Laboratorul 1 - C vs C++

Laboratorul 1

Programarea orientata pe obiecte in limbajul C++

Mai jos sunt prezentate cateva aspecte in legatura cu diferentele dintre limbajele C si C++.

Exemplul 1 Comenzi de intrare si iesire.

In limbajul C++ se pot folosi, in plus fata de functiile *scanf()* si *printf()*, metodele *cin*, respectiv *cout*. Acestea doua din urma nu necesita specificarea formatului datelor. Pentru a utiliza *cin* si *cout* se va include fisierul header *iostream*. In locul bibliotecii *stdlib.h* poate fi utilizata bibliotacea C++ *cstdlib.*

Cod C

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    char nume[15];
    char prenume[15];
    int varsta;

    printf("nume = ");
    scanf(" %s", nume);
    printf("prenume = ");
    scanf(" %s", prenume);
    printf("varsta = ");
    scanf("%d", &varsta);

    printf("%s %s are %d ani \n",
nume, prenume, varsta);

    system("pause");
    return 0;
}
```

Cod C++

```c
#include <iostream>
#include <stdlib.h>
using namespace std;

int main(){
    char nume[15];
    char prenume[15];
    int varsta;

    cout<<"nume = ";
    cin>>nume;
    cout<<"prenume = ";
    cin>>prenume;
    cout<<"varsta = ";
    cin>>varsta;

    cout<<nume<<" "<<prenume<<
" are "<<varsta<<" ani"<<endl;

    system("pause");
    return 0;
}
```

Exemplul 2. Transmiterea parametrilor prin referinta.

Cod C

```c
#include <stdio.h>
#include <stdlib.h>

void swap(int *a, int *b){
     int aux = *a;
     *a = *b;
     *b = aux;
}

int main(){
     int a, b;

     printf("a = "); scanf("%d", &a);
     printf("b = "); scanf("%d", &b);
     printf("\n %d %d \n", a, b);
     swap(&a, &b);
     printf("\n %d %d \n", a, b);

     system("pause");
     return 0;
}
```

Cod C++

```cpp
#include <iostream>
#include <stdlib.h>
using namespace std;
void swap(int &a, int &b){
     int aux = a;
     a = b;
     b = aux;
}

int main(){
     int a, b;

     cout<<"a = "; cin>>a;
     cout<<"b = "; cin>>b;
     cout<<endl<<a<<" "<<b<<endl;
     swap(a, b);
     cout<<endl<<a<<" "<<b<<endl;

     system("pause");
     return 0;
}
```

Exemplul 3. Alocarea dinamica a memoriei.

Se pot utiliza operatorii *new* si *delete*, conform exemplului de mai jos.

Cod C

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    int*p=(int*)malloc(sizeof(int));
    *p = 10;

    int*v=(int*)malloc(3*sizeof(int));
    v[2] = 5;

    printf("%d %d \n", *p, v[2]);

    free(p);
    free(v);

    system("pause");
    return 0;
}
```

