****

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**Calancea Catalin**

**MI-222**

**Raport**

**pentru lucrarea de laborator Nr.4**

**Tema: OOP în Python. Clase.**

***la cursul de “Programare Interactivă”***

Verificat:

lector universitar

Strună Vadim

**Chișinău – 20****23**

**CUPRINS**

[1.INTRODUCERE 3](#_Toc152578222)

[2.SARCINA 4](#_Toc152578233)

[3.1 SCRIPT CONDITIE I 5](#_Toc152578234)

[3.2 SCRIPT CONDITIE II 7](#_Toc152578235)

[4.CONCLUZIA 9](#_Toc152578236)

[4.WEBOGRAFIE 10](#_Toc152578237)

# 1.INTRODUCERE

# Programarea orientată pe obiect (OOP) reprezintă o paradigmă esențială în dezvoltarea software-ului, oferind o abordare structurală și modulară pentru rezolvarea problemelor complexe. În cadrul acestei lucrări de laborator, ne propunem să explorăm aspectele OOP specifice limbajului de programare Python. Vom analiza atât conceptele fundamentale ale OOP, cât și modul în care acestea sunt implementate în Python, un limbaj de programare versatile și puternic.

# Obiectivele principale ale acestei lucrări includ:

# 1. Studiul Paradigmei OOP în Python:

# - Înțelegerea principiilor de bază ale programării orientate pe obiect în contextul limbajului Python.

# 

# 2. Crearea Claselor, Atributelor, Metodelor și Instanțelor:

# - Explorarea procesului de definire a claselor în Python și identificarea caracteristicilor acestora.

# - Investigarea modului în care atributele sunt definite și gestionate în cadrul claselor Python.

# - Implementarea metodelor pentru manipularea datelor și comportamentelor asociate obiectelor create din aceste clase.

# - Crearea și utilizarea instanțelor pentru a exemplifica funcționalitatea claselor și a conceptelor OOP.

# Prin parcurgerea acestor obiective, vom obține o înțelegere detaliată a modului în care OOP este implementată în Python și vom putea aplica aceste concepte în rezolvarea problemelor practice de programare.

# 2.SARCINA

**I. A.**

**Numele clasei Time ; Descriere Timp**

**Când este implementată, clasa trebuie să conțină:**

**metode speciale:**

**- \_\_init\_\_(self,....) initializare cu parametrii necesari;**

**- \_\_str\_\_(self) prezentarea obiectului într-o formă convenabilă pentru om;**

**metode speciale de adunare, diferență și alte operații pe care clasa trebuie să le suporte;**

**metode de clasă:**

**- from\_string(cls,str\_value) creează un obiect pe baza str\_value**

**câmpuri, metode, proprietăți:**

**câmpurile necesare pentru clasa selectată;**

**metoda save(self , filename ) - salveaza obiectul in fisierul filename;**

**metoda load(self , filename) - încarcă un obiect fișier filename;**

**alte metode (cel puțin 3) și proprietăți identificate în etapa studierii clasei.**

**B. Creați o clasă container care va conține setul de obiecte din sarcina anterioară**

**Pentru o clasă de container, furnizați:**

**metode speciale:**

**\_\_init\_\_(self,...);**

**\_\_str\_\_(self);**

**\_\_getitem\_\_()**

**câmpuri, metode, proprietăţi:**

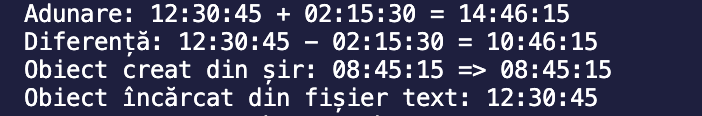
**câmpul \_data - conține un set de date;**

