Subiect 1

Să se implementeze ierarhia de clase din diagrama UML anexată.

Cerințe (+10p):

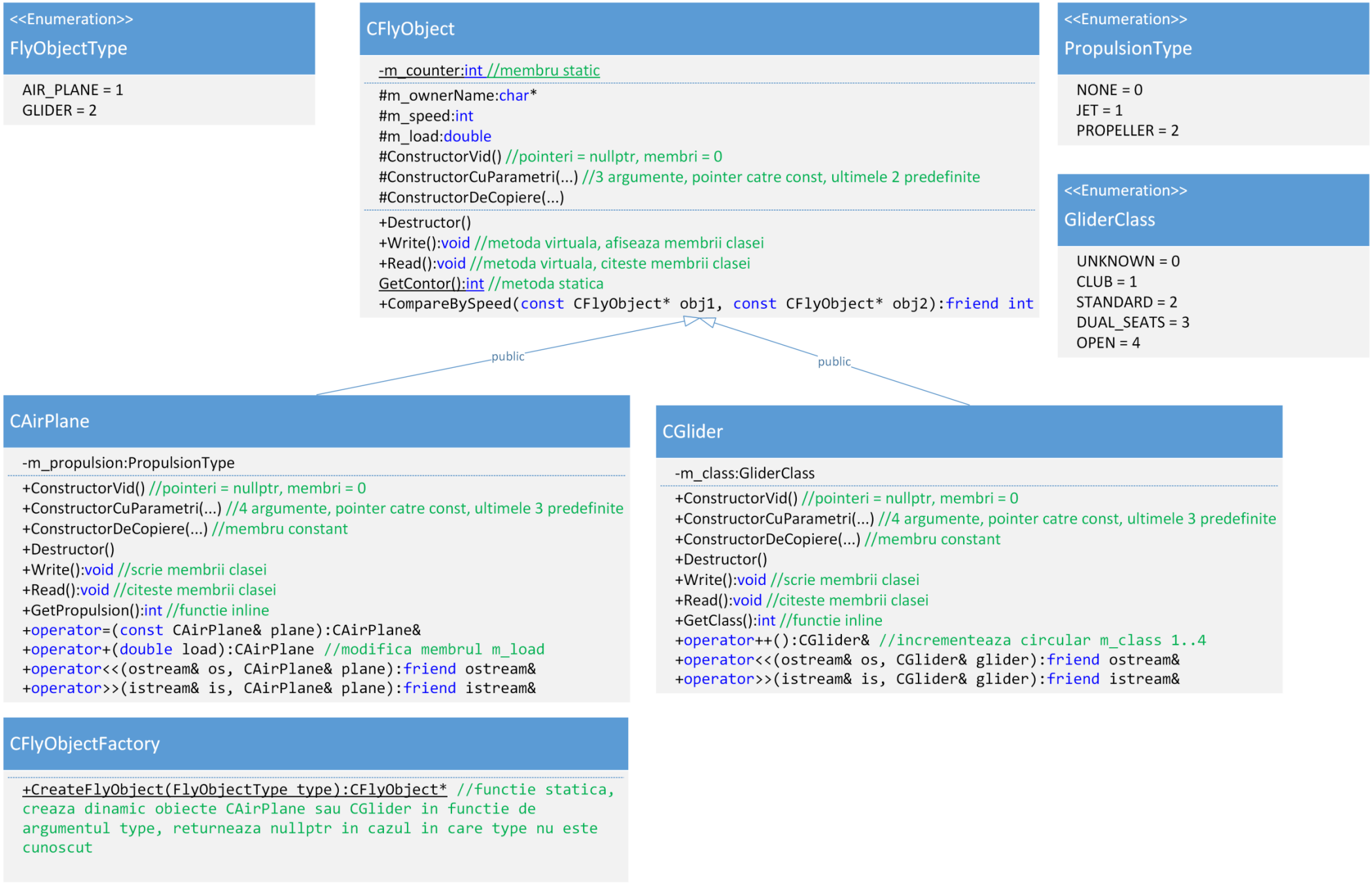
- instanțiați obiecte de tipul claselor derivate astfel încât să puneți în evidență apelul fiecărui constructor pentru fiecare clasă în parte **+ 5p**.
- afișati numărul instanțelor **+ 0.5p**.
- puneți în evidență apelul fiecărei funcții operator, inclusiv `operator<<` și `operator>>` **+ 1.5p**.
- în funcția `main()` alocați dinamic memorie pentru un vector de pointeri către clasa de bază.
    - parcurgeți vectorul și creați dinamic obiecte cu ajutorul funcției statice a clasei `Factory` **+ 0.5p**.
    - parcurgeți vectorul și citiți membrii obiectelor create (polimorfism) **+ 0.5p**
    - parcurgeți vectorul și afișați valorile membrilor obiectelor (polimorfism) **+ 0.5p**
    - sortați elementele vectorului folosind funcția prietenă `CompareBySpeed`(…) și reafișați vectorul **+ 1p**.
    - eliberați memoria **+ 0.5p** .