Cod C++

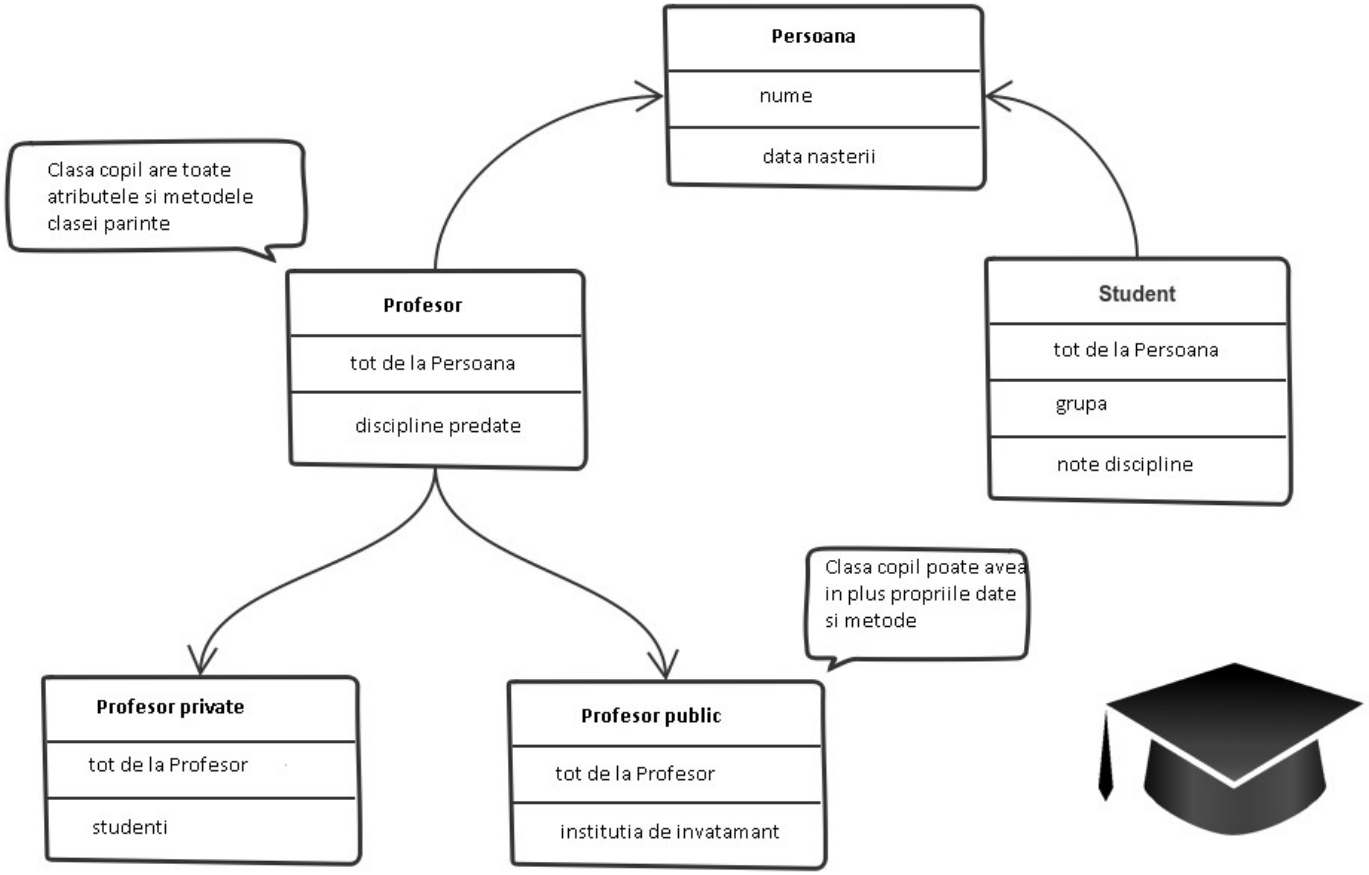
```cpp
#include <iostream>
#include <stdlib.h>
using namespace std;
int main(){
    int *p = new int;
    *p = 10;

    int *v = new int[3];
    v[2] = 5;

    cout<<*p<<" "<<v[2]<<endl;

    delete p;
    delete[] v;

    system("pause");
    return 0;
}
```

