Laborator 3

Exercițiul 1

Încapsularea nu reprezintă altceva decât proprietatea claselor de obiecte de a grupa sub aceeași structură datele și metodele aplicabile asupra datelor. În sensul programării orientată pe obiecte, încapsularea definește, de asemenea, modalitatea în care diversele obiecte și restul programului se pot referi la datele specifice obiectelor.

Clasele din C++ permit separarea datelor în date private și date publice. Programele utilizatorilor pot accesa și utiliza datele unui obiect, declarate private, numai prin utilizarea unor metode definite public. Separarea datelor în secțiuni de date publice și private determină protejarea datelor față de utilizarea lor eronată în programe.

În Python nu există modificatori de acces ca în limbaje precum Java sau C++ , ci toate atributele şi 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, şi nu va putea fi accesată din obiectul instanţiat. Putem simula acest comportament în Python, prefixând numele componentei respective cu dublu underscore __.

În Java, încapsularea este procesul de ascundere a-l informațiilor. Obiectele nu pot schimba starea internă a altor obiecte în mod direct (doar prin intermediul metodelor puse la dispozitie de acel obiect). Doar metodele proprii ale obiectului pot accesa starea acestuia. Procesul de compartimentare a elementelor unei abstractizari: structura și comportamentul. Avantajele sunt cele că poți face clasa read-only sau write-only (getteri si setteri) și că ai control asupra datelor (starea obiectului).

Sintaxa declarării unei clase în C++ este următoarea:

specificator_clasa Nume_clasa
{
[[private :] lista_membri_1]
[[public :] lista_membri_2]
};unde:
Specificatorul de clasă specificator_clasa poate fi:class, struct, union.

Descrierea propriu-zisă a clasei constă din cele doua liste de membri, prefixate de cuvintele cheie “private” şi/sau “public”. Membrii aparţinând secţiunii “public” pot fi accesaţi din orice punct al domeniului de existenţă al respectivei clase, iar cei care aparţin secţiunii “private” (atâtdate cât şi funcţii) nu pot fi accesaţi decât de către metodele clasei respective. Utilizatorul clasei nu va avea acces la ei decât prin intermediul metodelor declarate în secţiunea public (metodelor publice).

Avem următoarea clasa în Python:

```
class Impartire:
    def __init__(self, deimpartit, impartitor):
        self.deimpartit = deimpartit
        self.impartitor = impartitor
def executa(self):
        return self.deimpartit / self.impartitor
obj = Impartire(9, 3)
print(obj.executa())
```

Putem accesa oricare din cele două atribute ale obiectului clasei de mai sus prin sintaxa obj.impartitor. Dacă vrem să limităm accesul la aceste atribute nu trebuie decât să le redenumim cu dublu underscore în faţă:

```
class Impartire:
    def __init__(self, deimpartit, impartitor):
        self.__deimpartit = deimpartit
        self.__impartitor = impartitor
 def executa(self):
        return self.__deimpartit / self.__impartitor
obj = Impartire(9, 3)
print(obj.executa())
```

În Java, câmpurile dintr-o clasă care sunt create folosind modificatorul de acces *private,* deci care prin urmare nu pot fi accesate din afara clasei curente. 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 şi 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 numeşte *setter*.

În următorul exemplu, metodele *getNume()* şi *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 şi sunt în interiorul aceleiaşi clase cu câmpurile, ceea ce înseamnă că le poate accesa. Astfel, dacă suntem într-o altă clasă şi vrem să modificăm numele şi lungimea cozii unui obiect de tip Caine, tot ce trebuie să facem este:

```
public class Caine{
private String nume;
private int lungimeCoada;
public String getNume(){
        return nume;
    }
 public void setNume(String nume){
        this.nume = nume;
```

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

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

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

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