Constrângeri (-1p):

a) fișierele header vor conține **doar declarații** de clase, funcții, tipuri de date, constante etc.
b) toate funcțiile se vor **defini** numai în fișiere .cpp

Indicații:

a) folosiți `strcpy_s(char *dest, rsize_t sizeInBytes, const char *src);` pentru copierea unui șir, unde `sizeInBytes` este dimensiunea în octeți a șirului destinație
b) reutilizați constructorii din clasa de bază
c) reutilizați funcții din clasa de bază


<<Enumeration>>
FlyObjectType
AIR_PLANE = 1
GLIDER = 2
CFlyObject
-m_counter:int //membru static
#m_ownerName:char*
#m_speed:int
#m_load:double
#ConstructorVid() //pointeri = nullptr, membri = 0
#ConstructorCuParametri(...) //3 argumente, pointer catre const, ultimele 2 predefinite
#ConstructorDeCopiere(...)
+Destructor()
+Write():void //metoda virtuala, afiseaza membrii clasei
+Read():void //metoda virtuala, citeste membrii clasei
GetContor():int //metoda statica
+CompareBySpeed(const CFlyObject* obj1, const CFlyObject* obj2):friend int
<<Enumeration>>
PropulsionType
NONE = 0
JET = 1
PROPELLER = 2
<<Enumeration>>
GliderClass
UNKNOWN = 0
CLUB = 1
STANDARD = 2
DUAL_SEATS = 3
OPEN = 4
public
public
CAirPlane
-m_propulsion:PropulsionType
+ConstructorVid() //pointeri = nullptr, membri = 0
+ConstructorCuParametri(...) //4 argumente, pointer catre const, ultimele 3 predefinite
+ConstructorDeCopiere(...) //membru constant
+Destructor()
+Write():void //scrie membrii clasei
+Read():void //citeste membrii clasei
+GetPropulsion():int //functie inline
+operator=(const CAirPlane& plane):CAirPlane&
+operator+(double load):CAirPlane //modifica membrul m_load
+operator<<(ostream& os, CAirPlane& plane):friend ostream&
+operator>>(istream& is, CAirPlane& plane):friend istream&
CGlider
-m_class:GliderClass
+ConstructorVid() //pointeri = nullptr, membri = 0
+ConstructorCuParametri(...) //4 argumente, pointer catre const, ultimele 3 predefinite
+ConstructorDeCopiere(...) //membru constant
+Destructor()
+Write():void //scrie membrii clasei
+Read():void //citeste membrii clasei
+GetClass():int //functie inline
+operator++():CGlider& //incrementeaza circular m_class 1..4
+operator<<(ostream& os, CGlider& glider):friend ostream&
+operator>>(istream& is, CGlider& glider):friend istream&
CFlyObjectFactory
+CreateFlyObject(FlyObjectType type):CFlyObject* //functie statica, creaza dinamic obiecte CAirPlane sau CGlider in functie de argumentul type, returneaza nullptr in cazul in care type nu este cunoscut