**metoda add(self, value) - adaugă un element de valoare containerului;**

**metoda remove (self,index ) - sterge elementul din container dupa index**

**metoda save(self , filename)**

**metoda load ( self,filename)**

****

**II. Ierarhia claselor**

**Clasele:**

**TransferdeBani, TransferPoștal, TransferBancar, TransferValutar**

**Metodele clasei de bază:**

**indeplineste();**

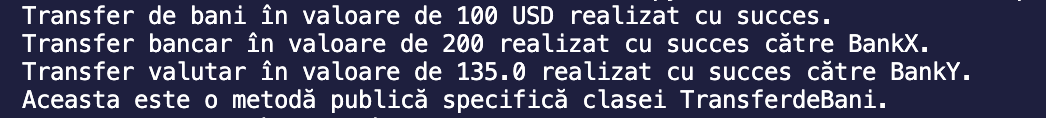
**construiți clasele într-o ierarhie, gândiți-vă la caracteristicile și acțiunile lor comune și distinctive;**

**Adăugați propria implementare a metodelor clasei de bază la fiecare dintre clase, oferind:**

**1.parametrii necesari pentru metodele de bază (de exemplu, un disc DVD abstract poate fi trecut la metoda de redare într-un DVD player);**

**2.câmpurile necesare pentru funcționarea metodelor de bază (de exemplu, la oprirea unui DVD player, este logic să salvați poziția curentă de redare); clasele trebuie să conțină cel puțin un domeniu/metodă public, nepublic și privat;**

**3.** **afișarea funcționării metodei pe ecran (de exemplu, apelarea metodei de oprire într-un DVD player ar trebui să indice pe ecran că playerul este instalat într-o anumită poziție).**

****

# 3.1 SCRIPT CONDITIE I

class Time:

def \_\_init\_\_(self, hours=0, minutes=0, seconds=0):

self.hours = hours

self.minutes = minutes

self.seconds = seconds

def \_\_str\_\_(self):

return f"{self.hours:02d}:{self.minutes:02d}:{self.seconds:02d}"

def \_\_add\_\_(self, other):

if not isinstance(other, Time):

raise ValueError("Can only add Time objects")

total\_seconds = self.hours \* 3600 + self.minutes \* 60 + self.seconds + \

other.hours \* 3600 + other.minutes \* 60 + other.seconds

new\_hours = total\_seconds // 3600

remaining\_seconds = total\_seconds % 3600

new\_minutes = remaining\_seconds // 60

new\_seconds = remaining\_seconds % 60

return Time(new\_hours, new\_minutes, new\_seconds)

def \_\_sub\_\_(self, other):

if not isinstance(other, Time):

raise ValueError("Can only subtract Time objects")

total\_seconds = self.hours \* 3600 + self.minutes \* 60 + self.seconds - \

other.hours \* 3600 + other.minutes \* 60 + other.seconds

new\_hours = total\_seconds // 3600

remaining\_seconds = total\_seconds % 3600

new\_minutes = remaining\_seconds // 60

new\_seconds = remaining\_seconds % 60

return Time(new\_hours, new\_minutes, new\_seconds)

@classmethod

def from\_string(cls, str\_value):

hours, minutes, seconds = map(int, str\_value.split(':'))

return cls(hours, minutes, seconds)

def save(self, filename):

with open(filename, 'w') as file:

file.write(str(self))

def load(self, filename):

with open(filename, 'r') as file:

time\_str = file.read()

return Time.from\_string(time\_str)

*# Exemplu de utilizare:*

time\_obj1 = Time(12, 30, 45)

time\_obj2 = Time(2, 15, 30)

*# Adunare*

result\_addition = time\_obj1 + time\_obj2

print(f"Adunare: {time\_obj1} + {time\_obj2} = {result\_addition}")

*# Diferență*

result\_subtraction = time\_obj1 - time\_obj2

print(f"Diferență: {time\_obj1} - {time\_obj2} = {result\_subtraction}")

*# Creare obiect din șir*

time\_str = "08:45:15"

time\_from\_str = Time.from\_string(time\_str)

print(f"Obiect creat din șir: {time\_str} => {time\_from\_str}")

*# Salvare în fișier text*

time\_obj1.save("time\_object.txt")

*# Încărcare din fișier text*

loaded\_time\_obj = Time().load("time\_object.txt")

print(f"Obiect încărcat din fișier text: {loaded\_time\_obj}")

