata, aplicatia ilustreaza si o prognoza meteo a zilei in curs cu ajutorul functiilor de alerta integrate.

Capitolul 2 – Analiza si proiectare pe obiecte a sistemului informatic

2.1 Diagrama obiectelor



2.2 Diagrama succesiunii evenimentelor



Capitolul 3 – Implementare si testare a sistemului informatic

3.1 Codul sursa

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<string.h>
#include<assert.h>

class utilizator
{
private: char*id;
static int nr;
public: utilizator();
utilizator(char*);
~utilizator();
static int get_nr();
};
int utilizator::nr=0;
utilizator::utilizator()
{
id=0;
++nr;
}
utilizator::utilizator(char*a)
{
id=new char[strlen(a)+1];
strcpy(id,a);
++nr;
}
```

```
utilizator::~utilizator()
{
delete id;
--nr;
}
int utilizator::get_nr()
{
cout<<"\n Numarul utilizatorilor activi:";
return nr;
}
class panou_control
{
public: panou_control();
~panou_control();
inline void oprire_panou()
{
cout<<"\n Panou de control dezactivat.";
}
inline void masuratoare_noua()
{
cout<<"\n Se initiaza o noua masuratoare...";
}
inline void email()
{
cout<<"\n Se trimite email cu datele inregistrate...";
}
};
panou_control::panou_control()
{
```

```
cout<<"\n Panou de control activat";
}
panou_control::~panou_control()
{}
class aparat
{
public: aparat();
inline void oprire_aparat()
{
cout<<"\n Aparat oprit.";
}
~aparat();
};
aparat::aparat()
{
cout<<"\n Pornire aparat...";
aparat::~aparat()
{}
class senzor
{
public: senzor();
virtual void analiza_date();
virtual float calibrare();
};
senzor::senzor()
{}
void senzor::analiza_date()
{
cout<<"\n Se analizeaza datele...";
}
float senzor::calibrare()
{
```

```
cout<<"\n Aparatul se calibreaza...";
return 0.0;
}
class senzor_presiune:public  senzor
{
private: char*tip_senzor;
float presiune;
public: senzor_presiune();
senzor_presiune(char*,  float);
~senzor_presiune();
void selectare();
void masoara();
void transmite_date();
void analiza_date();
void afisare_date();
float calibrare();
};
senzor_presiune::senzor_presiune()
{
tip_senzor="Senzor presiune";
presiune=0.0;
}
senzor_presiune::senzor_presiune(char*c,  float d)
{
tip_senzor=new char[strlen(c)+1];
strcpy(tip_senzor,c);
presiune=d;
}
senzor_presiune::~senzor_presiune()
{
delete tip_senzor;
}
void senzor_presiune::selectare()
{
```

```
cout<<"\n Senzor presiune selectat";
}
void senzor_presiune::masoara()
{
cout<<"\n Se masoara presiunea...";
}
void senzor_presiune::analiza_date()
{
if (presiune<695)
cout<<"\n ALERTA - Va aflati in zona depresionara!";
else if (presiune>803)
cout<<"\n ALERTA - Anticiclon!";
}
void senzor_presiune::transmite_date()
{
cout<<"\n Se transmit date catre panoul de control...";
}
void senzor_presiune::afisare_date()
{
cout<<"\n Presiune inregistrata:"<<presiune<<"mmHg";
}
float senzor_presiune::calibrare()
{
presiune=0.0;
cout<<"\n Presiune:";
return presiune;
}
class senzor_temperatura:public  senzor
{
private: char*tip_senzor;
float temperatura;
public: senzor_temperatura();
senzor_temperatura(char*, float);
~senzor_temperatura();
```

```
void selectare();
void masoara();
void transmite_date();
void analiza_date();
void afisare_date();
float calibrare();
};
senzor_temperatura::senzor_temperatura()
{
tip_senzor="Senzor temperatura";
temperatura=0.0;
}
senzor_temperatura::senzor_temperatura(char*e,  float f)
{
tip_senzor=new char[strlen(e)+1];
strcpy(tip_senzor,e);
temperatura=f;
}
senzor_temperatura::~senzor_temperatura()
{
delete tip_senzor;
}
void senzor_temperatura::selectare()
{
cout<<"\n Senzor temperatura selectat";
}
void senzor_temperatura::masoara()
{
cout<<"\n Se masoara temperatura...";
}
void senzor_temperatura::transmite_date()
{
cout<<"\n Se transmit date catre panoul de control...";
}
```

