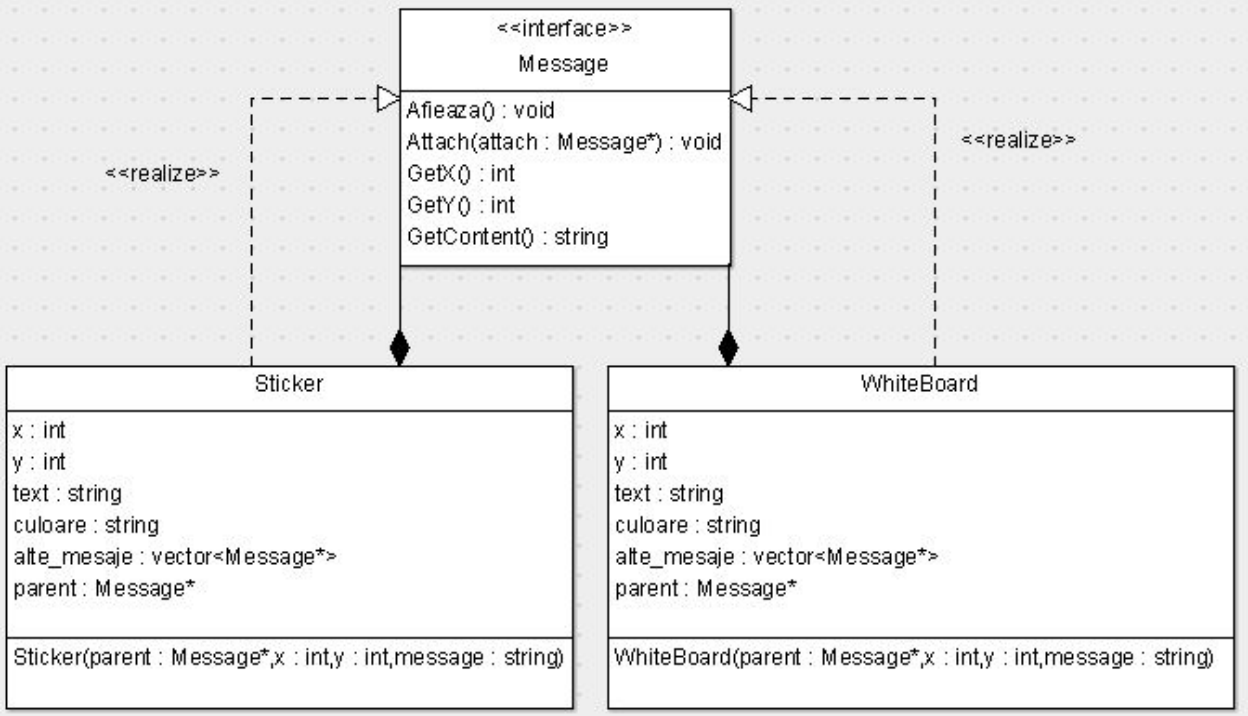
Problema 1 (VDC)

Mind-Maps



Realizati fisierele in C++ corespunzatoare diagramei UML si completati functiile a.i. codul din functia main sa functioneze.

```
int main(void)
{
	Message *wall = new WhiteBoard(0,0,0,"MindMap","violet");
	Message *stick1 = new Sticker(wall, 10, 10, "Vacanta", “rosu");
	Message *stick2 = new Sticker(wall, 10, 50, "Munte", "albastru");
	Message *stick3 = new Sticker(wall, 50, 10, "Tabara", "roz");
	Message *stick4 = new Sticker(wall, 50, 50, "Examen PA", "rosu inchis");
	Message *stick5 = new Sticker(wall, 150, 10, "Examen POO", "maro");
	Message *stick6 = new Sticker(wall, 100, 10, "Familie", "galben");
	Message *stick7 = new Sticker(stick6, 20, 30, "Iesire la Ciric", "verde");
	Message *stick8 = new Sticker(stick7, 20, 20, “Gratar", "rosu aprins");

	wall->Attach(stick1);
	wall->Attach(stick2);
	wall->Attach(stick3);
	wall->Attach(stick4);
	wall->Attach(stick5);
	wall->Attach(stick6);
```

```
        stick6->Attach(stick7);
        stick7->Attach(stick8);

        wall->Afiseaza();

        return 0;
}
```

