AI VIET NAM – Warmup 2025

# Python OOP – Exercise

Ngày 22 tháng 2 năm 2025

## 1 Giới Thiệu Về Lập Trình Hướng Đối Tượng (OOP)

Lập trình hướng đối tượng (Object-Oriented Programming - OOP) là một phương pháp lập trình tổ chức code thành các đối tượng có thuộc tính (data) và phương thức (hành vi). OOP giúp cho việc tái sử dụng, mở rộng, và bảo trì code dễ dàng hơn. Python là một ngôn ngữ hỗ trợ OOP một cách linh hoạt và hiệu quả.

## 2 Bốn Nguyên Tắc Của OOP

- **Đóng gói (Encapsulation)**: Giới hạn truy cập vào dữ liệu, chỉ cho phép truy cập thông qua các phương thức của class.
- **Kế thừa (Inheritance)**: Cho phép một class sử dụng lại code từ class khác.
- **Đa hình (Polymorphism)**: Cùng một phương thức nhưng hoạt động khác nhau dựa trên đối tượng sử dụng.
- **Trừu tượng (Abstraction)**: Chỉ cung cấp những gì cần thiết và ẩn bớt những chi tiết không quan trọng.

## 3 Cấu Trúc Cơ Bản Của OOP Trong Python

### 3.1 Lớp (Class) và Đối Tượng (Object)

Lớp là khuôn mẫu chứa thuộc tính và phương thức. Đối tượng là một thể hiện của lớp.

```
class Animal:
def init(self, name):
self.name = name

def make_sound(self):
    return "Some generic sound"

Tạo đối tượng từ lớp Animal

dog = Animal("Buddy")
print(dog.name)  # Output: Buddy
print(dog.make_sound())  # Output: Some generic sound
```

### 3.2 Đóng Gói (Encapsulation)

```
class BankAccount:
def init(self, owner, balance):
self.owner = owner
self.__balance = balance  # Thuộc tính private

def deposit(self, amount):
    self.__balance += amount
    return f"Deposited {amount}, New balance: {self.__balance}"

Sử dụng class

account = BankAccount("Alice", 1000)
print(account.deposit(500))  # Output: Deposited 500, New balance: 1500
```

### 3.3 Kế Thừa (Inheritance)

```
class Dog(Animal):  # Lớp Dog kế thừa từ Animal
def make_sound(self):
return "Woof! Woof!"

dog = Dog("Max")
print(dog.make_sound())  # Output: Woof! Woof!
```

### 3.4 Đa Hình (Polymorphism)

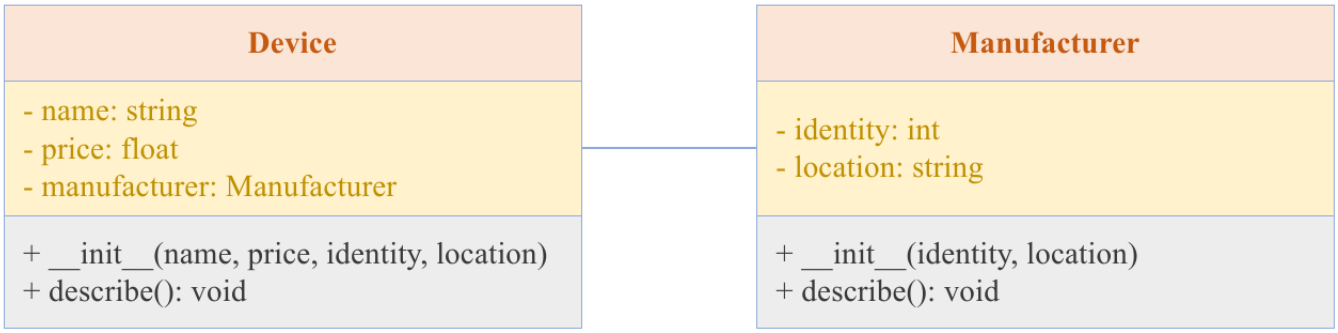
```
class Cat(Animal):
def make_sound(self):
return "Meow! Meow!"

animals = [Dog("Buddy"), Cat("Whiskers")]
for animal in animals:
print(f"{animal.name} says {animal.make_sound()}")
```

