**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ Кафедра прикладних інформаційних систем**

ЗВІТ

Про виконання лабораторної роботи №7

з дисципліни: «Технології розроблення програмних систем»

Студент групи ПП-42

Мельник Валентин

Київ – 2022

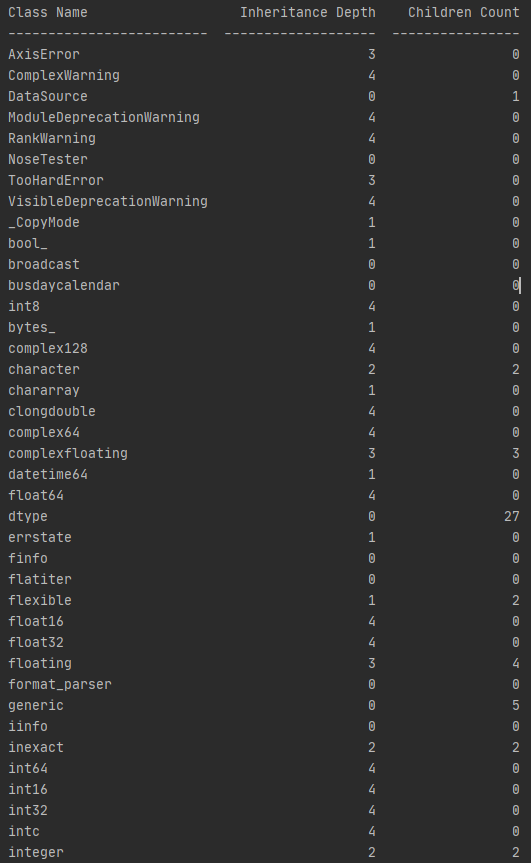
**Тема**: Обʼєктно-орієнтовані метрики коду

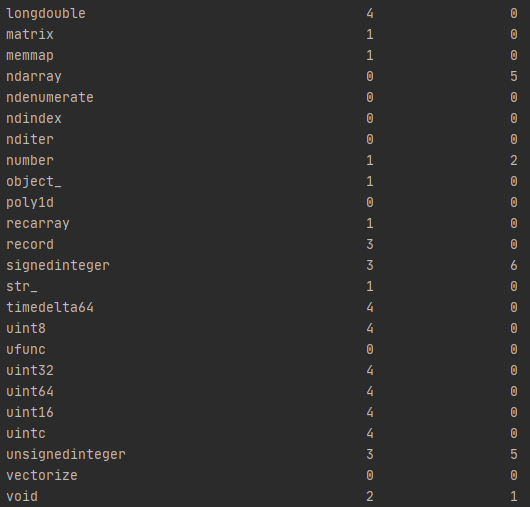
**Код програми:**

Для перевірки обрав бібліотекe numpy

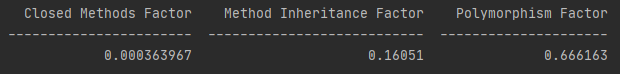
import inspect  
import sys  
  
import tabulate  
import numpy  
  
class ClassStats:  
 def \_\_init\_\_(self):  
 self.inheritance\_depth = 0  
 self.child\_count = 0  
 self.inherited\_methods\_count = 0  
 self.overridden\_methods\_count = 0  
 self.new\_methods\_count = 0  
 self.visible\_methods\_count = 0  
 self.private\_methods\_count = 0  
  
class MetricCounter:  
 def \_\_init\_\_(self):  
 self.\_\_cached\_inheritance\_depths: dict[type, int] = {}  
 self.classes\_stats: dict[type, ClassStats] = {}  
  
 def count\_class(self, clazz: type) -> ClassStats:  
 class\_metrics = ClassStats()  
 class\_metrics.child\_count = len(clazz.\_\_subclasses\_\_())  
 class\_metrics.inheritance\_depth = self.count\_class\_inheritance\_depth(clazz)  
 self.count\_props(clazz, class\_metrics)  
 self.classes\_stats[clazz] = class\_metrics  
 return class\_metrics  
  
