

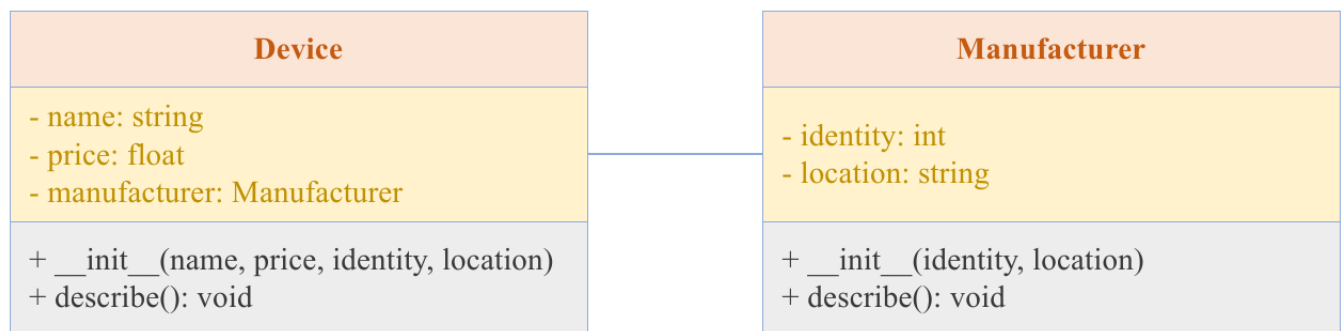
AI VIET NAM – COURSE 2022

Python OOP – Exercise

Ngày 14 tháng 5 năm 2023

I Bài Tập

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



Hình 1: Class Diagram

```
1 # Examples 1
2 device1 = Device(name="mouse", price=2.5, identity=9725, location="Vietnam")
3 device1.describe()
4 >> Name: mouse - Price: 2.5
5 Identity: 9725 - Location: Vietnam
6
7 device2 = Device(name="monitor", price=12.5, identity=11, location="Germany")
8 device2.describe()
9 >> Name: monitor - Price: 12.5
10 Identity: 11 - Location: Germany
```

