Laboratorul 3

1) Identificați modul de folosire a încapsulării folosind clase in limbajele C++, Python si Java.

O clasă este o colecție de obiecte care au proprietăți, operații și comportamente comune. O clasă este o combinație de stări(date) și comportamente(metode). În POO, o clasă este un tip de date, iar obiectele sunt instanțe ale acestui tip de date.

De exemplu, putem crea o clasă Laptop, care este o colecție a tuturor laptop-urilor din lume. Laptop-urile au anumite caracteristici specifice cum ar fi un model, un sistem de operare, o anumită memorie, un procesor, etc. Toate aceste caracteristici sunt încapsulate în clasa Laptop.

Încapsularea presupune ascunderea unei stări interne a obiectelor și necesitatea ca toate interacțiunile să fie efectuate utilizând metoda obiectelor. Cu alte cuvinte, câmpurile unei clase nu vor putea fi accesate (și modificate) în mod direct, ele nefiind vizibile(pentru că sunt private), dar modificările vor putea fi făcute prin intermediul metodelor.

Obiectele nu pot schimba starea interna a altor obiecte in mod direct (doar prin intermediul metodelor puse la dispoziție de acel obiect), doar metodele proprii ale obiectului pot accesa starea acestuia.

În Python nu există modificatori de acces ca în limbaje precum Java sau C++ , ci toate atributele și metodele sunt publice. Public înseamnă că sunt accesibile din exteriorul clasei (printr-un obiect), din orice modul.

Un atribut sau o metodă privată va putea fi folosită doar în interiorul unei clase, și nu va putea fi accesată din obiectul instanțiat. Putem simula acest comportament în Python, prefixând numele componentei respective cu dublu underscore __.

Pe scurt, încapsularea este procesul prin care "înfășurăm" câmpurile unei clase prin niște metode cu scopul de a avea mai mult control asupra valorilor acestora. În mod normal, acest procedeu este folosit pentru câmpurile declarate private.

A) Încapsularea în C++ - exemplu

Când se folosește class{ }; tot ce e înăuntru acestei clase este default privat, adică doar funcțiile din interiorul clasei pot accesa datele;

```
class Student
{
string mNume;
public:
float medie;
private:
int x;
}
```

mNume si x sunt variabile private, iar medie e public (nu e recomandat sa faci public o variabila)

Exemplu de accesare a datelor private prin functii, si de manipulare a acestora:

```
class Student
{
string mNume;
float mMedie;
public:
Student(){};
```

```
Student(char *nume, float medie);
void set_nume(char *nume);
void set_medie(float medie);
char *get_nume();
float get_medie();
bool restantier();
};

#include “Student.h”
bool Student::restantier()
{
if ( mMedie < 5 )
return true;
else
return false;
}
void Student::set_nume(char *nume)
{
mNume = nume;
}
void Student::set_medie(float medie)
{
mMedie = medie;
}
Student::Student(char *nume, float medie)
{
mNume = nume;
mMedie = medie;
}
char* Student::get_nume()
{
return &mNume[0];
}

float Student::get_medie()
{
return mMedie;
}
```

B) Încapsulare în Python- exemplu

La nivelul clasei Persoana putem implementa incapsularea prin prefixarea numelui atributelor de instanta (nume, varsta)cu underscore. Limbajul Python protejeaza membrii declarati cu dublu underscore prin modificarea numelor acestora cu numele clasei.

```
class Persoana():

 """Clasa Persoana"""
 numar_persoane = 0;
 def __init__(self, nume, varsta):
 self.__nume = nume
 self.__varsta = varsta
```

```
__class__.numar_persoane += 1
def __str__(self):
return "{0}, {1}, {2}".format(self.__nume, self.__varsta, __class__.numar_persoane)
def se_prezinta(self):
print("Buna! Numele meu este {} si am {} ani.".format(self.get_nume(), self.__varsta))
def set_nume(self, nume):
self.__nume = nume
def get_nume(self):
return self.__nume
```

Pentru citirea sau modificarea datelor membre protejate in clase pot fi definite metode de instante de tip set()/get(), care pot fi apelate din orice punct al domeniului de definitie al clasei.

```
>>> marianpopescu = Persoana("Marian Popescu", 35)
>>> marianpopescu
<__main__.Persoana object at 0x03B0A970>
>>> print(marianpopescu)
Marian Popescu, 35, 1
>>> marianpopescu.nume
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in
    marianpopescu.nume
AttributeError: 'Persoana' object has no attribute 'nume'
>>> marianpopescu.get_nume()
'Marian Popescu'
>>> marianpopescu.set_nume('Marian C. Popescu')
>>> print(marianpopescu)
Marian C. Popescu, 35, 1
```

C) Încapsulare în Java- exemplu

```
public class Caine{
   private String nume;
   private int lungimeCoada;

   public String getNume(){
      return nume;
   }
   public void setNume(String nume){
      this.nume = nume;
   }
   public void getLungimeCoada(){
      return lungimeCoada
   }
   public int setLungimeCoada(int lungimeCoada){
      this.lungimeCoada = lungimeCoada;
   }
}
```