```
void senzor_temperatura::analiza_date()
{
if (temperatura>30)
cout<<"\n ALERTA - Canicula!";
else if (temperatura<0)
cout<<"\n ALERTA - Inghet!";
}
void senzor_temperatura::afisare_date()
{
cout<<"\n Temperatura inregistrata:"<<temperatura<<"grade Celsius";
}
float senzor_temperatura::calibrare()
{
temperatura=0.0;
cout<<"\n Temperatura:";
return temperatura;
}
class senzor_precipitatii:public senzor
{
private: char*tip_senzor;
float precipitatii;
public: senzor_precipitatii();
senzor_precipitatii(char*g, float h);
~senzor_precipitatii();
void selectare();
void masoara();
void transmite_date();
void analiza_date();
void afisare_date();
float calibrare();
};
senzor_precipitatii::senzor_precipitatii()
{
tip_senzor="Senzor precipitatii";
```

```
precipitatii=0.0;
}
senzor_precipitatii::senzor_precipitatii(char*i,  float j)
{
tip_senzor=new char[strlen(i)+1];
strcpy(tip_senzor,i);
precipitatii=j;
}
senzor_precipitatii::~senzor_precipitatii()
{
delete tip_senzor;
}
void senzor_precipitatii::selectare()
{
cout<<"\n Senzor precipitatii selectat";
}
void senzor_precipitatii::masoara()
{
cout<<"\n Se masoara nivelul de precipitatii...";
}
void senzor_precipitatii::transmite_date()
{
cout<<"\n Se transmit date catre panoul de control...";
}
void senzor_precipitatii::analiza_date()
{
if (precipitatii>=10.0)
cout<<"\n ALERTA - Inundatie!";
else if (precipitatii>1.0)
cout<<"\n ALERTA - Ploaie!";
}
void senzor_precipitatii::afisare_date()
{
cout<<"\n Precipitatii inregistrate:"<<precipitatii<<"mm";
```

```
}
float senzor_precipitatii::calibrare()
{
precipitatii=0.0;
cout<<"\n Precipitatii:";
return precipitatii;
}
class senzor_lumina:public  senzor
{
private: char*tip_senzor;
float lumina;
public: senzor_lumina();
senzor_lumina(char*,  float);
~senzor_lumina();
void selectare();
void masoara();
void transmite_date();
void afisare_date();
float calibrare();
};
senzor_lumina::senzor_lumina()
{
tip_senzor="Senzor  lumina";
lumina=0.0;
}
senzor_lumina::senzor_lumina(char*k,  float l)
{
tip_senzor=new char[strlen(k)+1];
strcpy(tip_senzor,k);
lumina=l;
}
senzor_lumina::~senzor_lumina()
{
delete tip_senzor;
```

```
}
void senzor_lumina::selectare()
{
cout<<"\n Senzor lumina selectat";
}
void senzor_lumina::masoara()
{
cout<<"\n Se masoara intensitatea luminoasa...";
}
void senzor_lumina::transmite_date()
{
cout<<"\n Se transmit date catre panoul de control...";
}
void senzor_lumina::afisare_date()
{
cout<<"\n Grad luminos inregistrat:"<<lumina<<"lumeni";
}
float senzor_lumina::calibrare()
{
lumina=0.0;
cout<<"\n Lumina:";
return lumina;
}
class senzor_ceata:public senzor
{
private: char*tip_senzor;
float ceata;
public: senzor_ceata();
senzor_ceata(char*, float);
~senzor_ceata();
void selectare();
void masoara();
void transmite_date();
void analiza_date();
```

```
void afisare_date();
float calibrare();
};
senzor_ceata::senzor_ceata()
{
tip_senzor="Senzor ceata";
ceata=0.0;
}
senzor_ceata::senzor_ceata(char*m,  float n)
{
tip_senzor=new char[strlen(m)+1];
strcpy(tip_senzor,m);
ceata=n;
}
senzor_ceata::~senzor_ceata()
{
delete tip_senzor;
}
void senzor_ceata::selectare()
{
cout<<"\n Senzor ceata selectat";
}
void senzor_ceata::masoara()
{
cout<<"\n Se masoara nivelul de ceata...";
}
void senzor_ceata::transmite_date()
{
cout<<"\n Se transmit date catre panoul de control...";
}
void senzor_ceata::analiza_date()
{
if (ceata>=20.0)
cout<<"\n ALERTA - Ceata!";
```