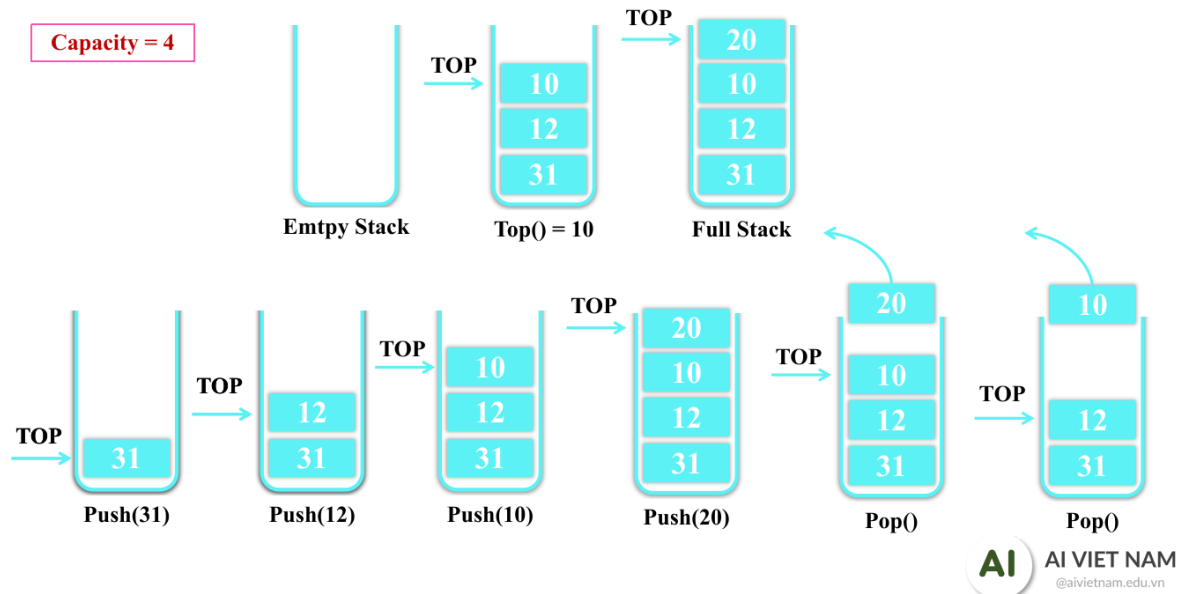
2. Một Ward gồm có name (string) và danh sách của mọi người trong Ward. Một người person có thể là student, doctor, hoặc teacher. Một student gồm có name, yob (int) (năm sinh), và grade (string). Một teacher gồm có name, yob, và subject (string). Một doctor gồm có name, yob, và specialist (string). Lưu ý cần sử dụng a list để chứa danh sách của mọi người trong Ward.
- 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.
 - 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.
 - Viết countDoctor() method để đếm số lượng doctor trong ward.
 - 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)
 - Viết aveTeacherYearOfBirth() method để tính trung bình năm sinh của các teachers trong ward.

```

1 # Examples
2 # 2(a)
3 student1 = Student(name="studentA", yob=2010, grade="7")
4 student1.describe()
5 #output
6 >> 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

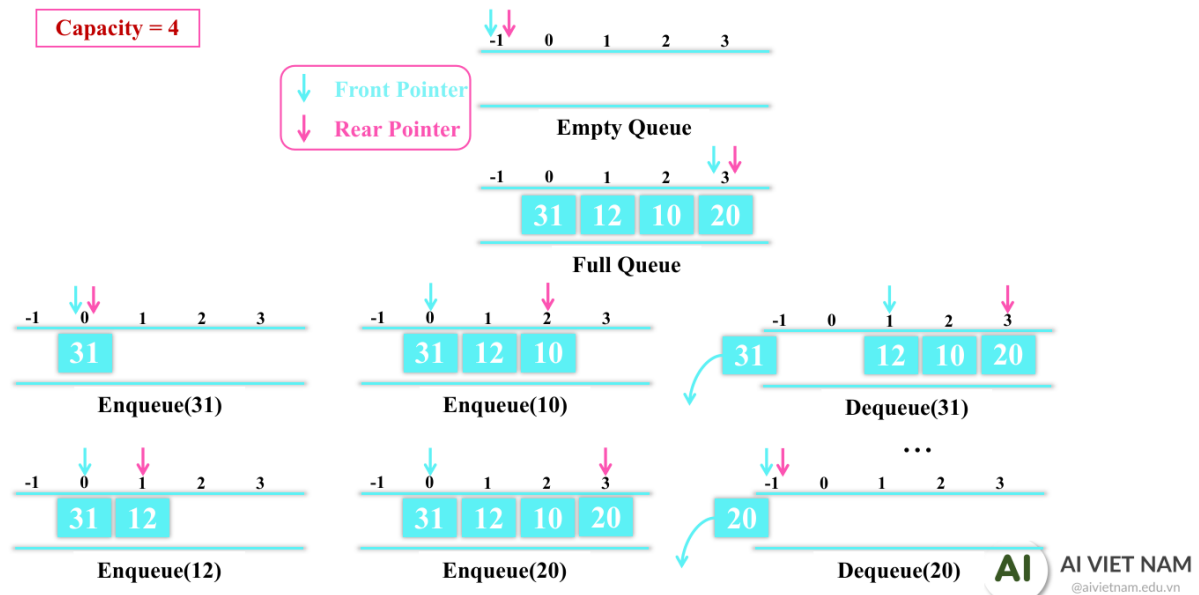
3. Thực hiện xây dựng class Stack với các chức năng (method) sau đây

- **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