### 3.5 Trừu Tượng (Abstraction)

```
from abc import ABC, abstractmethod

class Vehicle(ABC):
@abstractmethod
def move(self):
pass

class Car(Vehicle):
def move(self):
return "Car is moving on the road"

class Boat(Vehicle):
def move(self):
return "Boat is sailing on the water"

car = Car()
boat = Boat()
print(car.move())  # Output: Car is moving on the road
print(boat.move())  # Output: Boat is sailing on the water
```

# 4 Bài Tập

1. **Thực hiện theo yêu cầu như class diagram bên dưới.**




Hình 1: Class Diagram

```
# Examples 1
device1 = Device(name="mouse", price=2.5, identity=9725, location="Vietnam")
device1.describe()
>> Name: mouse - Price: 2.5
Identity: 9725 - Location: Vietnam

device2 = Device(name="monitor", price=12.5, identity=11, location="Germany")
device2.describe()
>> Name: monitor - Price: 12.5
Identity: 11 - Location: Germany
```

2. **Một Ward gồm có name (string) và danh sách của mọi người trong Ward. Một người person có thể là student, doctor, hoặc teacher. Một student gồm có name, yob (int) (năm sinh), và grade (string). Một teacher gồm có name, yob, và subject (string). Một doctor gồm có name, yob, và specialist (string). Lưu ý cần sử dụng a list để chứa danh sách của mọi người trong Ward.**

   (a) Thực hiện các class student, doctor, và teacher theo mô tả trên. Thực hiện describe() method để print ra tất cả thông tin của các objects.

   (b) Viết addPerson(person) method trong Ward class để add thêm một người mới với nghề nghiệp bất kỳ (student, teacher, doctor) vào danh sách người của ward. Tạo ra một ward object, và thêm vào 1 student, 2 teacher, và 2 doctor. Thực hiện describe() method để in ra tên ward (name) và toàn bộ thông tin của từng người trong ward.

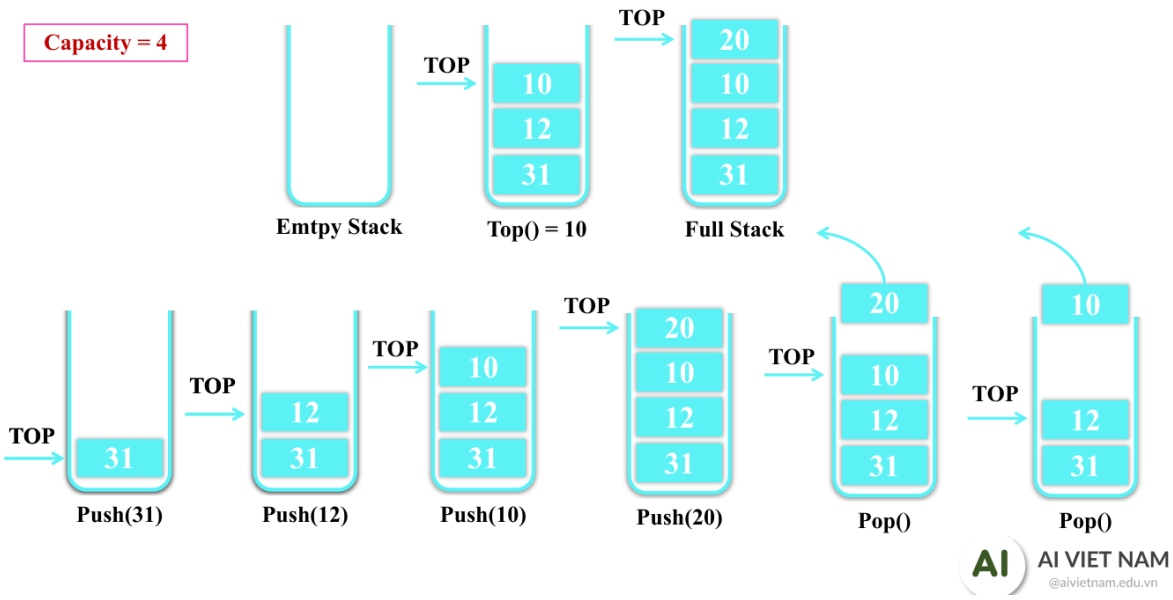
   (c) Viết countDoctor() method để đếm số lượng doctor trong ward.