```
}
void senzor_ceata::afisare_date()
{
cout<<"\n Grad de ceata inregistrat:"<<ceata<<"%";
}
float senzor_ceata::calibrare()
{
ceata=0.0;
cout<<"\n Ceata:";
return ceata;
}
class senzor_umiditate:public senzor
{
private: char*tip_senzor;
float umiditate;
public: senzor_umiditate();
senzor_umiditate(char*, float);
~senzor_umiditate();
void selectare();
void masoara();
void transmite_date();
void analiza_date();
void afisare_date();
float calibrare();
};
senzor_umiditate::senzor_umiditate()
{
tip_senzor="Senzor umiditate";
umiditate=0.0;
}
senzor_umiditate::senzor_umiditate(char*o, float p)
{
tip_senzor=new char[strlen(o)+1];
strcpy(tip_senzor,o);
```

```
umiditate=p;
}
senzor_umiditate::~senzor_umiditate()
{
delete tip_senzor;
}
void senzor_umiditate::selectare()
{
cout<<"\n Senzor umiditate selectat";
}
void senzor_umiditate::masoara()
{
cout<<"\n Se masoara umiditatea...";
}
void senzor_umiditate::transmite_date()
{
cout<<"\n Se transmit date catre panoul de control...";
}
void senzor_umiditate::analiza_date()
{
if (umiditate>=50.0)
cout<<"\n ALERTA - Umezeala!";
}
void senzor_umiditate::afisare_date()
{
cout<<"\n Grad umiditate inregistrat:"<<umiditate<<"%";
}
float senzor_umiditate::calibrare()
{
umiditate=0.0;
cout<<"\n Umiditate:";
return umiditate;
}
class monitorizare
```

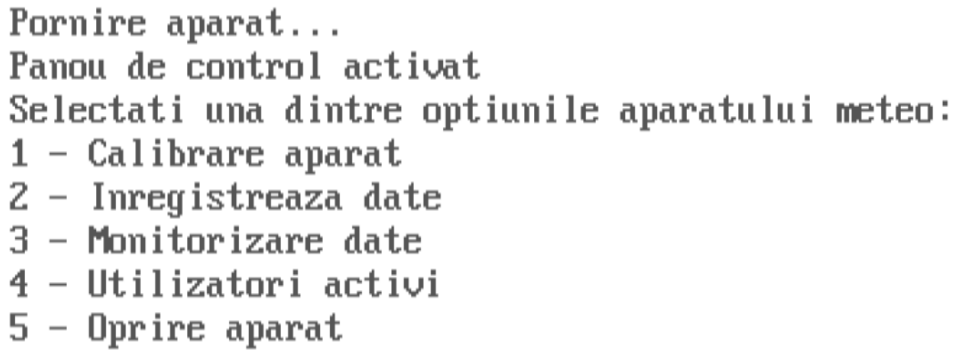
```
{
public: monitorizare();
~monitorizare();
virtual void afisare_date();
};
monitorizare::monitorizare()
{
cout<<"\n Monitorizare date";
}
monitorizare::~monitorizare()
{
cout<<"\n Monitorizare incheiata.";
}
void monitorizare::afisare_date()
{
cout<<"\n Datele inregistrate sunt:";
}
void main()
{
clrscr();
int x;
aparat a0;
panou_control pc0;
cout<<"\n Selectati una dintre optiunile aparatului meteo:";
cout<<"\n 1 - Calibrare aparat";
cout<<"\n 2 - Inregistreaza date";
cout<<"\n 3 - Monitorizare date";
cout<<"\n 4 - Utilizatori activi";
cout<<"\n 5 - Oprire aparat";
cin>>x;
utilizator u1("admin");
senzor_presiune sp0, sp1("Senzor presiune", 780);
senzor_presiune sp2("Senzor presiune", 660);
senzor_presiune sp3("Senzor presiune", 810);
```

