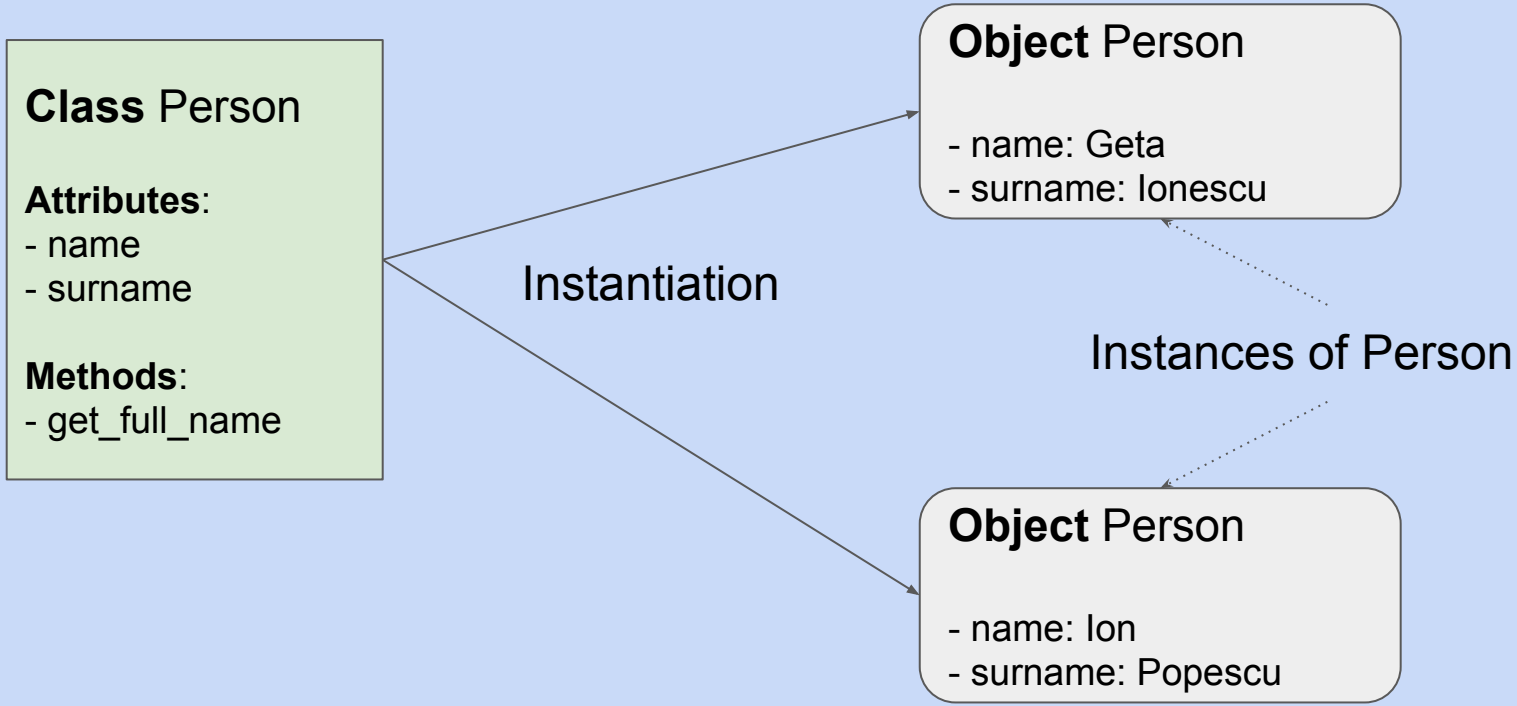
Python training - lab 12

Object-Oriented Programming (OOP)



Object-Oriented Programming (OOP)

```
class Person:

    def __init__(self, name, surname):
        """Constructor of the class"""
        self.name = name
        self.surname = surname

    def get_fullname(self):
        return f'{self.name} {self.surname}'


p1 = Person('Geta', 'Ionescu')
p2 = Person('Ion', 'Popescu')
print(p1.get_fullname())
print(Person.get_fullname(p2))
```