   (d) Viết sortAge() method để sort mọi người trong ward theo tuổi của họ với thứ tự tăng dần. (hint: Có thể sử dụng sort của list hoặc viết thêm function đều được)

   (e) Viết aveTeacherYearOfBirth() method để tính trung bình năm sinh của các teachers trong ward.

```
# Examples
# 2(a)
student1 = Student(name="studentA", yob=2010, grade="7")
student1.describe()
#output
>> Student - Name: studentA - YoB: 2010 - Grade: 7
```

```
7
8  teacher1 = Teacher(name="teacherA", yob=1969, subject="Math")
9  teacher1.describe()
10 #output
11 >> Teacher - Name: teacherA - YoB: 1969 - Subject: Math
12
13 doctor1 = Doctor(name="doctorA", yob=1945, specialist="Endocrinologists")
14 doctor1.describe()
15 #output
16 >> Doctor - Name: doctorA - YoB: 1945 - Specialist: Endocrinologists
17
18
19 # 2(b)
20 print()
21 teacher2 = Teacher(name="teacherB", yob=1995, subject="History")
22 doctor2 = Doctor(name="doctorB", yob=1975, specialist="Cardiologists")
23 ward1 = Ward(name="Ward1")
24 ward1.addPerson(student1)
25 ward1.addPerson(teacher1)
26 ward1.addPerson(teacher2)
27 ward1.addPerson(doctor1)
28 ward1.addPerson(doctor2)
29 ward1.describe()
30
31 #output
32 >> Ward Name: Ward1
33 Student - Name: studentA - YoB: 2010 - Grade: 7
34 Teacher - Name: teacherA - YoB: 1969 - Subject: Math
35 Teacher - Name: teacherB - YoB: 1995 - Subject: History
36 Doctor - Name: doctorA - YoB: 1945 - Specialist: Endocrinologists
37 Doctor - Name: doctorB - YoB: 1975 - Specialist: Cardiologists
38
39 # 2(c)
40 print(f"\nNumber of doctors: {ward1.countDoctor()}")
41
42 #output
43 >> Number of doctors: 2
44
45 # 2(d)
46 print("\nAfter sorting Age of Ward1 people")
47 ward1.sortAge()
48 ward1.describe()
49
50 #output
51 >> After sorting Age of Ward1 people
52 Ward Name: Ward1
53 Student - Name: studentA - YoB: 2010 - Grade: 7
54 Teacher - Name: teacherB - YoB: 1995 - Subject: History
55 Doctor - Name: doctorB - YoB: 1975 - Specialist: Cardiologists
56 Teacher - Name: teacherA - YoB: 1969 - Subject: Math
57 Doctor - Name: doctorA - YoB: 1945 - Specialist: Endocrinologists
58
59 # 2(e)
60 print(f"\nAverage year of birth (teachers): {ward1.aveTeacherYearOfBirth()}")
61
62 #output
63 >> Average year of birth (teachers): 1982.0
```

