Nume Prenume : **Data:**
Grupa: **Ora:**

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 tuturor constructorilor pentru fiecare clasă derivata în parte **+ 5p**.
- puneti in evidenta utilizarea functiilor ChangeXxxxxxxx(…) **+ 0.5p**.
- puneti in evidenta utilizarea functiilor operator **+ 0.5p**.
- în funcția *main()* creati o instanta a clasei *CCompany*.
    - adaugati adrese ale obiectelor instantiate dinamic, apel: *Hire(new …)* **+ 1p**.
    - afisati elementele adaugate, apel: *Display()* (polimorfism) **+ 0.5p**
    - eliminati 2 elemente precizand ID-ul, apel: *Fire(…)* **+ 0.5p**
    - modificati datele (nume, salariu, pozitie, language/task) a 2 elemente (angajati), apel: *GetEmployee(…)* și reafișați vectorul **+ 1p**.
    - eliberați memoria **+ 1p** .

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) nu presupuneti ci intrbati in caz de neclaritate sau informatii incomplete

CEmployee

<<Enumeration>>
+PositionID

+UNKNOWN = 0
+ENTRY_LEVEL
+JUNIOR
+TEAM_LEADER
+CEO
+REGIONAL
+CEO

-m_szName: char*
-m_dbSalary: double
-m_iEmploymentCount: int //static member, counts the number of class instances, it is only incremented never decremented
-m_iUniqueID: int //employee unique ID (holds the current class instance - the static member value, not to be set by user)

#m_iPosition: int //the position inside the company, one of the above enum members
#Print(): void //writes on screen the employee name, salary, ID and position

+VoidList**Constructor**() //pointers = nullptr, members = 0
+Parametrized**Constructor**(...) //default parameters (iPosition - JUNIOR, dbSalary - 1400)
+Copy**Constructor**(...)
+Destructor() //must be virtual
+ChangePosition(iNewPosition: int): void //change the position inside the company, one of the above enum members
+GetPosition(): int //return the *m_iPosition* member class
+GetName(): char* //returns the *m_szName* member class
+SetName(szName:char*): void //sets the *m_szName* member class
+GetSalary(): double //returns the *m_dbSalary* member class
+SetSalary(dbSalary: double): void //sets the *m_dbSalary* member class
+Write(): void //pure virtual function
+operator=(const CEmployee& other): CEmployee&; // assignment operator, increments the *m_iEmploymentCount* and stores on *m_iUniqueID*

CProgrammer

-m_szLanguage: char* //the programming language

+VoidList**Constructor**() //pointers = nullptr, members = 0
+Parametrized**Constructor**(...) //default parameters (iPosition - JUNIOR, dbSalary - 1400)
+Copy**Constructor**(...)
+Destructor()
+ChangeLanguage(szLanguage: char*): void //change the programming language of the programmer
+operator=(const CProgrammer& other): CProgrammer&; //calls the function operator= from the base class
+Write(): void //displays on screen the programmer informations