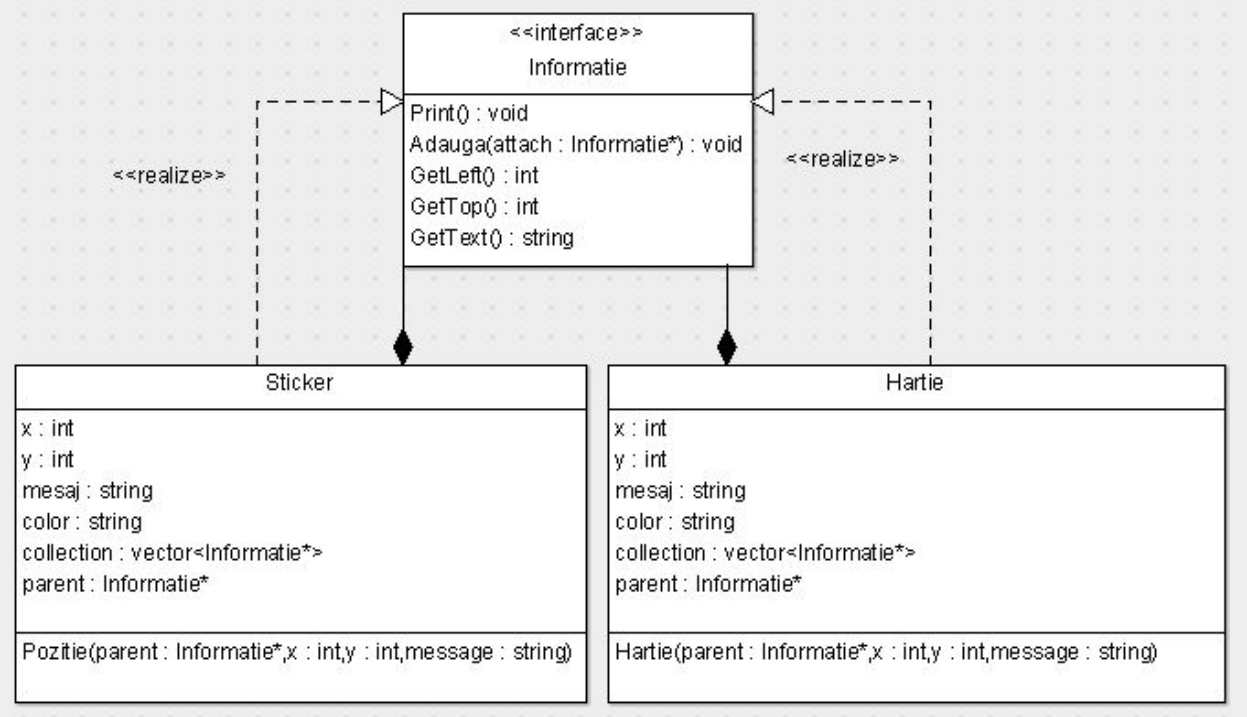
Specificatii:

- Metodele GetX() si GetY() vor intoarce coordonatele absolute ale obiectelor (relative la tabla)
- Metoda Attach((atasament) va inregistra pentru obiectul curent, noul obiect
- Metoda Afiseaza va afisa detaliile obiectului primit ca parametru (orice element copil va fi afisat impreuna cu toate detaliile - <content, culoare, pozitie absoluta, numar atasamente, content parinte>)

Punctaj:
Scrierea corecta a headerelor(ului) cu include-urile necesare si alte elemente lipsa – 4p
Daca in final programul compileaza cu success – 2p
Metodele GetX(), GetY(), GetContent() – cate 0.5p fiecare – 2 x (0.5 x 3) - 3p
Metodele Afiseaza() si Attach() cate 1.5p fiecare – 2 x (2 x 1.5) - 6p