```
senzor_temperatura st0, st1("Senzor temperatura", 40);
senzor_temperatura st2("Senzor temperatura", 13);
senzor_temperatura st3("Senzor temperatura", -10);
senzor_precipitatii spr0, spr1("Senzor precipitatii", 1.4);
senzor_precipitatii spr2("Senzor precipitatii", 11);
senzor_precipitatii spr3("Senzor precipitatii", 6);
senzor_lumina sl0, sl1("Senzor lumina", 70);
senzor_lumina sl2("Senzor lumina", 100);
senzor_lumina sl3("Senzor lumina", 150);
senzor_ceata sc0, sc1("Senzor ceata", 25);
senzor_ceata sc2("Senzor ceata", 14);
senzor_ceata sc3("Senzor ceata", 50);
senzor_umiditate su0, su1("Senzor umiditate", 15);
senzor_umiditate su2("Senzor umiditate", 12);
senzor_umiditate su3("Senzor umiditate", 52);
senzor s0;
switch(x)
{
case 1: senzor*ptr;
ptr=&s0;
cout<<ptr->calibrare();
ptr=&sp0;
cout<<ptr->calibrare();
ptr=&st0;
cout<<ptr->calibrare();
ptr=&spr0;
cout<<ptr->calibrare();
ptr=&sl0;
cout<<ptr->calibrare();
ptr=&sc0;
cout<<ptr->calibrare();
ptr=&su0;
cout<<ptr->calibrare();
break;
```

```
case 2: pc0.masuratoare_noua();
sp1.masoara();
sp1.transmite_date();
sp1.afisare_date();
st3.masoara();
st3.afisare_date();
spr1.masoara();
spr1.afisare_date();
sl1.masoara();
sl1.afisare_date();
sc2.masoara();
sc2.afisare_date();
su1.masoara();
su1.afisare_date();
senzor*ptrx;
ptrx=&s0;
ptrx->analiza_date();
ptrx=&sp1; ptrx->analiza_date();
ptrx=&st3; ptrx->analiza_date();
ptrx=&spr1; ptrx->analiza_date();
ptrx=&sl1; ptrx->analiza_date();
ptrx=&sc2; ptrx->analiza_date();
ptrx=&su1; ptrx->analiza_date();
break;
case 3: cout<<"\n Ultimele date inregistrate:";
sp2.afisare_date();
st2.afisare_date();
spr3.afisare_date();
sl2.afisare_date();
sc3.afisare_date();
su3.afisare_date();
senzor*ptrz;
ptrz=&s0;
ptrz->analiza_date();
```

```
ptrz=&sp2; ptrz->analiza_date();
ptrz=&st2; ptrz->analiza_date();
ptrz=&spr3; ptrz->analiza_date();
ptrz=&sl2; ptrz->analiza_date();
ptrz=&sc3; ptrz->analiza_date();
ptrz=&su3; ptrz->analiza_date();
pc0.email();
break;
case 4: cout<<utilizator::get_nr();
break;
case 5: pc0.oprire_panou();
a0.oprire_aparat();
break;
default: cout<<"\n Selectati o alta optiune"; }
cout<<"\n Press ENTER key to continue...";
getchar(); }
```

3.2 Seturi de date de test – rezultate si erori



Dupa cum se poate observa, programul functioneaza fara erori pana la etapa de selectare a uneia dintre functiile aparatului.

Datele de intrare ale programului au fost introduse in cod, mai exact s-au creat cate trei obiecte pentru fiecare clasa prevazuta, cu valori diferite, care vor testa toate functiile programului.

File Edit Search Run Compile Debug Project Options Window

PROJECT.CPP

```
utilizator u1("admin");
senzor_presiune sp0, sp1("Senzor presiune", 780);
senzor_presiune sp2("Senzor presiune", 660);
senzor_presiune sp3("Senzor presiune", 810);
senzor_temperatura st0, st1("Senzor temperatura", 40);
senzor_temperatura st2("Senzor temperatura", 13);
senzor_temperatura st3("Senzor temperatura", -10);
senzor_precipitatii spr0, spr1("Senzor precipitatii", 1.4);
senzor_precipitatii spr2("Senzor precipitatii", 11);
senzor_precipitatii spr3("Senzor precipitatii", 6);
senzor_lumina sl0, sl1("Senzor lumina", 70);
senzor_lumina sl2("Senzor lumina", 100);
senzor_lumina sl3("Senzor lumina", 150);
senzor_ceata sc0, sc1("Senzor ceata", 25);
senzor_ceata sc2("Senzor ceata", 14);
senzor_ceata sc3("Senzor ceata", 50);
senzor_umiditate su0, su1("Senzor umiditate", 15);
senzor_umiditate su2("Senzor umiditate", 12);
senzor_umiditate su3("Senzor umiditate", 52);
senzor s0;
```

