Curs 2 - POO

1. **Implementare C++ conceptul POO -> abstractizare -> clasa**
   a. **Abstractizare**: Stabileste setul de date relevante pentru un nou tip de date +
   b. Stabilese setul de operatii care se executa asupra setului de date

   Exemplu:

   Tipul de data student -> Nume, Prenume, An, Grupa, nrCredite

   Tipul de data student -> Afisare, Initializare -> valori pentru datele membru, calculNrCredite

   Observatie: Spre deosebire de tipurile predefinite (int, float, char…), acest nou tip de data, nu are reprezentare unica.

   Int x:

   - Dimensiunea
   - Setul de valori
   - Setul de operatii (+ - x / %)

Implementarea abstractizarii (CLASA)

Sintaxa:

```
class NumeTipData {
        public:
                date membre + metode membre;
        private:
                date membre + metode membre;
        protected:
                date membre + metode membre;
};
```

Domenii de access

public: *membrii se pot accesa din afara clasei;*
private: *membrii se pot accesa **doar** din interiorul clasei;*
protected: *mebrii se pot accesa **doar** din clasele care se afla in acelasi ierarhie.*

Observatie: Implicit (daca lipsesc modificatori de acces) clasa este ***PRIVATE***.

Exemplu 1.

```
class Student {
        private:
                char nume[50];
                int nrMatricola;
                int anStudii;
        public:
                void afisare() { cout << nume << " " << nrMatricol << " " << anStudii << endl; }
                void initializare(char sirNume, int nMat, int anS) {
                        strcpy(nume, sirNume);
                        nrMatricola = nMat;
                        anStudii = anS;
                }
};
```

Exemplu 2. -> Definiti un tip de date pentru a modula un nrComplex.(z = a + bi) | (a,b)

```
class Complex {
        private:
                double re;
                double im;
        public:
                void init(double a, double b) {
                        re = a;
                        im = b;
                }
                Void afis() { cout << re << " " << im << endl; }
};
```

2. **Implementare C++ conceptul POO -> incapsulare -> obiect**

Implementarea incapsularii (OBIECT)

Incapsularea: Datele si metodele membre definite intr-o clasa se **incapsuleaza** intru-un tot unitar, numit **obiect**.

Obiectul **este o valoare a clasei** (Blueprint/Sablon al clasei).

Consecinta a incapsularii → ***PRINCIPIUL ASCUNDERII***: zona privata din obiect nu este vizibila

Declararea unui obiect / Sintaxa:

Sa presupunem o clasa definita C

C ob; ←→ int x; = *valoare reziduala*

C *ob; ←→ int *x; = *NULL*

C &ob; ←→ int &x; = *adresa a unei zone de memorie*

C tab[10]; ←→ int tab[10]; = *10 valori de tip int cu valori reziduale;*

Student ob; C++ → ***ob are date cu valori reziduale;***

Complex z; C++ → ***z are date cu valori reziduale;***

Accesare membri pentru un obiect

a. ***Obiectul este alocat static:*** prin operatorul „ . ” de accesare (**ob.accesare()**) ***[DOAR MEMBRII PUBLICI]***
b. ***Obiectul este alocat dinamic (HEAP)***: prin operatorul „ → ” de accesare (**ob→afisare()**)

```
Student *ob = new Student();
```

Definitia 1: Stare obiect: setul de valori ale datelor membre (reziduala, Popescu Maria, 27934, 2)

Definitia 2: Comportament obiect (ce se poate invoca pentru un obiect?): setul de membrii publici

3. Pointerul THIS

```
Student s1, s2;
s1.init(„Popescu Maria”, 27934, 2);
s2.init(„Matei Alex”, 27854, 2);
```

Fiecare metoda din cadrul unei clase are un argument implicit care retine adresa obiectului pentru care se apeleaza o metoda.

Adresa obiectului este de fapt => pointerul „this”

Pas 1:

```
Class Student {
        Public:
                Void afisare() {
                        Cout << this→nume << „ „ << this→nrMatricola << „ „ <<
                        this→anStudiu << endl;
                }
};
```

Pas 2:

```
Student ob;
ob.afisare(); → this = *ob;
```

Utilitatea pointerului THIS

i) *Posibilitatea de a utiliza denumiri de argumente identice cu cele ale datelor membre;*

```
Void init(char *nume, int nrMatricola, int anStudii){
    Strcpy(nume,nume);
    Strcpy(this→nume, nume);
    This→nrMatricola = nrMatricola;
    This→anStudii = anStudii;
}
```