Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра АСУ



**Звіт**

до виконаної лабораторної роботи №4

з дисципліни “Об’єктно-орієнтоване програмування”

на тему:

**“Наслідування класів”**

Виконав

студент групи КН-118

Cтельмах Віталій

Прийняв

Цимбал Ю.В.

Львів – 2021

## Лабораторна робота № 4

*Тема роботи:*  **Наслідування класів**

***Мета роботи:***

Ознайомитись зі способами та механізмами наслідування класів й навчитись використовувати їх для побудови об’єктно-орієнтованих програм.

**Порядок роботи**

1. Ознайомитись з теоретичним матеріалом про наслідування класів.

2. Для заданого варіанта завдання написати програму з використанням наслідування.

3. Оформити звіт про роботу за такою структурою:

* назва роботи;
* мета роботи;
* послідовність роботи;
* індивідуальне завдання;
* текст програми;
* контрольний приклад та результати роботи програми;
* висновки.

20. Створити базовий клас ОЛІМПІАДНІ ЗАВДАННЯ (дані про учасника, кількість тестових прикладів, кількість пройдених тестів). Створити похідний клас ЗАВДАННЯ «ЧИМ БІЛЬШЕ, ТИМ КРАЩЕ» (задається максимальна кількість балів за завдання (бали учаснику виставляються пропорційно відсотку пройдених тестів). Для заданих прикладів завдань, які розв’язували учасники, впорядкувати учасників за зростанням набраних балів і визначити сумарну кількість балів, набраних учасниками олімпіади.

**Код програми**

mass=[]

class OlimpTask:

name=""

count\_all=0

count\_complate=0

def \_\_init\_\_(self):

self.name

self.count\_all

self.count\_complate

def get\_name(self):

return self.name

def get\_count\_all(self):

return self.count\_all

def get\_count\_complate(self):

return self.count\_complate

def set\_name(self,nm):

self.name=nm

def set\_count\_all(self,all):

self.count\_all=all

def set\_count\_complate(self,comp):

self.count\_complate=comp

class TheMoreTheBetter(OlimpTask):

max\_point=0

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.max\_point

def get\_max(self):

return self.max\_point

def set\_max(self,max):

self.max\_point=max

def CalcPoint(self):

return ((self.get\_count\_complate()\*self.get\_max())/self.get\_count\_all())

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

return "Ім'я - {} " \

"Кількість прикладів = {} " \

"Кількість вирішених = {} " \

"Максимальна кількість балів = {} " \

"Отримані бали = {}".format(mass[i].get\_name(),

mass[i].get\_count\_all(),

mass[i].get\_count\_complate(),

mass[i].get\_max(),

round(mass[i].CalcPoint(),2))

max = int(input("Максимальна кількість балів для учасників олімпіади\n"))

menu = 1

while menu == 1:

a = 1

n = int(input("Виберіть дію \n"

"1-додати учасника \n"

"2-вивести список \n"

"3-відсортувати по сумі балів\n"

"4-сума балів, які здобули всі учасники\n"

"0-вийти з програми\n"))

if(n==1):

while a==1:

name = str(input("ім'я\n"))

all = int(input("кількість прикладів\n"))

comp = int(input("кількість вирішених\n"))

b=1

while b == 1:

if int(all)>=int(comp):

obj = TheMoreTheBetter()

obj.set\_name(name)

obj.set\_count\_all(all)

obj.set\_count\_complate(comp)

obj.set\_max(max)

mass.append(obj)

b=0

else:

comp = input("кількість вирішених, не може бути більшою за загальну введіть ще раз\n")

b=1

a = int(input("Продовжити - 1\n"))

if (n == 2):

for i in range (len(mass)):

print (mass[i])

if (n == 3):

N = len(mass)

for i in range(N - 1):

m = mass[i].CalcPoint()

p = i

for j in range(i + 1, N):

if m > mass[j].CalcPoint():

m = mass[j].CalcPoint()

p = j

if p != i:

t = mass[i]

mass[i] = mass[p]

mass[p] = t

print("Відсортовані учасники по балах")

for i in range (len(mass)):

print(mass[i])

if (n == 4):

print("Сума всіх балів набраних учасниками")

