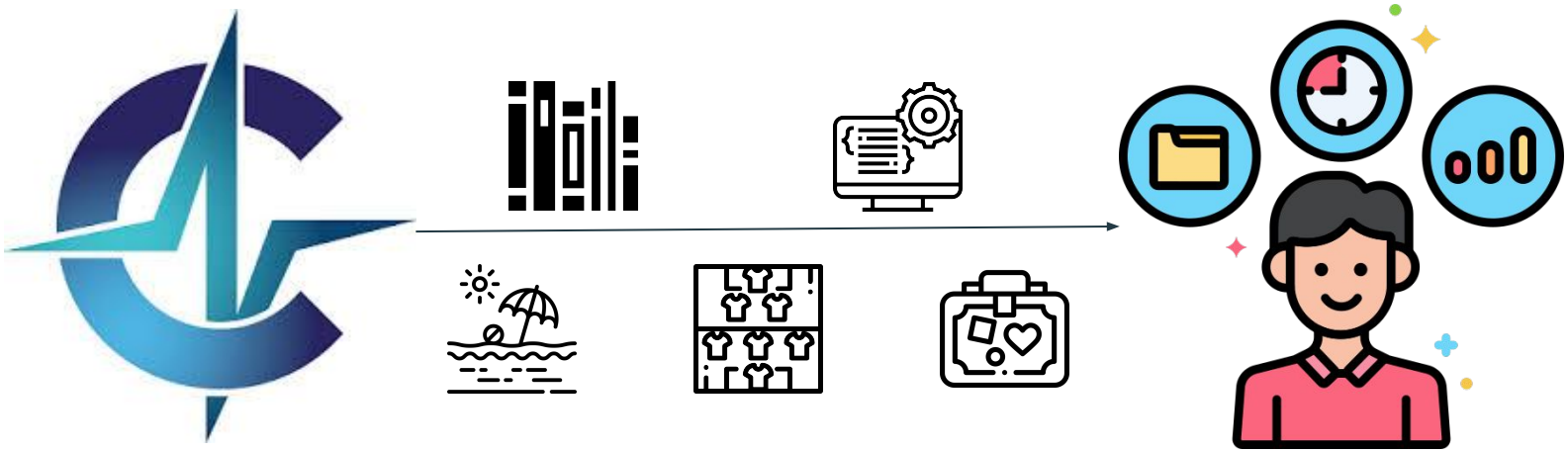
POO - Introducere, Clase, Obiecte

Daniel Chiș - 2020, UPB, ACS, An II, Seria AC

Get to know

Short Intro



Organizare

Detalii de organizare

Timeline:

11 laboratoare: 10 predare și învățare + 1 pentru prezentare proiecte

30 Noiembrie e zi liberă -> nu facem laborator în acea săptămână

19 Decembrie vacanța de Crăciun - primire temă proiect în săptămâna premergătoare

24-29 Ianuarie prezentare proiecte

Detalii de Organizare

Punctaj laborator: **2 puncte** din care:

- **1 punct**: exercițiile de la fiecare laborator (0.1p/laborator). **Prezența este obligatorie!**
- **1 punct:** proiect

Trebuie să luați minim **50%** pe laborator.

Exercițiile se vor trimite pe mail într-un interval de o săptămâna, înainte de a începe laboratorul din săptămâna următoare. Fiecare student trimite un email cu o arhivă care să conțină codul sursă (fișierele .cpp) plus screenshots cu rezultatul afișat pentru fiecare exercițiu. Dacă se depășește perioada nu se punctează.

La email treceți la subiect: **Lab_X_Nume_Prenume_Grupa**. Același nume să îl puneți și la arhivă.

Email: chisdanielioan@gmail.com

Industria IT

Roluri

Management: Project Manager, Product Manager, Product Owner, Scrum Master, Business Analyst

UI/UX (Figma, Bootstrap, CSS, HTML)

Engineering:

- FE: JavaScript; framework-uri: React, Angular, Vue etc.
- BE: C# (.NET), Java (Spring), Python (Django), Ruby (Ruby on Rails), PHP (laravel) etc.
- Firmware, SDK, Console - C++
- Low-level: C
- Databases: SQL (MySQL, Oracle, PostgreSQL), NoSQL (MongoDB)
- Data Analytics & Machine Learning: Python, R
- Big Data: Python, Scala
- QA: Java, Python etc.
- DevOps: bash, linux
- Security: scripting
- Cloud Engineering (AWS, Azure, GCP)