Observăm că ambele câmpuri din clasa de mai sus sunt declarate folosind modificatorul de acces private, prin urmare, acestea nu pot fi accesate din afara clasei curente. Cu toate acestea, ne-am dori să avem totuși o cale de a accesa și modifica aceste câmpuri și din exteriorul clasei, fără a modifica, totuși, modificatorul de acces ales. Pentru a rezolva această problemă este suficient să creăm o metodă declarată cu modificatorul de acces public (sau alt modificator de acces care ar

permite accesarea metodei din afara clasei curente) care să returneze valoarea câmpului nostru și o altă metodă declarată tot public care să schimbe valoarea câmpului cu o valoare primită ca parametru.

Metoda care returnează valoarea câmpului poartă numele de getter, iar cea care care modifică valoarea câmpului se numește setter.

În exemplu, metodele getNume() și getLungimeCoada() sunt getteri, iar metodele setNume(), respectiv setLungimeCoada() sunt setteri. Aceste metode au modificatorul de acces public, deci pot fi apelate din orice altă clasă și sunt în interiorul aceleiași clase cu câmpurile, ceea ce înseamnă că le poate accesa. Astfel, dacă suntem într-o altă clasă și vrem să modificăm numele și lungimea cozii unui obiect de tip Caine, tot ce trebuie să facem este:

```
public class ClasaPrincipala{
   public static void main(String[] args){
      Caine bobita = new Caine();
      bobita.setNume("Bobita");
      bobita.setLungimeCoada(10);

      System.out.println(bobita.getNume());
      System.out.println(bobita.getLungimeCoada());
   }
}
```

În urma rulării obținem:

```
Bobita
10
```

2) Definiți o clasa cu minim 2 atribute, cu minim 2 constructori, cu un destructor, cu 2 accesorii, 2 modificatori si încă 1-2 operații.

```
#include<iostream>
#include<string.h>

using namespace std;
// definirea clasei de baza carte

class carte

{
  public:
    carte(char *nume)
     {
       strcpy(carte::nume, nume);

     };
    void afisaza_nume()

     {
       cout<<"\n Titlul cartii:"<<nume<<endl;

     };
  protected :
```

```
   int nrpag ;
   float pret ;
   void afisaza_nrpag(void)

    {
      cout<< "\n numarul de pagini : "<<nrpag<<endl ;

    }
   void afisaza_pret(void)
    {
      cout<< "\n pretul cartii : "<<pret<<endl ;
    }
 private:
   char nume[40];
};
// definirea clasei derivate FisaLibrarie

class FisaLibrarie: public carte

{
  public:
    FisaLibrarie(char *nume, char *aut1, char *aut2, char *ed):carte(nume)
    {
      strcpy(FisaLibrarie ::autor1, aut1) ;
      strcpy(FisaLibrarie ::autor2, aut2) ;
      strcpy(FisaLibrarie ::editura, ed) ;
      cout<<"\n dati numarul de pagini : " ;
      cin>>nrpag ;
      cout<<"\n dati pretul cartii : " ;
      cin>>pret ;
};
  void afiseaza_Libraria()
    {
      cout<<"\n   Lista cartilor din librarie:"<<endl;
      cout<<"\n ==================================="<<endl;
      afisaza_nume() ;
      cout <<"\n numele autorului: "<<autor1;
      cout <<"\n prenumele autorului: "<<autor2;
      cout <<"\n editura: "<<editura;
      afisaza_nrpag() ;
      afisaza_pret() ;
      cout<<"\n ==================================="<<endl;
    };
 private:
   char autor1[40];

   char autor2[40];

   char editura[40];
};
int main()
 {
```

```
    FisaLibrarie fisa();
    char wnume[40],wautor1[40],wautor2[40],weditura[40];
    char r='d';
    while(r=='d')
     {
        cout<<"\n denumirea cartii:";
        cin>>wnume;
        cout<<"\n numele autorului:";
        cin>>wautor1;
        cout<<"\n prenumele autorului:";
        cin>>wautor2;
        cout<<"\n editura:";
        cin>>weditura;
        FisaLibrarie fisa(wnume,wautor1,wautor2,weditura);
        fisa.afiseaza_Libraria();
        cout<<"\n continuati?(d/n):";
        cin>>r;
     }
}
```

C:\Users\Asus\Desktop\Well-Code\lb3\bin\Debug\lb3.exe

```
denumirea cartii:Povestea

numele autorului:Obama

prenumele autorului:Michelle

editura:Litera

dati numarul de pagini : 620

dati pretul cartii : 45

  Lista cartilor din librarie:

==================================

Titlul cartii:Povestea

numele autorului: Obama
prenumele autorului: Michelle
editura: Litera
numarul de pagini : 620

pretul cartii : 45

==================================

continuati?(d/n):
```