AI VIET NAM – COURSE 2024

Object Oriented Programming - Day 12

Ngày 22 tháng 6 năm 2024

Ngày thực hiện:	22/06/2024
Người thực hiện:	Đinh Thị Tâm
Nguồn:	AIO2024 - Day 12
Nguồn dữ liệu (nếu có):	Link Sources code
Từ khóa:	Object Oriented Programming
Người tóm tắt:	Đinh Thị Tâm

1. Mô tả
   Lập trình hướng đối tượng (OOP) là một phương pháp lập trình tập trung vào việc tạo ra các "đối tượng", là những đơn vị cơ bản đại diện cho các thực thể trong thế giới thực. Mỗi đối tượng có các thuộc tính (data) và phương thức (hành vi) riêng. OOP mang lại nhiều lợi ích cho việc lập trình, bao gồm:

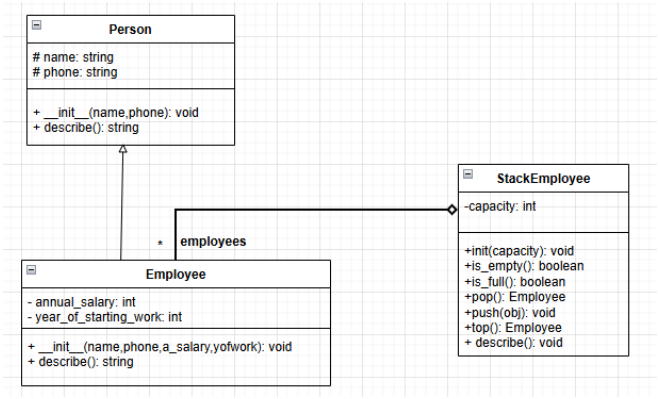
   - Tái sử dụng mã: OOP cho phép bạn viết mã có thể được sử dụng lại nhiều lần cho các mục đích khác nhau. Điều này giúp tiết kiệm thời gian và công sức, đồng thời giúp mã dễ dàng bảo trì hơn.
   - Tính linh hoạt: OOP giúp bạn dễ dàng thay đổi và mở rộng mã của mình. Khi bạn cần thêm tính năng mới, bạn chỉ cần tạo ra các đối tượng mới hoặc sửa đổi các đối tượng hiện có.
   - Tính bảo trì: OOP giúp mã của bạn dễ đọc và dễ hiểu hơn. Điều này giúp bạn dễ dàng tìm và sửa lỗi, đồng thời giúp những người khác dễ dàng hiểu mã của bạn.

2. **Bài tập:** Tạo lớp `Person` có thuộc tính là `name`, `phone` và lớp `Employee` kế thừa từ lớp `Person` có các thuộc tính riêng như: `annual_salary`, `year_of_starting_work`. Hãy tạo class mới có tên `StackEmployee` gồm các phương thức sau:

   - `initialization method` nhận một input `capacity`: dùng để khởi tạo stack với `capacity` là số lượng element mà stack có thể chứa
   - `is_empty()` kiểm tra stack có đang rỗng
   - `is_full()` kiểm tra stack đã đầy chưa
   - `pop()` loại bỏ top element và trả về giá trị đó
   - `push(obj)` thêm object vào trong stack
   - `top()` lấy giá trị top element hiện tại của stack, nhưng không loại bỏ giá trị đó

   Hãy thực hiện kiểm tra với `StackEmployee` với `capacity` là 5, và vẽ biểu đồ UML Class diagram.

