Arv 2

vt 24

Övning

- Skapa klasserna `Person`, `Teacher`, `Student` och `Course`
- Klasserna `Teacher` och `Student` ska ärva från `Person`
- Till din hjälp har du klassdiagram på följande sidor
- Målet är att göra ett program som kan lägga till elever till kurser och ge dem beytg.
- Man ska även kunna räkna ut elevernas merit
- Sist i bland sidorna hittar du även hur olika utskrifter ska se ut

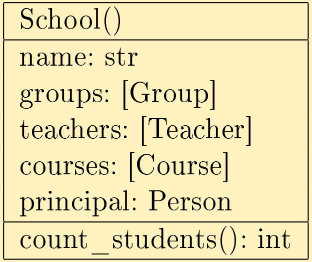
Person
name: str birth_year: int
__str__(): str __repr__():str

Course
name: str points: int students: [Student] teachers: [Teachers]
add_student(Student): void add_teacher(Teacher): void set_student_grade(Student, str) __repr__(): str

Teacher(Person)
name: str birth_year: int school: str subjects: [str]
__str__(): str

Student(Person)
name: str birth_year: int school: str grades: {[str,int]} group: str
calculate_merit(): float add_grade(course: str, grade: str, points: int) __str__(): str __repr__():str

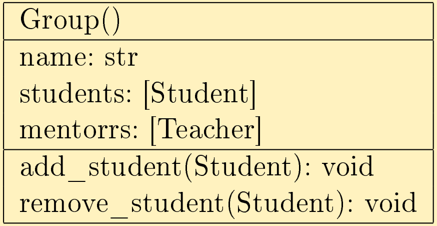
Klassdiagram



School()

name: str
groups: [Group]
teachers: [Teacher]
courses: [Course]
principal: Person

count_students(): int



Group()

name: str
students: [Student]
mentorrs: [Teacher]

add_student(Student): void
remove_student(Student): void

Utskrifter

```python
s = Student("Nisse", 2006, "Spyken", "Na3b")
t = Teacher("Calle", 1991, "Hedda", ["Matematik", "
    Programmering"])
c = Course("Programmering 1", 100)
c.add_teacher(t)

print(s)
print(t)
print(c)
```

```
Nisse 18 Spyken Na3b
Calle 33 Hedda Matematik Programmering
Programmering 1 100 (Calle)
```

Pseudokod

Person

```
class Person():
    func __init__(self, name, year):
        self.name := name
        self.birth_year := year
    func __str__(self):
        return self.name + " " + (2024-self.birth_year)
    func __repr__(self):
        return str(self)
```

Pseudokod

Course

```
class Course():
    def __init__(self, name, points):
        self.name := name
        self.points := points
        self.students := []
        self.teachers := []
    def add_student(self, stud):
        if stud not in self.students:
            self.students.append(stud)
    def add_teacher(self, teacher):
        if teacher not in self.teachers:
            self.teachers.append(teacher)
```

Fortsätter på nästa slide

Pseudokod

Course fortsättning

```
    def set_student_grade(self, stud, grade):
        for s in self.students:
            if s = stud:
                s.add_grade(self.name, [grade, self.points])

    def __repr__(self):
        out := self.name + " " + str(self.points) + "("
        for t in self.teachers:
            out := out + " "+t.name
        return out +")"
```

Pseudokod

Teacher

```
class Teacher(Person):

    def __init__(self, name, birth_year, school, subjects):
        super().__init__(name, birth_year)
        self.school := school
        self.subjects := subjects

    def __str__(self):
        out := super().__str__()
        for sub in self.subjects:
            out := out + " "+sub
        return out
```

Pseudokod

Student

```
class Student(Person):

    def __init__(self, name, birth_year, school, group):
        super().__init__(name, birth_year)

        self.school = school
        self.group = group

        self.grades = {}

    def add_grade(self, course, grade):
        self.grades[course] = grade
```

Pseudokod

Student fortsättning

```python
    def calculate_merit(self):
        system = {"A": 20, "B": 17.5, "C": 15, "D": 12.5, "E": 10, "F": 0}
        point_sum = 2400
        merit = 0
        for course in self.grades:
            if "Gymnasiearbete" not in course:
                merit += (system[self.grades[course][0]]*self.grades[course][1])/point_sum
        return merit
    def __str__(self):
        out = super().__str__() + " " + self.school + " " + self.group
        return out
```