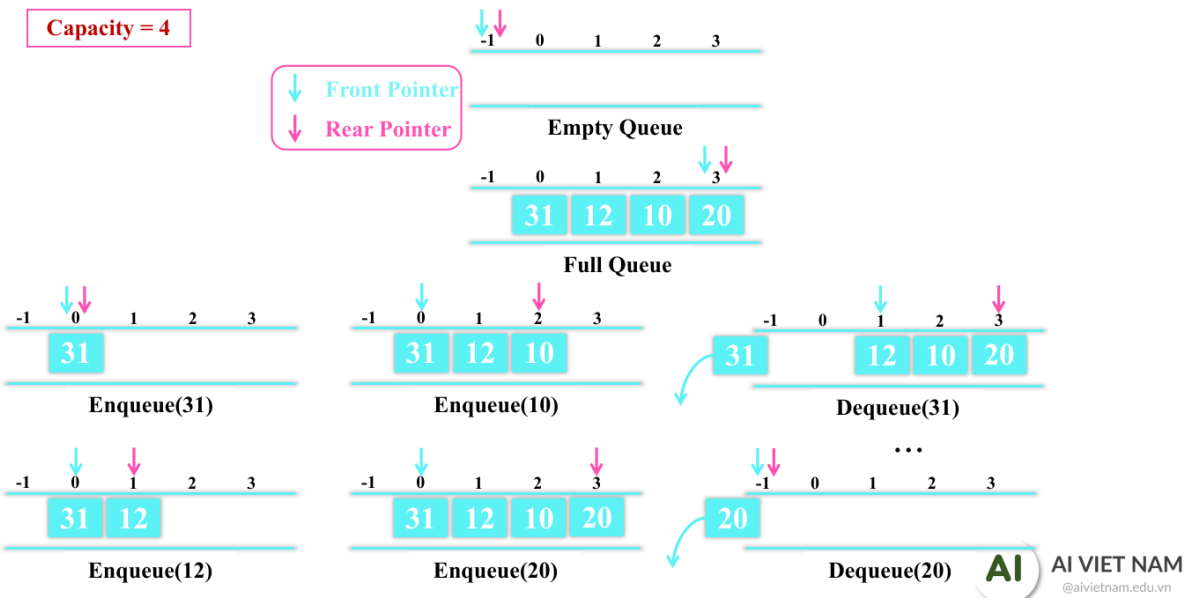



Hình 2: Stack

3. **Thực hiện xây dựng class Stack với các chức năng (method) sau đây**

   - **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa
   - **.isEmpty()**: kiểm tra stack có đang rỗng
   - **.isFull()**: kiểm tra stack đã full chưa
   - **.pop()**: loại bỏ top element và trả về giá trị đó
   - **.push(value)** add thêm value vào trong stack
   - **.top()** lấy giá trị top element hiện tại của stack, nhưng không loại bỏ giá trị đó
   - **Không cần thiết phải thực hiện với pointer như trong hình minh họa**

```
stack1 = MyStack(capacity=5)

stack1.push(1)

stack1.push(2)

print(stack1.isFull())
>> False

print(stack1.top())
>>2

print(stack1.pop())
>> 2

print(stack1.top())
>> 1

print(stack1.pop())
>> 1

print(stack1.isEmpty())
>> True
```




Hình 3: Queue

4. **Thực hiện xây dựng class Queue với các chức năng (method) sau đây**

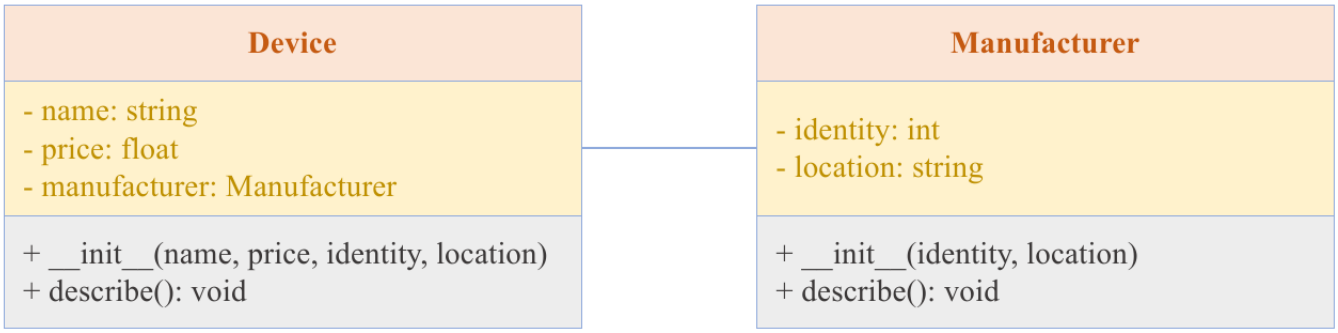
   - **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa
   - **.isEmpty()**: kiểm tra queue có đang rỗng
   - **.isFull()**: kiểm tra queue đã full chưa
   - **.dequeue()**: loại bỏ first element và trả về giá trị đó
   - **.enqueue(value)** add thêm value vào trong queue
   - **.front()** lấy giá trị first element hiện tại của queue, nhưng không loại bỏ giá trị đó
   - **Không cần thiết phải thực hiện với các pointers như trong hình minh họa**

```
queue1 = MyQueue(capacity=5)

queue1.enqueue(1)

queue1.enqueue(2)

print(queue1.isFull())
>> False

print(queue1.front())
>> 1

print(queue1.dequeue())
>> 1

print(queue1.front())
>> 2

print(queue1.dequeue())
>> 2

print(queue1.isEmpty())
>> True
```