   (a) sơ đồ class



Hình 1: Sơ đồ UML

(b) Code

```
class Person():
    def __init__(self, name, phone):
        self._name = name
        self._phone = phone

    def describe(self):
        return ('Name: {}, phone: {}').format(self._name, self._phone)

    def __eq__(self, other):
        if isinstance(other, Person):
            return self._phone == other._phone
        return False

    def __hash__(self):
        return hash(self.phone)


class Employee(Person):
    def __init__(self, name, phone, annual_salary, year_of_starting_work):
        super().__init__(name, phone)
        self.__annual_salary = annual_salary
        self.__year_of_starting_work = year_of_starting_work

    def describe(self):
        info = 'Employee: ' + super().describe()
        info = info + (' annual_salary: {}, year_of_starting_work: {}').
    format(
            self.__annual_salary, self.__year_of_starting_work)
        return info


class StackEmployee():
    def __init__(self, capacity=5):
        self.__capacity = capacity
        self.__employees = []

    def is_full(self):
        return len(self.__employees) == self.__capacity

    def is_empty(self):
        return len(self.__employees) == 0
```

```
41
42      def pop(self):
43          return self.__employees.pop(-1)
44
45      def push(self, obj):
46          if self.is_full():
47              print('Stack is full. Do not push obj')
48          elif self.__employees.__contains__(obj):
49              print('This obj have same phone with another in stack. Do not
    push obj')
50          else:
51              self.__employees.append(obj)
52
53      def top(self):
54          if self.is_empty():
55              print('Stack is empty. No element in stack')
56          else:
57              return self.__employees[-1]
58
59      def describe(self):
60          for x in self.__employees:
61              print(x.describe())
62
63
64  # test
65  stack_employee = StackEmployee()
66  employee1 = Employee('An', '0912345678', 250, 2002)
67  employee2 = Employee('Binh', '0912225678', 500, 2001)
68  employee3 = Employee('Nghia', '0912225678', 350, 2002)
69  employee4 = Employee('Tin', '0912335678', 700, 2000)
70  employee5 = Employee('Tai', '0914445678', 850, 2003)
71  employee6 = Employee('Binh', '0955225678', 900, 2004)
72  employee7 = Employee('Binh', '0956225678', 800, 2002)
73  # push to stack
74  stack_employee.push(employee1)
75  stack_employee.push(employee2)
76  stack_employee.push(employee3)
77  stack_employee.push(employee4)
78  stack_employee.push(employee5)
79  stack_employee.push(employee6)
80  stack_employee.push(employee7)
81  # print stack
82  print('-'*70)
83  print('List element of stack after push 6 element is: ')
84  stack_employee.describe()
85  # pop
86  obj = stack_employee.pop()
87  print('-'*70)
88  print('Element on top of stack is: \n   ', obj.describe())
89  print('-'*70)
90  print('List element of stack after pop element is: ')
91  stack_employee.describe()
92  # top
93  obj = stack_employee.top()
94  print('-'*70)
95  print('Element on top of stack is: \n', obj.describe())
96  print('List element of stack is: ')
97  stack_employee.describe()
```

(c) Kết quả

```
This obj have same phone with another in stack. Do not push obj
Stack is full. Do not push obj
------------------------------------------------------------------------
List element of stack after push 6 element is:
Employee: Name: An, phone: 0912345678 annual_salary: 250, year_of_starting_work: 2002
Employee: Name: Binh, phone: 0912225678 annual_salary: 500, year_of_starting_work: 200
Employee: Name: Tin, phone: 0912335678 annual_salary: 700, year_of_starting_work: 2000
Employee: Name: Tai, phone: 0914445678 annual_salary: 850, year_of_starting_work: 2003
Employee: Name: Binh, phone: 0955225678 annual_salary: 900, year_of_starting_work: 200
------------------------------------------------------------------------
Element on top of stack is:
    Employee: Name: Binh, phone: 0955225678 annual_salary: 900, year_of_starting_work:
------------------------------------------------------------------------
List element of stack after pop element is:
Employee: Name: An, phone: 0912345678 annual_salary: 250, year_of_starting_work: 2002
Employee: Name: Binh, phone: 0912225678 annual_salary: 500, year_of_starting_work: 200
Employee: Name: Tin, phone: 0912335678 annual_salary: 700, year_of_starting_work: 2000
Employee: Name: Tai, phone: 0914445678 annual_salary: 850, year_of_starting_work: 2003
------------------------------------------------------------------------
Element on top of stack is:
 Employee: Name: Tai, phone: 0914445678 annual_salary: 850, year_of_starting_work: 200
List element of stack is:
Employee: Name: An, phone: 0912345678 annual_salary: 250, year_of_starting_work: 2002
Employee: Name: Binh, phone: 0912225678 annual_salary: 500, year_of_starting_work: 200
Employee: Name: Tin, phone: 0912335678 annual_salary: 700, year_of_starting_work: 2000
Employee: Name: Tai, phone: 0914445678 annual_salary: 850, year_of_starting_work: 2003
```