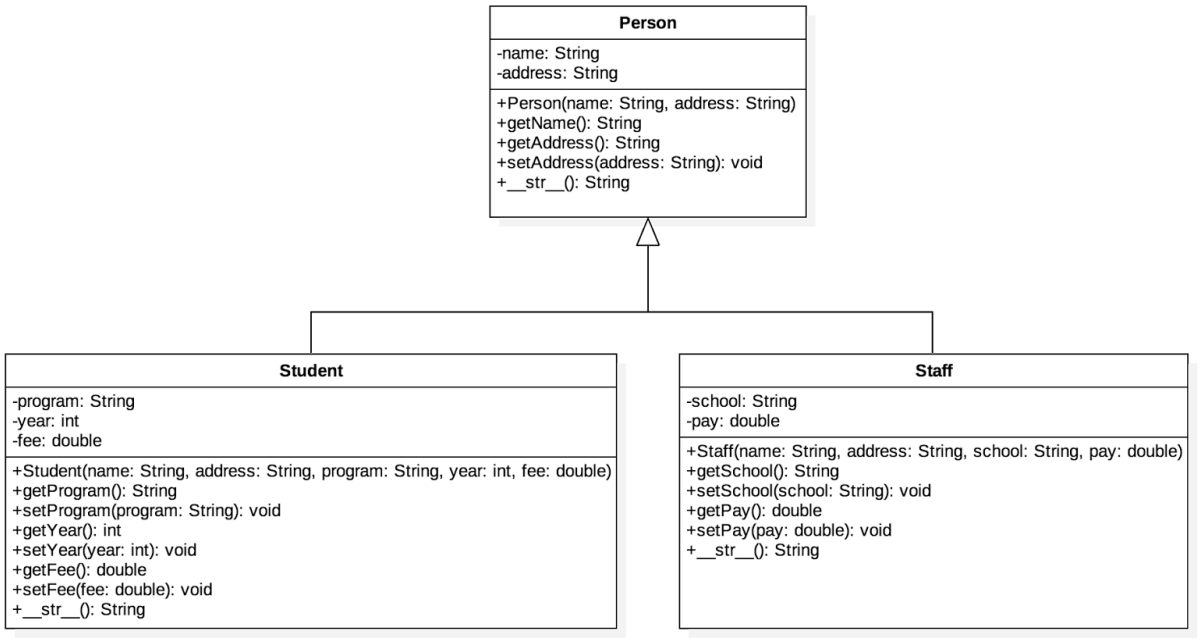
Inlämningsuppgift 2: OOP

Mål: implementera tre klasser med arv enligt given UML design.

Bakgrund:

MYH (Myndigheten för yrkeshögskolan) bygger ett nytt system för hantering av alla individer (anställda och studerande) i olika YH-skolor och vill att du ska bygga de första OOP klasserna.

UML Design:



Uppgiften är att följa designen och bygga tre klasser: Person, Student och Staff. Student och Staff ska ärva från Person (som pilen indikerar).

Del 1 (G-nivå)

Ni börjar med:

- En UML klass diagram (ovan)
- En *main.py* fil med ett antal enhetstester som testar alla metoder för Person, Student och Staff

Ni ska:

- Börja med ett projekt med min *main.py* (i PyCharm eller annat)
- Implementera samtliga tre klasser med korrekt:
    - Konstruktor
    - Attribut
    - Metoder
    - Arv
- Implementera samtliga klasser i egna Python filer (moduler) för att hålla isär de
    - Min main.py förväntar att filerna heter “person.py”, “student.py” och “staff.py” (med småbokstäver)

Hints:

- UML designen säger att “fee” ska vara en *double* - Python skiljer inte mellan *float* och *double* så double finns inte i språket. Det är med andra ord OK att hantera “fee” som en *float*.
- Använd super() för att komma åt basklassen på rätt sätt
- För att mina tester ska gå igenom måste ni implementera __str__() metoden i samtliga tre klasser så att de ger en string som ser ut på ett viss sätt. Dvs:
    - Person:
        - `Person[name=?,address=?]`
    - Student:
        - `Student[Person[name=?,address=?],program=?,year=?,fee=?]`
    - Staff:
        - `Staff[Person[name=?,address=?],school=?,pay=?]`

Del 2 (frivillig, VG-nivå)

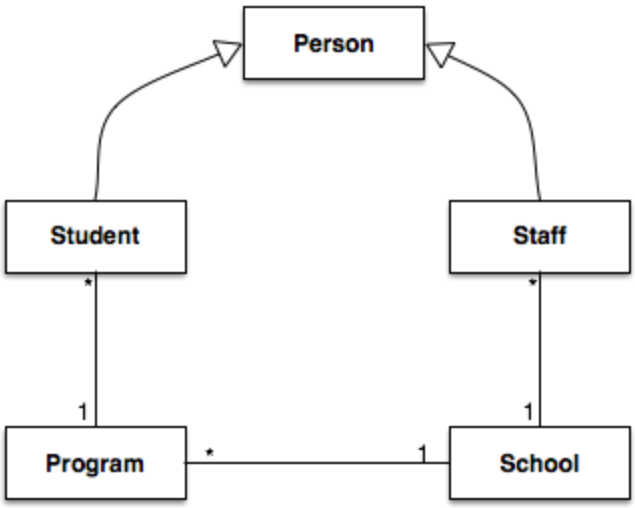
När du är färdig med del 1 så kan du fortsätta med del 2 (om du vill). Del 2 är svårare än del 1 då den kräver att ni själv designar några klasser för att lösa problemet utan en UML design - samt komma på hur du vill testa dina klasser.

Mål:

MYH vill nu testa dina tre klasser med en enkel applikation som hjälper räkna ut om en viss skola går med vinst eller förlust.

Ni börjar med:

- En fungerande del 1
- En skiss på klass relationer:



Ni ska:

- Designa alla attribut och metoder för två nya klasser:
    - Program
    - School
- Designa hur alla klasser hör ihop så att:
    - Varje School objekt kan ha ett antal Staff objekt (0 eller fler)
    - Varje School objekt kan ha ett antal Program objekt (0 eller fler)
    - Varje Program objekt har ett antal Student objekt (0 eller fler)
- School klassen bör ha en metod som räknar ut om den går med vinst eller förlust
    - Dvs: om summan av alla *fee* attribut för alla Student objekt är mer än summan av alla *pay* attribut för alla Staff objekt
- Skriva egna testfall (eller annan lösning) för att kunna testa koden som du har skrivit.
    - Det är OK att skippa unittest modulen och bara skriva lite Python kod som skapar alla objekt som du vill ha och sedan testa om skolan går med vinst eller förlust.
    - Förmodligen vill du testa båda fall (dvs när den går med vinst, samt när den går med förlust)

Hints:

- Använd “good OOP practices” - dvs, använd klasshierarkier och skapa metoder som hjälper dig lösa uppgiften.

    - Det vore fel om t.ex School hade all logik för att räkna ut intäkterna - det vore mer “OOP” om varje Program kunde räkna ut sina intäkter
- Det vore bra om t.ex School hade metoder för att kunna lägga till Program och Staff objekt dynamiskt
- Men i övrigt får ni inget mer hjälp med del 2 - tanken är att ni ska kunna lista ut vad som behövs och hur det ska testas! Från kursplanen för VG-nivå:
    - *Med mycket god förståelse och metodiskt kunna programmera och ta fram arkitekturer för applikationer*
    - ***Självständigt** kunna utveckla enligt anvisningar med det valda programmeringsspråket.*
- Lycka till!