## II. Câu hỏi trắc nghiệm

- Đọc tự luận trước để nắm được idea tổng quát (sẽ không yêu cầu nhưng khuyến khích các bạn tự làm tự luận) và các bài này sẽ được giải trong buổi TA.
- Các bạn phải làm phần trắc nghiệm
  - Các câu hỏi có ký hiệu **(LT)**: là các câu hỏi lý thuyết và/hoặc đã có sẵn trong file hint nên các bạn chỉ cần hiểu và chạy lại để chọn được đáp án đúng
  - Các câu hỏi có ký hiệu **(Code)**: là câu hỏi yêu cầu các bạn phải trực tiếp code vào phần bị khuyết để có thể chọn được đáp án đúng
  - **Lưu ý:** Đối với dạng câu hỏi **(Code)** trong file hint luôn có 1 test casse bắt đầu với từ khóa assert nếu các bạn chạy không báo lỗi có nghĩa các bạn đã vượt qua được test case này và chạy lệnh tiếp theo để trả lời câu hỏi trắc nghiệm
  - **Lưu ý:** Đọc kỹ các code gợi ý và code ví dụ mẫu ở tự luận có thể sẽ có ích cho các bạn khi làm trắc nghiệm

**Câu hỏi 1 (LT)** : Theo thiết kế này thì mối quan hệ giữa 2 class này được gọi là gì?




Hình 4: Class Diagram

a) Aggregation
b) Composition
c) Inheritance
d) Tất cả đều sai

**Câu hỏi 2 (LT)** : Đầu ra của chương trình sau đây là gì?

```
class Manufacturer:
    def __init__(self, identity:int, location: str ):
        self.__identity = identity
        self.__location = location

    def describe(self):
        print(f"Identity: {self.__identity} - Location: {self.__location}")

manu1 = Manufacturer(identity=100, location='Vietnam')
manu1.describe()
```

a) Identity: 100 - Location: Vietnam
b) Identity: Vietnam - Location: 100
c) None
d) Raise an error

**Câu hỏi 3 (LT)** : Đoạn code sau đây thể hiện mối quan hệ gì?

```
class Manufacturer:
    def __init__(self, identity:int, location: str ):
        self.__identity = identity
        self.__location = location

    def describe(self):
        print(f"Identity: {self.__identity} - Location: {self.__location}")


class Device:
    def __init__(self, name:str, price:float, identity:int, location:str ):
        self.__name = name
        self.__price = price
        self.__manufacturer = Manufacturer(identity, location)

    def describe(self):
        print(f"Name: {self.__name} - Price: {self.__price}")
        self.__manufacturer.describe()
device1 = Device(name="touchpad", price=3.3, identity=1111, location="Vietnam")
device1.describe()
```

