รายงานเรื่อง

Basic education management system

เสนอ

อาจารย์ ศักดิ์ระพี ไพศาลนันทน์

จัดทำโดย

นายธนภัทร ธำรงค์สกุลศิริ 630910167

นายนฤสรณ์ เขียวชะอุ่ม 630910336

นายบัลลังก์ คุณโรจน์อังกูร 630910340

นายวรภพ สินบุญยะมะ 630910359

รายงานนี้ เป็นส่วนหนึ่ง ของรายวิชา 618445 การออกแบบระบบเชิงวัตถุสำหรับวิศวกร

ปีการศึกษา 2566/2

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์และระบบคอมพิวเตอร์

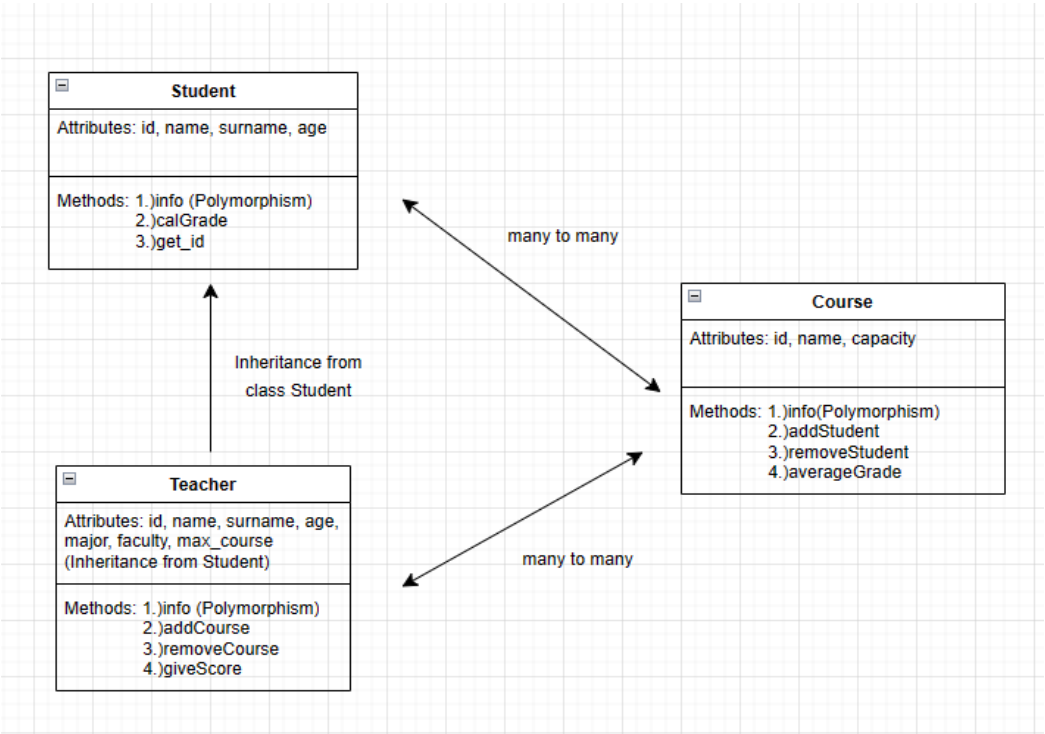
ภาควิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม

มหาวิทยาลัยศิลปากร

Description

UML Design



Student 1 คน สามารถลงเรียนได้หลาย Course และ Course 1 Course ก็สามารถมี Student ได้หลายคน (many to many)

Teacher 1 คน สามารถสอนได้หลาย Course และ Course 1 Course ก็สามารถมี Teacher ได้หลายคน (หลายกลุ่มเรียน) (many to many)

คลาส Student มี Attributes 5 Attributes ดังภาพ โดย self.__id จะ encapsulate ไว้

```python
class Student:
    def __init__(self, id, name, surname, age):
        self.__id = id
        self.name = name
        self.surname = surname
        self.age = age
        self.grade = {}
```

และมี methods 3 methods ดังนี้

```python
def info(self):
    print("###### Student info ######")
    print(f"Student ID: {Student.get_id(self)}\nStudent name \"{self.name} {self.surname}\" and {self.age} years old.")
    print(f"Course enrolled: {self.grade}\n")
```

