

OOP CURS 3

Silviu Ojog - <https://www.youtube.com/@SilviuOjog>

***LINK*Academy**

Exerciții

Ce se printează în output?

```
def fun(x,y,z):  
    return x + y**2 + z//2  
print(fun(z=2,x=3,y=6))
```

- a) **TypeError**
- b) **14**
- c) **44**
- d) **40**

Exerciții

Care variantă printează "nsn" în output?

```
def fun(x,y,z):
```

```
    return x + y**2 + z//2
```

```
print(fun(z=2,x=3,y=6))
```

a) TypeError

b) 14

c) 44

d) 40

a) # se ține cont de fiecare termen în parte

a) # print(fun(z=2,x=3,y=6)) este echivalent a print(fun(3, 6, 2))



Exerciții

Ce se printează în output?

```
def fun(x,y,*z):  
    return (x + sum(z))**y
```

```
print(fun(1,2,1,1,2))
```

- a) **SyntaxError**
- b) **TypeError**
- c) **25**
- d) **8**

Exerciții

Ce se printează în output?

```
def fun(x,y,*z):  
    return (x + sum(z))**y  
print(fun(1,2,1,1,2))
```

- a) SyntaxError
- b) TypeError
- c) 25
- a) $(1 + \text{sum}(1,1,2)) ** 2$
- b) $(1 + 4) ** 2$
- d) 8

Exerciții

Ce se printează în output?

```
def fun(**a):  
    return (min(a)+min(a))/2  
print(fun(11,29,51,-31,20))
```

- a) **SyntaxError**
- b) **TypeError**
- c) **10**
- d) **41**

Exerciții

Ce se printează în output?

```
def fun(**a):  
    return (min(a)+min(a))/2  
print(fun(11,29,51,-31,20))
```

- a) SyntaxError
- b) **TypeError**
 - a) takes 0 positional arguments but 5 were given
 - a) Rezolvare: *a în loc de **a
- c) 10
- d) 41

Exerciții

Ce se printează în output?

```
def fun(**a, *b):  
    return sum(b)/len(b)
```

```
print(fun(11,29))
```

- a) **SyntaxError**
- b) **TypeError**
- c) **20**
- d) **40**

Exerciții

Ce se printează în output?

```
def fun(**a, *b):  
    return sum(b)/len(b)
```

```
print(fun(11,29))
```

- a) **SyntaxError**
 - a) **SyntaxError: invalid syntax**
 - b) **Corect ar fi def fun(*b, **a)**
- b) **TypeError**
- c) **20**
- d) **40**



Exerciții

Ce se printează în output?

```
def fun(*b, **a):  
    return sum(b)/len(b)
```

```
print(fun(a=11,b=29))
```

- a) **SyntaxError**
- b) **ZeroDivisionError**
- c) **20**
- d) **40**

Exerciții

Ce se printează în output?

```
def fun(*b, **a):  
    return sum(b)/len(b)  
print(fun(a=11,b=29))
```

- a) SyntaxError
- b) ZeroDivisionError

a) *b - nu are niciun parametru

**a - are următorii parametri a["a"] și a["b"]

- c) 20
- d) 40

Exerciții

Care variantă este corectă?

```
def fun(x = 3, y = 3):
```

```
    y -= 1
```

```
    z = x * y * 1
```

```
    x *= 2
```

```
    return z
```

```
x = 1
```

```
print(fun())
```

- a) 2
- b) 12
- c) 6
- d) SyntaxError
- e) TypeError

Exerciții

Care variantă este corectă?

```
def fun(x = 3, y = 3):
    y -= 1
    z = x * y * 1
    x *= 2      # Nu influențează cu nimic rezultatul final
    return z

x = 1          # Nu influențează cu nimic rezultatul final
print(fun())
```

- a) 2
- b) 12
- c) 6 # $x = 3$ și $y = 3$, atunci $z = 3 * (3 - 1) * 1 = 6$
- d) SyntaxError
- e) TypeError

Recapitulare?

- Ce știm până acum despre Python?
 - Limbaj interpretat?
 - Limbaj obiect orientat? (Object-Oriented-Programming)

Recapitulare

- Ce știm până acum despre OOP?
 - Obiect vs clasa?
 - Metode vs functii?
 - Ce înseamnă self?
 - Ce înseamnă `__init__`?