a) Aggregation
b) Composition
c) Inheritance
d) Tất cả đều sai

**Câu hỏi 4 (LT)** : Đoạn code sau đây thể hiện mối quan hệ gì?

```
class Manufacturer:
    def __init__(self, identity:int, location: str ):
        self.__identity = identity
        self.__location = location

    def describe(self):
        print(f"Identity: {self.__identity} - Location: {self.__location}")


class Device:
    def __init__(self, name:str, price:float, manu:Manufacturer):
        self.__name = name
        self.__price = price
        self.__manufacturer = manu

    def describe(self):
        print(f"Name: {self.__name} - Price: {self.__price}")
        self.__manufacturer.describe()

manu1 = Manufacturer(identity=1111, location='Vietnam')
device1 = Device(name="touchpad", price=3.3, manu=manu1)
device1.describe()
```

a) Aggregation

b) Composition
c) Inheritance
d) Tất cả đều sai

**Câu hỏi 5** (Code) : Một người (person) có thể là student, doctor, hoặc teacher. Một student gồm có name (string), yob (int) (năm sinh), và grade (string). Các bạn thực hiện viết class Student theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object) và kết quả đầu ra là gì? Chọn đáp án đúng nhất bên dưới.

```
from abc import ABC, abstractmethod

class Person(ABC):
    def __init__(self, name:str, yob:int):
        self._name = name
        self._yob = yob

    def getYoB(self):
        return self._yob

    @abstractmethod
    def describe(self):
        pass


class Student(Person):
    def __init__(self, name:str, yob:int, grade:str):
        #########YOUR CODE HERE#########

        ################################
    def describe(self):
        #########YOUR CODE HERE#########

        ################################

student1 = Student(name="studentZ2023", yob=2011, grade="6")
student1.describe()
```

a) Student - Name: studentZ2023 - YoB: 2011 - Grade: 6
b) Student - Name: studentZ2023 - YoB: 6 - Grade: 2011
c) Student - Name: 6 - YoB: studentZ2023 - Grade: 2011
d) Tất cả đều sai

**Câu hỏi 6** (Code) : Một người (person) có thể là student, doctor, hoặc teacher. Một teacher gồm có name (string), yob (int), và subject (string). Các bạn thực hiện viết class Teacher theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object) và kết quả đầu ra là gì? Chọn đáp án đúng nhất bên dưới.

```
from abc import ABC, abstractmethod

class Person(ABC):
    def __init__(self, name:str, yob:int):
        self._name = name
        self._yob = yob

    def getYoB(self):
        return self._yob

    @abstractmethod
```

```
    def describe(self):
        pass


class Teacher(Person):
    def __init__(self, name:str, yob:int, subject:str):
        #########YOUR CODE HERE#########

        ################################

    def describe(self):
        #########YOUR CODE HERE#########

        ################################

teacher1 = Teacher(name="teacherZ2023", yob=1991, subject="History")
teacher1.describe()
```

a) Teacher - Name: teacherZ2023 - YoB: 1991 - Subject: History
b) Teacher - Name: 1991 - YoB: teacherZ2023 - Subject: History
c) Teacher - Name: History - YoB: teacherZ2023 - Subject: 1991
d) Tất cả đều sai

**Câu hỏi 7** (Code) : Một người (person) có thể là student, doctor, hoặc teacher. Một doctor gồm có name (string), yob (string), và specialist (string). Các bạn thực hiện viết class Teacher theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object) và kết quả đầu ra là gì? Chọn đáp án đúng nhất bên dưới.

```
from abc import ABC, abstractmethod

class Person(ABC):
    def __init__(self, name:str, yob:int):
        self._name = name
        self._yob = yob

    def getYoB(self):
        return self._yob

    @abstractmethod
    def describe(self):
        pass


class Doctor(Person):
    def __init__(self, name:str, yob:int, specialist:str):
        #########YOUR CODE HERE#########

        ################################

    def describe(self):
        #########YOUR CODE HERE#########

        ################################

doctor1 = Doctor(name="doctorZ2023", yob=1981, specialist="Endocrinologists")
doctor1.describe()
```

a) Doctor - Name: doctorZ2023 - YoB: 1981 - Specialist: Endocrinologists
b) Doctor - Name: 1981 - YoB: doctorZ2023 - Specialist: Endocrinologists

c) Doctor - Name: Endocrinologists - YoB: 1981 - Specialist: doctorZ2023
d) Tất cả đều sai