Oportunități

Unde puteți lucra:

- Freelancing
- Start-ups
- Scale-ups
- Corporations

First steps:

Internships

Hackatons

POO
(OOP)

POO - Introducere

Orice produs software trebuie să răspundă unei probleme de business. POO a fost introdus pentru a răspunde industriei și a reprezentat o schimbare de paradigmă

.Beneficiile POO sunt următoarele:

1. Modularitate
2. Reutilizarea codului
3. Flexibilitate
4. Eficacitate în rezolvarea problemelor

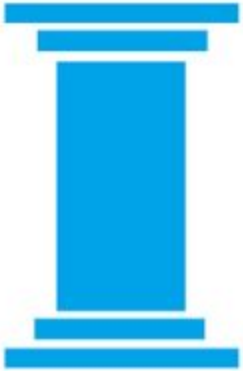
DRY - Don’t repeat yourself!

Este un principiu în programare prin care părți ale codului sunt scrise o singură dată într-un singur loc și după reutilizate. OOP ajută la îndeplinirea acestui principiu, codul devenind mai ușor de menținut, extins și depănat (debugged).

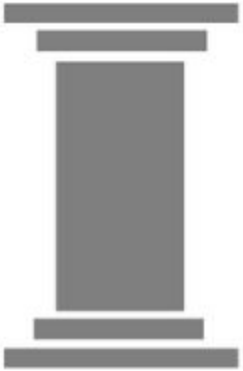
ENCAPSULATION

ABSTRACTION

INHERITANCE



POLYMORPHISM



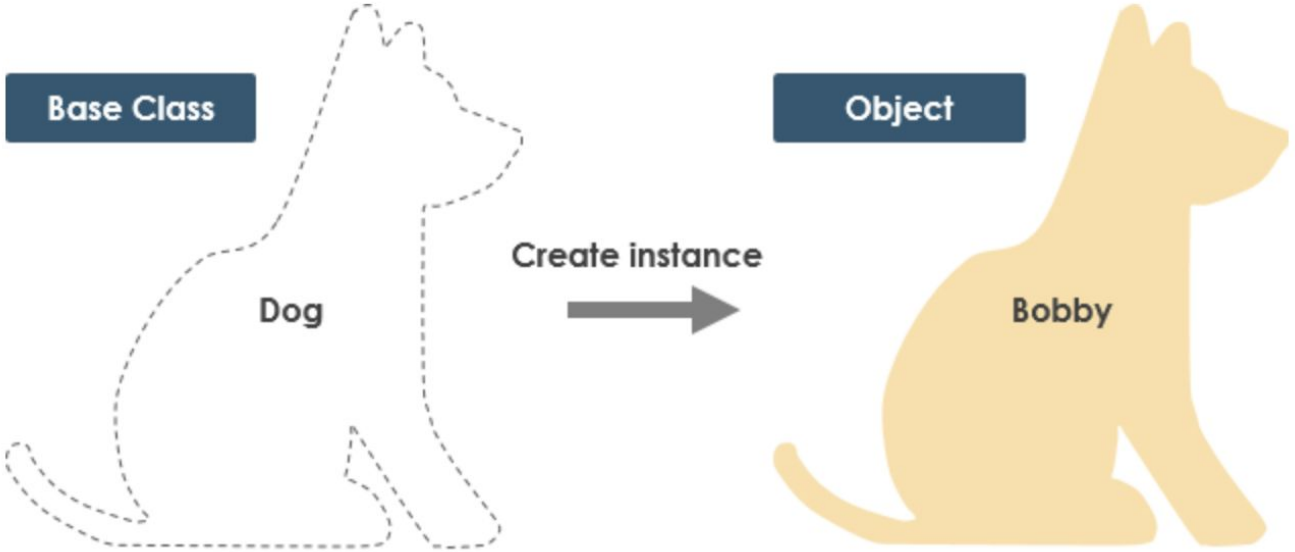
Clase și Obiecte

Clase și Obiecte

POO introduce termenul de **clase (class)**. O clasă reprezintă un blue-print (o matriță) care să reprezinte cum va arăta un **obiect (object)** și cum se va comporta acesta. Prin clase vom putea construi obiecte.