Programare orientata pe obiecte

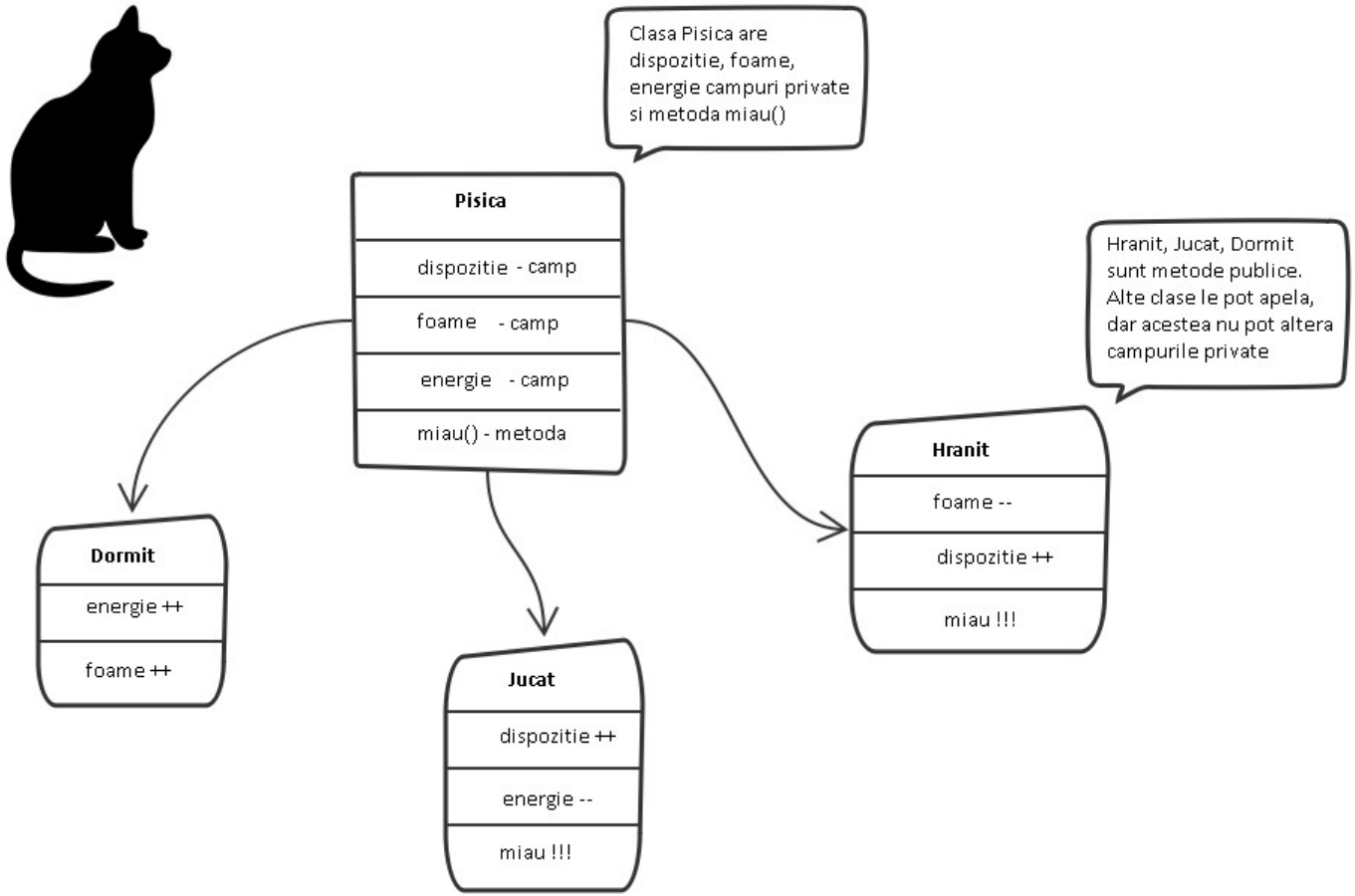
Paradigma de programare orientata pe obiecte are la baza conceptul de obiect. Obiectele descriu entitati, actiuni, obiecte, fenomene din viata reala. Cu alte cuvinte, obiectele reprezinta structuri de date menite sa contina informatii despre notiuni pe care le modeleaza.

Principalele caracteristici la Programarii Orientate pe Obiecte:

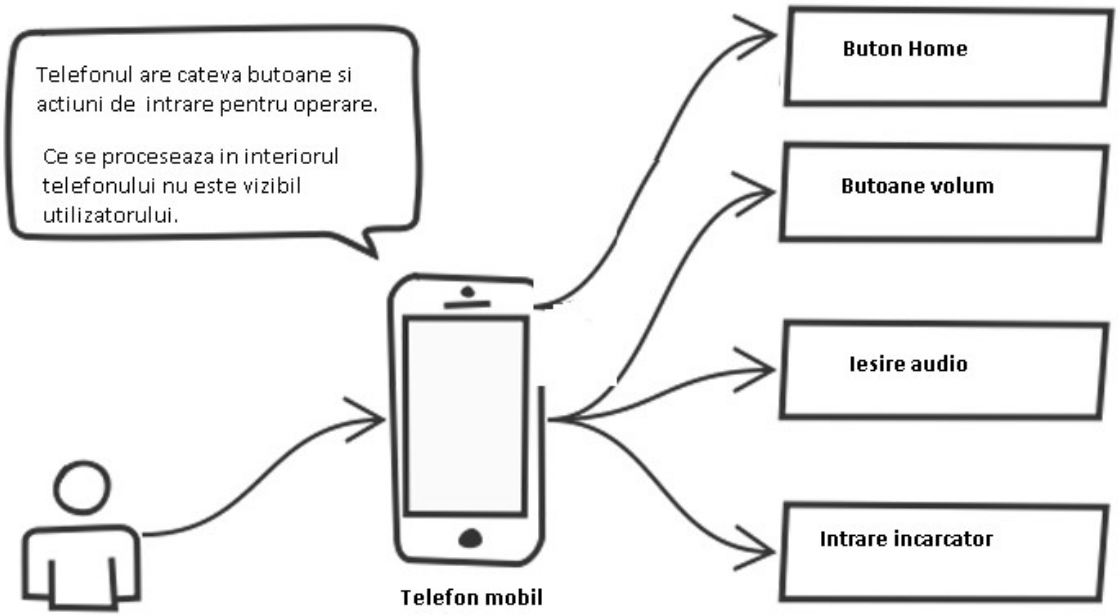
1. Mostenire



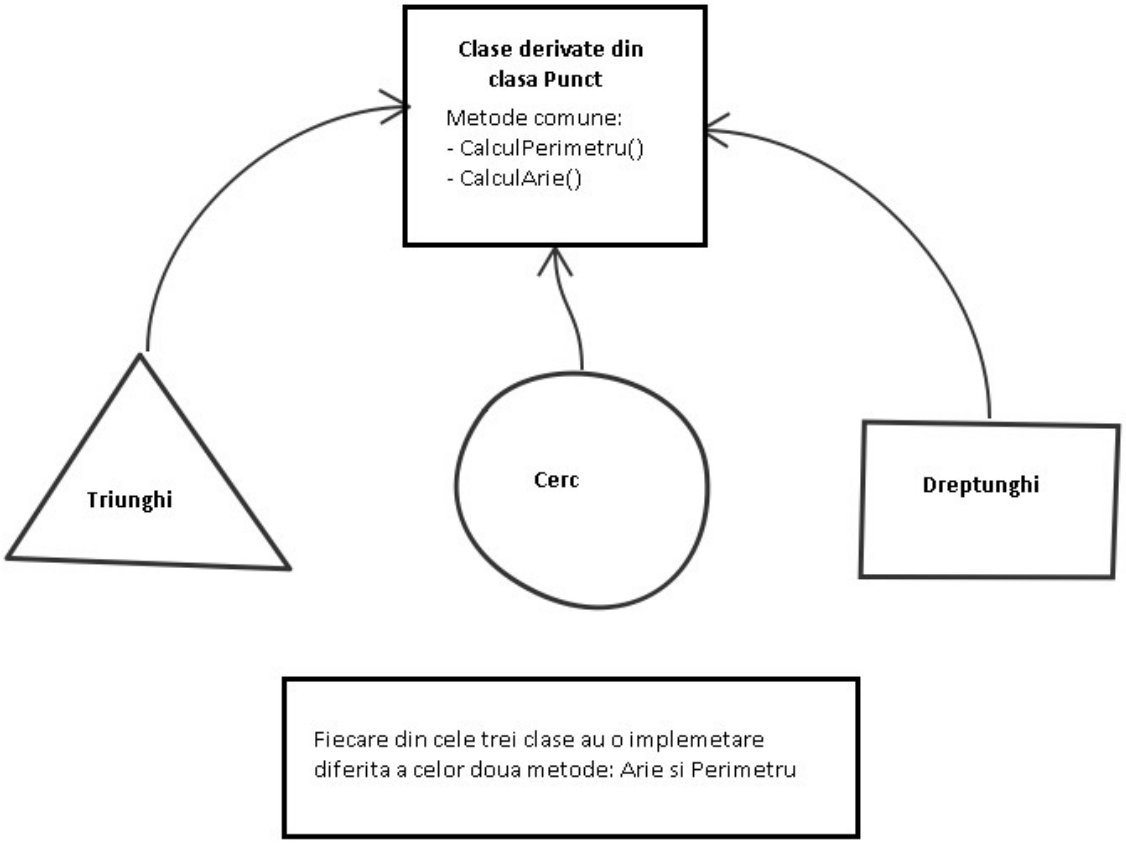
2. Incapsulare



3. Abstractizare



4.Polimorfism



Schema dupa care sunt create obiectele se numeste clasa. Clasele contin caracteristicile (atributele) si actiunile (metodele) unui obiect. Acele atribute si metode care pot fi accesate in afara clasei se vor declara cu specificatorul public. Crearea unui obiect dupa modelul clasei poarta denumirea de instantiere.

In cadrul unei clase se pot defini doua tipuri speciale de metode: constructori si destructori. Astfel de metode nu au tip definit (nu sunt nici de tip void) si sunt apelate implicit la crearea unui obiect (instantiere), respectiv la distrugerea acestuia. Constructorul poarta numele clasei. Destructorul are denumirea formata din simbolul ~si denumirea clasei.

Mai jos este un exemplu in care se defineste o clasa menita sa modeleze conceptul de figura geometrica de tip cerc. Caracteristicile unui obiect de tip cerc sunt date de centrul si raza acestuia. Acestea au fost declarate private pentru a nu se putea accesa si modifica in afara clasei.

Exercitiul 1.1.

#include <iostream>

```
#include <cstdlib>
using namespace std;
struct Punct{
       float x;
       float y;
       };
class Cerc{
      private:
         struct Punct centru;
         float raza;

      public:
            Cerc(struct Punct centruCoord, float valRaza){
                  centru = centruCoord;
                  raza = valRaza;
                  }

            float getRaza(){
                  return raza;
                  }
            struct Punct getCentru(){
               return centru;
```