Problema 2 (VDC)

Planificare



Realizati fisierele in C++ corespunzatoare diagramei UML si completati functiile a.i. codul din functia main sa functioneze.

```
int main(void)
{
	Informatie*tabla = new Hartie(0,0,0,"Plan","alb");
	Informatie*t1 = new Pozitie(tabla, 10, 10, "Text1", "portocaliu");
	Informatie*he = new Pozitie(tabla, 10, 50, "Hello", "albastru");
	Informatie*ms = new Pozitie(tabla, 50, 10, "Mesaj", "roz");
	Informatie*ia = new Pozitie(tabla, 50, 50, "Iasi", "rosu inchis");
	Informatie*va = new Pozitie(tabla, 150, 10, "Vacanta", "maro");
	Informatie*ua = new Pozitie(tabla, 100, 10, "UAIC", "galben");
	Informatie*ex = new Pozitie(ua, 20, 30, "Examen", "verde");
	Informatie*po = new Pozitie(ex, 20, 20, "POO", "rosu aprins");

	tabla->Adauga(t1);
	tabla->Adauga(he);
	tabla->Adauga(ms);
	ua->Adauga(ex);
	ex->Adauga(po);
	tabla->Adauga(ua);
```

```
    tabla->Adauga(ia);
    tabla->Adauga(va);

    tabla->Print();

    return 0;
}
```

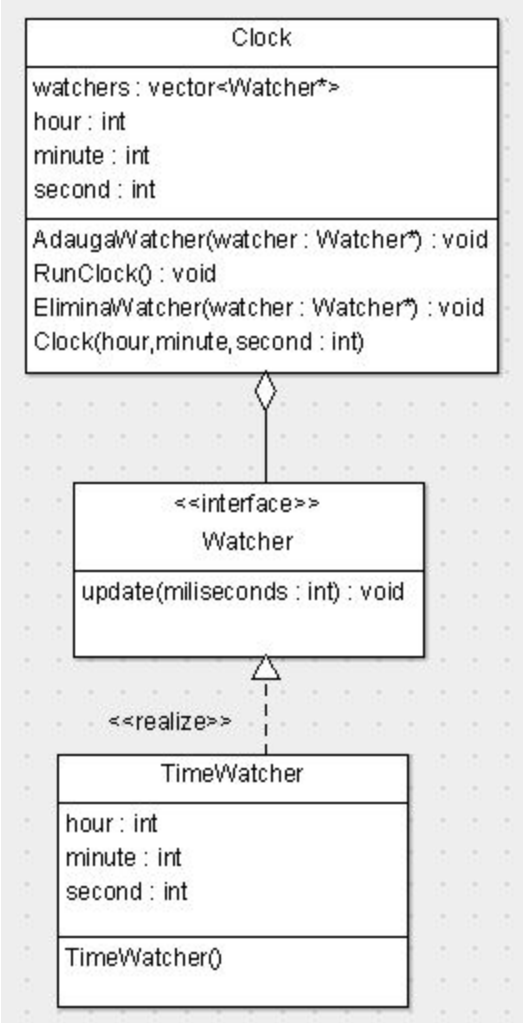
Specificatii:

- Metodele GetLeft() si GetTop() vor intoarce coordonatele absolute ale obiectelor (relative la tabla)
- Metoda Adauga(atasament) va adauga obiectului curent, obiectul primit ca parametru
- Metoda Print va afisa detaliile de pe obiectul primit ca parametru (daca acesta are atasamente, atunci si ele vor fi afisate impreuna cu toate detaliile - <nume, culoare, pozitie absoluta, numar atasamente, cine este obiectul parinte>)

Punctaj:
Scrierea corecta a headerelor(ului) cu include-urile necesare si alte elemente lipsa – 4p
Daca in final programul compileaza cu success – 2p
Metodele GetLeft(), GetTop(), GetText() – cate 0.5p fiecare – 2 x (0.5 x 3) - 3p
Metodele Print() si Adauga() cate 1.5p fiecare – 2 x (2 x 1.5) - 6p

Problema 3 (VDC)