```
Pornire aparat...
Panou de control activat
Selectati una dintre optiunile aparatului meteo:
1 - Calibrare aparat
2 - Inregistreaza date
3 - Monitorizare date
4 - Utilizatori activi
5 - Oprire aparat1

Aparatul se calibreaza...0
Presiune:0
Temperatura:0
Precipitatii:0
Lumina:0
Ceata:0
Umiditate:0
Press ENTER key to continue...
```

```
1 - Calibrare aparat
2 - Inregistreaza date
3 - Monitorizare date
4 - Utilizatori activi
5 - Oprire aparat2

Se initiaza o noua masuratoare...
Se masoara presiunea...
Se transmit date catre panoul de control...
Presiune inregistrata:780mmHg
Se masoara temperatura...
Temperatura inregistrata:-10grade Celsius
Se masoara nivelul de precipitatii...
Precipitatii inregistrate:1.4mm
Se masoara intensitatea luminoasa...
Grad luminos inregistrat:70lumeni
Se masoara nivelul de ceata...
Grad de ceata inregistrat:14%
Se masoara umiditatea...
Grad umiditate inregistrat:15%
Se analizeaza datele...
ALERTA - Inghet!
ALERTA - Ploaie!
Se analizeaza datele...
Press ENTER key to continue...
```

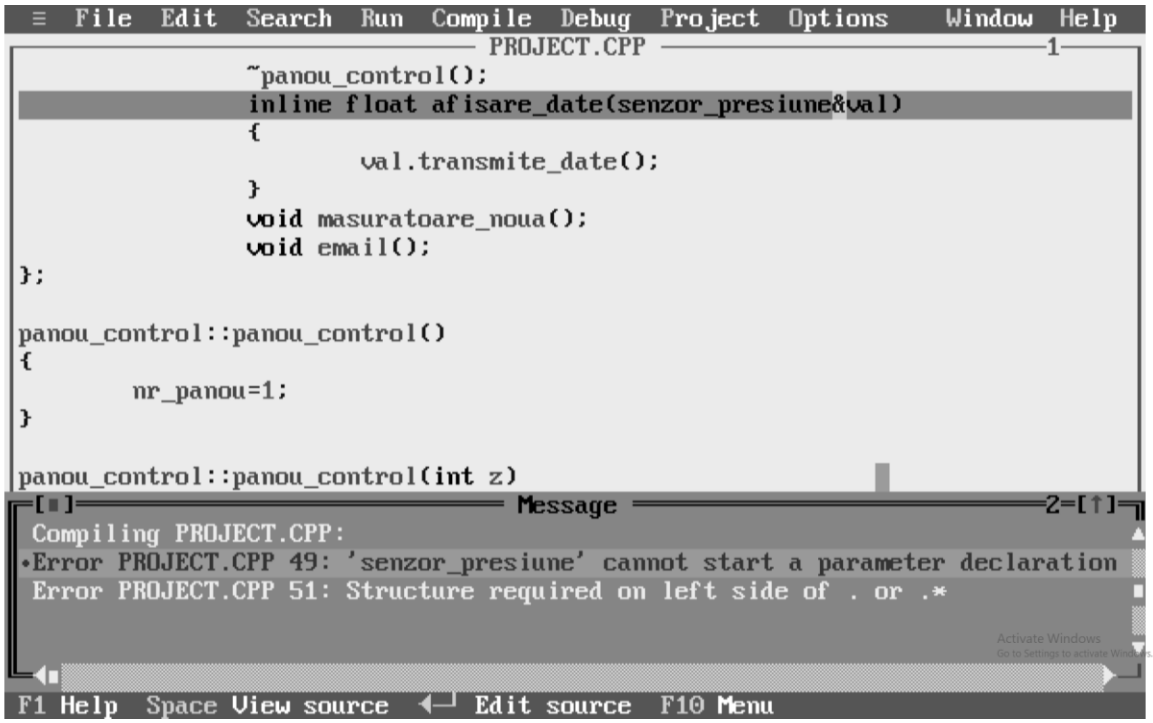
```
1 - Calibrare aparat
2 - Inregistreaza date
3 - Monitorizare date
4 - Utilizatori activi
5 - Oprire aparat2

