Zadanie K
CallCounter

Twoim zadaniem jest zaimplementowanie metaklasy `CallCounter`. Każda klasa będąca typu `CallCounter` ma umożliwiać udostępnianie informacji o liczbie wywoływanych metod tej klasy. Dokładniej, wywołanie `x.foo.calls` ma zwracać liczbę wywołań metody `foo` klasy `x`.

Ponadto, metoda metaklasy `CallCounter` o nazwie `ranking(number)` powinna wypisywać pierwsze `number` niezerowych pozycji z rankingu (wywołanie wszystkich metod wszystkich klas) uporządkowanych najpierw po liczbie wywołań metody, a następnie leksykograficznie po nazwie klasy i nazwie metody. Jeżeli `number` jest za duże, metoda powinna powinna wyrzucać wyjątek `Exception` z wiadomością `Too few methods have been called!`. Dokładna specyfikacja metody `ranking(number)` wynika z testów przykładowych.

Ponadto, metoda `NazwaKlasy.clear()` powinna wyzerować w rankingu użycie wszystkich metod klasy `NazwaKlasy`.

Ostatnią funkcjonalnością, którą chcemy udostępnić, jest możliwość utworzenia klasy *idealnej*, wyposażonej w trzy najczęściej wykonywane metody, pod warunkiem, że takie metody istnieją. Gdy nie możemy utworzyć klasy idealnej z trzema metodami należy wyrzucić wyjątek `Exception` z wiadomością `Too few methods have been called!`. Chcemy, by klasę idealną zwracała metoda `CallCounter.ideal()`. Sposób nazywania metod w klasie idealnej proszę wywnioskować z przykładu.

Wysyłanie rozwiązania

Masz wysłać jeden plik o nazwie **met_count.py**, w którym będzie znajdowała się definicja metaklasy `CallCounter`. Testami są pliki ***.py**, które importują metaklasę `CallCounter`, wykonują wyżej opisane operacje i sprawdzają ich poprawność.

Dostępna pamięć: 512MB

Przykład

Dla testu:

```
from met_count import CallCounter
class A(metaclass = CallCounter):
    def foo(self):
        print("A foo")
    def bar(self):
        print("A bar")
class B(metaclass = CallCounter):
    def foo(self):
        print("B foo")
a = A()
b = B()
d = A()
CallCounter.ranking(0)
try:
    CallCounter.ranking(1)
except Exception as e:
    print(e)
a.foo();b.foo();
a.bar();a.bar();a.foo();d.foo()
print(B.foo.calls)
CallCounter.ranking(3)
A.clear()
try:
    CallCounter.ranking(2)
except Exception as e:
    print(e)
CallCounter.ranking(1)
a.foo();a.foo();a.foo();a.bar();d.bar()
CallCounter.ranking(3)
b = CallCounter.ideal()
w = b()
w.A_foo()
w.A_bar()
w.B_foo()
CallCounter.ranking(3)
```

Poprawną odpowiedzią jest:

```
Too few methods have been called!
A foo
B foo
A bar
A bar
A foo
A foo
1
A.foo() --> 3
A.bar() --> 2
B.foo() --> 1
Too few methods have been called!
B.foo() --> 1
A foo
A foo
A foo
A bar
A bar
A.foo() --> 3
A.bar() --> 2
B.foo() --> 1
A foo
A bar
B foo
A.foo() --> 4
A.bar() --> 3
B.foo() --> 2
```