Exercitiu - 1

- Construiti clasa Account(**cont bancar**) care sa abstractizeze un cont bancar
- Clasa deține următoarele implementari cu verificarile necesare:
 - **withdraw(scoateBani)** - se scot bani din portofel
 - **addMoney(adaugaBani)** - se adaga bani in portofel
 - **showBalance(arataBalanta)** - se printeaza balanta curenta

OOP - Denumiri

- Clasa (Class):
 - un template (șablon) pentru crearea de obiecte de un anumit tip
- Metodele:
 - sunt funcții care aparțin unei clase
- Atribute:
 - sunt variabile care aparțin unei clase
- Obiect:
 - o instanță specifică de clasă

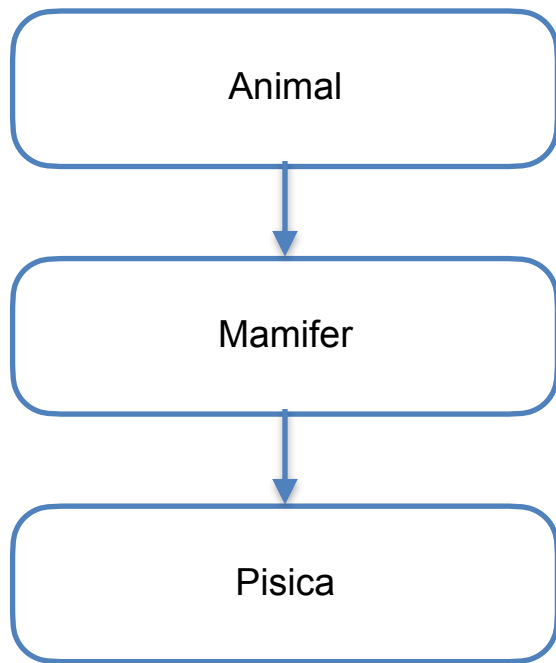
Moștenirea



Moștenirea

- Clasa: Animal
 - Atribute:
 - varsta →
 - greutate
 - Metode:
 - mananca
- Clasa: Mamifer
 - Atribute:
 - varsta →
 - greutate
 - Metode
 - doarme
 - naste
 - mananca
- Clasa: Pisica
 - Atribute:
 - varsta
 - nume →
 - culoare par
 - greutate
 - Metode
 - miauna
 - toarce
 - doarme
 - naste
 - mananca


Moștenirea



Principiul DRY?

- Clasa: Animal
 - Atribute:
 - varsta
 - greutate
 - Metode:
 - mananca
- Clasa: Mamifer
 - Atribute:
 - varsta
 - greutate
 - Metode
 - doarme
 - naste
 - mananca
- Clasa: Pisica
 - Atribute:
 - varsta
 - nume
 - culoare par
 - greutate
 - Metode
 - miauna
 - toarce
 - doarme
 - naste
 - mananca

Principiul DRY?

- Clasa: Animal
 - Atribute:
 - varsta
 - greutate
 - Metode:
 - mananca
- 
- Clasa: Mamifer: Animal
 - Atribute:
 - ~~varsta~~
 - greutate
 - Metode
 - doarme
 - naste
 - ~~mananca~~

Principiul DRY?

- Clasa: Mamifer

- Atribute:

- varsta
 - greutate

- Metode

- doarme
 - naste
 - mananca



- Clasa: Pisica: Mamifer

- Atribute:

- varsta
 - nume
 - culoare par
 - greutate

- Metode

- miauna
 - toarce
 - doarme
 - naste
 - mananca



Moștenirea

- Clasa: Mamifer:
Animal

- Atribute:
 - varsta
 - greutate
- Metode
 - doarme
 - naste
 - mananca



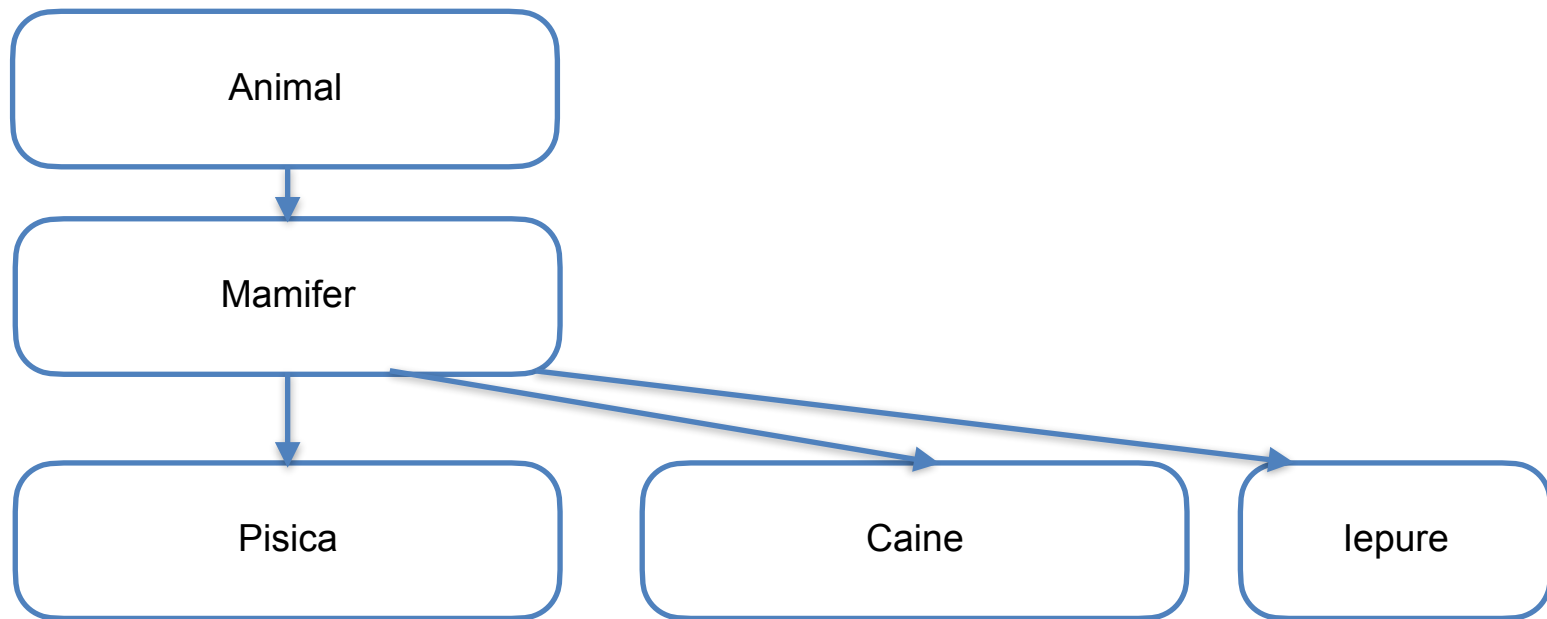
- Clasa: Pisica: Mamifer

- Atribute:
 - varsta
 - nume
 - culoare par
 - greutate
- Metode
 - miauna
 - toarce
 - doarme
 - naste
 - mananca