**Câu hỏi 8** (Code) : Một Ward gồm có name (string) và danh sách của mọi người trong Ward. Một người person có thể là student, doctor, hoặc teacher và cần sử dụng một list để chứa danh sách của mọi người trong Ward. Viết addPerson(person) method trong Ward class để add thêm một người mới với nghề nghiệp bất kỳ (student, teacher, doctor) vào danh sách người của ward. Tạo ra một ward object, và thêm vào 1 student, 2 teacher, và 2 doctor (sử dụng code ở câu 5,6,7 để tạo person mong muốn). Thực hiện describe() method để in ra tên ward (name) và toàn bộ thông tin của từng người trong ward. Chọn đáp án đúng nhất bên dưới.

```
class Ward:
    def __init__(self, name:str):
        #########YOUR CODE HERE#########

        ################################

    def addPerson(self, person:Person):
        #########YOUR CODE HERE#########

        ################################

    def describe(self):
        #########YOUR CODE HERE#########

        ################################

student1 = Student(name="studentK-111", yob=2012, grade="5")
teacher1 = Teacher(name="teacherK-222", yob=1966, subject="Math")
doctor1 = Doctor(name="doctorK-333", yob=1965, specialist="Endocrinologists")
teacher2 = Teacher(name="teacherK-444", yob=1945, subject="History")
doctor2 = Doctor(name="doctorK-555", yob=1975, specialist="Cardiologists")
ward1 = Ward(name="Ward11")
ward1.addPerson(student1)
ward1.addPerson(teacher1)
ward1.addPerson(teacher2)
ward1.addPerson(doctor1)
ward1.addPerson(doctor2)
ward1.describe()
```

a) Ward Name: Ward11
Student - Name: studentK-111 - YoB: 2012 - Grade: 5
Teacher - Name: teacherK-222 - YoB: 1966 - Subject: Math
Teacher - Name: teacherK-444 - YoB: 1945 - Subject: History
Doctor - Name: doctorK-333 - YoB: 1965 - Specialist: Endocrinologists
Doctor - Name: doctorK-555 - YoB: 1975 - Specialist: Cardiologists
b) Ward Name: Ward22
Student - Name: studentK-111 - YoB: 2012 - Grade: 5
Teacher - Name: Math - YoB: teacherK-222 - Subject: 1966
Teacher - Name: teacherK-444 - YoB: 1945 - Subject: History
Doctor - Name: 1965 - YoB: doctorK-333 - Specialist: Endocrinologists
Doctor - Name: doctorK-555 - YoB: 1975 - Specialist: Cardiologists
c) Tất cả đều đúng
d) Tất cả đều sai

**Câu hỏi 9** (LT) : Thực hiện xây dựng class Stack với các chức năng (method) sau đây: **initialization**

method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa. **.isEmpty()**: kiểm tra stack có đang rỗng. Kết quả đầu ra là gì?

```
class MyStack:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__stack = []

    def isEmpty(self):
        return len(self.__stack) == 0

stack1 = MyStack(capacity=5)
print(stack1.isEmpty())
```

a) True
b) False
c) None
d) Raise an error

**Câu hỏi 10** (LT) : Thực hiện xây dựng class Stack với các chức năng (method) sau đây: **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa (tại đây stack cũng đã được thêm vào các element). **.isFull()**: kiểm tra stack đã full chưa. Kết quả đầu ra là gì?

```
class MyStack:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__stack = [1,2,3,4,5]

    def isFull(self):
        return len(self.__stack) == self.__capacity

stack1 = MyStack(capacity=5)
print(stack1.isFull())
```

a) True
b) False
c) None
d) Raise an error

**Câu hỏi 11** (Code) : Thực hiện xây dựng class Stack với các chức năng (method) sau đây: **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa. **.isFull()**: kiểm tra stack đã full chưa. **.push(value)** add thêm value vào trong stack. Kết quả đầu ra là gì?

```
class MyStack:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__stack = []

    def isFull(self):
        return len(self.__stack) == self.__capacity

    def push(self, value):
        ##########YOUR CODE HERE#########

        ################################