1. info(self) จะแสดงข้อมูลของ Student (methods นี้ เป็น Polymorphism คือมีชื่อ methods เหมือนกัน ใน class อื่นๆ แต่แสดงผลต่างกัน)

Example

```
###### Student info ######
Student ID: 550
Student name "Man Above" and 19 years old.
Course enrolled: {'S-10': {'Course': 'Physic', 'Grade': 85}, 'G-50': {'Course': 'Social', 'Grade': 37}}
```

โดย attributes self.grade จะเก็บข้อมูลเป็นลักษณะ Nested Dictionary ซึ่งจะถูกเพิ่มข้อมูลผ่าน Methods giveScore ของ Class Teacher

2. calGrade เช็ค คะแนนของ Course ที่ระบุและ แสดงเกรดที่ได้รับใน Course นั้นๆ

```python
def calGrade(self, course):
    if course.id in self.grade:
        if self.grade[course.id]["Grade"] >= 80 and self.grade[course.id]["Grade"] <= 100:
            print(f"\"{self.name} {self.surname}\" got grade \"A\" in course \"{course.name}\" .")
        elif self.grade[course.id]["Grade"] >= 70:
            print(f"\"{self.name} {self.surname}\" got grade \"B\" in course \"{course.name}\" .")
        elif self.grade[course.id]["Grade"] >= 60:
            print(f"\"{self.name} {self.surname}\" got grade \"C\" in course \"{course.name}\" .")
        elif self.grade[course.id]["Grade"] >= 50:
            print(f"\"{self.name} {self.surname}\" got grade \"D\" in course \"{course.name}\" .")
        else:
            print(f"\"{self.name} {self.surname}\" got grade \"F\" in course \"{course.name}\" .")
    else:
        print(f"\"{self.name} {self.surname}\" didn't enroll course \"{course.name}\" .")
```

Example

```
"Mary Fox" got grade "C" in course "Physic" .
"Mary Fox" got grade "A" in course "Math" .
"Mary Fox" got grade "F" in course "English" .
"Mary Fox" got grade "B" in course "Social" .
"Mary Fox" didn't enroll course "Industial"
```

3. get_id ใช้สำหรับ รับค่า self.__id ที่ถูก encapsulate ไว้

```python
def get_id(self):
    return self.__id
```

class Teacher เป็น sub-class ของ Student (Inheritance from class Student) ซึ่งจะมี Attributes เพิ่มมาคือ major, faculty, max_course

คลาส Teacher เป็น sub-class ของ Student และ มี Attributes เพิ่ม 4 Attributes โดย self.teach จะแสดงวิชาที่สอนซึ่งเก็บข้อมูลแบบ list

```
class Teacher(Student):
    def __init__(self, id, name, surname, age, major, faculty, max_course):
        super().__init__(id, name, surname, age)
        self.major = major
        self.faculty = faculty
        self.teach = []
        self.max_course = max_course
```

และมี 4 methods ดังนี้

1.) info(self) จะแสดงข้อมูลของ Teacher (methods นี้ เป็น Polymorphism คือมีชื่อ methods เหมือนกัน ใน class อื่นๆ แต่แสดงผลต่างกัน)

```
def info(self):
    print("###### Teacher info ######")
    print(f"Teacher ID: {Teacher.get_id(self)}, Teacher name \"{self.name} {self.surname}\" and {self.age} years
    print("Course responsibilty:")
    for x in range(0,len(self.teach)):
        print(" -",self.teach[x].name)
    print("")
```

Example

```
###### Teacher info ######
Teacher ID: 100-00, Teacher name "Gary Alpha" and 32 years old
From faculty Sciene, Major Math, Max course: 3
Course responsibilty:
 - Physic
 - Math
 - English
```

2.) addCourse จะทำการเพิ่ม instance ของคลาส Course ที่กำหนดลงใน self.teach

```python
def addCourse(self, course):
    if self.max_course <= len(self.teach):
        print("Schedule of teacher is full!")
    else:
        self.teach.append(course)
        print("Course added successfully")
```