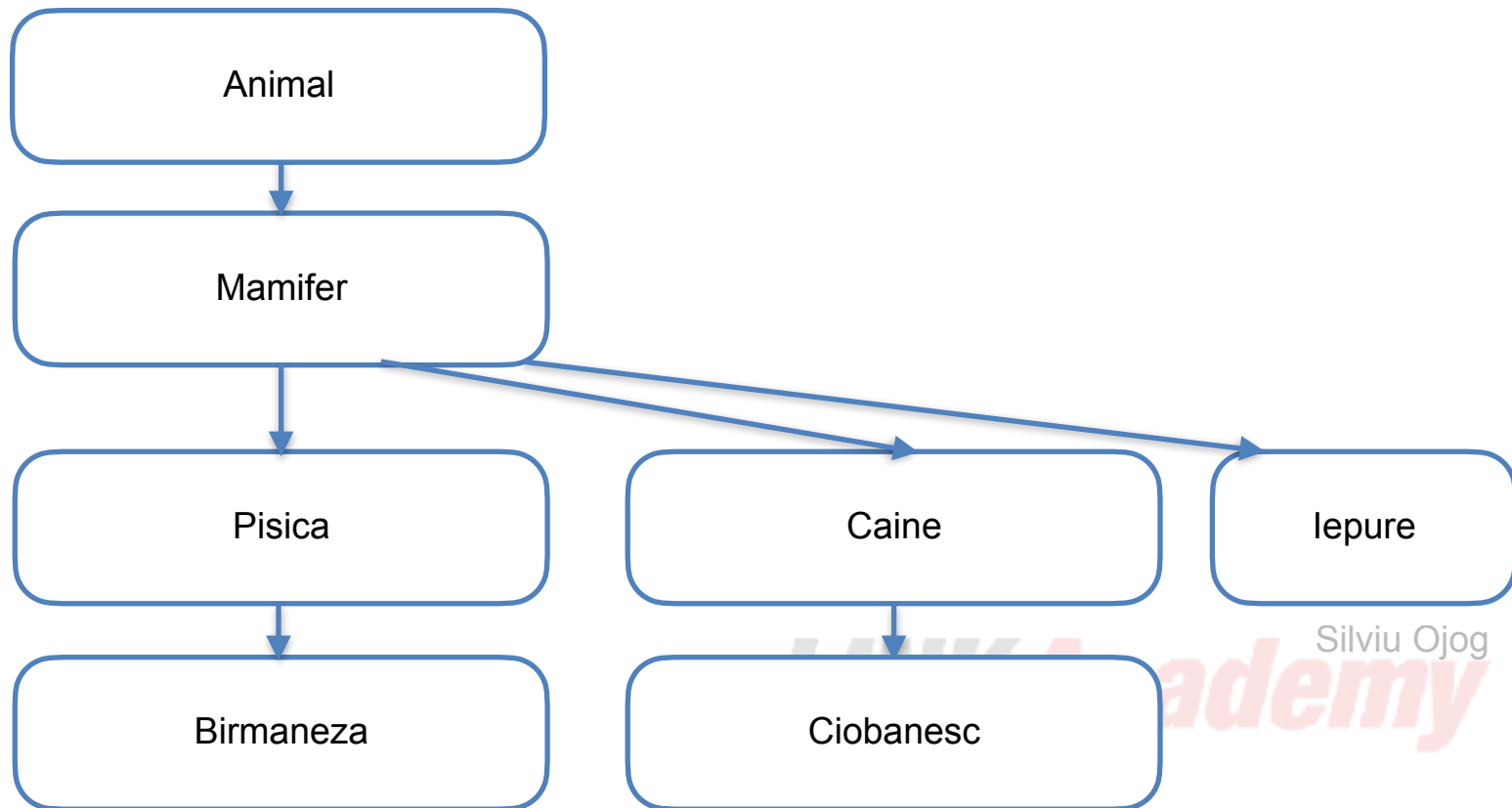
Moștenirea

- O modalitate de refolosire a codului si incapsulare (limitarea accesului din exterior)
 - Structura mostenirii este data de catre dezvoltator (nu de program/compiler etc)