CManager

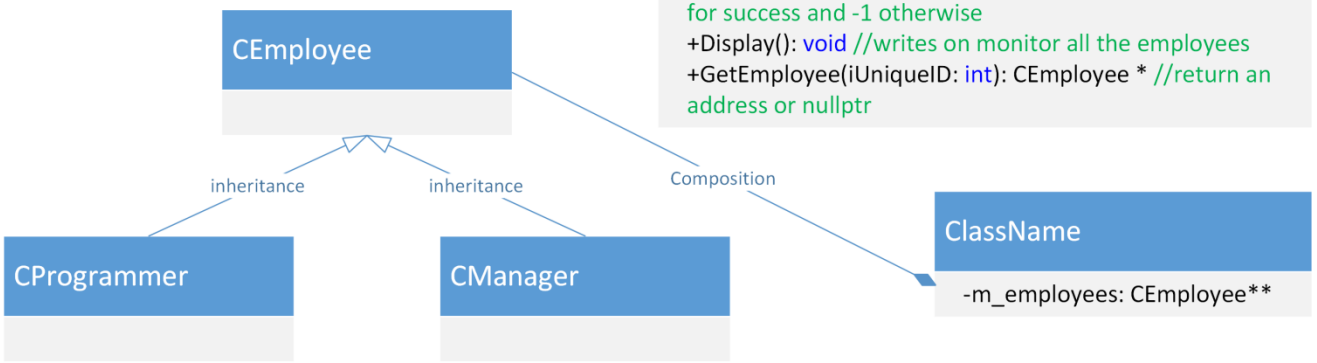
-m_szTask: char* //the manager task

+VoidList**Constructor**() //pointers = nullptr, members = 0
+Parametrized**Constructor**(...) //default parameters (iPosition - JUNIOR, dbSalary - 1400)
+Copy**Constructor**(...)
+Destructor()
+ChangeTask(szTask: char*): void //change the task of the programmer
+operator=(const CProgrammer& other): CProgrammer&; //calls the function operator= from the base class
+Write(): void //displays on screen the manager informations

CCompany

-m_employees: CEmployee**
-m_iSize: int //the vector size

+VoidList**Constructor**() //memory allocation for m_employees vector of pointers in chunks of COMPANY_SIZE elements and initialize the elements with nullptr
+Destructor() //release the memory for every valid pointer element (!=nullptr) and then free the vector
+Hire(CEmployee *employee): int //adds an CEmployee pointer to the m_employees vector and returns the employee ID. If there is a nullptr element in vector that element will store the address. If no elements are nullptr then a memory realocation must be done
+Fire(iUniqueID: int): int //release the memory and writes nullptr in vector for the coresponding element. Returns 0 for success and -1 otherwise
+Display(): void //writes on monitor all the employees
+GetEmployee(iUniqueID: int): CEmployee * //return an address or nullptr



Nume Prenume : **Data:**
Grupa: **Ora:**

Subiect 2

Să se implementeze ierarhia de clase din diagrama UML anexată.

Cerințe (+10p):

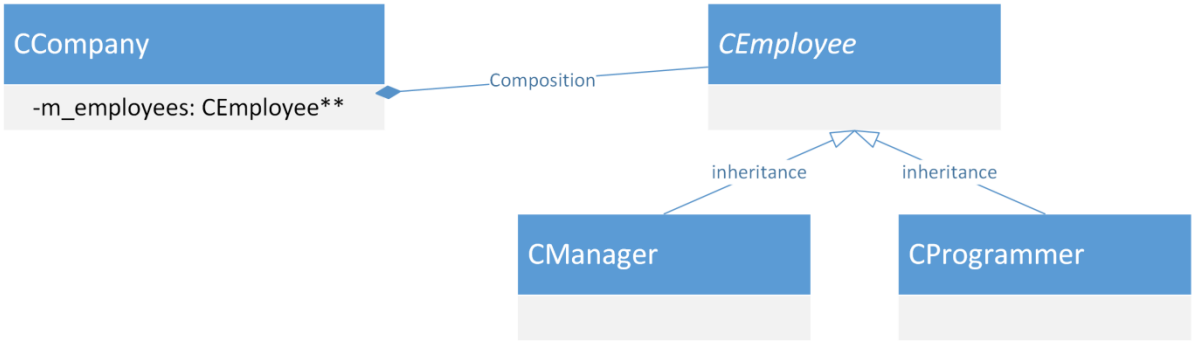
- instanțiați obiecte de tipul claselor derivate astfel încât să puneți în evidență apelul tuturor constructorilor pentru fiecare clasă derivata în parte **+ 5p**.
- puneti in evidenta utilizarea functiilor operator **+ 0.5p**.
- puneti in evidenta utilizarea functiilor *ChangeAaaa(…)* **+ 0.5p**.
- în funcția *main()* creati o instanta a clasei *CCompany*.
    - eliminati 2 elemente instantiate dinamic precizand ID-ul, apel: *Fire(…)* **+ 0.5p**
    - afisati elementele adaugate, apel: *Display()* (polimorfism) **+ 0.5p**
    - adaugati adrese ale obiectelor instantiate dinamic, apel: *Hire(new …)* **+ 1p**.
    - modificati datele (nume, salariu, pozitie, language/task) a 2 elemente (angajati), apel: *GetEmployee(…)* și reafișați vectorul **+ 1p**.
    - eliberați memoria **+ 1p** .

