|  |  |  |
| --- | --- | --- |
| C++ | Pyhton | Java |
| La baza POO stau conceptele de clasa si de obiect.  Definitia 1  Clasa, intr-un anumit limbaj de programare, reprezinta definirea unui tip de obiecte abstracte sau concrete, adica descrierea proprietatilor, a datelor si a metodelor, a prelucrarilor, posibile asupra datelor. Clasa este, de fapt, o notiune abstracta, care defineste un anumit tip de obiecte, sau, altfel spus, o clasa reprezinta multimea posibila a mai multor obiecte de acelasi tip.  Definitia 2  Obiect, reprezinta o multime de date care descrie un anumit obiect concret sau abstract, numite si proprietati, impreuna cu procedurile, functiile, de prelucrare a cestora, numite metode.  Crearea unui obiect presupune specificarea clasei din care face parte, astfel identificandu-se proprietatile obiectului si modul in care acestea pot fi folosite si prelucrate  Observatie:Crearea sau declararea unui obiect este similara cu crearea sau declararea unor variabile. Declararea unei variabile se face, dupa cum se stie, prin specificarea identificatorului acesteia, precedat de tipul variabilei iar declararea unui obiect presupune precizarea identificatorului acestuia, precedat de clasa din care face parte, clasa care in mod implicit determina proprietatile si metodele obiectului definit.  O clasa in limbajul C++ poate fi reprezentata ca o extindere a conceptului de structura, care, ca si o structura, descrie un sablon pentru viitoarele declarari de variabile obiect. | În Python, cuvântul "obiect" nu se referă neapărat la instanțierea unei clase. Clasele în sine sunt obiecte, iar, în sens mai larg, în Python toate tipurile de date sunt obiecte. Există tipuri de date care nu sunt clase: numerele întregi, listele, fișierele. O clasă este creată folosind cuvântul cheie class. Atributele şi metodele clasei sunt listate întrun bloc indentat  Clasele şi metodele au o diferenţă specifică faţă de funcţiile obişnuite - ele trebuie să aibă un prefix suplimentar care trebuie adăugat la începutul listei de parametri, dar nu trebuie să-i daţi o valoare când apelaţi metoda, Python o va furniza. Această variabilă specială se referă la obiectul însuşi (engl. self) şi prin convenţie este numită self. Când ne referim la membrii clasei, vom folosi self.membru, într-un mod asemănător cu folosirea "this" din Java.  Metoda specială \_\_init\_\_() este apelată la instanțierea clasei (crearea unui obiect de tipul clasei) și poate fi asemănată cu un constructor.  Metoda specială \_\_del\_\_() este apelată când nu mai sunt referințe la acel obiect (mecanism de garbage collector) și poate fi asemănată cu un destructor.  Instanțierea se face prin apelarea obiectului clasă, posibil cu argumente | Clasele reprezinta principiile de baza ale programarii orientate obiect. Clasele reprezinta notiuni abstracte, povesti, modele (blueprints), ce descriu:   * caracteristicile, atributele unui obiect; reprezinta ceea ce stie sau ce este obiectul; * ce stie sa faca obiectul, comportamentul acestuia descris de metodele sale;   Pe baza claselor,programatorul poate defini ceva concret, obiecte. Astfel, prin instantierea clasei (atributele definite in clasa, sunt initializate cu valori) se obtin variabile de tip obiect. Pentru cei care au cunostinte de C, o clasa este asemenea unei structuri, definita prin *struct*, in care poti defini si metode.  Termenul de Programare Orientata Obiect descrie o metodologie prin care aplicatia software poate fi descompusa in obiecte, ce reprezinta componente dezvoltate pentru a rezolva probleme mai mici. Obiectivul Programarii Orientata Obiect este de a descompune solutia complexa in probleme mai mici (clase) si mai usor de gestionat. Alte concepte ale Programarii Orientata Obiect (ce vor fi detaliate in posturi separate):   * incapsulare – metodologie prin care se ascunde cat mai mult din mecanismele interne ale clasei; atributele obiectelor nu sunt accesibile direct, ci doar prin intermediul interfetei (colectie de functii); in afara clasei se stie doar ce face clasa, insa nu se stie cum si nici nu ai posibilitatea de a modifica acest lucru; * derivare/mostenire – permite extinderea unei clase existente si construirea unei noi solutii pe baza unei solutii existente fara a o modifica pe aceasta; * polimorfism – permite implementarea de solutii diferite sub aceeasi denumire. |

L

|  |  |  |  |
| --- | --- | --- | --- |
| **Definirea generala unei clase in C++**  **class [*nume\_clasa*]**  **{**  **specificator\_acces1;**  ***date\_si\_functii1*;**  **specificator\_acces2;**  ***date\_si\_functii2*;**  **………………………..**  **specificator\_accesn;**  ***date\_si\_functiin*;**  **}[*obiect1*] [,*obiect2*] … [,*obiectm*];** | **Exemplu:**  **definirea clasei Complex**  **class Complex:**  **def \_\_init\_\_(self, real, imag): self.r = real**  **self.i = imag**  **z = Complex(3.0, -4.**5) | **Class nume\_clasa**  **{**  **//inceput bloc clasa**  **//atribute**  **//metode**  **//blocuri de cod**  **//alte clase**  **}//sfarsit bloc clasa** |  |

**Cod c++**

#include<iostream>

#include<string.h>

#include<stdio.h>

using namespace std;

class universitate

{

public:

char universitatea[40];

char facultatea[40];

char specializarea[40];

int anul;

void citeste\_universitate(char \*universitateauniversitate,char \*facultateauniversitate,char \*specializareauniversitate,int \*anu);

void afiseaza\_universitate(int i);

};

void universitate::citeste\_universitate(char \*universitateauniversitate,char \*facultateauniversitate,char \*specializareauniversitate,int \*anu)

{

strcpy(universitatea,universitateauniversitate);

strcpy(facultatea,facultateauniversitate);

strcpy(specializarea,specializareauniversitate);

anul=\*anu;

}

void universitate::afiseaza\_universitate(int i)

{

cout<<"\n universitatea"<<i<<"= "<<universitatea<<endl;

cout<<"\n facultatea= "<<facultatea<<endl;

cout<<"\n specializarea= "<<specializarea<<endl;

cout<<"\n anul= "<<anul<<endl;

}

int i=0;

int main()

{

universitate universitatecitita;

char wuniversitatea[40],wfacultatea[40],wspecializarea[40];

int wanul;

char r='d';

while(r=='d')

{

printf("\n denumirea universitatii= ");

gets(wuniversitatea);

printf("\n denumirea facultatii= ");

gets(wfacultatea);

printf("\n denumire specializare= ");

gets(wspecializarea);

cout<<"\n anul= ";

cin>>wanul;

universitatecitita.citeste\_universitate(wuniversitatea,wfacultatea,wspecializarea,&wanul);

universitatecitita.afiseaza\_universitate(i);

cout<<"\n continuati?(d/n)=";

cin>>r;

}

}