```

1 stack1 = MyStack(capacity=5)
2
3 stack1.push(1)
4
5 stack1.push(2)
6
7 print(stack1.isFull())
8 >> False
9
10 print(stack1.top())
11 >> 2
12
13 print(stack1.pop())
14 >> 2
15
16 print(stack1.top())
17 >> 1
18
19 print(stack1.pop())
20 >> 1
21
22 print(stack1.isEmpty())
23 >> 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

```

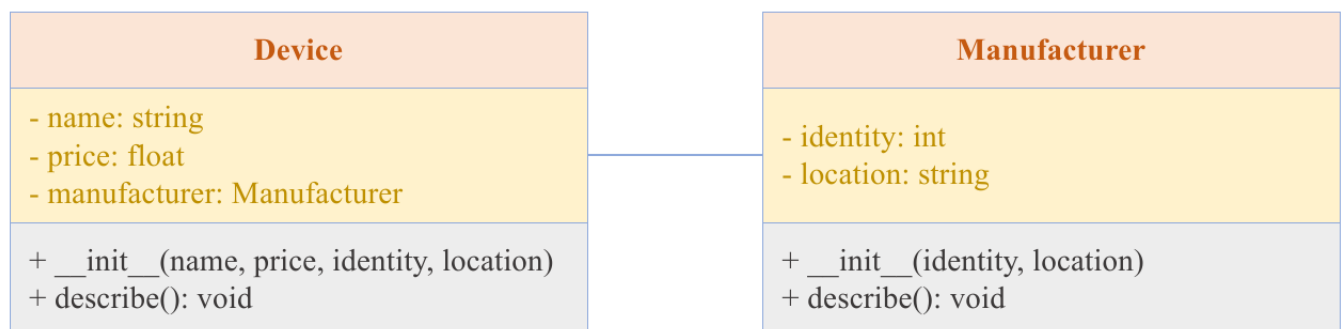
1 queue1 = MyQueue(capacity=5)
2
3 queue1.enqueue(1)
4
5 queue1.enqueue(2)
6
7 print(queue1.isFull())
8 >> False
9
10 print(queue1.front())
11 >> 1
12
13 print(queue1.dequeue())
14 >> 1
15
16 print(queue1.front())
17 >> 2
18
19 print(queue1.dequeue())
20 >> 2
21
22 print(queue1.isEmpty())
23 >> True

```

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 case 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

- Aggregation
- Composition
- Inheritance
- Tất cả đều sai

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

```

1 class Manufacturer:
2     def __init__(self, identity:int, location: str ):
3         self.__identity = identity
4         self.__location = location
5
6     def describe(self):
7         print(f"Identity: {self.__identity} - Location: {self.__location}")
8
9 manu1 = Manufacturer(identity=100, location='Vietnam')
10 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?

```

1 class Manufacturer:
2     def __init__(self, identity:int, location: str ):
3         self.__identity = identity
4         self.__location = location
5
6     def describe(self):
7         print(f"Identity: {self.__identity} - Location: {self.__location}")
8
9
10 class Device:
11     def __init__(self, name:str, price:float, identity:int, location:str ):
12         self.__name = name
13         self.__price = price
14         self.__manufacturer = Manufacturer(identity, location)
15
16     def describe(self):
17         print(f"Name: {self.__name} - Price: {self.__price}")
18         self.__manufacturer.describe()
19 device1 = Device(name="touchpad", price=3.3, identity=1111, location="Vietnam")
20 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?

```

1 class Manufacturer:
2     def __init__(self, identity:int, location: str ):
3         self.__identity = identity
4         self.__location = location
5
6     def describe(self):
7         print(f"Identity: {self.__identity} - Location: {self.__location}")
8
9
10 class Device:
11     def __init__(self, name:str, price:float, manu:Manufacturer):
12         self.__name = name
13         self.__price = price
14         self.__manufacturer = manu
15
16     def describe(self):
17         print(f"Name: {self.__name} - Price: {self.__price}")
18         self.__manufacturer.describe()
19
20 manu1 = Manufacturer(identity=1111, location='Vietnam')
21 device1 = Device(name="touchpad", price=3.3, manu=manu1)
22 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.

```

1 from abc import ABC, abstractmethod
2
3 class Person(ABC):
4     def __init__(self, name:str, yob:int):
5         self._name = name
6         self._yob = yob
7
8     def getYoB(self):
9         return self._yob
10
11     @abstractmethod
12     def describe(self):
13         pass
14
15
16 class Student(Person):
17     def __init__(self, name:str, yob:int, grade:str):
18         #####YOUR CODE HERE#####
19
20         #####
21     def describe(self):
22         #####YOUR CODE HERE#####
23
24         #####
25
26 student1 = Student(name="studentZ2023", yob=2011, grade="6")
27 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.

```

1 from abc import ABC, abstractmethod
2
3 class Person(ABC):
4     def __init__(self, name:str, yob:int):
5         self._name = name
6         self._yob = yob
7
8     def getYoB(self):
9         return self._yob
10
11     @abstractmethod

```



```

12     def describe(self):
13         pass
14
15
16 class Teacher(Person):
17     def __init__(self, name:str, yob:int, subject:str):
18         #####YOUR CODE HERE#####
19
20         #####
21
22     def describe(self):
23         #####YOUR CODE HERE#####
24
25         #####
26
27 teacher1 = Teacher(name="teacherZ2023", yob=1991, subject="History")
28 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.