```

```
stack1 = MyStack(capacity=5)
stack1.push(1)
stack1.push(2)
print(stack1.isFull())
```

a) True
b) False
c) None
d) Raise an error

**Câu hỏi 12** (Code) : Thực hiện xây dựng class Stack với các chức năng (method) sau đây: **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa. **.isEmpty()**: kiểm tra stack có đang rỗng. **.isFull()**: kiểm tra stack đã full chưa. **.push(value)** add thêm value vào trong stack. **.top()** lấy giá trị top element hiện tại của stack, nhưng không loại bỏ giá trị đó. Kết quả đầu ra là gì?

```
class MyStack:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__stack = []

    def isEmpty(self):
        return len(self.__stack) == 0

    def isFull(self):
        return len(self.__stack) == self.__capacity

    def push(self, value):
        ##########YOUR CODE HERE#########

        ##################################

    def top(self):
        ##########YOUR CODE HERE#########

        ##################################

stack1 = MyStack(capacity=5)
stack1.push(1)
stack1.push(2)
print(stack1.top())
```

a) 1
b) 2
c) None
d) Raise an error

**Câu hỏi 13** (LT) : Thực hiện xây dựng class Queue với các chức năng (method) sau đây **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa..**isEmpty()**: kiểm tra queue có đang rỗng. Kết quả đầu ra là gì?

```
class MyQueue:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__queue = []

    def isEmpty(self):
        return len(self.__queue) == 0
```

```

queue1 = MyQueue(capacity=5)
print(queue1.isEmpty())
```

a) True
b) False
c) None
d) Raise an error

**Câu hỏi 14** (LT) : Thực hiện xây dựng class Queue với các chức năng (method) sau đây **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa. **.isFull()**: kiểm tra queue đã full chưa. Kết quả đầu ra là gì?

```
class MyQueue:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__queue = []

    def isFull(self):
        return len(self.__queue) == self.__capacity


queue1 = MyQueue(capacity=5)
print(queue1.isFull())
```

a) True
b) False
c) None
d) Raise an error

**Câu hỏi 15** (Code) : Thực hiện xây dựng class Queue với các chức năng (method) sau đây **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa. **.isFull()**: kiểm tra queue đã full chưa. **.enqueue(value)** add thêm value vào trong queue. Kết quả đầu ra là gì?

```
class MyQueue:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__queue = []

    def isFull(self):
        return len(self.__queue) == self.__capacity

    def enqueue(self, value):
        #########YOUR CODE HERE#########

        ################################

queue1 = MyQueue(capacity=5)
queue1.enqueue(1)
queue1.enqueue(2)
print(queue1.isFull())
```

a) True
b) False
c) None
d) Raise an error

**Câu hỏi 16** (Code) : Thực hiện xây dựng class Queue với các chức năng (method) sau đây **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa. **.isFull()**: kiểm tra queue đã full chưa. **.enqueue(value)** add thêm value vào trong queue. **.front()** lấy giá trị first element hiện tại của queue, nhưng không loại bỏ giá trị đó . Kết quả đầu ra là gì?

```
class MyQueue:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__queue = []

    def isEmpty(self):
        return len(self.__queue) == 0

    def isFull(self):
        return len(self.__queue) == self.__capacity

    def enqueue(self, value):
        ##########YOUR CODE HERE#########

        ##################################

    def front(self):
        ##########YOUR CODE HERE#########

        ##################################
queue1 = MyQueue(capacity=5)
queue1.enqueue(1)
queue1.enqueue(2)
print(queue1.front())
```

a) 1
b) 2
c) None
d) Raise an error

### Đáp án

Câu 1: D
Câu 2: B
Câu 3: B
Câu 4: A
Câu 5: A
Câu 6: A
Câu 7: A
Câu 8: A
Câu 9: B
Câu 10: A
Câu 11: B
Câu 12: A
Câu 13: A
Câu 14: B
Câu 15: B
Câu 16: A