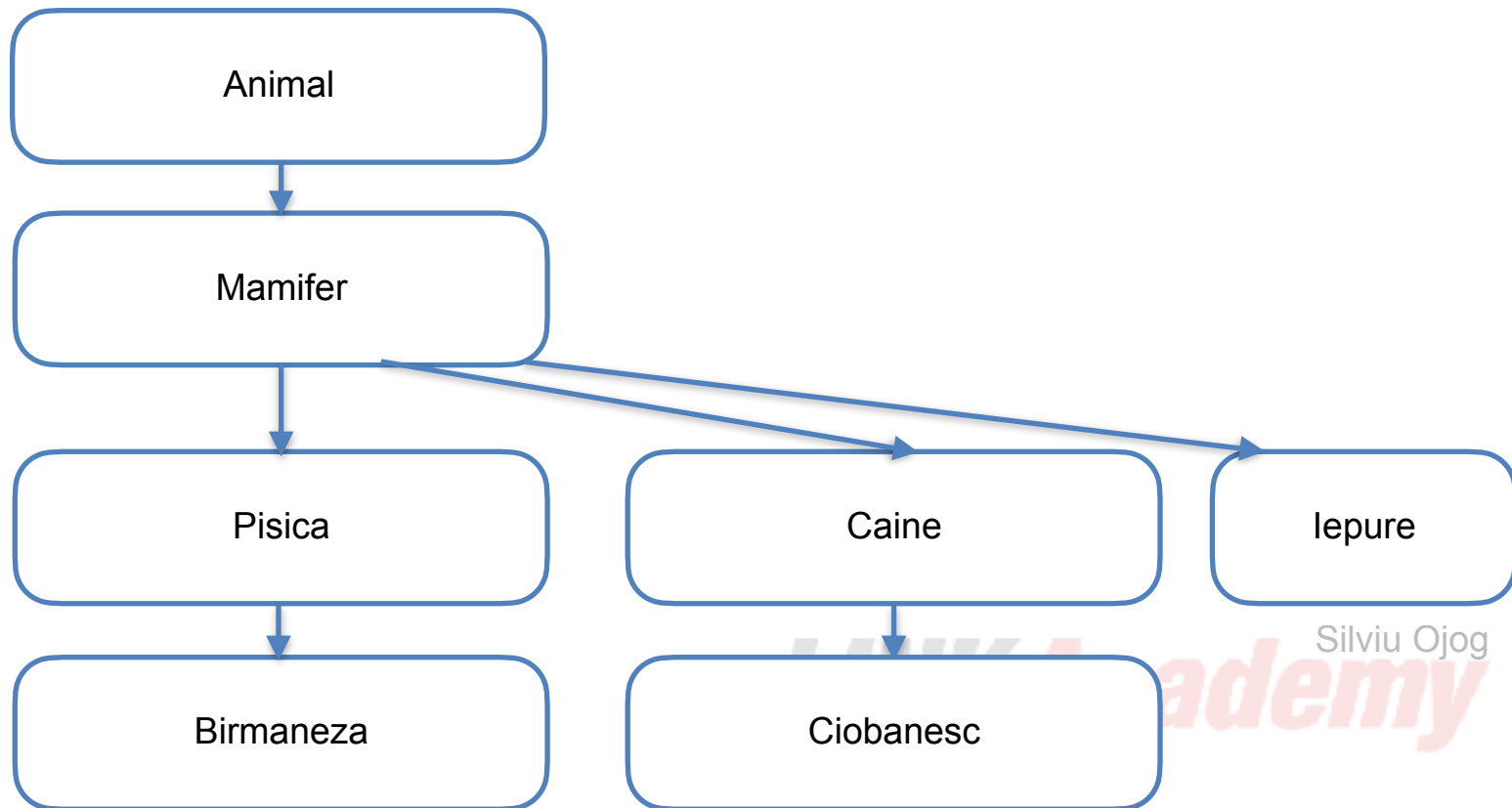
Moștenirea



Moștenirea

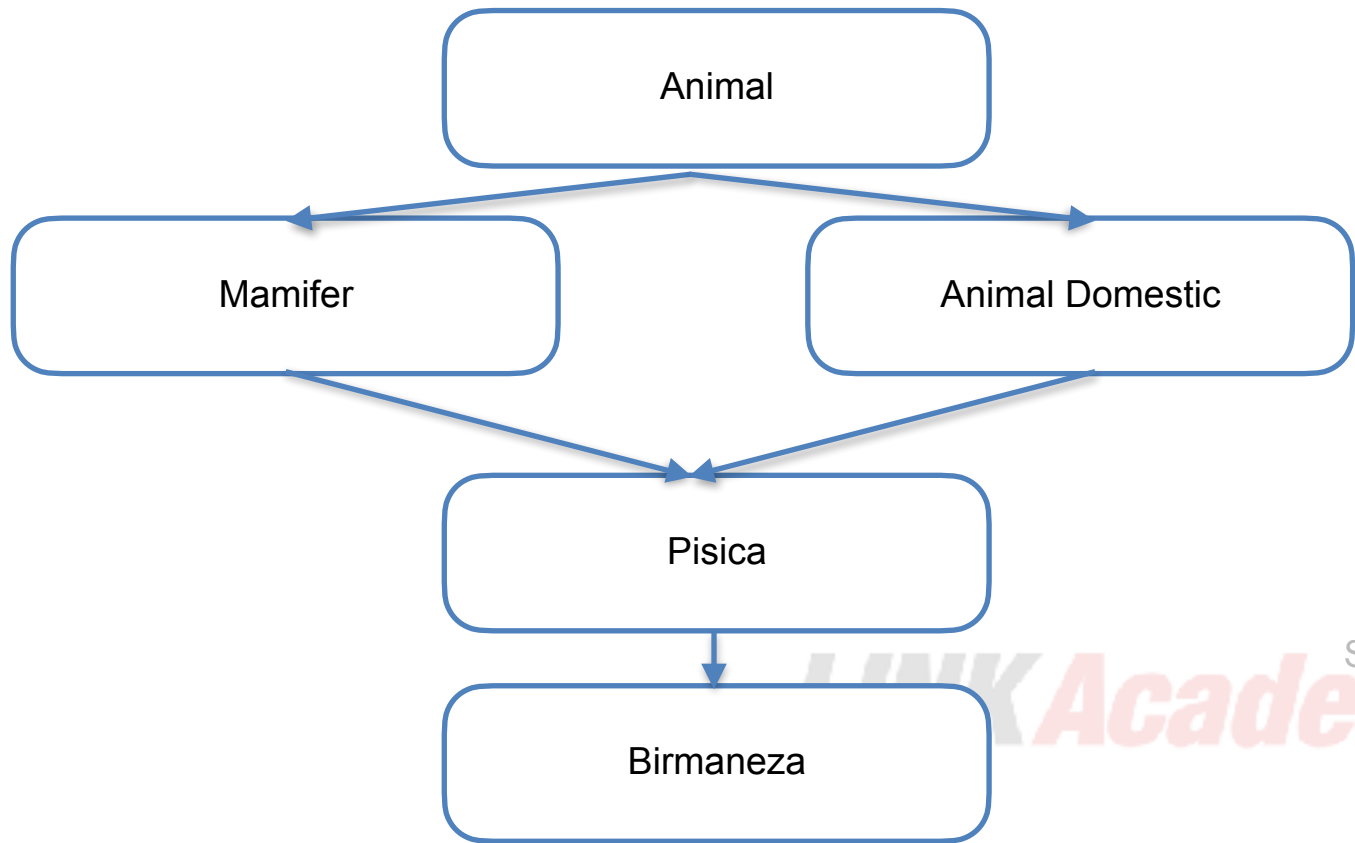


Moștenirea





Moștenirea multiplă



Moștenirea

- Moștenirea reprezintă o relație de tipul **is-a**
 - Pisica **este** un mamifer. (mostenire)
- Moștenirea **NU** reprezintă o relație de tipul **has-a**
 - Pisica **are** o blana, nume etc. (attribute)
- Moștenirea **NU** reprezintă o relație de tipul **does**
 - Pisica **face ceva** anume (metode/functii)

OOP - Denumiri

- Clasa (Class):
 - un template (șablon) pentru crearea de obiecte de un anumit tip
- Metodele:
 - sunt funcții care aparțin unei clase
- Atribute:
 - sunt variabile care aparțin unei clase
- Obiect:
 - o instanță specifică de clasă
- Mostenire:
 - o modalitate prin care o clasă poate moșteni comportament de la o altă clasă
- Compoziția:
 - crearea de obiecte complexe cu ajutorul altor obiecte

Exercitiu 2

- Trebuie creată clasa **Calculator** cu două proprietăți: aș.
- Clasa deține următoarele metode:
 - **add** - ca rezultat returnează suma a doi operanzi;
 - **sub** - ca rezultat returnează suma diferenței a doi operanzi;
 - **mul** - ca rezultat returnează produsul a doi operanzi;
 - **div** - ca rezultat returnează împărțirea celor doi operanzi.



Temă

- Modificați funcția din exercițiul anterior, a.î. să accepte și al treilea parametru cu lista de puncte pentru desen.
- Creați funcția care va returna lista de puncte pe baza punctului inițial și final.

```
render(20,15,path([2,3],[17,12]))
```



- Șirul de puncte trebuie să conțină calea de la punctul inițial la cel final.



LINK

jog

Temă cu Punctul

- Trebuie creată clasa **Point** care sa abstractizeze un punct dintr-un plan (coordonate x si y)
- Clasa deține următoarele caracteristici:
 - **Punctul x** si **punctul y**
- Clasa deține următoarele implementari:
 - **show** - se printeaza coordonatele punctului;
 - **move** - se muta coordonatele;
 - **setToOrigin** - se seteaza coordonatele (0, 0);
 - **dist** - distanta intre doua puncte

Temă cu Punctul



https://www.youtube.com/watch?v=mxdlhG3Tcml&t=2373s&ab_channel=SilviuOjog

Silviu Ojog

LINK Academy