sumall=0

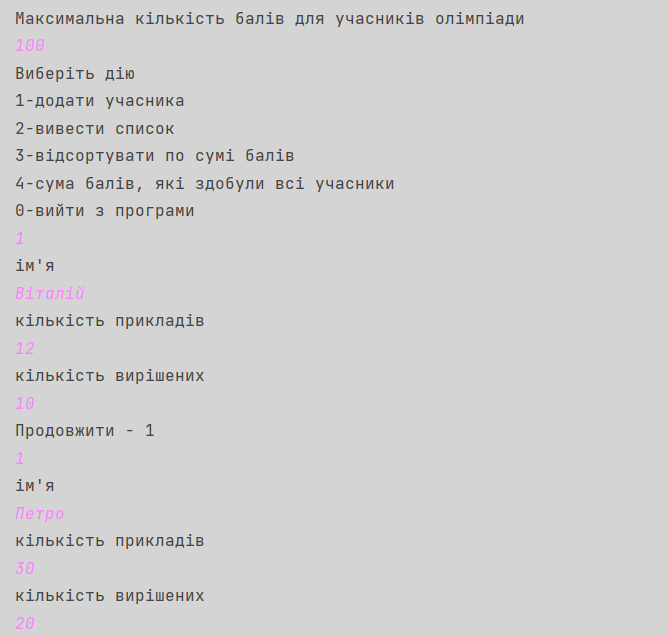
for i in range (len(mass)):

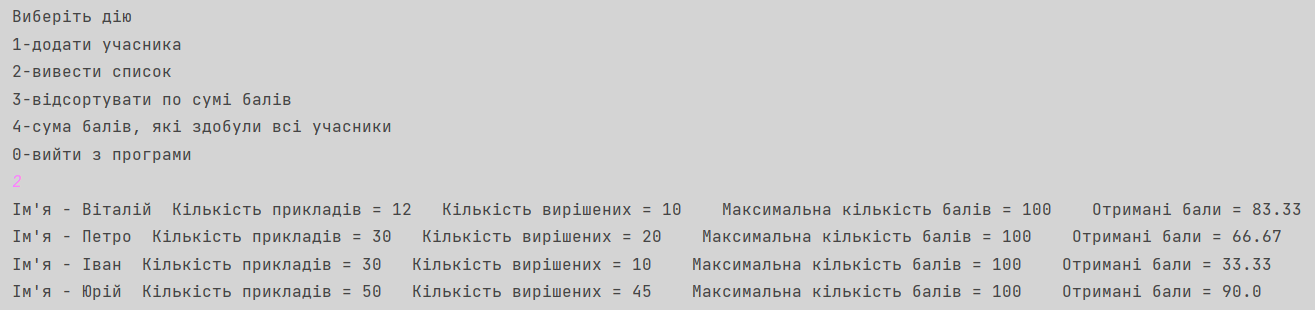
sumall+=mass[i].CalcPoint()

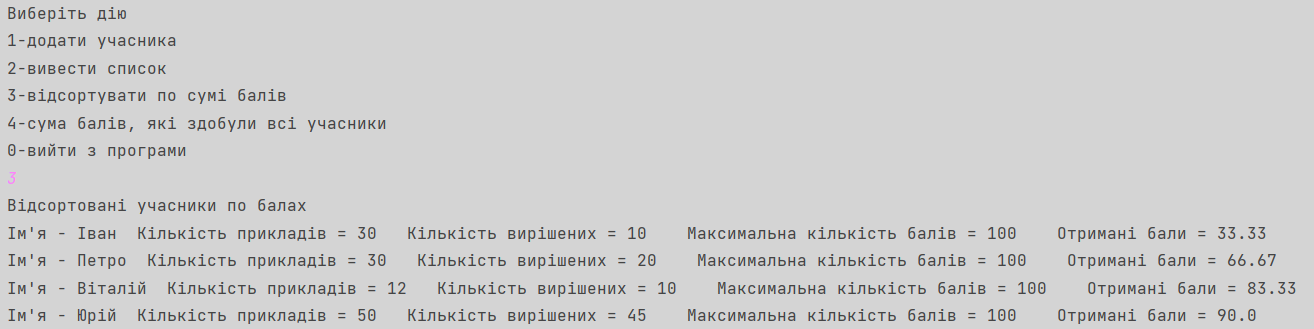
print(str(round(sumall,2)))

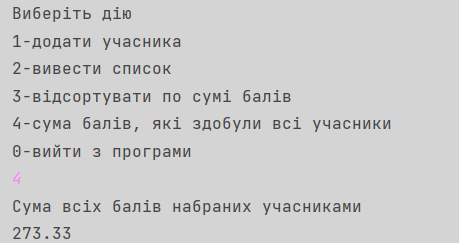
if (n == 0):

menu=0









***Висновок:***

На даній лабораторній роботі я ознайомився зі способами та механізмами наслідування класів й навчився використовувати їх для побудови об’єктно-орієнтованих програм.