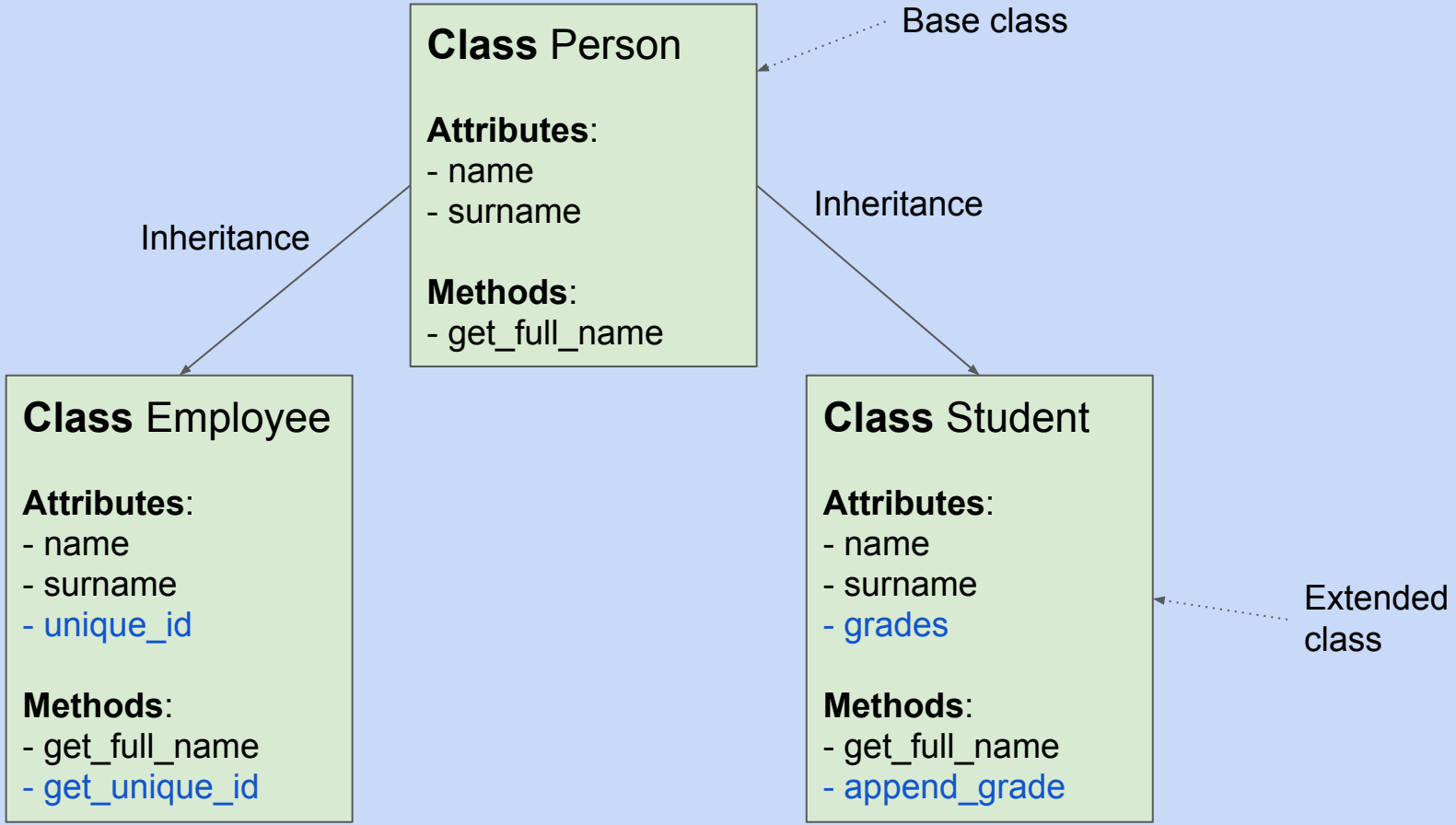
Object-Oriented Programming (OOP)

```
p1 = Person('Geta', 'Ionescu')

# access attributes
print(p1.name)
p1.name = 'Georgeta'

# add attribute on the fly; usually not ok
p1.new_something = 1234

# del attribute on the fly; not ok
del p1.surname
```

OOP - private vs public

```
class Person:

    def __init__(self, name, surname):
        # private attributes
        self.__name = name
        self.__surname = surname

    def get_name(self):
        return self.__name

    def set_name(self, name):
        self.__name = name

    def get_fullname(self):
        return f'{self.__name} {self.__surname}'


p1 = Person('Geta', 'Ionescu')
print(p1.__name)  # error
```

OOP - simple inheritance



OOP - simple inheritance

```
class Student(Person):

    def __init__(self, name, surname, grades):
        super().__init__(name, surname)
        self.grades = grades

    def get_fullname(self):
        return f'{self.name} {self.surname}'.upper()

    def get_average(self):
        if len(self.grades) == 0:
            return 0.0
        return round(sum(self.grades) / len(self.grades), 2)


p1 = Student('Geta', 'Ionescu', [3, 5, 6, 7])
print(p1.get_fullname())
print(p1.get_average())

print(type(p1))  # Student
print(isinstance(p1, Person))  # True. A Student is also a Person
```

OOP - simple inheritance