Constrângeri (-1p):

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

Indicații:

b) nu presupuneti ci intrbati in caz de neclaritate sau informatii incomplete



CProgrammer

-m_szLanguage: char* //the programming language

+VoidList**Constructor**() //pointers = nullptr, members = 0
+Parametrized**Constructor**(...) //default parameters (iPosition - JUNIOR, dbSalary - 1400)
+Copy**Constructor**(...)
+Destructor()
+ChangeLanguage(szLanguage: char*): void //change the programming language of the programmer
+operator=(const CProgrammer& other): CProgrammer&; //calls the function operator= from the base class
+Write(): void //displays on screen the programmer informations

CManager

-m_szTask: char* //the manager task

+VoidList**Constructor**() //pointers = nullptr, members = 0
+Parametrized**Constructor**(...) //default parameters (iPosition - JUNIOR, dbSalary - 1400)
+Copy**Constructor**(...)
+Destructor()
+ChangeTask(szTask: char*): void //change the task of the programmer
+operator=(const CProgrammer& other): CProgrammer&; //calls the function operator= from the base class
+Write(): void //displays on screen the manager informations

CCompany

-m_employees: CEmployee**
-m_iSize: int //the vector size

+VoidList**Constructor**() //memory allocation for m_employees vector of pointers in chunks of COMPANY_SIZE elements and initialize the elements with nullptr
+Destructor() //release the memory for every valid pointer element (!=nullptr) and then free the vector
+Hire(CEmployee *employee): int //adds an CEmployee pointer to the m_employees vector and returns the employee ID. If there is a nullptr element in vector that element will store the address. If no elements are nullptr then a memory realocation must be done
+Fire(iUniqueID: int): int //release the memory and writes nullptr in vector for the coresponding element. Returns 0 for success and -1 otherwise
+Display(): void //writes on monitor all the employees
+GetEmployee(iUniqueID: int): CEmployee * //return an address or nullptr

CEmployee

<<Enumeration>>
+PositionID

+UNKNOWN = 0
+ENTRY_LEVEL
+JUNIOR
+TEAM_LEADER
+CEO
+REGIONAL
+CEO

-m_szName: char*
-m_dbSalary: double
-m_iEmploymentCount: int //static member, counts the number of class instances, it is only incremented never decremented
-m_iUniqueID: int //employee unique ID (holds the current class instance - the static member value, not to be set by user)

#m_iPosition: int //the position inside the company, one of the above enum members
#Print(): void //writes on screen the employee name, salary, ID and position

+VoidList**Constructor**() //pointers = nullptr, members = 0
+Parametrized**Constructor**(...) //default parameters (iPosition - JUNIOR, dbSalary - 1400)
+Copy**Constructor**(...)
+Destructor() //must be virtual
+ChangePosition(iNewPosition: int): void //change the position inside the company, one of the above enum members
+GetPosition(): int //return the *m_iPosition* member class
+GetName(): char* //returns the *m_szName* member class
+SetName(szName:char*): void //sets the *m_szName* member class
+GetSalary(): double //returns the *m_dbSalary* member class
+SetSalary(dbSalary: double): void //sets the *m_dbSalary* member class
+Write(): void //pure virtual function
+operator=(const CEmployee& other): CEmployee&; // assignment operator, increments the *m_iEmploymentCount* and stores on *m_iUniqueID*