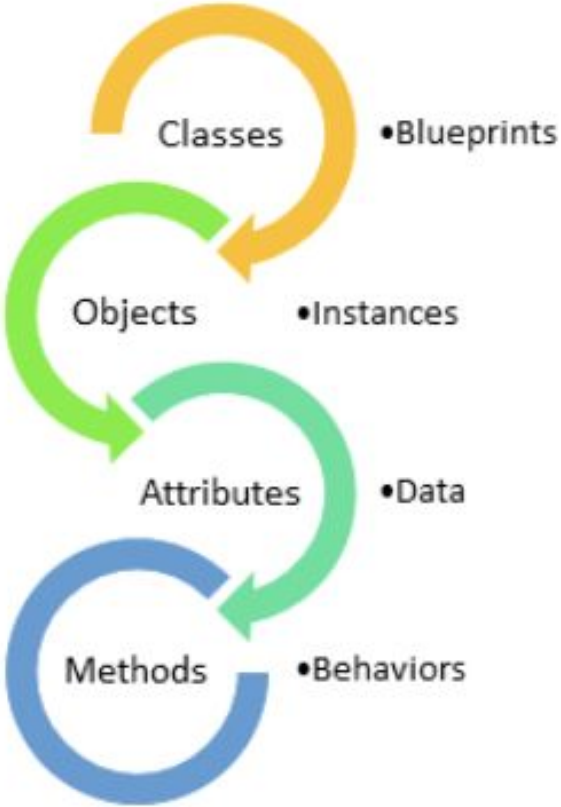
O clasă conține date și funcții pentru manipularea acestora, aceste elemente fiind considerate **membrii ai clasei**.

Obiectul reprezintă **instanțierea** unei **clase**. Când un obiect este creat, acesta se poate folosi de toate funcțiile și metodele oferite de clasa respectivă.



Properties	Methods
Color	Sit
Eye Color	Lay Down
Height	Shake
Length	Come
Weight	

Property Values	Methods
Color: Yellow	Sit
Eye Color: Brown	Lay Down
Height: 17 in	Shake
Length: 35 in	Come
Weight: 24 pounds	


Classes
•Blueprints
Objects
•Instances
Attributes
•Data
Methods
•Behaviors

Crearea unei clase

class - este folosit pentru a crea o clasă

public - este un specificator de acces (într-un laborator viitor)

attributes - exampleNumber și exampleString sunt atribute ale unei clase vor fi folosite pentru a stoca date

; - nu uitați de ele la final de clasă

```
13  class ExampleClass {        // The name of the class
14    public:                 // Access specifier type
15      int exampleNumber;        // Attribute 1 (int variable)
16      string exampleString;  // Attribute 2 (string variable)
17  };
```

```cpp
#include <iostream>
#include <string>

using namespace std;

class ExampleClass {       // The name of the class
  public:             // Access specifier type
    int exampleNumber;        // Attribute 1 (int variable)
    string exampleString;  // Attribute 2 (string variable)
};


int main(){
    //creating the object from the class
    ExampleClass objectExample;
    //access the atributes from that class and set the values to them
    objectExample.exampleNumber = 100;
    objectExample.exampleString = "This is OOP";
    //print attribute stored values for the created object
    cout<<objectExample.exampleNumber<<" "<<objectExample.exampleString<<endl;
}
```

Crearea unui obiect și atribuirea de valori

Summary

De reținut

Cei 4 piloni ai OOP sunt: moștenire, încapsulare, abstractizare și polimorfism.

Clasa reprezintă un blue-print al unui tip de date.

Clasele au atribute și metode.

Prin intermediul unei clase putem crea obiecte.

DRY - Principiu de programare


OBJECT ORIENTED PROGRAMMING
BECAUSE YOU'VE GOT CLASS

Exerciții

Exerciții

1. Pornind de la exemplul din slide 19, realizați o clasă **Student** cu următoarele **atribute**: nume, prenume, vârstă, oraș natal, hobby. Creați un obiect care să fie o reflexie a voastră și afișați aceste date. **5p**
2. Porning de la codul din slide-ul următor, adăugați la metoda **sortingAlg** o metodă de sortare a vectorului. **4p**

```cpp
#include <stdio.h>
#include <iostream>

using namespace std;

class Sorter{
public:
   void sortingAlg(int arr[], int n)
   {
      //TO DO
   }
   void printArray(int arr[], int n)
   {
      int i;
      for (i = 0; i < n; i++)
      cout << arr[i] << " ";
      cout << endl;
   }

};

  int main()
{
   int arr[5] = { 10, 2, 22, 5, 1 };
   int n = 5;
   Sorter sorting;
   sorting.sortingAlg(arr, n);
   sorting.printArray(arr, n);

   return 0;
}
```