# 3.2 SCRIPT CONDITIE II

class Transfer:

def \_\_init\_\_(self, sender, receiver, amount):

self.sender = sender

self.receiver = receiver

self.amount = amount

def indeplineste(self):

*# Implementarea comună a metodei pentru toate clasele*

print("Transfer realizat cu succes.")

def \_metoda\_protejata(self):

*# Metodă protejată*

print("Aceasta este o metodă protejată.")

def \_\_metoda\_privata(self):

*# Metodă privată*

print("Aceasta este o metodă privată.")

class TransferdeBani(Transfer):

def \_\_init\_\_(self, sender, receiver, amount, currency):

super().\_\_init\_\_(sender, receiver, amount)

self.currency = currency

def indeplineste(self):

print(f"Transfer de bani în valoare de {self.amount} {self.currency} realizat cu succes.")

def metoda\_publica\_specific\_transfer\_bani(self):

print("Aceasta este o metodă publică specifică clasei TransferdeBani.")

class TransferPostal(Transfer):

def \_\_init\_\_(self, sender, receiver, package\_weight):

super().\_\_init\_\_(sender, receiver, package\_weight)

def indeplineste(self):

print(f"Transfer poștal pentru pachetul cu greutatea de {self.amount} kg realizat cu succes.")

class TransferBancar(Transfer):

def \_\_init\_\_(self, sender, receiver, amount, bank\_name):

super().\_\_init\_\_(sender, receiver, amount)

self.bank\_name = bank\_name

def indeplineste(self):

print(f"Transfer bancar în valoare de {self.amount} realizat cu succes către {self.bank\_name}.")

class TransferValutar(TransferBancar):

def \_\_init\_\_(self, sender, receiver, amount, bank\_name, exchange\_rate):

super().\_\_init\_\_(sender, receiver, amount, bank\_name)

self.exchange\_rate = exchange\_rate

def indeplineste(self):

converted\_amount = self.amount \* self.exchange\_rate

print(f"Transfer valutar în valoare de {converted\_amount} realizat cu succes către {self.bank\_name}.")

*# Exemplu de utilizare:*

transfer\_bani = TransferdeBani("John", "Alice", 100, "USD")

transfer\_postal = TransferPostal("Bob", "Charlie", 5)

transfer\_bancar = TransferBancar("Eve", "Mallory", 200, "BankX")

transfer\_valutar = TransferValutar("Carol", "Dave", 150, "BankY", 0.9)

transfer\_bani.indeplineste()

transfer\_bancar.indeplineste()

transfer\_valutar.indeplineste()

transfer\_bani.metoda\_publica\_specific\_transfer\_bani()

# 4.CONCLUZIA

În urma acestei lucrări de laborator, am explorat cu succes paradigma programării orientate pe obiect în limbajul Python. Am înțeles importanța claselor, a atributelor, a metodelor și a instanțelor în crearea unei structuri robuste și eficiente pentru proiectele software. Capacitatea de a utiliza OOP în Python aduce beneficii semnificative, precum modularitatea crescută, reutilizarea codului și o mai mare înțelegere a relațiilor dintre diferitele componente ale programului. De asemenea, am experimentat cu aplicarea conceptelor OOP în rezolvarea problemelor concrete, consolidând astfel cunoștințele dobândite. În continuare, aceste abilități vor fi esențiale în dezvoltarea de programe mai complexe și în gestionarea eficientă a codului sursă în mediul Python. Lucrarea de laborator reprezintă astfel o bază solidă pentru explorarea ulterioară a programării orientate pe obiect în contextul Python.

# 4.WEBOGRAFIE

1. Python Documentation. Disponibil online la: https://docs.python.org/

2. Stack Overflow. Comunitatea programatorilor. Disponibil online la: https://stackoverflow.com/

3. W3Schools. Tutoriale Python. Disponibil online la: https://www.w3schools.com/python/

4. GeeksforGeeks. Resurse Python. Disponibil online la: https://www.geeksforgeeks.org/python-programming-language/