 def count\_class\_inheritance\_depth(self, clazz: type) -> int:  
 if clazz in self.\_\_cached\_inheritance\_depths:  
 return self.\_\_cached\_inheritance\_depths[clazz]  
 if clazz.\_\_base\_\_ == object:  
 inheritance\_depth = 0  
 else:  
 inheritance\_depth = self.count\_class\_inheritance\_depth(clazz.\_\_base\_\_) + 1  
 self.\_\_cached\_inheritance\_depths[clazz] = inheritance\_depth  
 return inheritance\_depth  
  
 def count\_props(self, clazz: type, out\_stats: ClassStats):  
 new\_methods = 0  
 inherited\_methods = 0  
 overridden\_methods = 0  
 visible\_methods = 0  
 private\_methods = 0  
 for name, obj in inspect.getmembers(clazz):  
 if inspect.isroutine(obj):  
 if name not in clazz.\_\_dict\_\_:  
 inherited\_methods += 1  
 elif any(name in super\_class.\_\_dict\_\_ for super\_class in clazz.mro()[1:]):  
 overridden\_methods += 1  
 else:  
 new\_methods += 1  
 if name.startswith(f'\_{clazz.\_\_name\_\_}') and not name.endswith("\_\_"):  
 private\_methods += 1  
 else:  
 visible\_methods += 1  
 out\_stats.new\_methods\_count = new\_methods  
 out\_stats.overridden\_methods\_count = overridden\_methods  
 out\_stats.inherited\_methods\_count = inherited\_methods  
 out\_stats.visible\_methods\_count = visible\_methods  
 out\_stats.private\_methods\_count = private\_methods  
  
 def get\_polymorphism\_factor(self) -> float:  
 overriden\_methods = 0  
 denom = 0  
 for clazz, stats in self.classes\_stats.items():  
 overriden\_methods += stats.overridden\_methods\_count  
 denom += stats.new\_methods\_count \* stats.child\_count  
 if overriden\_methods == 0 or denom == 0:  
 return 0.0  
 return overriden\_methods / denom  
  
 def get\_method\_inheritance\_factor(self) -> float:  
 inherited\_methods = 0  
 all\_methods = 0  
 for clazz, stats in self.classes\_stats.items():  
 inherited\_methods += stats.overridden\_methods\_count  
 all\_methods += stats.new\_methods\_count + stats.inherited\_methods\_count + stats.overridden\_methods\_count  
 if inherited\_methods == 0 or all\_methods == 0:  
 return 0.0  
 return inherited\_methods / all\_methods  
  
 def get\_closed\_methods\_factor(self) -> float:  
 private\_methods = 0  
 all\_methods = 0  
 for clazz, stats in self.classes\_stats.items():  
 private\_methods += stats.private\_methods\_count  
 all\_methods += stats.visible\_methods\_count + stats.private\_methods\_count  
 if private\_methods == 0 or all\_methods == 0:  
 return 0.0  
 return private\_methods / all\_methods  
  
  
def class\_stats\_to\_row(clazz: type, stats: ClassStats):  
 return [clazz.\_\_name\_\_, stats.inheritance\_depth, stats.child\_count]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 counter = MetricCounter()  
 for name, obj in inspect.getmembers(sys.modules['numpy']):  
 if inspect.isclass(obj):  
 counter.count\_class(obj)  
 table\_headers = ["Class Name", "Inheritance Depth", "Children Count"]  
 print(tabulate.tabulate([class\_stats\_to\_row(clazz, stats) for clazz, stats in counter.classes\_stats.items()],  
 headers=table\_headers))  
  
 lib\_factors = {"Closed Methods Factor": [counter.get\_closed\_methods\_factor()],  
 "Method Inheritance Factor": [counter.get\_method\_inheritance\_factor()],  
 "Polymorphism Factor": [counter.get\_polymorphism\_factor()]}  
 lib\_factors\_headers = ["Closed Methods Factor", "Method Inheritance Factor", "Polymorphism Factor"]  
 print(tabulate.tabulate(lib\_factors, headers="keys"))

**Результат роботи програми:**





Також виводжу метрики коду



**Висновок**: в межах цієї лабораторної роботи я навчився обраховувати обʼєктно-орієнтовані метрики коду програм написаних на мові Python