Clock-Watcher



Realizati fisierele in C++ corespunzatoare diagramei UML si completati functiile a.i. codul din functia main sa functioneze. Atentie la specificatii!

```
int main(void)
{
	Watcher *ck = new TimeWatcher();
	Clock ceas(16,34,51);
	ceas.AdaugaWatcher(ck);
	ceas.RunClock();
	return 0;
}
```

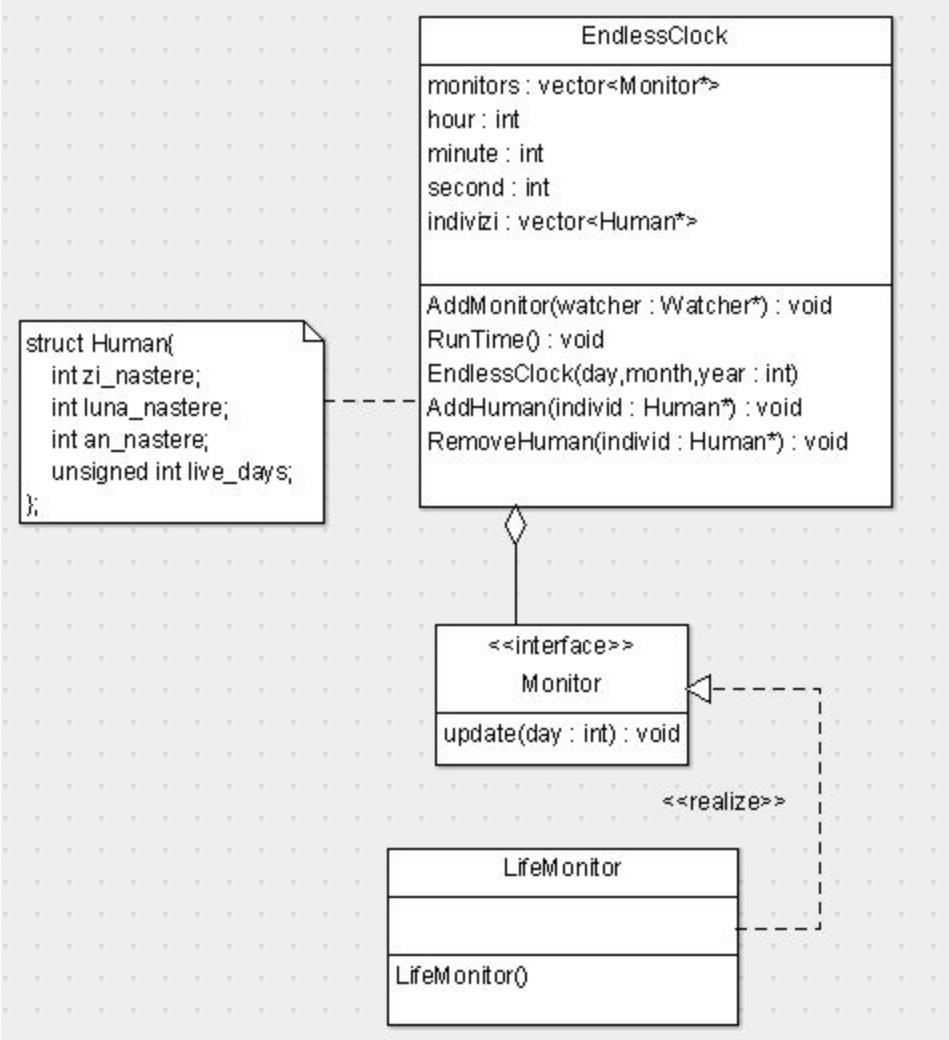
Specificatii:

- Metodele Adauga() si Elimina() vor modifica elementele din watchers
- Constructorul Clock() are ca parametri un timp de pornire (ora, minut, secunda)
- Metoda Run(), pornind de la timpul curent, va calcula milisecunda de pornire, apoi intr-o bucla infinita va actualiza milisecunda la fiecare iteratie, apeland Sleep(1). La fiecare secunda trecuta, se apeleaza si watcherul. Pentru a folosi Sleep(), includeti <windows.h>
- Metoda update() din TimeWatcher va afisa ora curenta la consola, in formatul Ora:Minut:Secunda