3.) removeCourse จะทำการลบ instance ของคลาส Course ที่กำหนดออกจาก self.teach

```python
def removeCourse(self, course):
    for x in range(0, len(self.teach)):
        if course == self.teach[x]:
            print(f"Course \"{course.name}\" have been remove from schedule of teacher name \"{self.name}\" .")
            del self.teach[x]
            return True
    print("Course isn't found!")
```

4.) giveScore จะทำการให้คะแนน student.grade โดย Check ผ่าน Course ที่ Teacher สอน และ Student ต้องลง Course นั้น

```python
def giveScore(self, course, student, score):
    for x in range(0, len(self.teach)):
        if course == self.teach[x]:
            for i in range(0, len(course.student)):
                if course.student[i] == student:
                    student.grade[course.id] = {"Course" : course.name, "Grade" : score }
                    print(f"Score added successfully")
                    return None
            print(f"\"{student.name} {student.surname}\" isn't enroll for course \"{course.name}\"")
            return None
    print(f"Course \"{course.name}\" is out of teacher \"{self.name} {self.surname}\" responsibility!")
    return None
```

Example

```
Score added successfully
Score added successfully
Score added successfully
Course "Social" is out of teacher "Gary Alpha" responsibility!
Score added successfully
"Jimmy Geo" isn't enroll for course "Physic"
Course "Physic" is out of teacher "German Oliver" responsibility!
"Mary Fox" isn't enroll for course "Industial"
```

คลาส Course มี Attributes 4 Attributes ดังภาพ โดย self.student จะเก็บข้อมูล student ที่ลงเรียน Course แบบ list

```
class Course:
    def __init__(self, id, name, capacity):
        self.id = id
        self.name = name
        self.capacity = capacity
        self.student = []
```

มี 4 methods ดังนี้

1.) info(self) จะแสดงข้อมูลของ Course (methods นี้ เป็น Polymorphism คือมีชื่อ methods เหมือนกัน ใน class อื่นๆ แต่แสดงผลต่างกัน)

```
def info(self):
    print("####### Course info #######")
    print(f"Course ID: {self.id}, Course name: {self.name}\nCourse capacity: {self.capacity}")
    print("Students enroll this course:")
    for x in range(0,len(self.student)):
        print(" -",self.student[x].name ,self.student[x].surname)
    print("")
```

Example

```
####### Course info #######
Course ID: S-10, Course name: Physic
Course capacity: 3
Students enroll this course:
 - Bill Gate
 - Mary Fox
 - Man Above
```

2.) addStudent จะทำการเพิ่ม Student ลงใน self.student

```python
def addStudent(self, student):
    if self.capacity <= len(self.student):
        print("Course is full right now.")
    else:
        self.student.append(student)
        print("Student added successfully.")
```

Example

```
Student added successfully.
Student added successfully.
Student added successfully.
Student added successfully.
Student added successfully.
Course is full right now.
```

3.) removeStudent ทำการลบ student ออกจาก self.student

```python
def removeStudent(self, student):
    for x in range(0, len(self.student)):
        if student == self.student[x]:
            print(f"Student name \"{student.name} {student.surname}\" have been remove from course name \"{self.name}\" .")
            del self.student[x]
            return True
    print("Student isn't found!")
```

4.) averageGrade จะทำการคำนวณคะแนนเฉลี่ยของ Course

```python
def averageGrade(self): # Careful! make sure you give score to all student in the course before call this method.
    averageGrade = 0
    num = 0
    for x in range(0, len(self.student)):
        if self.student[x].grade[self.id]["Course"] == self.name:
            averageGrade = averageGrade + self.student[x].grade[self.id]["Grade"]
            num += 1
    if num == 0:
        return None
    else:
        print(f"Average grade of course \"{self.name}\" is {averageGrade/num} .")
        return averageGrade/num
```

ข้อควรระวังคือ ต้องทำการเพิ่มคะแนนให้กับ student ที่ลงใน Course ครบทุกคนก่อนจึงจะสามารถเรียกใช้ methods นี้

Example

```
Average grade of course "Physic" is 65.0 .
Average grade of course "Social" is 66.2 .
Average grade of course "English" is 11.0 .
```

สามารถดูตัวอย่าง Code ได้ที่ main.py