Pornire aparat...
Panou de control activat
Selectati una dintre optiunile aparatului meteo:
1 - Calibrare aparat
2 - Inregistreaza date
3 - Monitorizare date
4 - Utilizatori activi
5 - Oprire aparat3

Ultimele date inregistrate:
Presiune inregistrata:660mmHg
Temperatura inregistrata:13grade Celsius
Precipitatii inregistrate:6mm
Grad luminos inregistrat:100lumeni
Grad de ceata inregistrat:50%
Grad umiditate inregistrat:52%
Se analizeaza datele...
ALERTA - Va aflati in zona depresionara!
ALERTA - Ploaie!
```

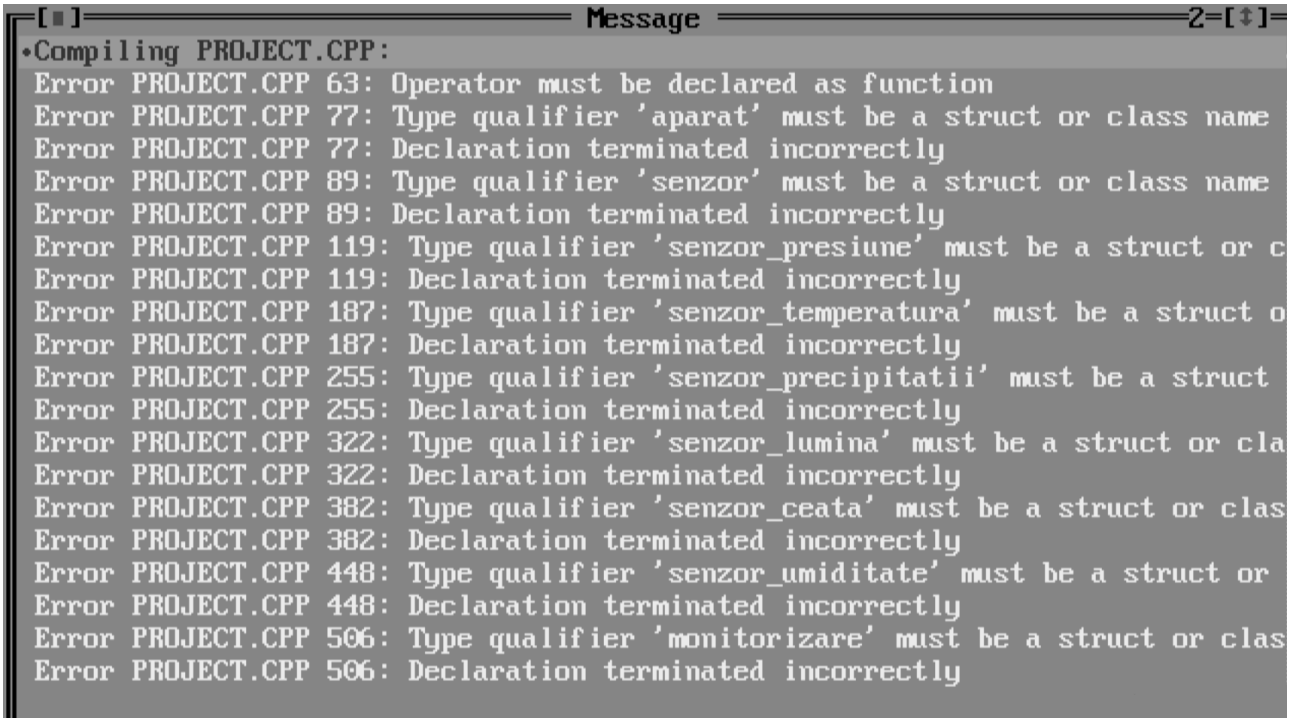
```
Pornire aparat...
Panou de control activat
Selectati una dintre optiunile aparatului meteo:
1 - Calibrare aparat
2 - Inregistreaza date
3 - Monitorizare date
4 - Utilizatori activi
5 - Oprire aparat4