```

1 from abc import ABC, abstractmethod
2
3 class Person(ABC):
4     def __init__(self, name:str, yob:int):
5         self._name = name
6         self._yob = yob
7
8     def getYoB(self):
9         return self._yob
10
11     @abstractmethod
12     def describe(self):
13         pass
14
15
16 class Doctor(Person):
17     def __init__(self, name:str, yob:int, specialist:str):
18         #####YOUR CODE HERE#####
19
20         #####
21
22     def describe(self):
23         #####YOUR CODE HERE#####
24
25         #####
26
27 doctor1 = Doctor(name="doctorZ2023", yob=1981, specialist="Endocrinologists")
28 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.

```

1 class Ward:
2     def __init__(self, name:str):
3         #####YOUR CODE HERE#####
4
5         #####
6
7     def addPerson(self, person:Person):
8         #####YOUR CODE HERE#####
9
10        #####
11
12    def describe(self):
13        #####YOUR CODE HERE#####
14
15        #####
16
17 student1 = Student(name="studentK-111", yob=2012, grade="5")
18 teacher1 = Teacher(name="teacherK-222", yob=1966, subject="Math")
19 doctor1 = Doctor(name="doctorK-333", yob=1965, specialist="Endocrinologists")
20 teacher2 = Teacher(name="teacherK-444", yob=1945, subject="History")
21 doctor2 = Doctor(name="doctorK-555", yob=1975, specialist="Cardiologists")
22 ward1 = Ward(name="Ward11")
23 ward1.addPerson(student1)
24 ward1.addPerson(teacher1)
25 ward1.addPerson(teacher2)
26 ward1.addPerson(doctor1)
27 ward1.addPerson(doctor2)
28 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?

```

1 class MyStack:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__stack = []
5
6     def isEmpty(self):
7         return len(self.__stack) == 0
8
9 stack1 = MyStack(capacity=5)
10 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?

```

1 class MyStack:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__stack = [1,2,3,4,5]
5
6     def isFull(self):
7         return len(self.__stack) == self.__capacity
8
9 stack1 = MyStack(capacity=5)
10 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?

```

1 class MyStack:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__stack = []
5
6     def isFull(self):
7         return len(self.__stack) == self.__capacity
8
9     def push(self, value):
10         #####YOUR CODE HERE#####
11
12         #####
13

```

```

14 stack1 = MyStack(capacity=5)
15 stack1.push(1)
16 stack1.push(2)
17 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?

```

1 class MyStack:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__stack = []
5
6     def isEmpty(self):
7         return len(self.__stack) == 0
8
9     def isFull(self):
10        return len(self.__stack) == self.__capacity
11
12    def push(self, value):
13        #####YOUR CODE HERE#####
14
15        #####
16
17    def top(self):
18        #####YOUR CODE HERE#####
19
20        #####
21
22 stack1 = MyStack(capacity=5)
23 stack1.push(1)
24 stack1.push(2)
25 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?

```

1 class MyQueue:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__queue = []
5
6     def isEmpty(self):
7         return len(self.__queue) == 0

```

```
8
9 queue1 = MyQueue(capacity=5)
10 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?

```
1 class MyQueue:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__queue = []
5
6     def isFull(self):
7         return len(self.__queue) == self.__capacity
8
9
10 queue1 = MyQueue(capacity=5)
11 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?

```
1 class MyQueue:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__queue = []
5
6     def isFull(self):
7         return len(self.__queue) == self.__capacity
8
9     def enqueue(self, value):
10         #####YOUR CODE HERE#####
11
12         #####
13
14 queue1 = MyQueue(capacity=5)
15 queue1.enqueue(1)
16 queue1.enqueue(2)
17 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?

```
1 class MyQueue:
2     def __init__(self, capacity):
3         self.__capacity = capacity
4         self.__queue = []
5
6     def isEmpty(self):
7         return len(self.__queue) == 0
8
9     def isFull(self):
10        return len(self.__queue) == self.__capacity
11
12    def enqueue(self, value):
13        #####YOUR CODE HERE#####
14
15        #####
16
17    def front(self):
18        #####YOUR CODE HERE#####
19
20        #####
21 queue1 = MyQueue(capacity=5)
22 queue1.enqueue(1)
23 queue1.enqueue(2)
24 print(queue1.front())
```

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