Punctaj:
Scrierea corecta a headerelor(ului) cu include-urile necesare si alte elemente lipsa – 2.5p
Daca in final programul compileaza cu success – 2.5p
Metodele AdaugaWatcher(), EliminaWatcher() – cate 1p fiecare – 2 x 1 - 2p
2 constructori (pentru clasele Clock si TimeWatcher) – cate 1p fiecare – 1 x 2 - 2p
1 metode update() - 3p
1 metoda RunClock() - 3p

Problema 4 (VDC)

LifeTime-monitor



Realizati fisierele in C++ corespunzatoare diagramei UML si completati functiile a.i. codul din functia main sa functioneze. Atentie la specificatii!

```
int main(void)
{
	Monitor *m = new LifeMonitor();
	EndlessClock chronos(0,0,0);

	Human decebal={11,7,30,27740};
	Human constantin_cel_mare={27,2,272,23725};
	Human stefan_cel_mare={2,7,1433,25915};
	Human milarepa={4,10,1052,28109};
	Human columb={21,9,1451,26774};
	Human eminescu={15,1,1850,14235};
```

```
        Human einstein={14,3,1879,27740};
        Human chuck_norris={1,1,0,2999999999};

        chronos.AddMonitor(m);
        chronos.AddHuman(decebal);
        chronos.AddHuman(constantin_cel_mare);
        chronos.AddHuman(stefan_cel_mare);
        chronos.AddHuman(milarepa);
        chronos.AddHuman(columb);
        chronos.AddHuman(eminescu);
        chronos.AddHuman(einstein);
        chronos.AddHuman(chuck_norris);

        chronos.RunTime();
        return 0;
}
```

Specificatii:

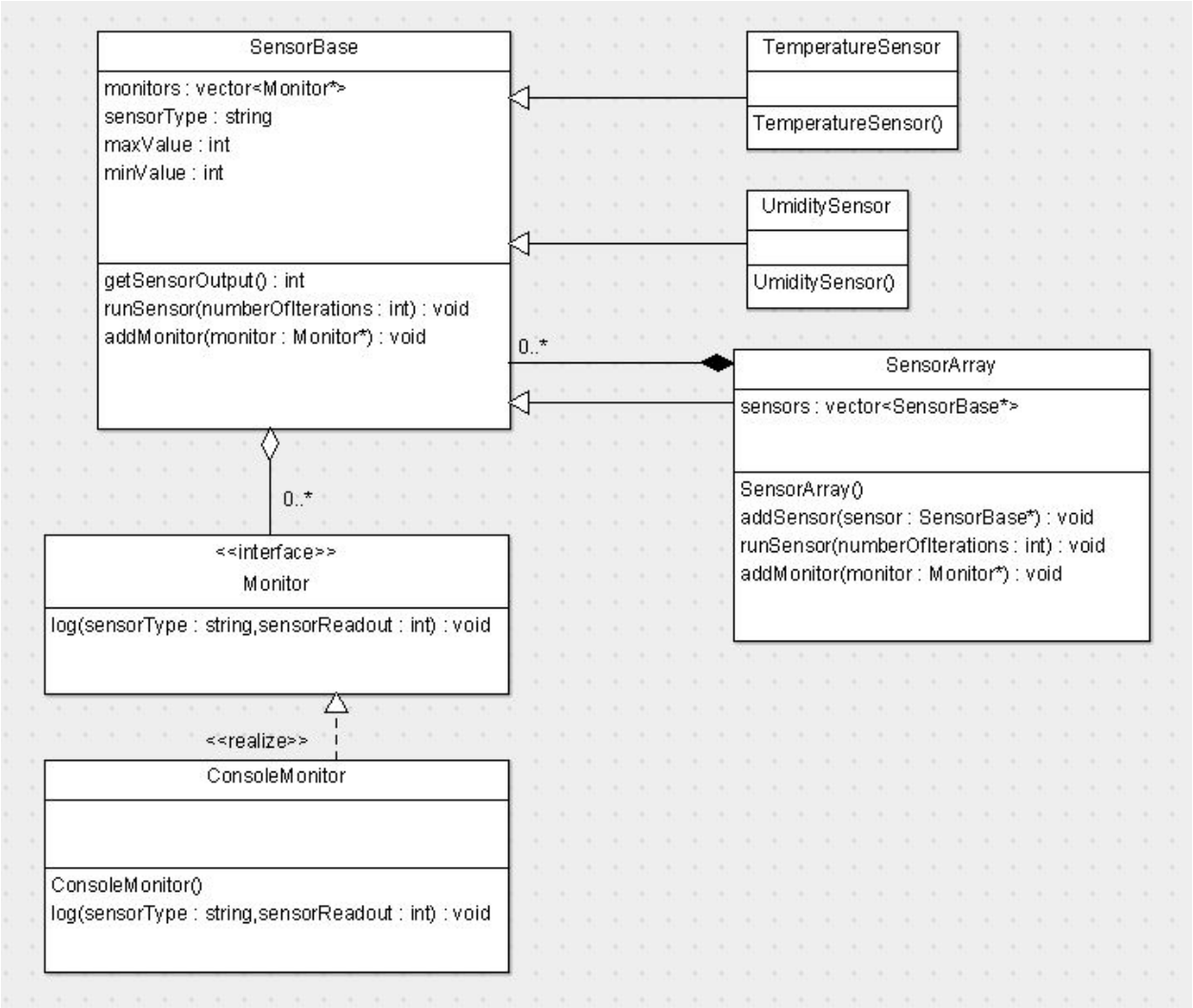
- Metoda AddMonitor() adauga un monitor nou in vectorul monitors
- Metodele AddHuman() si RemoveHuman() adauga/elimina oameni din vectorul indivizi
- Constructorul EndlessClock() are ca parametri o data de pornire (zi, luna, an)
- Metoda RunTime(), pornind de la timpul curent, va actualiza ziua la fiecare iteratie intr-o bucla care proceseaza timpul scurs pe o perioada de 3000 de ani. In fiecare zi noua (iteratie), se apeleaza metoda update.
- Metoda update() din LifeMonitor va afisa cand se naste sau moare o persoana. Cand moare, ea este eliminata din vectorul indivizi.

Punctaj:

Scrierea corecta a headerelor(ului) cu include-urile necesare si alte elemente lipsa – 2.5p
Daca in final programul compileaza cu success – 2.5p
Metodele AddMonitor(), AddHuman(), RemoveHuman() – cate 0.5p fiecare – 3 x 1 - 3p
2 constructori (pentru clasele EndlessClock si LifeMonitor – cate 1p fiecare – 1 x 2 - 2p
1 metoda update() - 2p
1 metoda RunTime() - 3p

Problema 5 (CIA)

SensorPlay



Realizati fisierele in C++ corespunzatoare diagramei UML si completati functiile a.i. codul din functia main sa functioneze. Atentie la specificatii!

```
int main()
{
	TemperatureSensor t1, t2;
	UmiditySensor u1, u2;

	SensorArray arr;
	arr.addSensor(&t1);
	arr.addSensor(&u1);

	ConsoleMonitor cm;
```

```
    arr.addMonitor(&cm);
    arr.runSensor(1);

    arr.addSensor(&t2);
    arr.runSensor(2);

    return 0;
}
```

Output:

Console:

Temperature: <nr>
Umidity: <nr>
Temperature: <nr>
Temperature: <nr>
Umidity: <nr>
Umidity: <nr>
Temperature: <nr>
Temperature: <nr>

Specificatii:

- Pentru metoda getSensorOutput, se va folosi functia “rand” din libraria “cstdlib” pentru a genera un output intre limitele stocate in variabilele maxValue si minValue. Exemplu: rand()%(maxValue - minValue)+minValue;
- Interfata Monitor si implementarile acesteia (ConsoleMonitor) reprezinta Observeri. Instantele se vor inregistra la un senzor.
- Clasa SensorArray va gazdui o serie de senzori, si va actiona ca si un senzor la randul ei. Fiecare observator adaugat va fi pasat mai departe la fiecare dintre senzorii pe care clasa ii inglobeaza. De asemenea, daca, dupa inregistrarea mai multor observeri, noi senzori sunt adaugati, si noii senzori vor beneficia de observerii deja prezenti.
- La rularea senzorului de tip SensorArray, acesta va rula toti senzorii pe care ii inglobeaza, in ordine.
- Metoda runSensor are rolul de a obtine multiple output-uri de la senzor, in functie de parametrul primit. De asemenea, aceasta metoda va lua in calcul si prezenta observerilor.
- Functia “log” de la fiecare Monitor va scrie rezultatele primite ca si parametrii intr-un anumit mediu (consola).
- String-ul sensorType va contine un string ce descrie tipul senzorului.

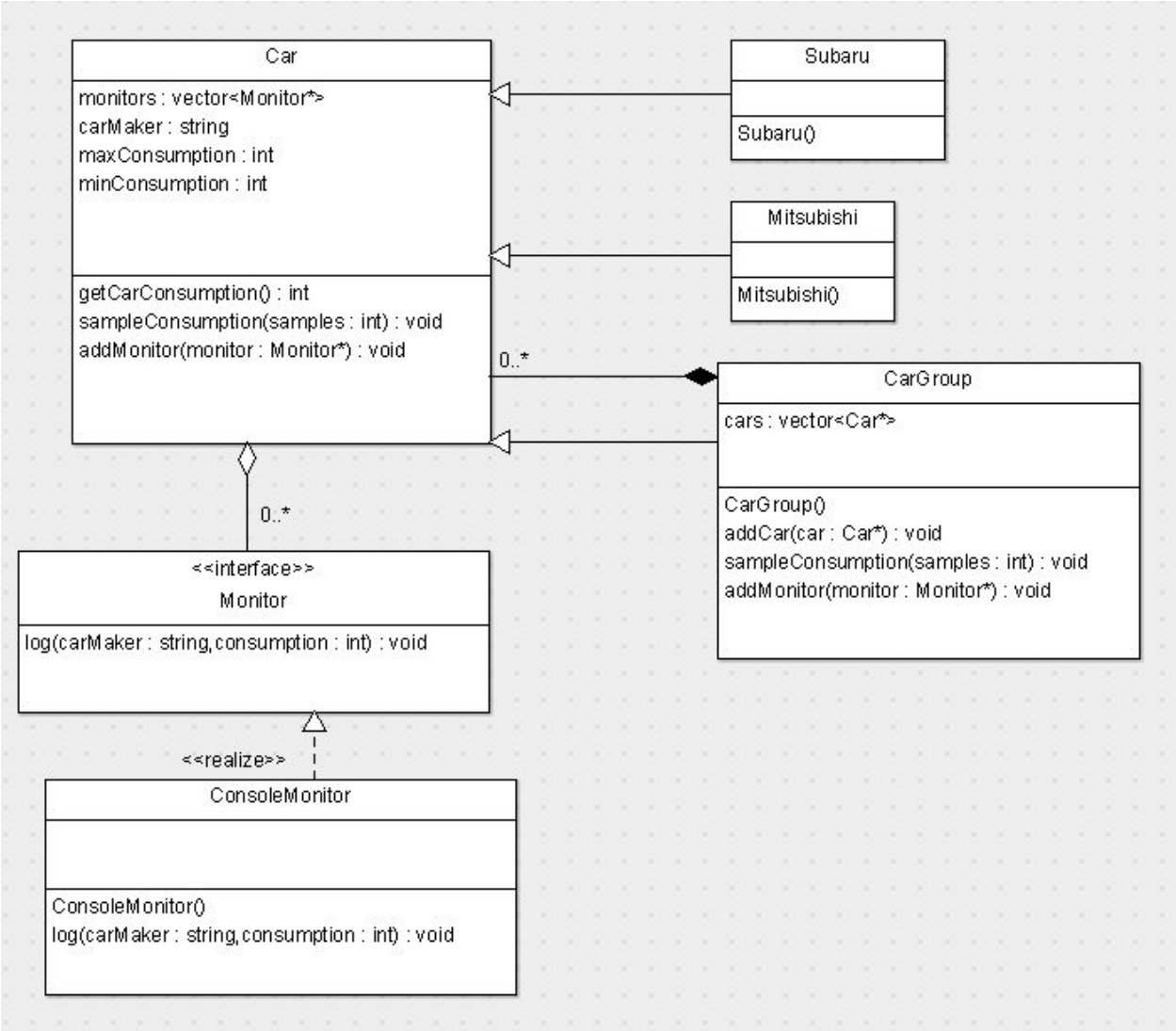
Punctaj:

- Implementarea Monitorului = 3 pct
- Implementarea constructorilor de la senzori (temperatura, umiditate, array) - 0.5 x 3 = 1.5 pct
- Implementarea functiilor getSensorOutput, runSensor, addMonitor aferente clasei SensorBase - 1.5 x 3 = 4.5 pct

- Implementarea functiilor addSensor, runSensor, addMonitor aferente clasei SensorArray - 2 x 3 = 6 pct

Problema 6 (CIA)

CarPlay



Realizati fisierele in C++ corespunzatoare diagramei UML si completati functiile a.i. codul din functia main sa functioneze. Atentie la specificatii!

```
int main()
{
        Subaru s1, s2;
        Mitsubishi m1, m2;
```

```
    CarGroup group;
    group.addCar(&s1);
    group.addCar(&m1);

    ConsoleMonitor cm;
    FileMonitor fm("cars.txt");

    group.addMonitor(&cm);
    group.sampleConsumption(1);

    group.addCar(&s2);
    group.addMonitor(&fm);

    group.sampleConsumption(2);

    return 0;
}
```

Output:

Console:

Subaru: <nr> l/km
Mitsubishi: <nr> l/km
Subaru: <nr> l/km
Subaru: <nr> l/km
Mitsubishi: <nr> l/km
Mitsubishi: <nr> l/km
Subaru: <nr> l/km
Subaru: <nr> l/km

Specificatii:

- Pentru metoda getCarConsumption, se va folosi functia “rand” din libraria “cstdlib” pentru a genera un output intre limitele stocate in variabilele maxConsumption si minConsumption Exemplu: rand()%(maxConsumption - minConsumption)+minConsumption;
- Interfata Monitor si implementarile acesteia (ConsoleMonitor) reprezinta Observeri. Instantele se vor inregistra la un senzor.
- Clasa CarGroup va gazdui o serie de masini, si va actiona ca un tot unitar (poate fi tratata ca si o masina). Fiecare observator adaugat va fi pasat mai departe la fiecare dintre masinile pe care clasa le inglobeaza. De asemenea, daca, dupa inregistrarea mai multor observeri, noi masini sunt adaugate, si noile masini vor beneficia de observerii deja prezenti.

- La interogarea consumului la o clasa de tip CarGroup, aceasta va interoga toate masinile pe care le inglobeaza, in ordine.
- Metoda sampleConsumption are rolul de a obtine multiple valori de la masina vizand consumul, in functie de parametrul primit. De asemenea, aceasta metoda va lua in calcul si prezenta observerilor.
- Functia “log” de la fiecare Monitor va scrie rezultatele primite ca si parametrii intr-un anumit mediu (consola).
- String-ul carMaker va contine un string ce descrie producatorul masinii.

Punctaj:

- Implementarea Monitorului = 3 pct
- Implementarea constructorilor de la masini si grup(Subaru, Mitsubishi, CarGroup) - 0.5 x 3 = 1.5 pct
- Implementarea functiilor getCarConsumption, sampleConsumption, addMonitor aferente clasei Car - 1.5 x 3 = 4.5 pct
- Implementarea functiilor addCar, sampleConsumption, addMonitor aferente clasei CarGroup - 2 x 3 = 6 pct