Numarul utilizatorilor activi:1
Press ENTER key to continue...
```

```
Pornire aparat...
Panou de control activat
Selectati una dintre optiunile aparatului meteo:
1 - Calibrare aparat
2 - Inregistreaza date
3 - Monitorizare date
4 - Utilizatori activi
5 - Oprire aparat5

Panou de control dezactivat.
Aparat oprit.
Press ENTER key to continue...
```

Au existat erori la implementarea aplicatiei, astfel:

- Erori de sintaxa si de utilizare a argumentelor in cod;
- Erori la apelul functiilor virtuale dar si a celorlalte tipuri de functii;
- Erori la nivelul implementarii structurilor de comparare a datelor, intrucat nu s-a reusit o utilizare corecta a referintelor;
- Erori la ordinea de scriere a instructiunilor atunci cand acestea erau mai multe;
- Dificultati de parcurgere a codului dupa depasirea unui anumit numar de linii de cod;


File Edit Search Run Compile Debug Project Options Window Help
PROJECT.CPP
1
~panou_control();
inline float afisare_date(senzor_presiune&val)
{
val.transmite_date();
}
void masuratoare_noua();
void email();
};
panou_control::panou_control()
{
nr_panou=1;
}
panou_control::panou_control(int z)
Message
2=[↑]
Compiling PROJECT.CPP:
•Error PROJECT.CPP 49: 'senzor_presiune' cannot start a parameter declaration
Error PROJECT.CPP 51: Structure required on left side of . or .*
F1 Help Space View source Edit source F10 Menu


Message
2=[↕]
•Compiling PROJECT.CPP:
Error PROJECT.CPP 63: Operator must be declared as function
Error PROJECT.CPP 77: Type qualifier 'aparat' must be a struct or class name
Error PROJECT.CPP 77: Declaration terminated incorrectly
Error PROJECT.CPP 89: Type qualifier 'senzor' must be a struct or class name
Error PROJECT.CPP 89: Declaration terminated incorrectly
Error PROJECT.CPP 119: Type qualifier 'senzor_presiune' must be a struct or c
Error PROJECT.CPP 119: Declaration terminated incorrectly
Error PROJECT.CPP 187: Type qualifier 'senzor_temperatura' must be a struct o
Error PROJECT.CPP 187: Declaration terminated incorrectly
Error PROJECT.CPP 255: Type qualifier 'senzor_precipitatii' must be a struct
Error PROJECT.CPP 255: Declaration terminated incorrectly
Error PROJECT.CPP 322: Type qualifier 'senzor_lumina' must be a struct or cla
Error PROJECT.CPP 322: Declaration terminated incorrectly
Error PROJECT.CPP 382: Type qualifier 'senzor_ceata' must be a struct or clas
Error PROJECT.CPP 382: Declaration terminated incorrectly
Error PROJECT.CPP 448: Type qualifier 'senzor_umiditate' must be a struct or
Error PROJECT.CPP 448: Declaration terminated incorrectly
Error PROJECT.CPP 506: Type qualifier 'monitorizare' must be a struct or clas
Error PROJECT.CPP 506: Declaration terminated incorrectly

Capitolul 4 – Concluzii

In concluzie, aplicatia pentru coordonarea unui aparat meteo autonom functioneaza suficient de bine incat sa reprezinte un instrument usor de folosit, atat utilizatorului, cat si ciclului de viata al aparatului fizic. Majoritatea erorilor din aplicatie au fost remediate, iar viitoarele tipuri de erori care pot sa apara sunt rezolvabile in aceeasi maniera.

In cadrul aplicatiei, se pot schimba si interschimba oricand valorile atributelor obiectelor, intrucat acestea nu reprezinta masuratori reale, ci valori aproximative datelor reale.

Cu toate acestea, aplicatia reprezinta o simulare apropiata de realitate a unui aparat meteo autonom aflat pe piata la momentul actual, si poate oricand sa devina baza dezvoltarii unui software care sa aiba si o parte electronica atasata acestuia. Codul este usor accesibil si se poate interveni oricand pentru a fi modificat, imbunatatit sau fixat, iar limbajul de programare este unul de baza si usor de abordat.

Bibliografie

- ✓ ro.wikipedia.org
- ✓ meteoromania.ro
- ✓ cantemir.ro
- ✓ ro.scribd.com
- ✓ lege5.ro
- ✓ scrigroup.com